



# External Memory Interfaces Intel® Agilex™ 7 F-Series and I-Series FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **23.2**

IP Version: **2.7.1**

## Contents

---

<b>1. About the External Memory Interfaces Intel® Agilex™ 7 F-Series and I-Series FPGA IP.....</b>	<b>8</b>
1.1. Release Information.....	8
<b>2. Intel Agilex™ 7 F-Series and I-Series EMIF IP – Introduction.....</b>	<b>9</b>
2.1. Intel Agilex 7 F-Series and I-Series EMIF IP Protocol and Feature Support.....	9
2.2. Intel Agilex 7 F-Series and I-Series EMIF IP Design Flow.....	9
2.3. Intel Agilex 7 F-Series and I-Series EMIF IP Design Checklist.....	10
<b>3. Intel Agilex 7 F-Series and I-Series EMIF IP – Product Architecture.....</b>	<b>12</b>
3.1. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Introduction.....	12
3.1.1. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Subsystem.....	13
3.1.2. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O SSM.....	14
3.1.3. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Bank.....	15
3.1.4. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Lane.....	29
3.1.5. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Input DQS Clock Tree...	37
3.1.6. Intel Agilex 7 F-Series and I-Series EMIF Architecture: PHY Clock Tree.....	38
3.1.7. Intel Agilex 7 F-Series and I-Series EMIF Architecture: PLL Reference Clock Networks.....	38
3.1.8. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Clock Phase Alignment.	40
3.2. Intel Agilex 7 F-Series and I-Series EMIF Sequencer.....	41
3.3. Intel Agilex 7 F-Series and I-Series EMIF Calibration.....	42
3.3.1. Intel Agilex 7 F-Series and I-Series Calibration Stages .....	42
3.3.2. Intel Agilex 7 F-Series and I-Series Calibration Stages Descriptions.....	43
3.3.3. Intel Agilex 7 F-Series and I-Series Calibration Flowchart.....	44
3.3.4. Intel Agilex 7 F-Series and I-Series Calibration Algorithms.....	44
3.4. Intel Agilex 7 F-Series and I-Series EMIF Controller.....	59
3.4.1. Hard Memory Controller.....	59
3.4.2. Intel Agilex 7 F-Series and I-Series Hard Memory Controller Rate Conversion Feature.....	63
3.5. User-requested Reset in Intel Agilex 7 F-Series and I-Series EMIF IP.....	63
3.6. Intel Agilex 7 F-Series and I-Series EMIF for Hard Processor Subsystem.....	66
3.6.1. Restrictions on I/O Bank Usage for Intel Agilex 7 F-Series and I-Series EMIF IP with HPS.....	66
3.7. Using a Custom Controller with the Hard PHY.....	72
<b>4. Intel Agilex 7 F-Series and I-Series EMIF IP – End-User Signals.....</b>	<b>73</b>
4.1. Intel Agilex 7 F-Series and I-Series EMIF IP Interface and Signal Descriptions.....	73
4.1.1. Intel Agilex 7 EMIF IP Interfaces for DDR4.....	73
4.1.2. Intel Agilex 7 EMIF IP Interfaces for QDR-IV.....	81
4.2. Intel Agilex 7 F-Series and I-Series EMIF IP AFI Signals.....	87
4.2.1. AFI Clock and Reset Signals.....	87
4.2.2. AFI Address and Command Signals.....	87
4.2.3. AFI Write Data Signals.....	88
4.2.4. AFI Read Data Signals.....	89
4.2.5. AFI Calibration Status Signals.....	89
4.2.6. AFI Tracking Management Signals.....	90
4.2.7. AFI Shadow Register Management Signals.....	90

4.3. Intel Agilex 7 F-Series and I-Series EMIF IP AFI 4.0 Timing Diagrams.....	91
4.3.1. AFI Address and Command Timing Diagrams.....	92
4.3.2. AFI Write Sequence Timing Diagrams.....	93
4.3.3. AFI Read Sequence Timing Diagrams.....	98
4.3.4. AFI Calibration Status Timing Diagram.....	100
4.4. Intel Agilex 7 F-Series and I-Series EMIF IP Memory Mapped Register (MMR) Tables...	101
4.4.1. ctrlcfg0.....	102
4.4.2. ctrlcfg1.....	102
4.4.3. dramtiming0.....	104
4.4.4. sbcfg1.....	104
4.4.5. caltiming0.....	104
4.4.6. caltiming1.....	105
4.4.7. caltiming2.....	105
4.4.8. caltiming3.....	105
4.4.9. caltiming4.....	106
4.4.10. caltiming9.....	106
4.4.11. dramaddrw.....	106
4.4.12. sideband0.....	107
4.4.13. sideband1.....	107
4.4.14. sideband4.....	107
4.4.15. sideband6.....	107
4.4.16. sideband7.....	107
4.4.17. sideband9.....	108
4.4.18. sideband11.....	108
4.4.19. sideband12.....	108
4.4.20. sideband13.....	109
4.4.21. sideband14.....	109
4.4.22. dramsts.....	110
4.4.23. niosreserve0.....	110
4.4.24. niosreserve1.....	110
4.4.25. sideband16.....	110
4.4.26. ecc3: ECC Error and Interrupt Configuration.....	111
4.4.27. ecc4: Status and Error Information.....	111
4.4.28. ecc5: Address of Most Recent SBE/DBE.....	112
4.4.29. ecc6: Address of Most Recent Correction Command Dropped.....	112
4.4.30. ecc7: Extension for Address of Most Recent SBE/DBE.....	113
4.4.31. ecc8: Extension for Address of Most Recent Correction Command Dropped...	113
<b>5. Intel Agilex 7 F-Series and I-Series FPGA IP – Simulating Memory IP.....</b>	<b>114</b>
5.1. Simulation Options.....	114
5.2. Simulation Walkthrough.....	115
5.2.1. Calibration Modes.....	116
5.2.2. Simulation Scripts.....	117
5.2.3. Functional Simulation with Verilog HDL.....	117
5.2.4. Functional Simulation with VHDL.....	118
5.2.5. Simulating the Design Example.....	118
5.3. Simulating the Design Example with Mentor Graphics* AXI4 Master BFM (Intel FPGA Edition) .....	119
5.3.1. Overview of Steps.....	119
5.3.2. Generating the EMIF Design Example.....	120
5.3.3. Editing the Simulation Design Example with the Platform Designer.....	121

5.3.4. Editing the ed_sim.v File.....	125
5.3.5. Obtaining the master_test_program.sv File.....	126
5.3.6. Running the Simulation .....	127
<b>6. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – DDR4 Support.....</b>	<b>129</b>
6.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Parameter Descriptions.....	129
6.1.1. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: General.....	129
6.1.2. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Memory.....	131
6.1.3. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Mem I/O.....	133
6.1.4. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: FPGA I/O.....	136
6.1.5. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Mem Timing..	138
6.1.6. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Controller.....	141
6.1.7. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Diagnostics....	144
6.1.8. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Example Designs.....	145
6.2. Intel Agilex 7 F-Series and I-Series External Memory Interfaces Intel Calibration IP Parameters.....	147
6.3. Register Map IP-XACT Support for Intel Agilex 7 F-Series and I-Series EMIF DDR4 IP..	148
6.4. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Pin and Resource Planning.....	149
6.4.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Interface Pins.....	149
6.4.2. Intel Agilex 7 FPGA EMIF IP Resources.....	151
6.4.3. Pin Guidelines for Intel Agilex 7 F-Series and I-Series FPGA EMIF IP.....	152
6.5. DDR4 Board Design Guidelines.....	162
6.5.1. Terminations for DDR4 with Intel Agilex 7 F-Series and I-Series Devices.....	163
6.5.2. Clamshell Topology.....	165
6.5.3. General Layout Routing Guidelines.....	166
6.5.4. Reference Stackup.....	170
6.5.5. Intel Agilex 7 F-Series and I-Series EMIF-Specific Routing Guidelines for Various DDR4 Topologies.....	172
6.5.6. DDR4 Routing Guidelines: Discrete (Component) Topologies.....	184
6.5.7. Intel Agilex 7 F-Series and I-Series EMIF Pin Swapping Guidelines.....	193
<b>7. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – QDR-IV Support.....</b>	<b>197</b>
7.1. Intel Agilex 7 FPGA EMIF IP Parameter Descriptions.....	197
7.1.1. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: General.....	197
7.1.2. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Memory.....	199
7.1.3. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: FPGA I/O....	200
7.1.4. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Mem Timing	202
7.1.5. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Controller...	202
7.1.6. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Diagnostics.	203
7.1.7. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Example Designs.....	204
7.2. Intel Agilex 7 F-Series and I-Series External Memory Interfaces Intel Calibration IP Parameters.....	205
7.3. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Pin and Resource Planning.....	206
7.3.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Interface Pins.....	206
7.3.2. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Resources.....	207
7.3.3. Pin Guidelines for Intel Agilex 7 F-Series and I-Series FPGA EMIF IP.....	208
7.4. QDR-IV Board Design Guidelines.....	215
7.4.1. General Layout Routing Guidelines.....	216
7.4.2. Reference Stackup.....	218
7.4.3. Routing Guidelines for QDR-IV Topology.....	222

7.4.4. Intel Agilex 7 EMIF Design Guideline Summary.....	224
<b>8. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Timing Closure.....</b>	<b>225</b>
8.1. Timing Closure .....	225
8.1.1. Timing Analysis.....	225
8.2. Optimizing Timing.....	226
<b>9. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – I/O Timing Closure.....</b>	<b>228</b>
9.1. I/O Timing Closure Overview.....	228
9.2. Collateral Generated with Your EMIF IP.....	230
9.3. SPICE Decks.....	230
9.3.1. Address/Command Simulation Deck.....	231
9.3.2. FPGA Write Operation Simulation Deck.....	234
9.3.3. FPGA Read Operation Simulation Deck.....	236
9.4. File Organization.....	237
9.5. Top-level Parameterization File .....	238
9.6. IP-Supplied Parameters that You Might Need to Override.....	239
9.7. Understanding the *_ip_parameters.dat File and Making a Mask Polygon.....	241
9.8. Multi-Rank Topology.....	244
9.9. Pin Parasitics.....	245
9.10. Mask Evaluation.....	245
<b>10. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Controller Optimization.....</b>	<b>247</b>
10.1. Interface Standard.....	247
10.2. Bank Management Efficiency.....	248
10.3. Data Transfer.....	248
10.4. Improving Controller Efficiency.....	248
10.4.1. Auto-Precache Commands.....	249
10.4.2. Additive Latency.....	251
10.4.3. Bank Interleaving.....	252
10.4.4. Additive Latency and Bank Interleaving.....	254
10.4.5. User-Controlled Refresh.....	255
10.4.6. Frequency of Operation.....	256
10.4.7. Series of Reads or Writes.....	256
10.4.8. Data Reordering.....	256
10.4.9. Starvation Control.....	257
10.4.10. Command Reordering.....	257
10.4.11. Bandwidth.....	259
10.4.12. Enable Command Priority Control.....	259
10.4.13. Controller Pre-pay and Post-pay Refresh (DDR4 Only).....	259
<b>11. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Debugging.....</b>	<b>261</b>
11.1. Interface Configuration Performance Issues.....	261
11.1.1. Interface Configuration Bottleneck and Efficiency Issues.....	261
11.2. Functional Issue Evaluation.....	262
11.2.1. Intel IP Memory Model.....	263
11.2.2. Vendor Memory Model.....	263
11.2.3. Transcript Window Messages.....	263
11.2.4. Modifying the Example Driver to Replicate the Failure.....	265
11.3. Timing Issue Characteristics.....	266
11.3.1. Evaluating FPGA Timing Issues.....	266
11.3.2. Evaluating External Memory Interface Timing Issues.....	267

11.4. Verifying Memory IP Using the Signal Tap Logic Analyzer.....	268
11.4.1. Signals to Monitor with the Signal Tap Logic Analyzer.....	269
11.5. Hardware Debugging Guidelines.....	269
11.5.1. Create a Simplified Design that Demonstrates the Same Issue.....	269
11.5.2. Measure Power Distribution Network.....	269
11.5.3. Measure Signal Integrity and Setup and Hold Margin.....	270
11.5.4. Vary Voltage.....	270
11.5.5. Operate at a Lower Speed.....	270
11.5.6. Determine Whether the Issue Exists in Previous Versions of Software.....	270
11.5.7. Determine Whether the Issue Exists in the Current Version of Software.....	270
11.5.8. Try A Different PCB.....	271
11.5.9. Try Other Configurations.....	271
11.5.10. Debugging Checklist.....	272
11.6. Categorizing Hardware Issues.....	272
11.6.1. Signal Integrity Issues.....	273
11.6.2. Hardware and Calibration Issues.....	275
11.7. Debugging with the External Memory Interface Debug Toolkit .....	276
11.7.1. Prerequisites for Using the EMIF Debug Toolkit.....	277
11.7.2. Configuring a Design to Use the Toolkit.....	277
11.7.3. Launching the EMIF Debug Toolkit.....	280
11.7.4. Using the EMIF Debug Toolkit.....	282
11.7.5. Exporting Tables.....	297
11.7.6. Viewing Reports Graphically in the Eye Viewer.....	297
11.8. Using the Default Traffic Generator.....	300
11.8.1. Reading the Default Traffic Generator Status.....	302
11.8.2. Running Infinite Traffic using the Default Traffic Generator.....	303
11.8.3. Changing the Reset Trigger of the Default Traffic Generator.....	304
11.8.4. Observing Generated Traffic with Signal Tap.....	304
11.9. Using the Configurable Traffic Generator (TG2) .....	305
11.9.1. Enabling the Traffic Generator in a Design Example.....	305
11.9.2. Traffic Generator Block Description.....	306
11.9.3. Default Traffic Pattern.....	307
11.9.4. Configuration and Status Registers.....	307
11.9.5. User Pattern.....	312
11.9.6. Traffic Generator Status.....	327
11.9.7. Starting Traffic with the Traffic Generator.....	329
11.9.8. Traffic Generator Configuration User Interface.....	330
11.10. EMIF On-Chip Debug Port.....	346
11.10.1. Enabling the On-Chip Debug Port.....	347
11.10.2. I/O SSM calbus Bridge Data Structures and Usage.....	347
11.10.3. I/O SSM User-RAM Data Structures and Usage.....	350
11.10.4. Debug Interface Structure.....	364
11.10.5. Example: Reading Calibration Results and Margins with the On-Chip Debug Port.....	371
11.11. Efficiency Monitor.....	376
11.11.1. Enabling the Efficiency Monitor in a Design Example.....	376
11.11.2. Efficiency Monitor Block Descriptions.....	376
11.11.3. Control and Status Registers.....	377
11.11.4. Opening the Efficiency Monitor Toolkit.....	379

<b>12. External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA IP User Guide Archives.....</b>	<b>383</b>
<b>13. Document Revision History for External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA IP User Guide.....</b>	<b>384</b>

## 1. About the External Memory Interfaces Intel® Agilex™ 7 F-Series and I-Series FPGA IP

---

### 1.1. 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	2.7.1
Intel Quartus Prime	23.2
Release Date	2023.06.26

#### Related Information

[External Memory Interfaces Intel Agilex 7 FPGA IP Core Release Notes](#)

## 2. Intel Agilex™ 7 F-Series and I-Series FPGA EMIF 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 Agilex™ 7 F-Series and I-Series FPGA IP* (referred to hereafter as the *Intel Agilex 7 F-Series and I-Series 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*, available here: <https://www.intel.com/content/www/us/en/programmable/support/support-resources/support-centers/external-memory-interfaces-support/emif.html>.

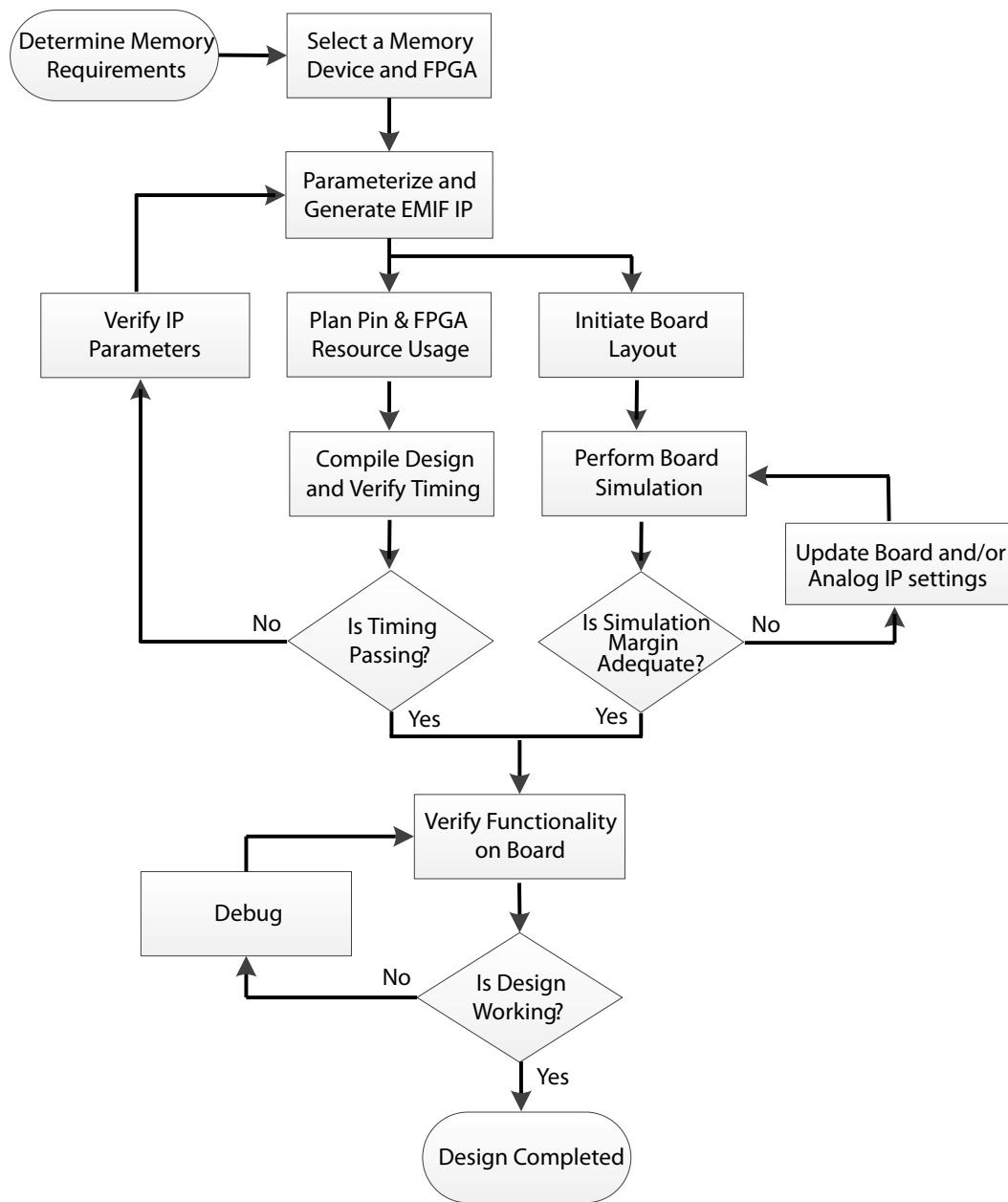
### 2.1. Intel Agilex 7 F-Series and I-Series EMIF IP Protocol and Feature Support

- The Intel Agilex 7 F-Series and I-Series FPGA EMIF IP supports DDR4 with hard memory controller and hard PHY.
- The Intel Agilex 7 F-Series and I-Series FPGA EMIF IP supports QDR-IV with soft memory controller and hard PHY.

### 2.2. Intel Agilex 7 F-Series and I-Series 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**


### 2.3. Intel Agilex 7 F-Series and I-Series EMIF IP Design Checklist

Refer to the following checklist as a quick reference for information about steps 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 the resources listed in the right column.	<ul style="list-style-type: none"> <li><a href="#">External Memory Interfaces Support Center</a></li> <li><a href="#">External Memory Interface Spec Estimator</a></li> </ul>
Parameterize the IP	Correct IP parameterization is important for good EMIF IP operation. The resources listed in the right column define the memory parameters during IP generation.	<ul style="list-style-type: none"> <li><a href="#">DDR4 Parameter Descriptions</a></li> <li><a href="#">QDR-IV Parameter Descriptions</a></li> </ul>
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.	<ul style="list-style-type: none"> <li><a href="#">Design Example Quick Start Guide</a></li> </ul>
Perform functional simulation	Simulation of the EMIF design helps to determine correct operation. The resources listed in the right column explain how to perform simulation and what differences exist between simulation and hardware implementation.	<ul style="list-style-type: none"> <li><a href="#">Design Example Quick Start Guide</a></li> <li><a href="#">Simulating Memory IP</a></li> </ul>
Make pin assignments	For guidance on pin placement, refer to the resources listed in the right column.	<ul style="list-style-type: none"> <li><a href="#">DDR4 Parameter Descriptions</a></li> <li><a href="#">QDR-IV Parameter Descriptions</a></li> <li><a href="#">Device Pin Tables</a></li> </ul>
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 the resources listed in the right column.	<ul style="list-style-type: none"> <li><a href="#">Board Design Guidelines</a></li> <li><a href="#">I/O Timing Closure</a></li> </ul>
Verify timing closure	For information regarding compilation, system-level timing closure and timing reports refer to the Timing Closure section of this User Guide.	<ul style="list-style-type: none"> <li><a href="#">Timing Closure</a></li> </ul>
Run the design on hardware	For instructions on how to program a FPGA refer to the Quick-Start section of the Design Example User Guide.	<ul style="list-style-type: none"> <li><a href="#">Design Example Quick Start Guide</a></li> </ul>
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. The resources listed in the right column contain information on typical debug procedures and available tools to help diagnose hardware issues.	<ul style="list-style-type: none"> <li><a href="#">Debugging</a></li> <li><a href="#">External Memory Interfaces Support Center</a></li> </ul>

## 3. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Product Architecture

This chapter describes the Intel Agilex 7 F-Series and I-Series FPGA EMIF IP product architecture.

### 3.1. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Introduction

The Intel Agilex 7 F-Series and I-Series 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.

*Note:* The current version of the External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA IP supports the DDR4 and QDR-IV memory protocols.

The following are key hardware features of the Intel Agilex 7 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, for interfaces placed on the same edge of the FPGA.

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

#### Hard PHY

The PHY circuitry in Intel Agilex 7 F-Series and I-Series 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 DDR4 memory protocol.

#### PHY-Only Mode

Protocols that use a hard controller provide a PHY-only option, which generates only the PHY and sequencer, but not the controller. This PHY-only mode is available if you want to implement your own custom controller in the FPGA fabric, rather than using the hardened controller in the I/O subsystem or the soft controllers.

### High-Speed PHY Clock Tree

Dedicated high speed PHY clock networks clock the I/O buffers in Intel Agilex 7 F-Series and I-Series 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.

### Related Information

[Using a Custom Controller with the Hard PHY](#) on page 72

## 3.1.1. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Subsystem

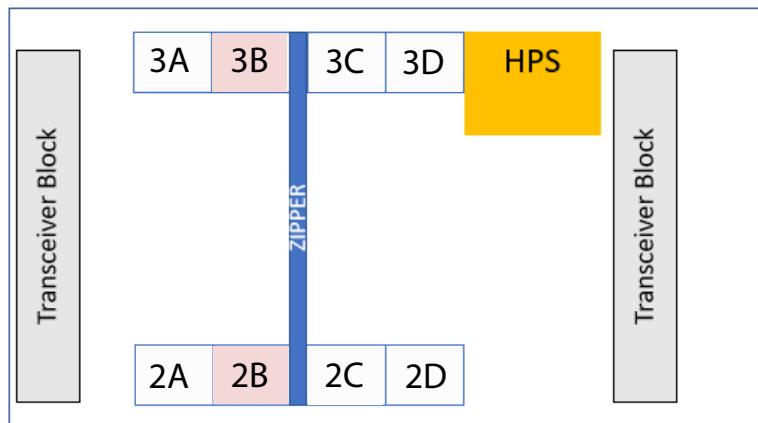
In Intel Agilex 7 F-Series and I-Series devices, the I/O subsystem consists of two rows at the edge of the core.

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
  - I/O Bank I/O PLL for external memory interfaces and user logic
  - Fabric-feeding for non-EMIF/non-LVDS SERDES IP applications
- True Differential Signaling
- External memory interface components, as follows:
  - Hard memory controller
  - Hard PHY
  - Hard Nios processor and calibration logic
  - DLL

## Figure 2. Intel Agilex 7 F-Series and I-Series I/O Subsystem

The following figure depicts the I/O subsystem structure for AGF014/AGF012 series devices. I/O banks 2B and 3B are the locations of calibration I/O SSMs in these devices.



### 3.1.2. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O SSM

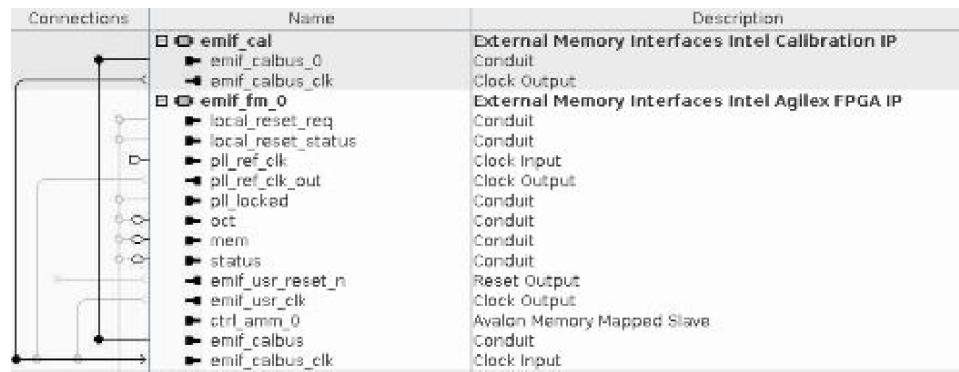
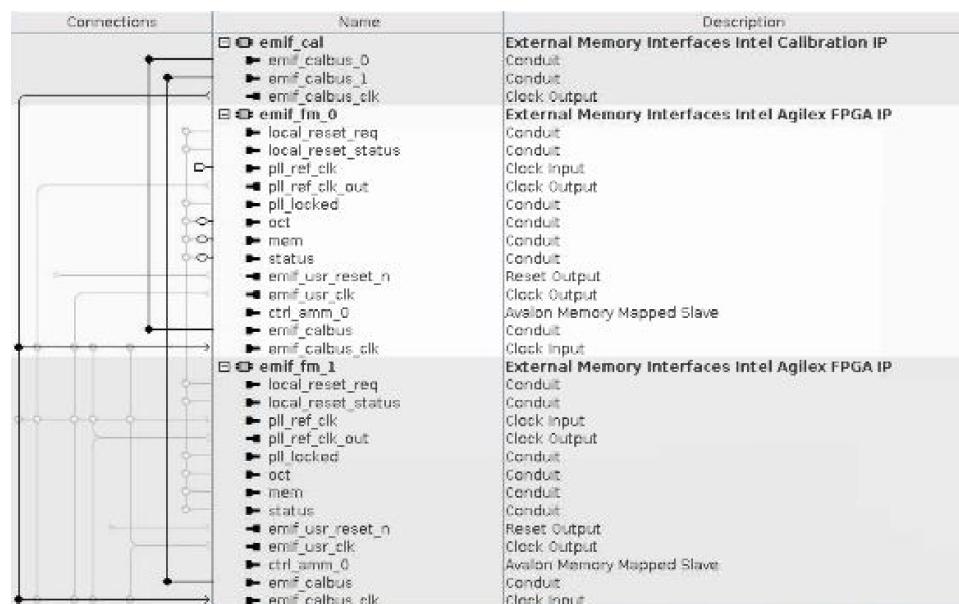
Each I/O row includes one I/O subsystem manager (I/O SSM), which contains a hardened Nios II processor with dedicated memory. The I/O SSM is responsible for calibration of all the EMIFs in the I/O row. There is one I/O SSM in the top row and bottom row. Its location is fixed in one of the I/O banks along each edge and depends on the base die.

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® memory-mapped interface bus.

The I/O SSM is clocked by the on-chip configuration network, and therefore does not consume a PLL.

Each EMIF instance must be connected to the I/O SSM through the External Memory Interfaces Calibration IP. The Calibration IP exposes a calibration bus master port, which must be connected to the slave calibration bus port on every EMIF instance.

Only one calibration IP is allowed for each I/O row. All the EMIFs in the same I/O row must be connected to the same calibration I/P. You can specify the number of EMIF interfaces to be connected to the calibration IP when parameterizing the IP. Connect the `emif_calbus` and `emif_calbus_clk` on the calibration IP to the `emif_calbus` and `emif_calbus_clk`, respectively, on the EMIF IP core.

**Figure 3. Connectivity Between Calibration IP and Single EMIF Interface****Figure 4. Connectivity Between Calibration IP and Multiple EMIF Interfaces on the Same I/O Row**

### 3.1.3. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Bank

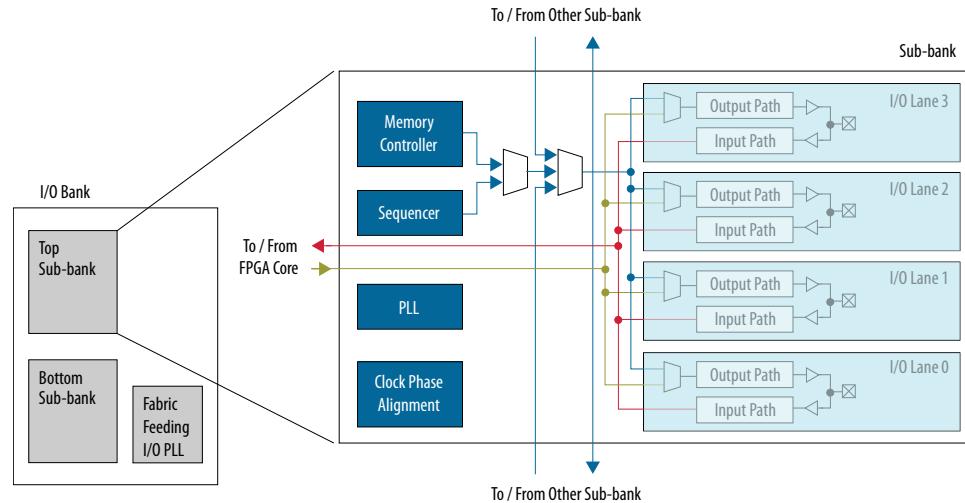
Each I/O row contains up to four I/O banks; the exact number of banks depends on device size and pin package.

Each I/O bank consists of two sub-banks, and each sub-bank contains the following components:

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

A single I/O sub-bank contains all the hardware needed to build an external memory interface. You can make a wider interface by connecting multiple adjacent sub-banks together.

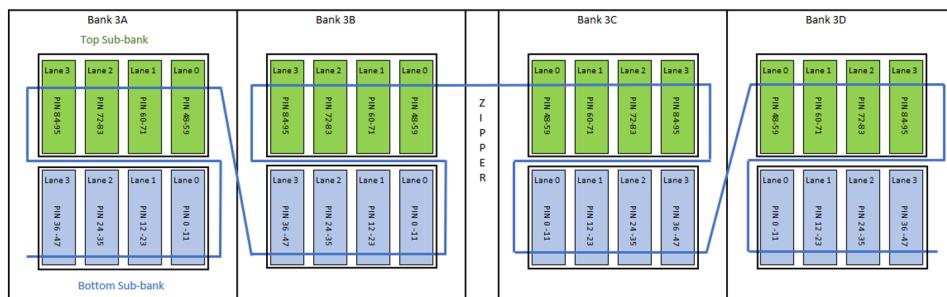
**Figure 5. I/O Bank Architecture in Intel Agilex 7 F-Series and I-Series Devices**



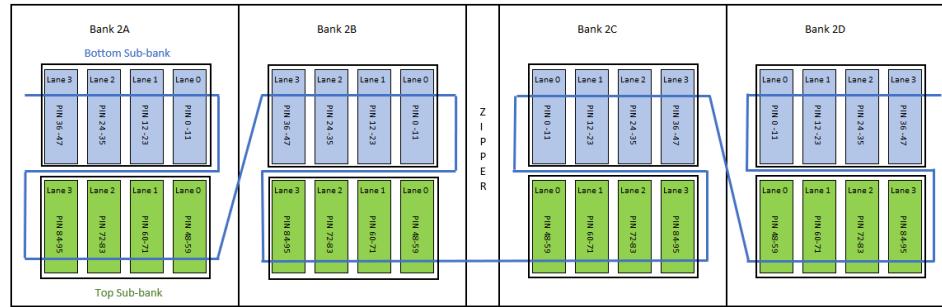
Within an I/O bank, the top sub-bank is placed near the edge of the die, and the bottom sub-bank is placed near the FPGA core.

There are interconnects between the sub-banks which chain the sub-banks into a row. The following figures show how I/O lanes in various sub-banks are chained together to form the top and bottom I/O rows in various Intel Agilex 7 F-Series and I-Series device variants. These figures represent the top view of the silicon die that corresponds to a reverse view of the device package.

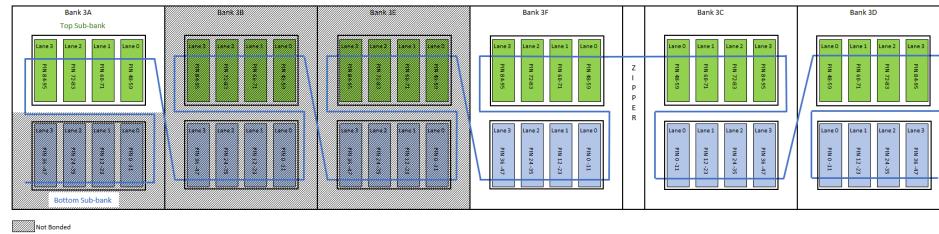
**Figure 6. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF012 and AGF014, package R24A/R24B**



**Figure 7. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF012 and AGF014, package R24A/R24B**

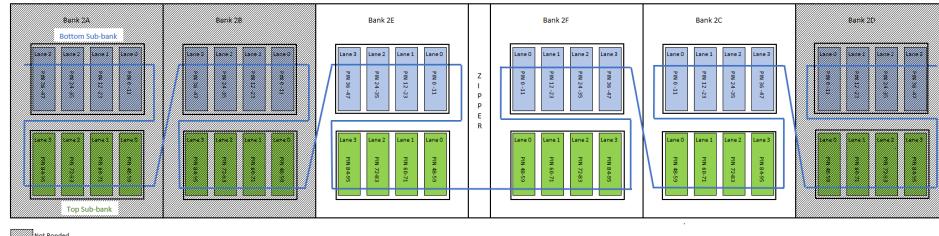


**Figure 10. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R25A**

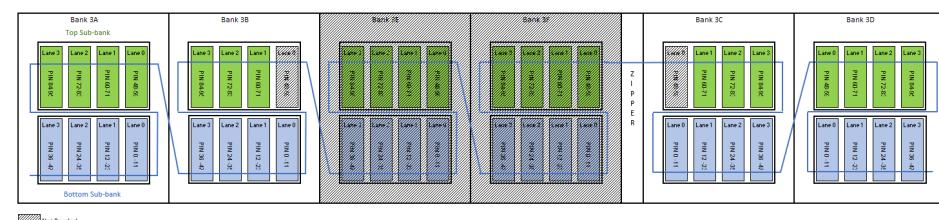


On the top I/O row in AGF022 and AGF027 package R25A , an EMIF interface cannot span across Bank 3A and Bank 3F, because the two I/O banks are not adjacent to each other.

**Figure 11. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R25A**

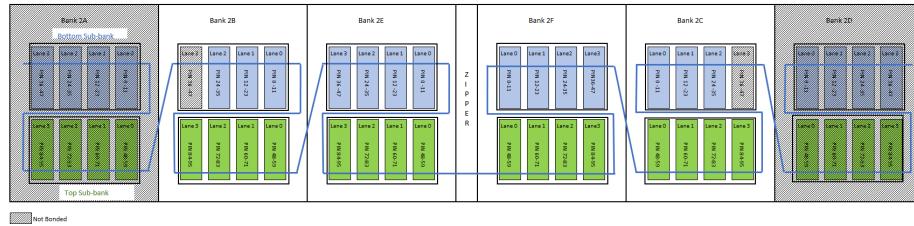


**Figure 12. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027 devices, package R29A**



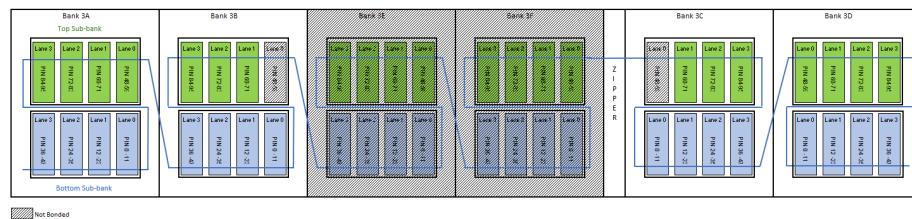
Bank 3E, Bank 3F, I/O Lane 0 in Top Sub-bank 3B and I/O Lane 0 in Top Sub-bank 3C are not bonded.

**Figure 13. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027 devices, package R29A**



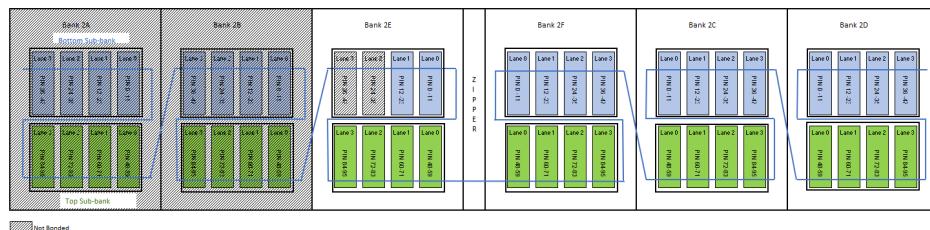
Bank 2A, Bank 2D, I/O Lane 3 in Bottom Sub-bank 2B and I/O Lane 3 in Bottom Sub-bank 2C are not bonded.

**Figure 14. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027 devices, package R31B**



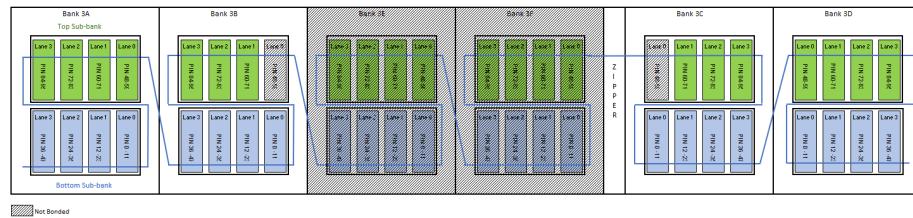
Bank 3E, Bank 3F, I/O Lane 0 in Top Sub-bank 3B and I/O Lane 0 in Top Sub-bank 3C are not bonded.

**Figure 15. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027 devices, package R31B**



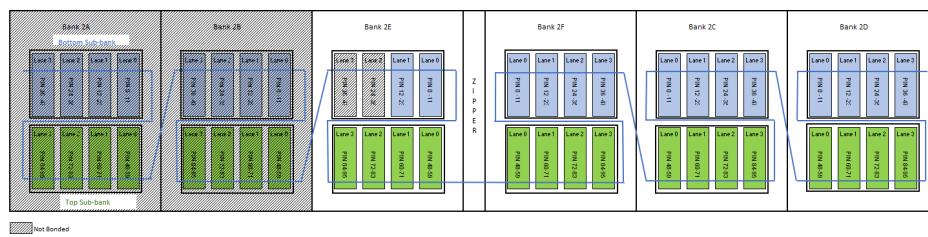
Bank 2A, Bank 2B, I/O Lane 3 and I/O Lane 2 in Bottom Sub-bank 2E are not bonded.

**Figure 16. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R31C**



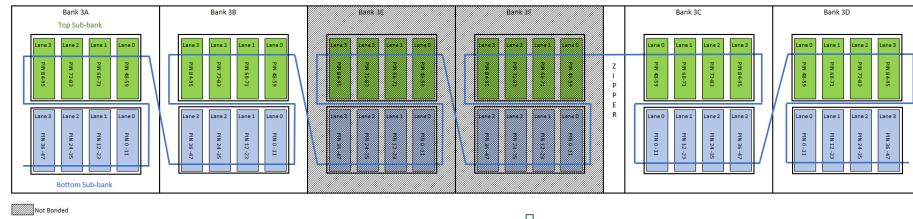
Bank 3E, Bank 3F, I/O Lane 0 in Top Sub-bank 3B and I/O Lane 0 in Top Sub-bank 3C are not bonded.

**Figure 17. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R31C**

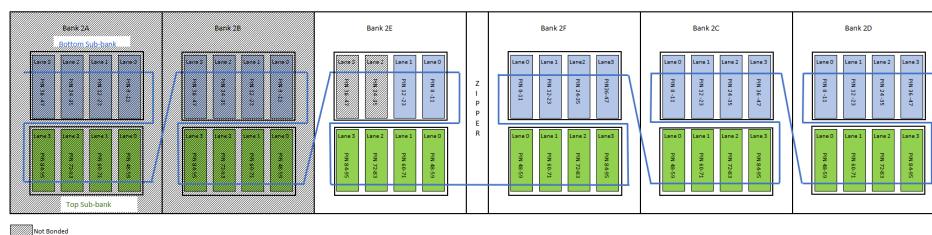


Bank 2A, Bank 2B, I/O Lane 3 and I/O Lane 2 in Bottom Sub-bank 2E are not bonded

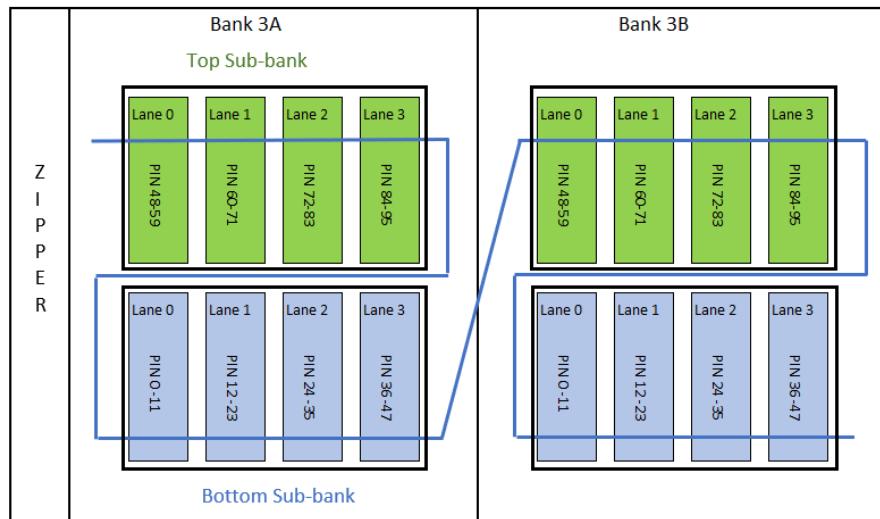
**Figure 18. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R24C**



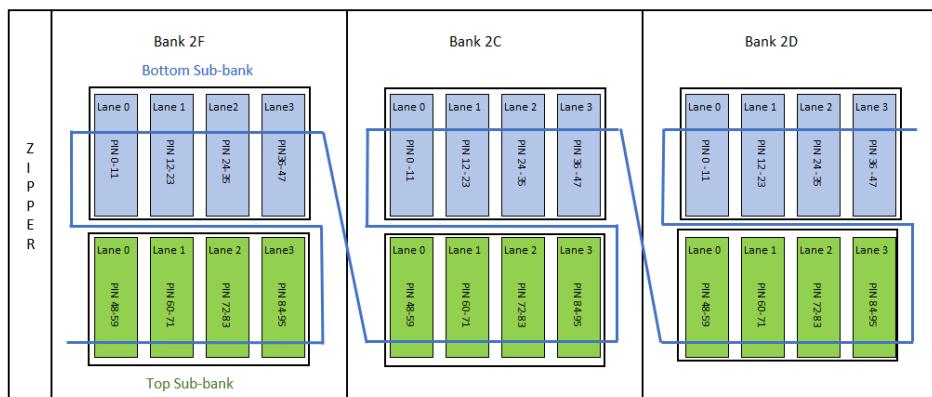
**Figure 19. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF022 and AGF027 devices, package R24C**



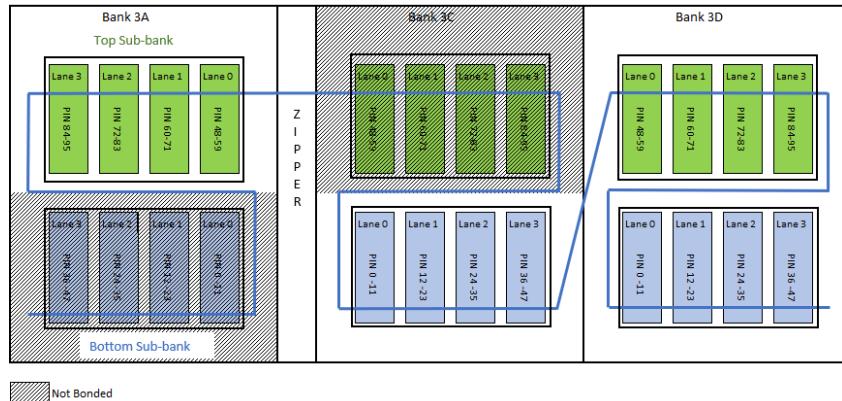
**Figure 20. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R25A**



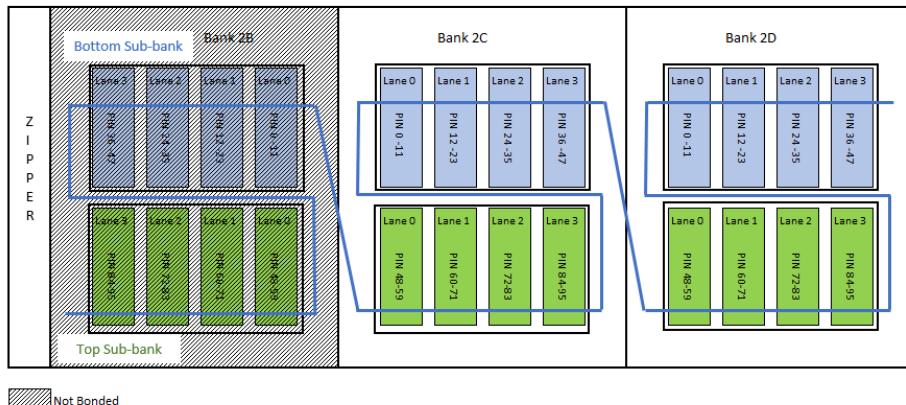
**Figure 21. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R25A**



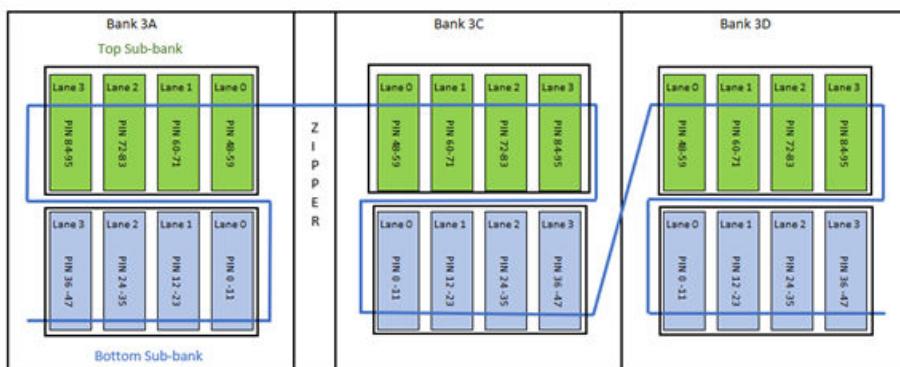
**Figure 22. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF006 and AGF008, package R16A**



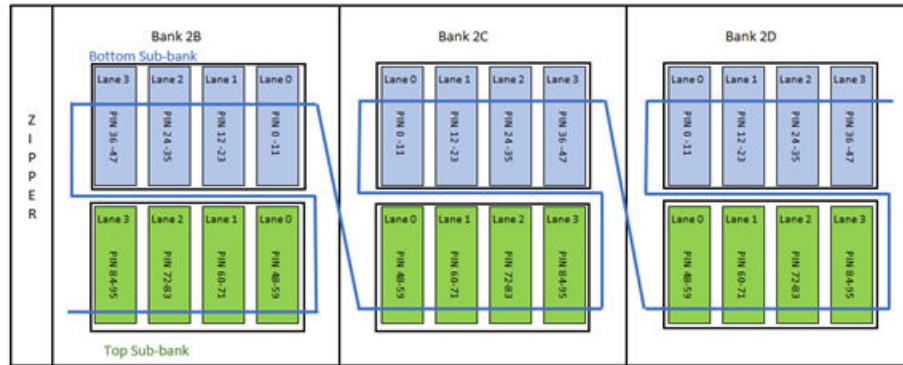
**Figure 23. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF006 and AGF008, package R16A**



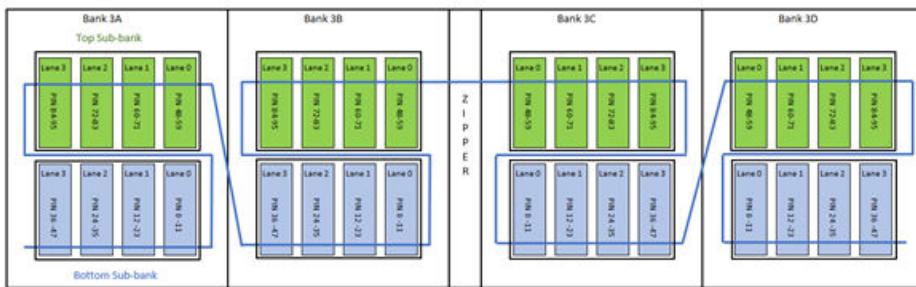
**Figure 24. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF006 and AGF008, package R24C**



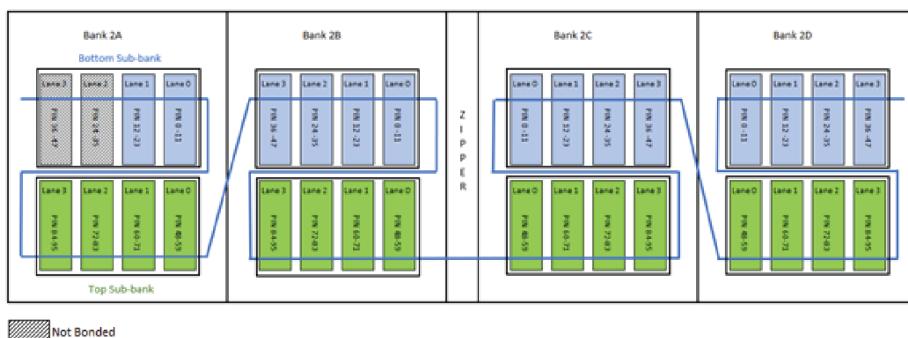
**Figure 25. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF006 and AGF008, package R24C**



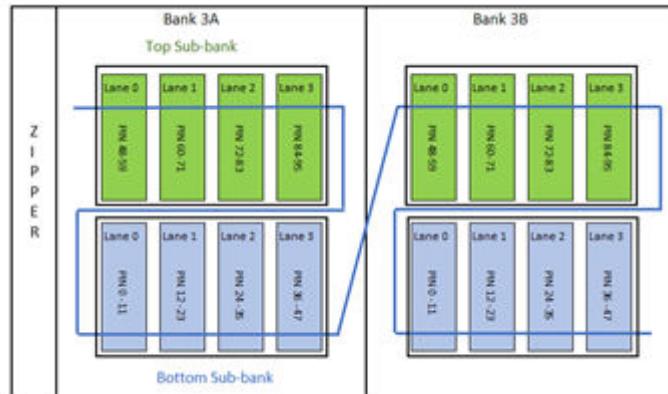
**Figure 26. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF012 and AGF014, package R24C**



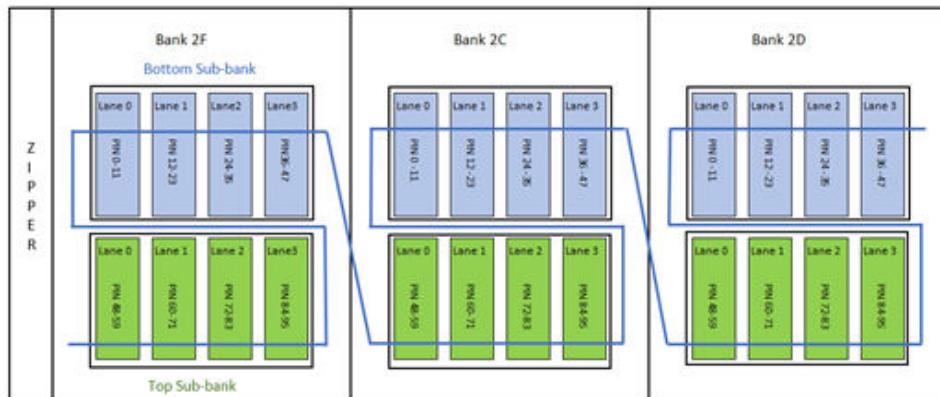
**Figure 27. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF012 and AGF014, package R24C**



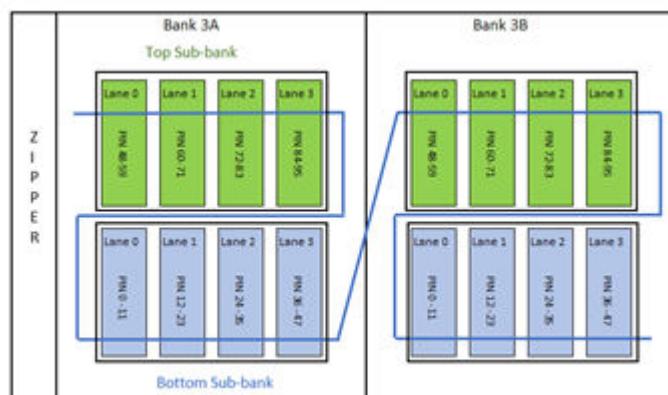
**Figure 28. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R24C**



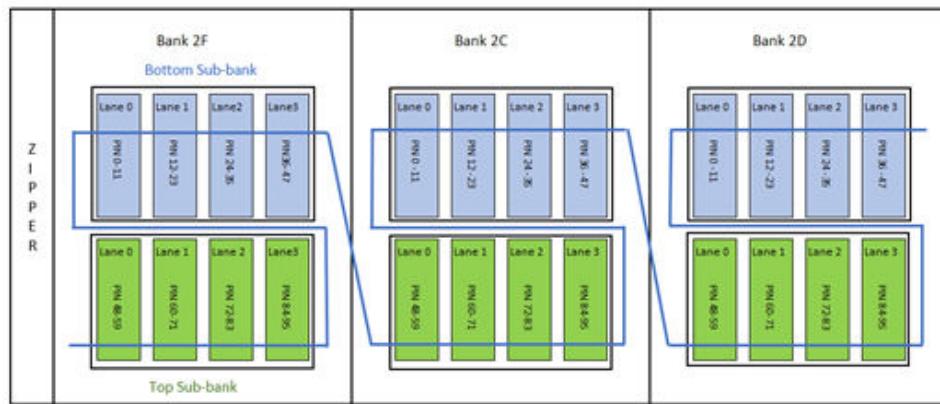
**Figure 29. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R24C**



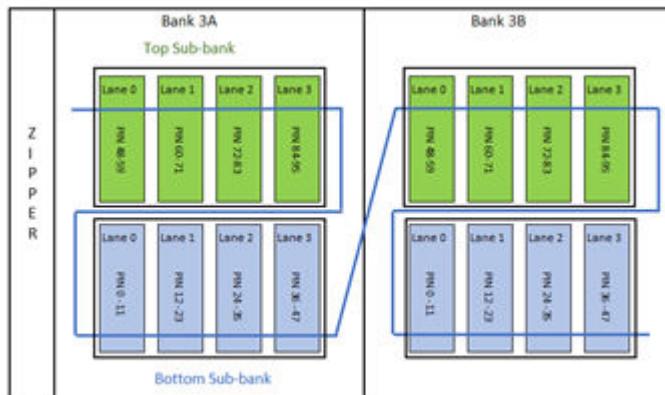
**Figure 30. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI019 and AGI023, package R31B**



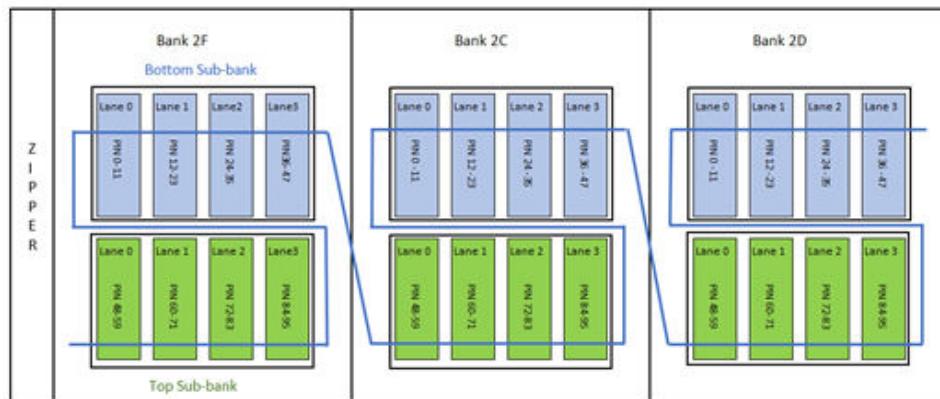
**Figure 31. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI019 and AGI023, package R31B**



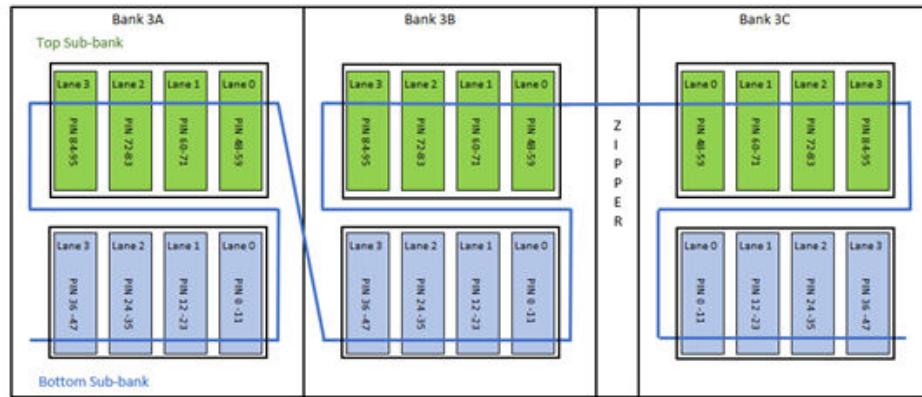
**Figure 32. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI019 and AGI023, package R18A**



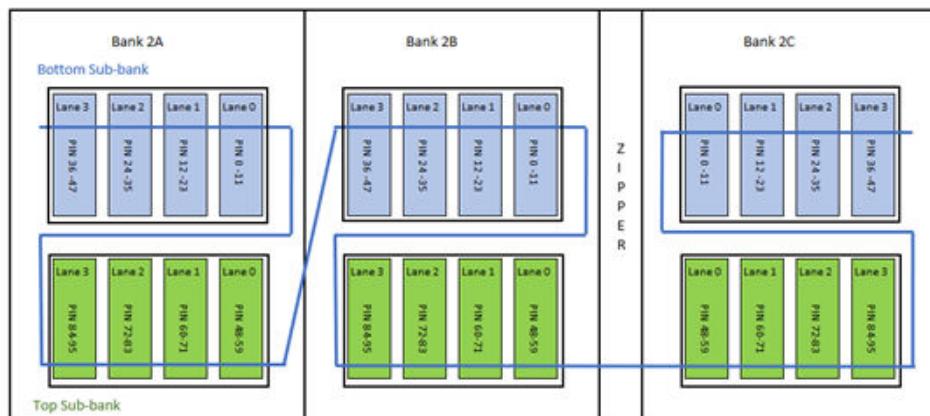
**Figure 33. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI019 and AGI023, package R18A**



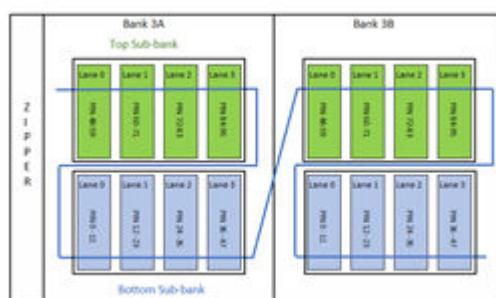
**Figure 34. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI035 and AGI040, package R39A**



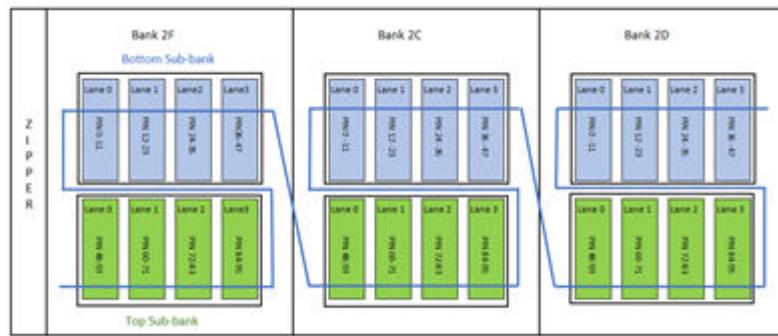
**Figure 35. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI035 and AGI040, package R39A**



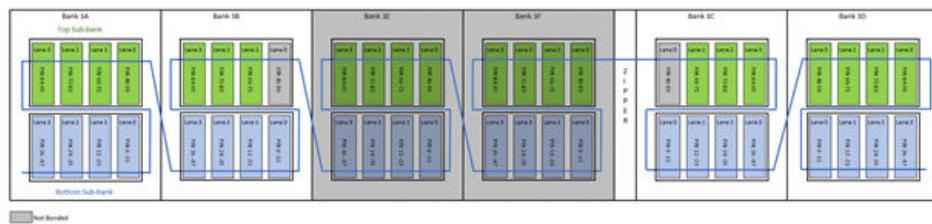
**Figure 36. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R31C**



**Figure 37. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGF019 and AGF023, package R31C**

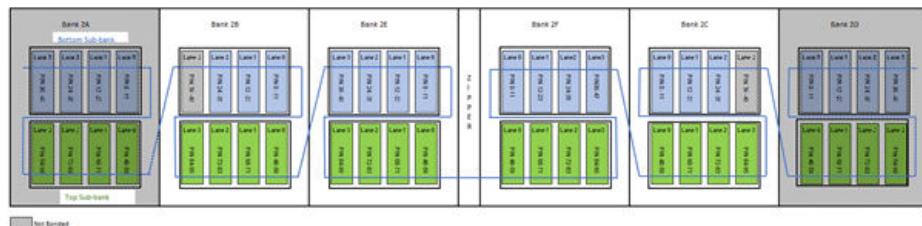


**Figure 38. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI027, package R29B**



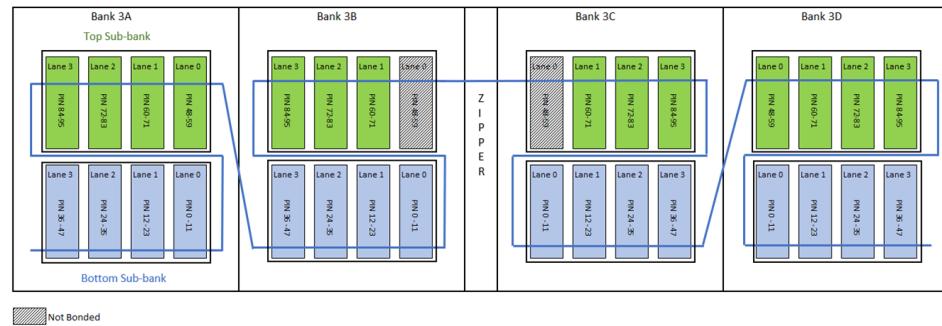
Bank 3E, Bank 3F, I/O Lane 0 in Top Sub-bank 3B and I/O Lane 0 in Top Sub-bank 3C are not bonded.

**Figure 39. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI027, package R29B**

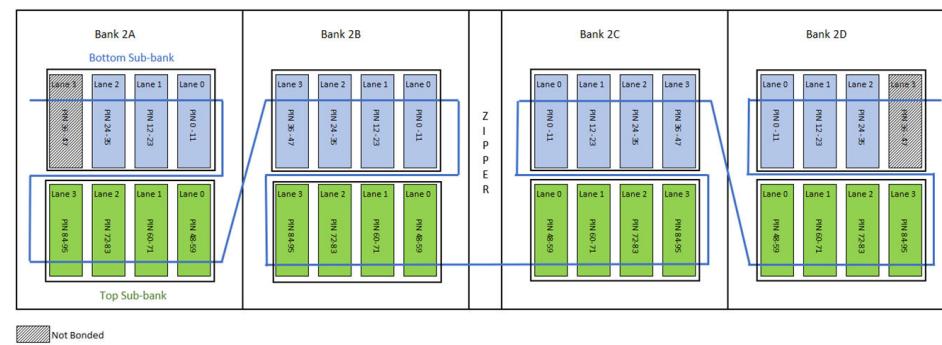


Bank 2A, Bank 2D, I/O Lane 3 in Bottom Sub-bank 2B and I/O Lane 3 in Bottom Sub-bank 2C are not bonded.

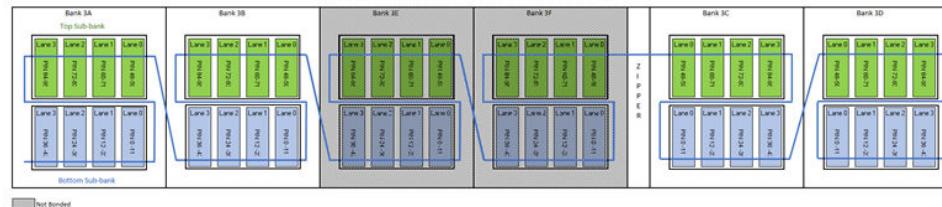
**Figure 40. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI041, package R29D**



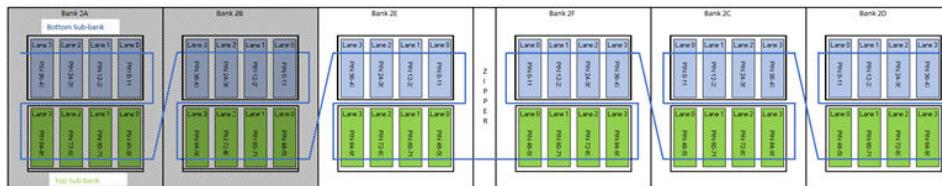
**Figure 41. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI041, package R29D**



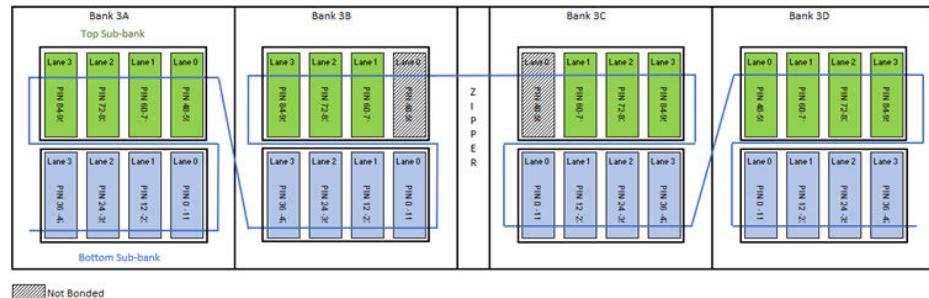
**Figure 42. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027, package R31A**



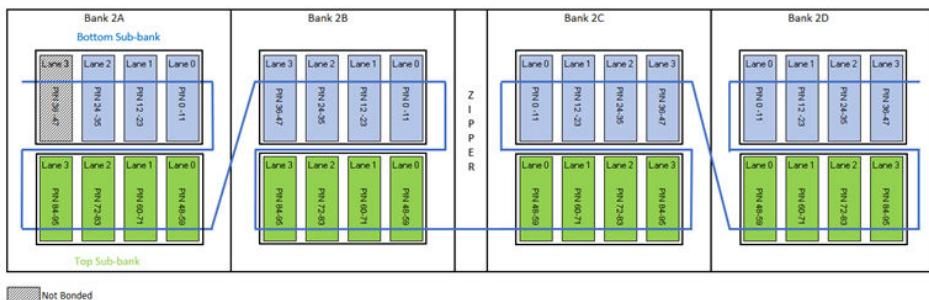
**Figure 43. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI022 and AGI027, package R31A**



**Figure 44. Sub-Bank Ordering in Top I/O Row in Intel Agilex 7 F-Series and I-Series AGI041, package R31B**



**Figure 45. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex 7 F-Series and I-Series AGI041, package R31B**



For AGI041 package R29D, pin JB26 and JH26 are available as REFCLK\_GXRR15C\_CH2p and REFCLK\_GXRR15C\_CH2n respectively on selected OPN. You must adhere to the layout guideline in "Restriction for using JB26, JH26, JP26 and JL27 on AGI041, Package R29D" section if you use the following pins:

- JB26 (REFCLK\_GXRR15C\_CH2p)
- JH26 (REFCLK\_GXRR15C\_CH2n)
- JP26 (I/O pin in Lane 2, Bottom Sub-bank of 2D)
- JL27 (I/O pin in Lane 2, Bottom Sub-bank of 2D)

The two sub-banks within an I/O bank are adjacent to each other when there is at least one I/O lane in each sub-bank that is bonded out and available for EMIF use. The blue line in the above figures shows the connectivity between the sub-banks.

For example, in the top row in Intel Agilex 7 F-Series and I-Series AGF012 and AGF014 devices (Figure 6):

- The top sub-bank in 3A is adjacent to the bottom sub-bank in 3A and the bottom sub-bank in 3B.
- The top sub-bank in 3B is adjacent to the bottom sub-bank in 3B and the top sub-bank in 3C.
  - The top sub-bank in 3B is adjacent to the top sub-bank in 3C even though there is a zipper block between the two sub-banks.
- The top sub-bank in 3B is not adjacent to the bottom sub-bank in 3A.

When an interface must occupy multiple sub-banks, ensure that those sub-banks are adjacent to one another. You can identify where a pin is located within an I/O bank based on its Index within I/O Bank value in the device pinout file.

### Zipper Block

The zipper is a block that performs necessary routing adjustments where routing wires cross the zipper.

### I/O Sub-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, DDR4 x8 interface, with only a single I/O sub-bank. A wider interface of up to 72 bits can be implemented by configuring multiple adjacent sub-banks in a multi-bank interface.

**Note:** A given sub-bank cannot be shared between multiple EMIFs.

**Note:** For DDR4 hard PHY-only configurations, you can implement a minimum of x16 interface width up to a maximum of x72 interface width.

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

To use a multi-bank Intel Agilex 7 F-Series and I-Series EMIF interface, you must observe the following rules:

- Designate one sub-bank as the address and command bank.
- The address and command sub-bank must contain all the address and command pins.
- The locations of individual address and command pins within the address and command sub-bank must adhere to the pin map defined in the pin table—regardless of whether you use the hard memory controller or not. You can find the pin tables at the following location: <https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html>.
- If you do use the hard memory controller, the address and command sub-bank contains the active hard controller.

All the sub-banks are capable of functioning as the address and command bank. For interfaces that span multiple sub-banks, the Intel Quartus Prime software requires that the address and command bank be placed in the center-most bank of the interface. The only exception to this rule is for the Hardened Processor Subsystem External Memory Interface.

### 3.1.4. Intel Agilex 7 F-Series and I-Series EMIF Architecture: I/O Lane

An I/O bank contains two sub-banks. Each sub-bank contains 48 I/O pins, organized into four I/O lanes of 12 pins each. You can identify where a pin is located within an I/O bank based on its Index within I/O Bank value in the device pinout.

**Table 3. Pin Index Mapping**

<b>Pin Index</b>	<b>Lane</b>	<b>Sub-bank Location</b>
0-11	0	Bottom
12-23	1	
24-35	2	
36-47	3	
48-59	0	Top
60-71	1	
72-83	2	
84-95	3	

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 a x18 group, you can use multiple lanes within the same sub-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.

For DQ and DQS pin assignments for various configurations, refer to the Intel Agilex 7 F-Series and I-Series device pin tables at the following location: <https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html>.

**Table 4. Lanes Used Per DQS Group**

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

Figure 46. x4 Group

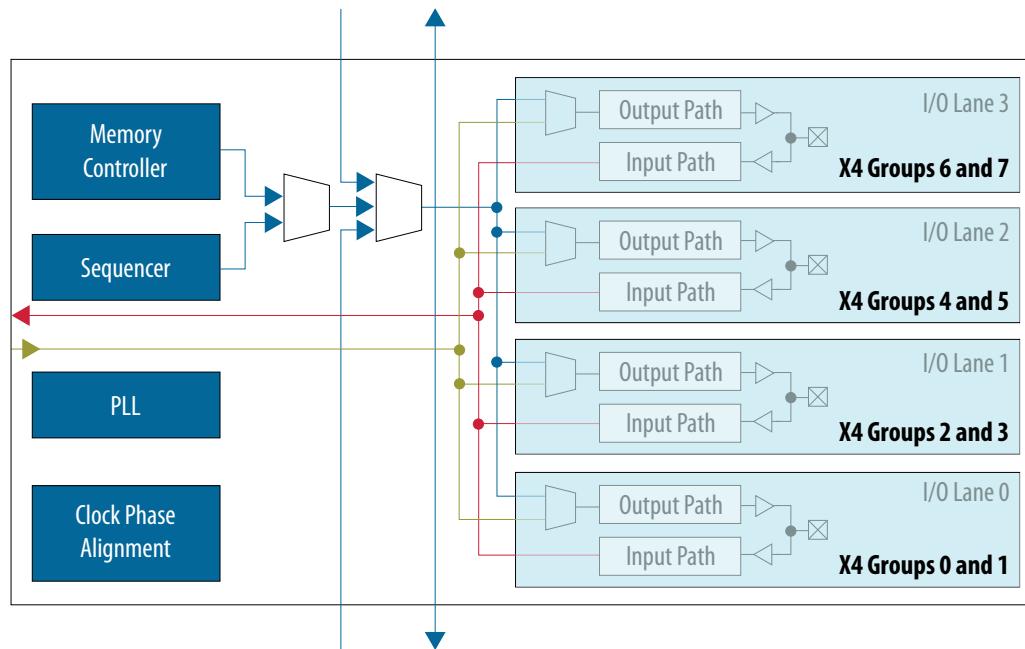
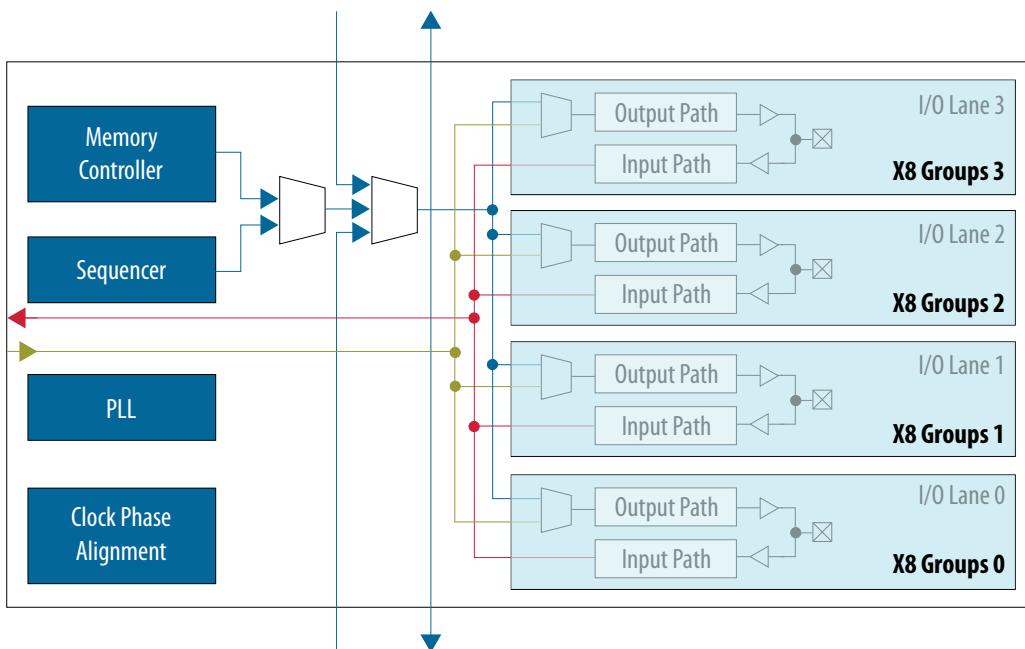
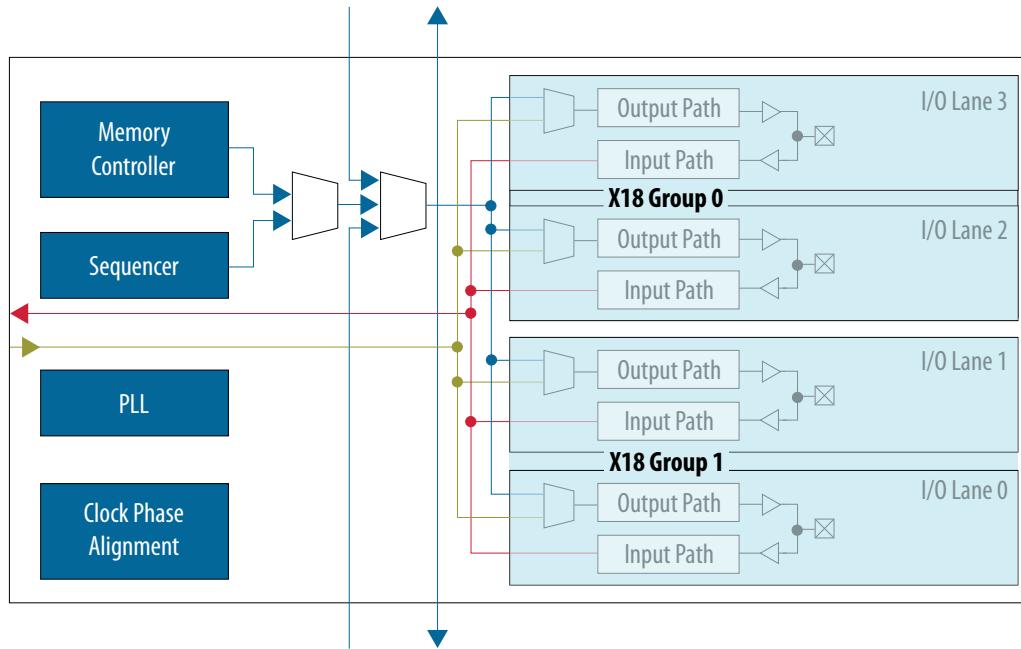


Figure 47. x8 Group



**Figure 48. x18 Group**

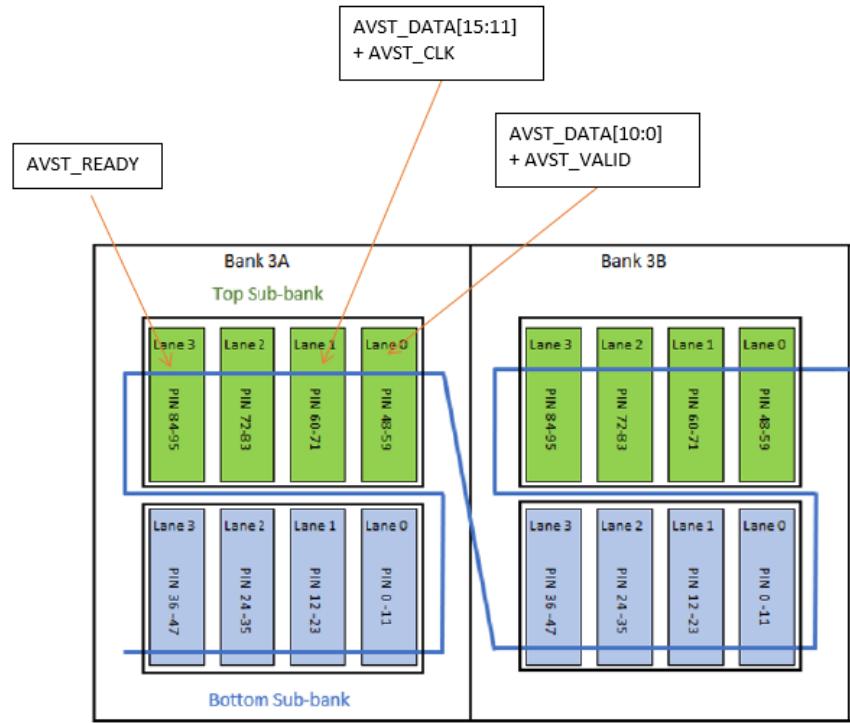


### 3.1.4.1. Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme

The AVST x8 configuration scheme uses the dedicated SDM I/O pins and does not impact the number of DDR4 x72 interfaces that can be implemented on the device. An AVST x32 configuration scheme uses all four of the I/O lanes in the top sub-bank in Bank 3A. This breaks the contiguity requirement and reduces the maximum number of DDR4 x72 interfaces that can be supported on the device.

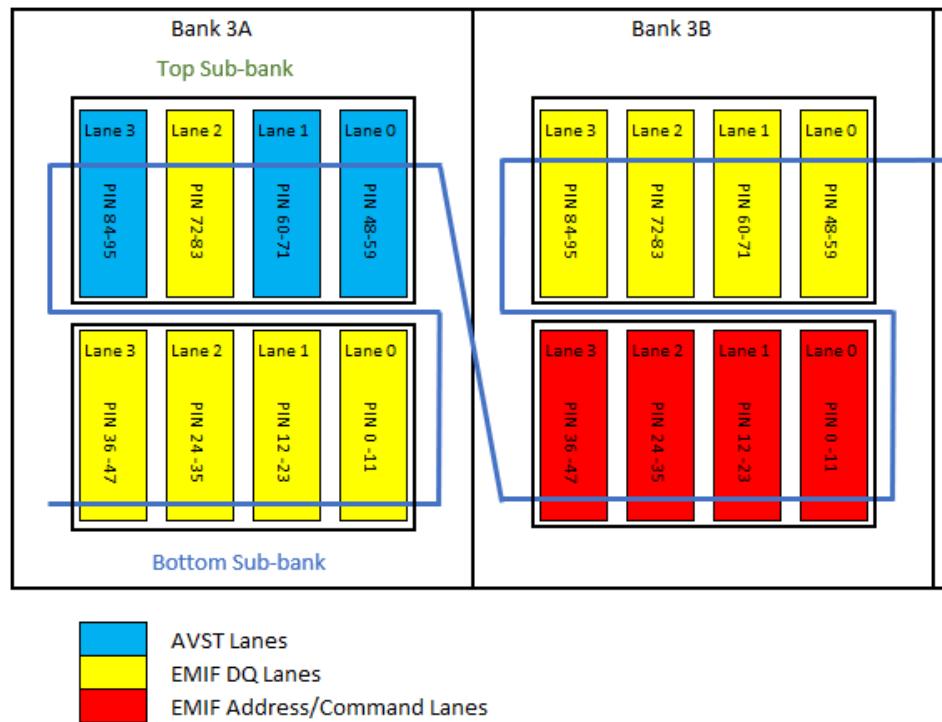
However, the AVST x16 configuration scheme only uses three I/O Lanes. I/O Lane 2 in the top sub-bank in Bank 3A is available for EMIF purposes and maintains the contiguity requirement. This I/O Lane can be used as a DQ Lane for EMIF purposes.

**Figure 49. Pin Assignment in Top Sub-bank in Bank 3A for AVST x16 Configuration Scheme**



To implement a DDR4 x72 interface using the top sub-bank in Bank 3A together with an AVST x16 configuration scheme, you must use the address/command scheme with four IO Lanes. The following figure shows the I/O lane assignment for implementing a DDR4 x 72 interface in such a scenario.

**Figure 50. I/O Lane Assignment for Implementing AVST x16 and DDR4 x72 Interface Using Bank 3A**



The following table shows the maximum number of DDR4 x72 interfaces that can be supported with different AVST configuration schemes.

**Table 5. DDR x72 EMIF With AVST and Address/Command Scheme with 4 I/O Lanes**

Device/ Package	1x DDR4 x72	2x DDR4 x72	3x DDR4 x72	4x DDR4 x72	6x DDR4 x72	8x DDR4 x72
AGF014/ AGF012, R24A/ R24B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16	N/A	N/A
AGF027/ AGF022, R24C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16	N/A	N/A
AGF027/ AGF022, R25A	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A	N/A
AGF027/ AGF022, R31C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI027/ AGI022, R29A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI027/ AGI022, R31B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF019/ AGF023, R25A	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGF006/ AGF008, R16A	AVST 8, 16, 32	N/A	N/A	N/A	N/A	N/A

*continued...*

Device/ Package	1x DDR4 x72	2x DDR4 x72	3x DDR4 x72	4x DDR4 x72	6x DDR4 x72	8x DDR4 x72
AGF006/ AGF008, R24C	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A	N/A
AGF012/ AGF014, R24C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16	N/A	N/A
AGF019/ AGF023, R24C	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGI019/ AGI023, R31B	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGI019/ AGI023, R18A	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGI035/ AGI040, R39A	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A	N/A
AGI019/ AGI023, R31C	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGI027, R29B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI041, R29D	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI022/ AGI027, R31A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF019/ AGF023, R31C	AVST 8, 16, 32	AVST 8, 16	N/A	N/A	N/A	N/A
AGI041, R31B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A

**Table 6. DDR x72 EMIF With AVST and Address/Command Scheme with 3 I/O Lanes**

Device/ Package	1x DDR4 x72	2x DDR4 x72	3x DDR4 x72	4x DDR4 x72	6x DDR4 x72	8x DDR4 x72
AGF014/ AGF012, R24A/ R24B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF027/ AGF022, R24C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF027/ AGF022, R25A	AVST 8, 16, 32	N/A	N/A			
AGF027/ AGF022, R31C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI027/ AGI022, R29A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI027/ AGI022, R31B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF019/ AGF023, R25A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGF006/ AGF008, R16A	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A	N/A
AGF006/ AGF008, R24C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF012/ AGF014, R24C	AVST 8, 16, 32	N/A	N/A			

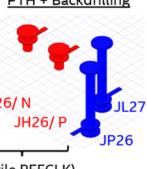
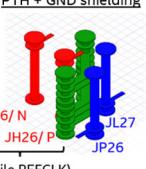
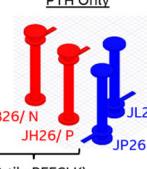
*continued...*

<b>Device/ Package</b>	<b>1x DDR4 x72</b>	<b>2x DDR4 x72</b>	<b>3x DDR4 x72</b>	<b>4x DDR4 x72</b>	<b>6x DDR4 x72</b>	<b>8x DDR4 x72</b>
AGF019/ AGF023, R24C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGI019/ AGI023, R31B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGI019/ AGI023, R18A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGI035/ AGI040, R39A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI019/ AGI023, R31C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGI027, R29B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI041, R29D	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGI022/ AGI027, R31A	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A
AGF019/ AGF023, R31C	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	N/A	N/A	N/A
AGI041, R31B	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8, 16, 32	AVST 8	N/A	N/A

#### **Restriction for using JB26, JH26, JP26 and JL27 on AGI041, Package R29D**

You must adhere to these guidelines when using pin JB26, JH26, JP26 and JL27 on AGI041, package R29D, to reduce crosstalk coupling from adjacent I/O pins (JP26 and JL27) to R-tile REFCLK pins (JB26 and JH26).

**Figure 51. Recommended PCB Layout for Using JB26, JH26, JP26 and JL27**

Graphical illustration (w)/ preferred option ranking	<b>PTH + Backdrilling</b>	<b>PTH + GND shielding</b>	<b>PTH Only</b>	<b>Legend</b>
				
REFCLK Routing Layer	Shallow (L3)	Shallow (L3)	Shallow (L3)	
Backdrilling for REFCLK	Yes	No	No	
DDR4 (JP26, JL27 pins) Routing Layer	Deep (L16)	Deep (L16)	Deep (L16)	
Backdrilling for DDR4 (JP26, JL27 pins)	No	No	No	
Ground Ring/ Guard	No	Yes	No	

Follow these guidelines:

- Implement one of the 3 layout options shown in the above figure.
- Route DDR/IO (JP26, JL27) at deep layer, L16 and R-tile REFCLK at shallow layer, L3. This is only applicable to pin JB26, JH26, JP26 and JL27.
- Keep maximum R-tile REFCLK route length < 3.7 inches.
- Refer to routing recommendations for REFCLK generator from manufacturer (if applicable).
- Keep at least 5xH separation between 100MHz R-tile REFCLK to DDR4/other GPIO signals.

**Figure 52. Separation between R-Tile REFCLK to DDR4/GPIO signals**



Note: H is the closest distance measuring from trace to ground reference plane in Z-direction.

### 3.1.5. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Input DQS Clock Tree

The input DQS clock tree is a balanced clock network that distributes the read capture clock (such as 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, or x18 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 7. Pins Usable as Read Capture Clock / Strobe Pair**

Group Size	Index of Lanes Spanned by Clock Tree <sup>1</sup>	Sub-Bank	Index of Pins Usable as Read Capture Clock / Strobe Pair	
			DQS p	DQS n
x4	0A	Bottom	4	5
x4	0B		6	7
x4	1A		16	17
x4	1B		18	19
x4	2A		28	29
x4	2B		30	31
x4	3A		40	41
x4	3B		42	43
x8 / x9	0		4	5
<i>continued...</i>				

Group Size	Index of Lanes Spanned by Clock Tree <sup>1</sup>	Sub-Bank	Index of Pins Usable as Read Capture Clock / Strobe Pair	
			DQS p	DQS n
x8 / x9	1	Top	16	17
x8 / x9	2		28	29
x8 / x9	3		40	41
x18	0, 1		4	5
x18	2, 3		28	29
x4	0A		52	53
x4	0B		54	55
x4	1A		64	65
x4	1B		66	67
x4	2A		76	77
x4	2B		78	79
x4	3A		88	89
x4	3B		90	91
x8 / x9	0		52	53
x8 / x9	1		64	65
x8 / x9	2		76	77
x8 / x9	3		88	89
x18	0,1		53	53
x18	2,3		76	77

Note: <sup>1</sup> A and B refer to the two nibbles within the lane.

### 3.1.6. Intel Agilex 7 F-Series and I-Series EMIF Architecture: PHY Clock Tree

Dedicated high-speed clock networks drive I/Os in Intel Agilex 7 F-Series and I-Series EMIF. Each PHY clock network spans only one sub-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 Agilex 7 F-Series and I-Series devices can run as fast as 1.6 GHz. All Intel Agilex 7 F-Series and I-Series external memory interfaces use the PHY clock trees.

### 3.1.7. Intel Agilex 7 F-Series and I-Series EMIF Architecture: PLL Reference Clock Networks

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

The PLL reference clock must be constrained to the address and command sub-bank only.

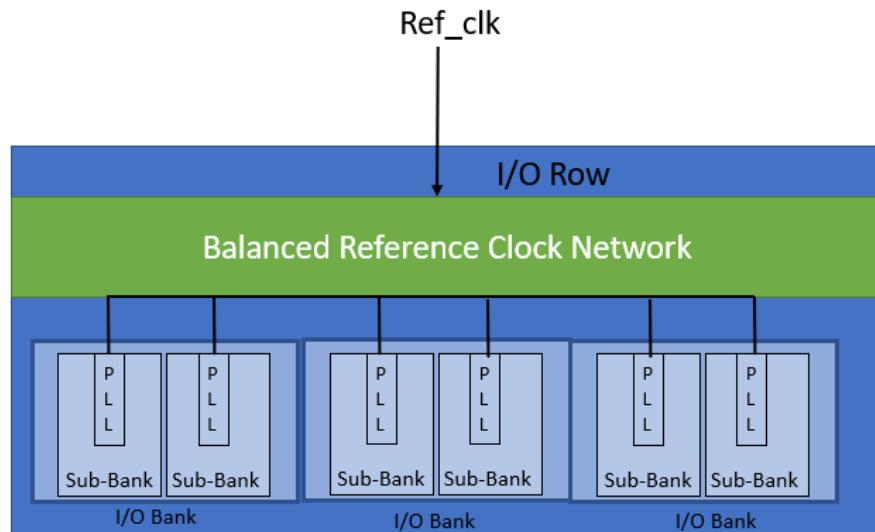
- A single-ended reference clock must be constrained to pin index 0 in lane 2. You cannot use pin index 1 in lane 2 as a general purpose I/O pin.
- Differential reference clocks must be constrained to pin indices 0 and 1 in lane 2.

Intel Agilex 7 F-Series and I-Series external memory interfaces that span multiple sub-banks use the PLL in each sub-bank. The Intel Agilex 7 F-Series and I-Series architecture allows for relatively short PHY clock networks, reducing jitter and duty-cycle distortion.

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

- A single PLL reference clock source feeds all I/O bank I/O 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. This clock must be free-running and stable prior to FPGA configuration.
- The EMIF IP sets the PLL configuration (counter settings, bandwidth settings, compensation and feedback mode setting) 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 Intel Agilex 7 F-Series and I-Series 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.

**Figure 53. PLL Balanced Reference Clock Tree**



### 3.1.8. Intel Agilex 7 F-Series and I-Series EMIF Architecture: Clock Phase Alignment

In Intel Agilex 7 F-Series and I-Series 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 Intel Agilex 7 F-Series and I-Series external memory interfaces employ clock phase alignment circuitry.

**Figure 54. Clock Phase Alignment Illustration**

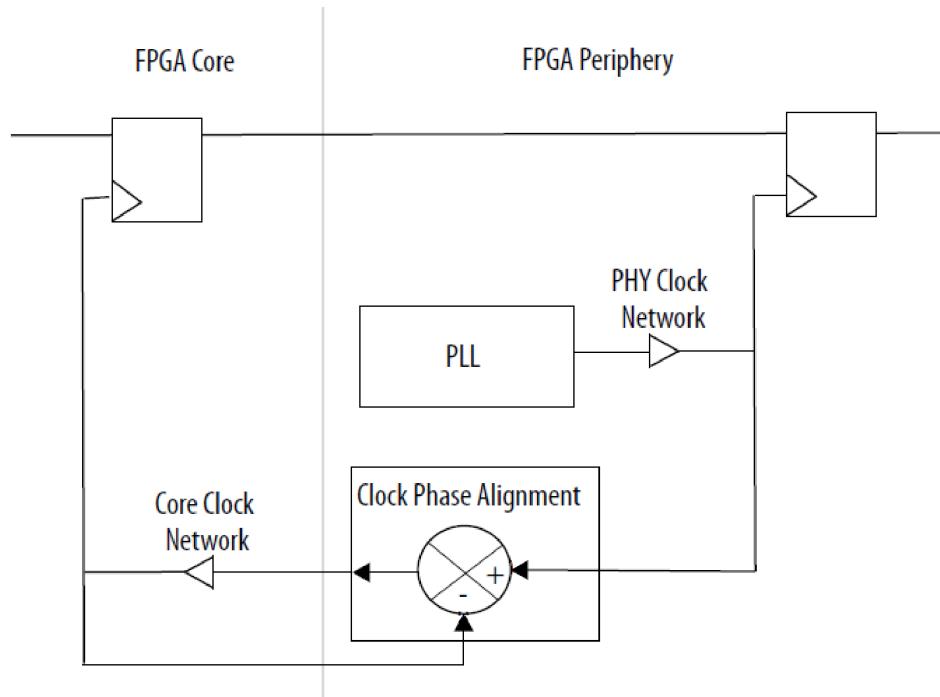
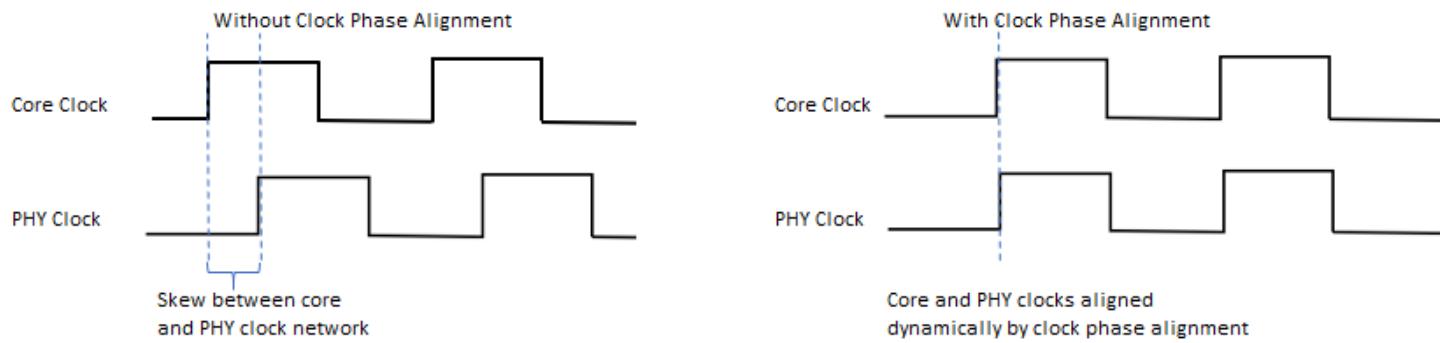


Figure 55. Effect of Clock Phase Alignment



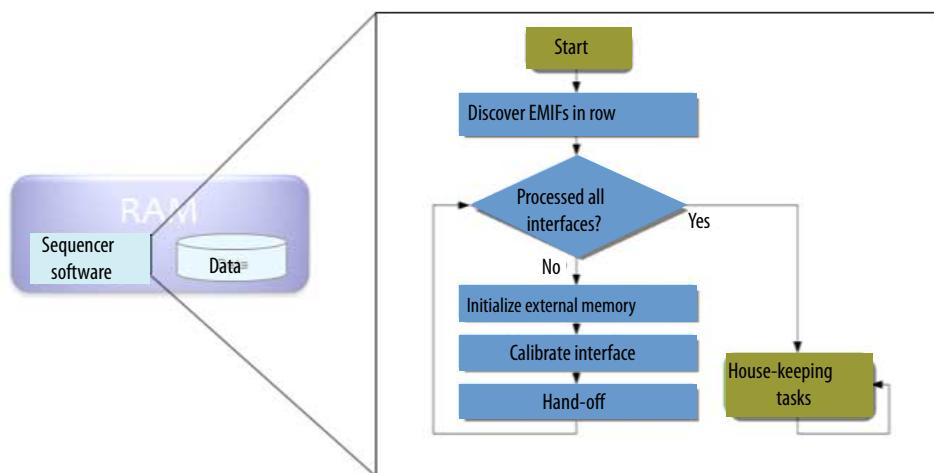
### 3.2. Intel Agilex 7 F-Series and I-Series EMIF Sequencer

The Intel Agilex 7 F-Series and I-Series EMIF sequencer is fully hardened in silicon, with executable code to handle protocols and topologies. Hardened RAM contains the calibration algorithm.

The Intel Agilex 7 F-Series and I-Series 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 56. Intel Agilex 7 F-Series and I-Series EMIF Sequencer Operation



### 3.3. Intel Agilex 7 F-Series and I-Series 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 on multiple pins in parallel whenever possible; however, some operations still operate on individual byte lanes sequentially. Interfaces in a row are calibrated in the order in which they are connected to the calibration IP (first the interface connected to calbus\_0, then the interface connected to calbus\_1, and so forth.)

**Note:** The calibration process is intended to maximize margins for robust EMIF operation; it cannot compensate for an inadequate PCB layout. Examples of PCB-related issues that cannot be calibrated, include the following:

- Excessive skew between signals within a byte lane.
- Inter-symbol interference caused by suboptimal trace topology, such as multiple vias, impedance mismatches, or discontinuities.
- Simultaneously-switching signal effects (victim/aggressor coupling caused by insufficient trace spacing, broadside coupling, or layer-to-layer coupling).
- Electrical noise effects such as improper plane referencing, split-plane crossing, routing signals too close to noisy sources such as switching power supplies or other high-frequency noise generators.
- Impedance mismatches, such as improper choices for FPGA/DRAM-side transmit/receive termination relative to PCB trace impedance, or excessive loading on the address/command or data buses due to multiple loads.

#### 3.3.1. Intel Agilex 7 F-Series and I-Series 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 8. Calibration Stages by Protocol**

Stage	DDR4	QDR-IV
<b>Address and Command</b>		
Leveling	Yes	—
Deskew	Yes	Yes
<b>Read</b>		
<i>continued...</i>		

Stage	DDR4	QDR-IV
DQSen	Yes	Yes
Deskew	Yes	Yes
VREF-In	Yes	Yes
LFIFO	Yes	Yes
<b>Write</b>		
Leveling	Yes	Yes
Deskew	Yes	Yes
VREF-Out	Yes	—

### 3.3.2. Intel Agilex 7 F-Series and I-Series 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 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

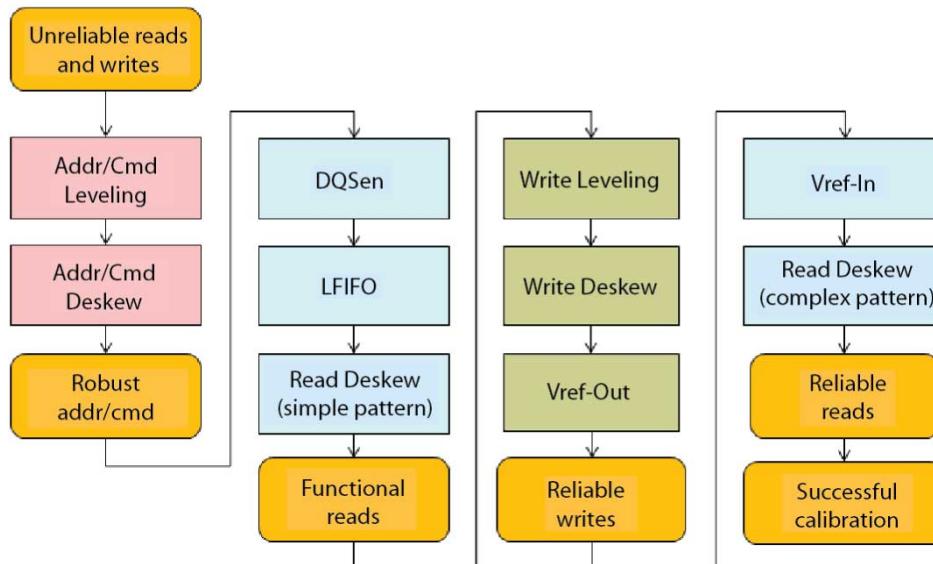
Write calibration consists of the following parts:

- 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 Agilex 7 F-Series and I-Series Calibration Flowchart

The following flowchart illustrates the Intel Agilex 7 F-Series and I-Series calibration flow.

**Figure 57. Calibration Flowchart**



### 3.3.4. Intel Agilex 7 F-Series and I-Series Calibration Algorithms

The calibration algorithms are specific to the targeted memory protocol.

#### 3.3.4.1. Calibration Algorithms for DDR4

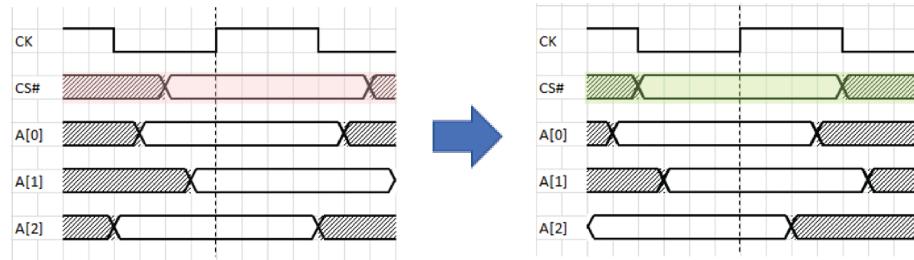
For DDR4, address and command calibration consists of leveling calibration and deskew calibration.

##### Leveling Calibration

The FPGA toggles the CS# and CAS#/A15 signals while keeping other address and command signals constant, to send READ commands to memory, and monitors whether the DQS signal toggles. 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. This stage moves the entire address and command bus with CS#.

**Figure 58. Address and Command Leveling moves the entire Address and Command Bus with CS#**

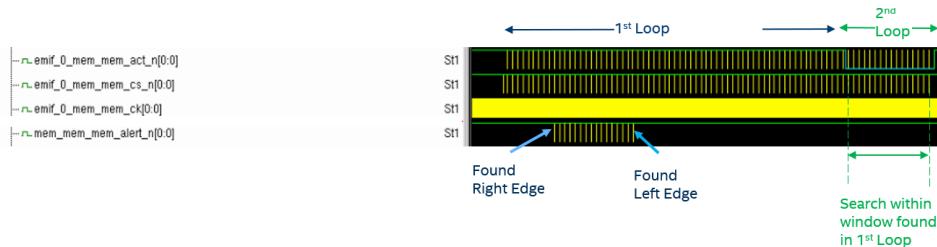


### Deskew Calibration

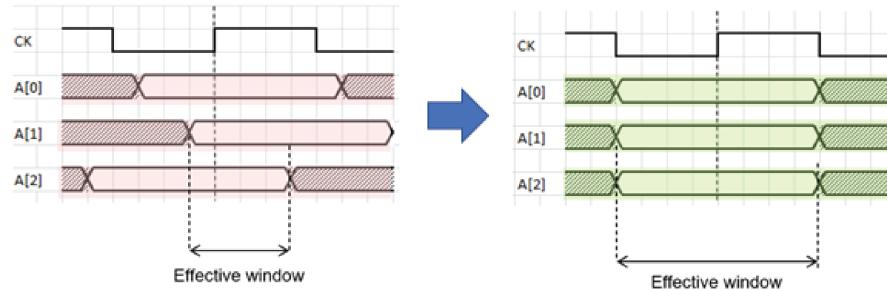
Deskew calibration uses the DDR4 address and command parity feature; you should not omit these pins from your design. The deskew calibration cannot deskew ODT and CKE pins because these signals are not included in the address and command parity calculation.

The deskew calibration test consists of two loops. In the first loop, the FPGA adjusts the delays on an address and command pin to send the address parity bit; the DDR4 memory device responds with an alert signal if it detects the parity bit. The test repeats this process for all other address and command signals, except for the ODT and CKE signals. The first loop detects the union of windows if there are multiple components within the same rank. For a sub-optimal board layout, the window detected in the first loop may be greater than 1 memory clock cycle.

**Figure 59. Different Parity Injection Pattern in First and Second Loops**



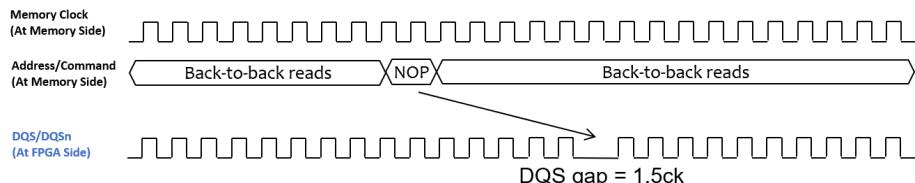
In the second loop, the algorithm inverts the parity injection pattern and sweeps the delay within the window found in the first loop. The alert signal does not toggle in the second loop when all memory components in the same rank receive the command correctly. This check detects the intersection of the windows and improves the accuracy of the address and command deskew.

**Figure 60. Address and Command Deskew**


### 3.3.4.1.1. DDR4 Read Calibration

#### DQSen Calibration

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 then repeats for all other read DQS groups.

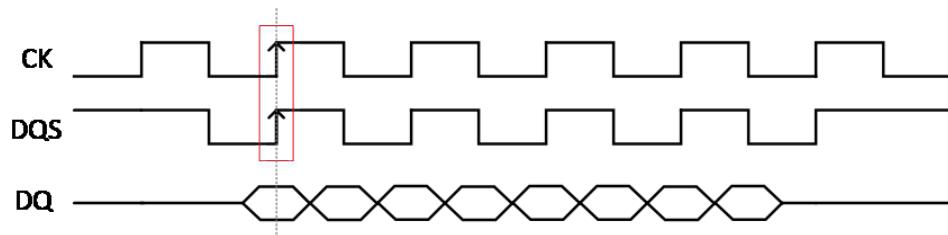
**Figure 61. DQS Enable Calibration using Hardware State Machine**


#### Deskew Calibration

Read deskew calibration occurs before write leveling, and must occur at least twice: once before write calibration using simple data patterns from guaranteed writes, and again after write calibration using complex data patterns.

To ensure that guaranteed writes work correctly, the Write Leveling Phase training occurs before the guaranteed writes. The goal of Write Leveling Phase training is to align the rising edge for write DQS with the rising edge of MEM\_CLK at the DRAM.

**Figure 62. Write Leveling Phase Training -Align Write DQS with MEM\_CLK at DRAM**

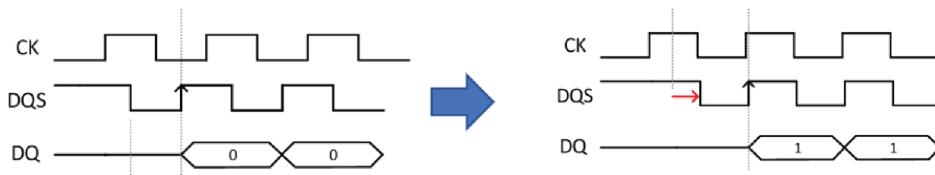


The algorithm uses the DRAM write leveling feature for Write Leveling Phase Training. In this mode the following actions occur:

- The algorithm adjusts the DQS output delay (at the FPGA side) while toggling write DQS signal.
- The DRAM samples the MEM\_CLK using the rising edge of write DQS and outputs the sampled value on DQ pins.
- The algorithm continues to adjust the DQS output while toggling the write\_DQS signal until it detects a 0 to 1 transition on the DQ pins.

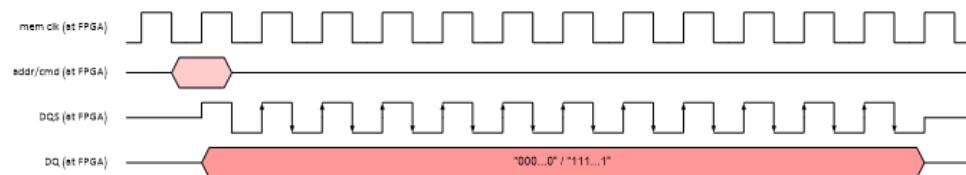
The following four figures illustrate an example of read deskew calibration. In this example, the DQS is within the read window before the read deskew calibration, and the FPGA gets the correct data from the back-to-back read operation.

**Figure 63. DQS and MEM\_CLK are phase aligned when 0 to 1 transition is detected on DQ pins**



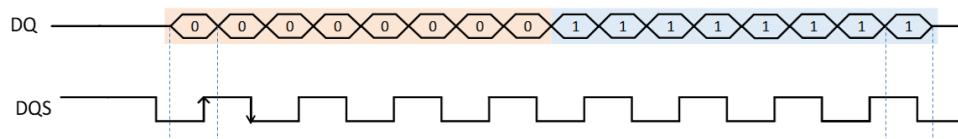
In guaranteed writes, the algorithm writes a burst of 0s to one location and a burst of 1s to another location. The data from the back-to-back reads from these two locations is used as a simple data pattern for read deskew calibration.

**Figure 64. Guarantee Write – Writing a Simple Data Pattern to Memory**

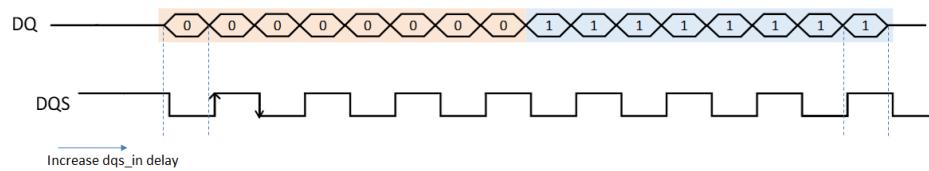


Before the write calibration, the deskew calibration algorithm performs a guaranteed write, and then sweeps dqs\_in delay values from low to high, to find the right-hand edge of the read window. The algorithm then sweeps dq\_in delay values from low to high, to find the left-hand edge of the read window. The algorithm then applies updated dqs\_in and dq\_in delay values to center the read window. The process then repeats for all data pins.

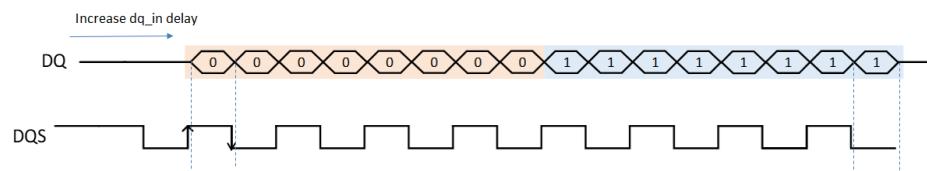
**Figure 65. A Passing Back-to-back Read with Simple Data Pattern**



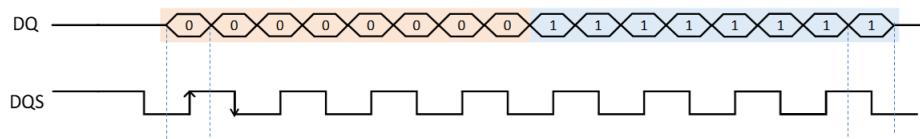
**Figure 66. Increase dqs\_in Delay Until Read Fails -Right Edge is Identified**



**Figure 67. Increase dq\_in Delay Until Read Fails-Left Edge is Identified**



**Figure 68. Centering the Read Window**



After the write path is calibrated, the algorithm performs another read-deskew calibration using a complex data pattern.

#### 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. Vref-In is generated from the VCCIO on the I/O banks used for DQ/DQS signals and calibrated internally in FPGA.

#### 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.

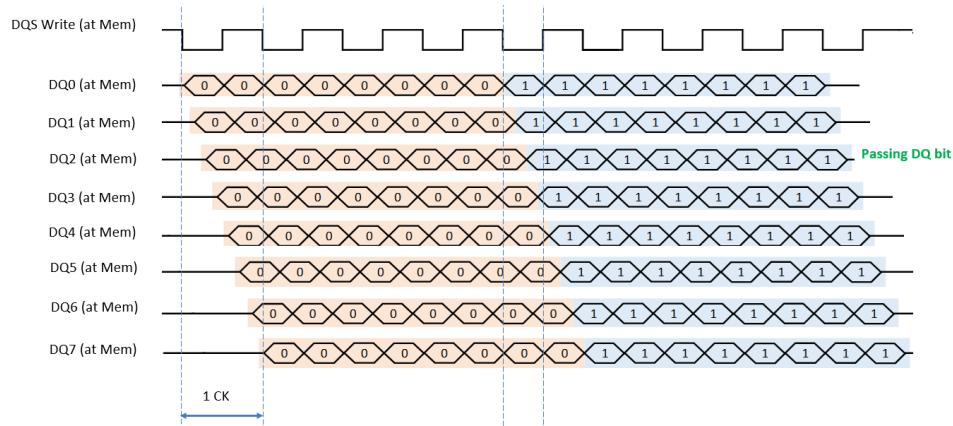
#### 3.3.4.1.2. DDR4 Write Calibration

### Write Leveling

Write leveling calibration aligns the write DQS 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.

Write leveling occurs before write deskew, therefore only one successful DQ bit is required to register a pass. Write leveling staggers the write DQ bus to ensure that at least one DQ bit falls within the valid write window.

**Figure 69. Staggering DQ bus during Write Leveling**



### Write Deskew

Write Deskew 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 DQS is aligned to the CK clock during write leveling.

### VREF-OUT Calibration

VREF-OUT calibration tunes the VREF setting at DDR4 memory device by using mode register set (MRS) commands. The VREF-OUT calibration algorithm is similar to the VREF-IN calibration algorithm. The algorithm picks the VREF-OUT setting that gives the best write window.

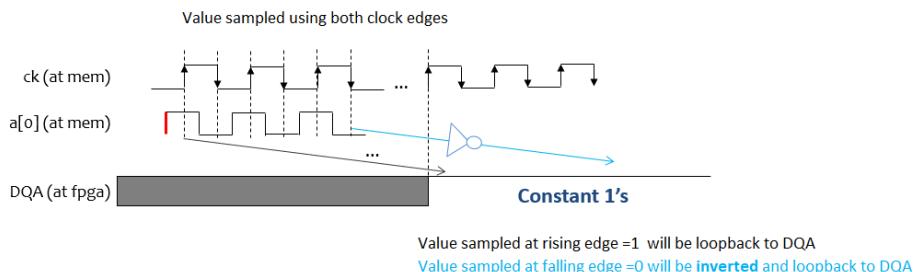
#### 3.3.4.2. Calibration Algorithms for QDR-IV

##### Address and Command Deskew

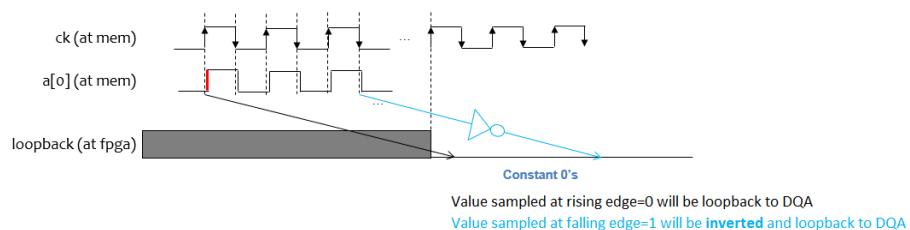
QDR-IV uses the loopback mode for address and command deskew. The FPGA sends address and command signals, and the memory device sends back the address and command signals which it captures, via the read data pin of port A. Each input pin is sampled on both the rising and falling clock edges of input CK/CK#. The output value on the rising edge of output clock QKA/QKA# is the value that was sampled on the rising clock edge of the input clock. The output value on the falling edge of the output clock of QKA/QKA# is the inverted value of what was sampled on the falling edge of the input clock.

By sweeping the output delay on the address and command pins on the FPGA, the algorithm determines the right and left edges of the window, and centers the signal accordingly. Deskew calibration can deskew all synchronous address and command signals,

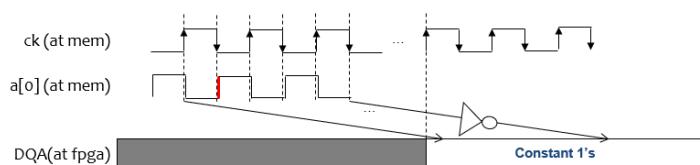
**Figure 70.** QDR-IV Address/Command Loopback



**Figure 71.** QDR-IV Address/Command Deskew – Right Edge Detected



**Figure 72.** QDR-IV Address/Command Deskew – Left Edge Detected



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

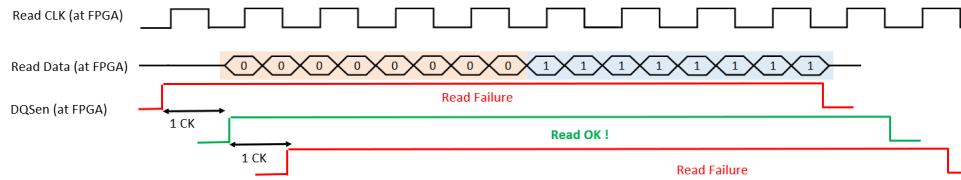
### 3.3.4.2.1. QDR-IV Read Calibration

#### DQSen Calibration

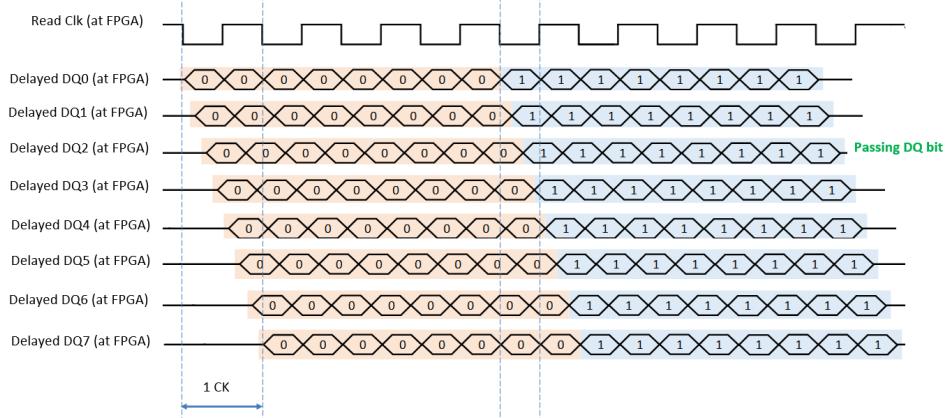
The calibration algorithm does not use a hardware state machine; rather, it calibrates cycle-level delays using software, and subcycle delays using DQS tracking hardware. The algorithm requires good data in memory, and therefore relies on guaranteed writes. The algorithm enables DQS tracking to calibrate 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. Because the DQSen calibration occurs before Read Deskew, only one successful data

bit is required to register a pass. The algorithm staggers the read DQ bus to ensure that at least one DQ bit falls within the valid Read window. The process then repeats for all other read groups.

**Figure 73. Sweep DQSen at Cycle-Level Until At Least One DQ Bit is Passing**



**Figure 74. Staggering DQ Bus to Ensure at Least One Passing DQ bit**

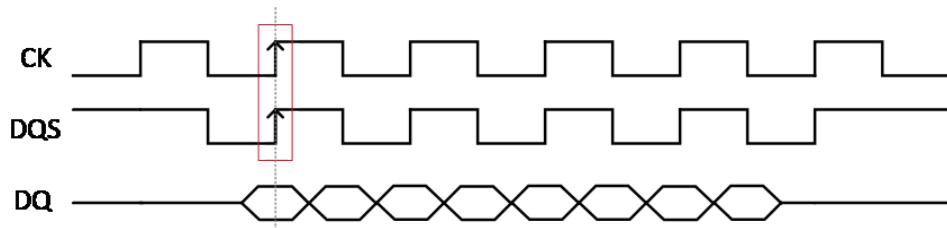


### Deskew Calibration

Read deskew calibration occurs before write leveling, and must occur at least twice: once before write calibration using simple data patterns from guaranteed writes, and again after write calibration using complex data patterns.

To ensure that guaranteed writes work correctly, the Write Leveling Phase training occurs before the guaranteed writes. The goal of Write Leveling Phase training is to align the rising edge for write DQS with the rising edge of MEM\_CLK at the DRAM.

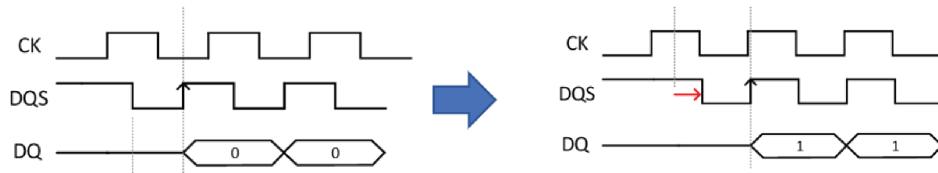
**Figure 75. Write Leveling Phase Training -Align Write DQS with MEM\_CLK at DRAM**



The algorithm uses the DRAM write leveling feature for Write Leveling Phase Training. In this mode the following actions occur:

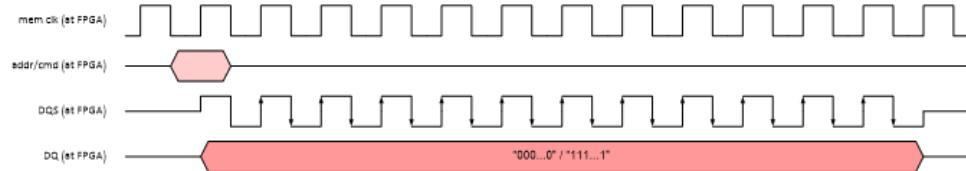
- The algorithm adjusts the DQS output delay (at the FPGA side) while toggling write\_DQS signal.
- The DRAM samples the MEM\_CLK using the rising edge of write DQS and outputs the sampled value on DQ pins.
- The algorithm continues to adjust the DQS output while toggling the write\_DQS signal until it detects a 0 to 1 transition on the DQ pins.

**Figure 76. DQS and MEM\_CLK are phase aligned when 0 to 1 transition is detected on DQ pins**



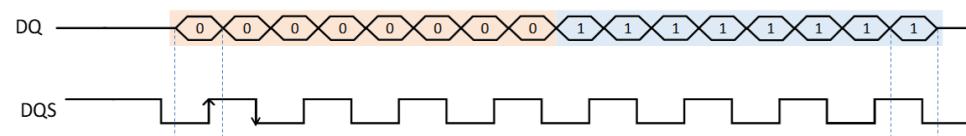
In guaranteed writes, the algorithm writes a burst of 0s to one location and a burst of 1s to another location. The data from the back-to-back reads from these two locations is used as a simple data pattern for read deskew calibration.

**Figure 77. Guarantee Write – Writing a Simple Data Pattern to Memory**

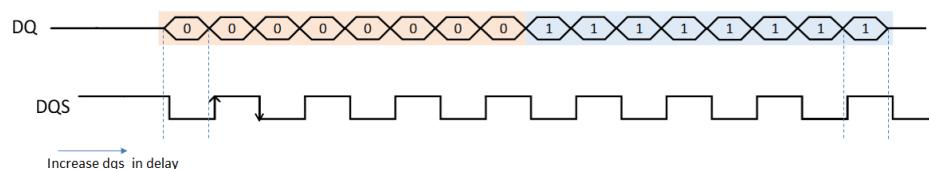


Before the write calibration, the deskew calibration algorithm performs a guaranteed write, and then sweeps dqs\_in delay values from low to high, to find the right-hand edge of the read window. The algorithm then sweeps dq\_in delay values from low to high, to find the left-hand edge of the read window. The algorithm then applies updated dqs\_in and dq\_in delay values to center the read window. The process then repeats for all data pins.

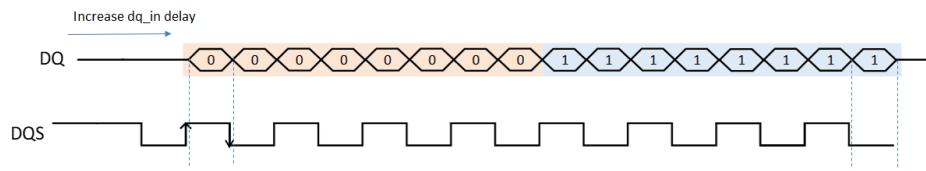
**Figure 78. A Passing Back-to-back Read with Simple Data Pattern**



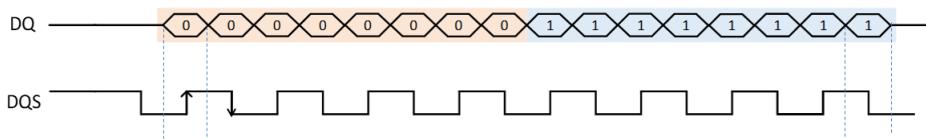
**Figure 79. Increase dqs\_in Delay Until Read Fails -Right Edge is Identified**



**Figure 80. Increase dq\_in Delay Until Read Fails-Left Edge is Identified**



**Figure 81. Centering the Read Window**



After the write path is calibrated, the algorithm performs another read-deskew calibration using a complex data pattern.

#### 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. Vref-in is generated from the VCCIO on the I/O banks used for DQ/DQS signals and calibrated internally in FPGA.

#### 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.

#### 3.3.4.2.2. QDR-IV Write Calibration

##### Write Leveling Calibration

The algorithm optimizes the CK versus DK relationship. It is covered by address and command deskew calibration using the loopback mode.

##### Write Deskew

The algorithm 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.

#### 3.3.4.3. Guidelines for Debugging Calibration Issues

The following topics provide general guidelines for debugging calibration-related issues.

## General Hardware Debugging for Calibration Issues

1. Verify that the correct memory component or DIMM is installed on the circuit board.
2. 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.
3. 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.
4. Ensure that the PCB has correct termination resistors on the address and command signals. Refer to the Board Layout Guidelines section of this user guide for more information on suggested termination values.
5. If you implement the EMIF interface on I/O lanes in Bank 3A that are utilized for AVST x16 or AVST x32 configuration mode, ensure that the MSEL pins are not set to AVST x16 or AVST x32 configuration. These configuration modes utilize the I/O lane in the top sub-bank 3A and may cause EMIF calibration issues if the EMIF interface uses the same I/O lane as the AVST x16 or AVST x32 configuration mode.
6. 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.
7. 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.
8. For DDR4 discrete memory components, the TEN pin on the memory component must not be left floating. If the TEN feature is not used, connect the TEN pin directly to GND. If the TEN feature is used, connect the pin to GND through a 1KΩ resistor.
9. 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.
10. Check the relevant voltage rails for absolute value and for worst case noise. Suggested rails are V<sub>CC</sub>, V<sub>CCP</sub>, V<sub>CCIO\_PIO</sub>, V<sub>CCPT</sub>, V<sub>CCA\_PLL</sub>, V<sub>REF</sub>, V<sub>TT</sub> and the power supplies at the DDR4 memory device.
11. Ensure that the reset signal to the DDR4 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.
12. Check to determine whether the calibration problem exists on more than one board.

13. 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. Refer to [Intel Agilex 7 F-Series and I-Series F-Series and I-Series FPGA EMIF IP – I/O Timing Closure](#) for more information.
14. Repeat the calibration multiple times without reconfiguring the device, to see whether the calibration can recover by recalibrating the interface.
15. Rerun the calibration by reconfiguring the device to see whether the calibration can recover after reconfiguring the device.

#### 3.3.4.3.1. 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.

#### 3.3.4.3.2. 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.

#### 3.3.4.3.3. 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<sub>T</sub>. Refer to the Board Design Guidelines section of 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.

### 3.3.4.3.4. Debugging DQS Enable Failure

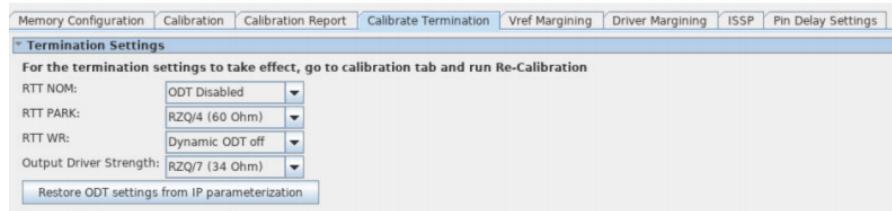
1. Verify that the correct resistor is connected between the RZQ pin of the FPGA and GND.
2. Verify that the correct resistor is connected between the ZQ pin of the memory component and GND.
3. Verify that there is no connectivity problem preventing the memory component from receiving the back-to-back 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)).

### 3.3.4.3.5. Debugging Read Deskew Calibration Failure

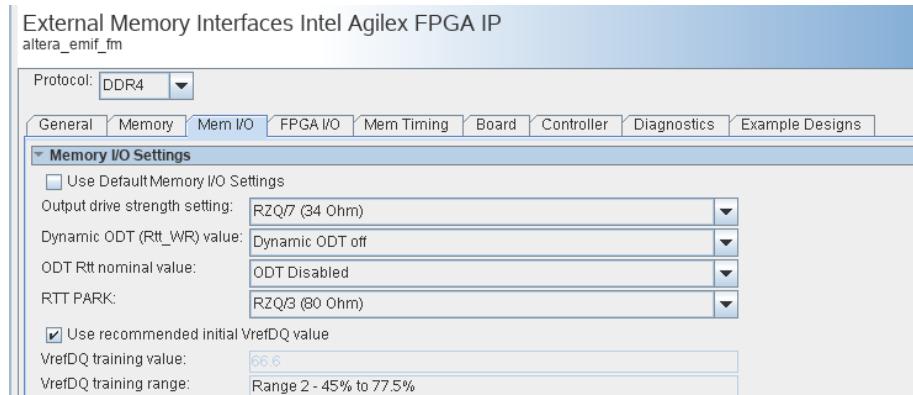
1. Ensure that you specify the correct memory timing parameter, CAS, and Write CAS latency when generating the EMIF IP. An incorrect parameter value can cause data corruption.
2. Determine which pins are failing.
  - If only certain DQ pins are failing, verify that there is no connectivity problem on 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 ( $R_{tt\_WR}$ ) 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 Debug Toolkit in your design, you can change the above settings on the *Calibrate Terminations* tab, without recompiling your design.

**Figure 82. Changing Termination Settings with the Debug Toolkit**

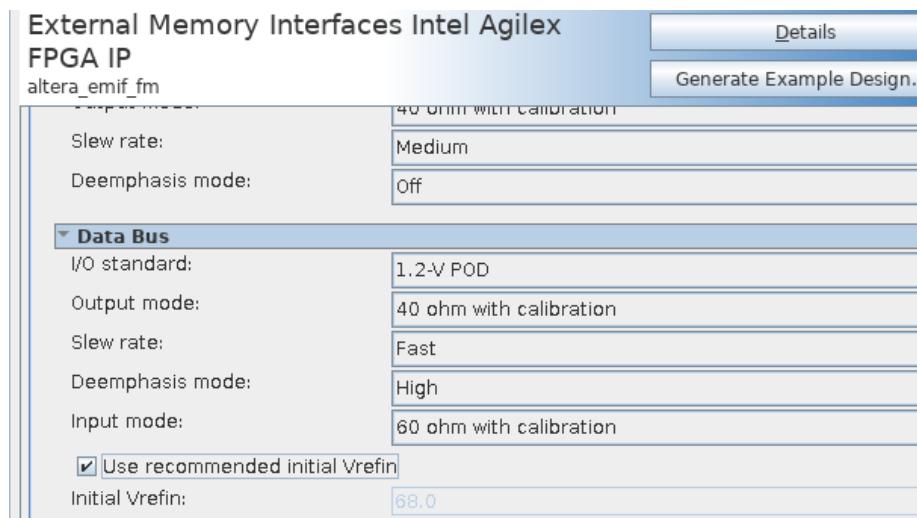


**Figure 83. Changing Termination Setting when Regenerating EMIF IP – Recompilation Required**



### 3.3.4.3.6. Debugging V<sub>REFIN</sub> Calibration Failure

1. Ensure that the V<sub>CCIO</sub> of the failing group is powered up to V<sub>CCIO</sub>=1.2V at the FPGA side.
2. Regenerate the EMIF IP with other Initial V<sub>REFIN</sub> values. It defaults to 68% when using the default FPGA I/O settings.

**Figure 84. Changing the Initial V<sub>REFIN</sub> Value**

### 3.3.4.3.7. 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.

### 3.3.4.3.8. 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.

### 3.3.4.3.9. Debugging Write Deskew Calibration Failure

If write deskew calibration is failing, perform the same checks as for [Debugging Read Deskew Calibration Failure](#).

### 3.3.4.3.10. Debugging V<sub>REFOUT</sub> 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.

## 3.4. Intel Agilex 7 F-Series and I-Series EMIF Controller

### 3.4.1. Hard Memory Controller

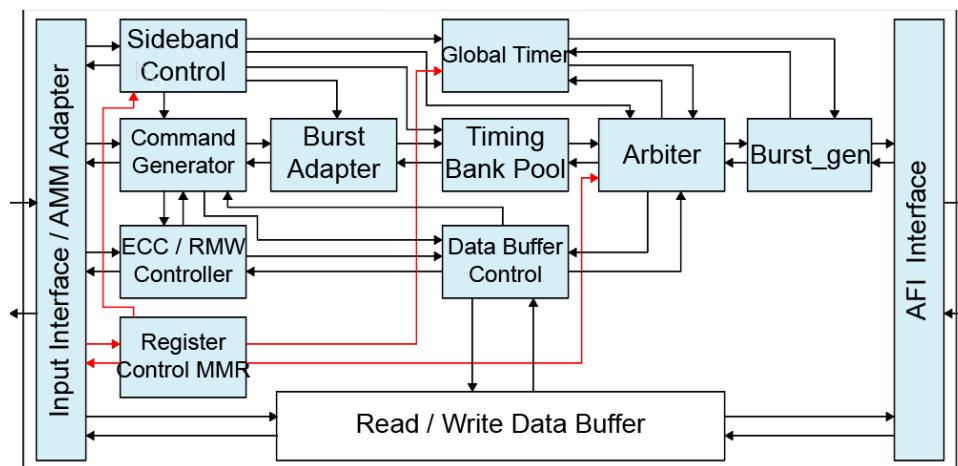
The Intel Agilex 7 F-Series and I-Series hard memory controller is designed for high speed, high performance, high flexibility, and area efficiency. The Intel Agilex 7 F-Series and I-Series hard memory controller supports the DDR4 memory standard.

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 sub-bank. The structure allows you to:

- Configure each I/O sub-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 85. 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 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 9. Features of the Intel Agilex 7 F-Series and I-Series Hard Memory Controller**

Feature	Description
Memory standards support	Supports DDR4 SDRAM.
Memory devices support	Supports the following memory devices: <ul style="list-style-type: none"> <li>• Discrete</li> <li>• UDIMM</li> <li>• RDIMM</li> <li>• LRDIMM</li> <li>• SODIMM</li> </ul>
3D Stacked Die support	Supports 2 and 4 height of 3D stacked die for DDR4 to increase memory capacity.
Memory controller bypass mode (Future support.)	You can use this configurable mode to bypass the hard memory controller and use your own customized controller.
Interface protocols support	<ul style="list-style-type: none"> <li>• Supports Avalon memory-mapped interface.</li> <li>• The PHY interface adheres to the AFI protocol.</li> </ul>
Rate support	The legal options are: <ul style="list-style-type: none"> <li>• HMC half-rate, user logic half-rate (extremely slow interfaces only)</li> <li>• HMC half-rate, user-logic quarter-rate</li> <li>• HMC quarter-rate, user-logic quarter-rate (extremely high-speed interfaces only)</li> </ul>
Configurable memory interface width	Supports data widths from 8 to 72 bits, in 8 bit increments
Multiple ranks support	Supports: <ul style="list-style-type: none"> <li>• 4 ranks with single slot</li> <li>• 2 ranks with dual slots</li> </ul>
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 memory-mapped interface specification, the maximum burst length is 64. No burst chop support for DDR4.

*continued...*

Feature	Description
Efficiency optimization features	<ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>Data reordering—the controller reorders read/write commands.</li> <li>Additive latency—the controller can issue a READ/WRITE command after the ACTIVATE command to the memory bank prior to <math>t_{RCD}</math>, which increases the command efficiency.</li> </ul>
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.
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	<ul style="list-style-type: none"> <li>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.</li> <li>Per-rank refresh—allows refresh for each individual rank.</li> <li>Controller-controlled refresh.</li> </ul>
ECC support	<ul style="list-style-type: none"> <li>8 bit ECC code; single error correction, double error detection (SECDED).</li> <li>User ECC supporting pass through user ECC bits as part of data bits.</li> <li>ECC is based on a Hamming coding scheme.</li> </ul>
Power saving features	<ul style="list-style-type: none"> <li>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.</li> <li>Automatic power down—puts the memory device in power down mode when the controller is idle. You can configure the idle waiting time.</li> <li>Memory clock gating.</li> </ul>
DDR4 features	<ul style="list-style-type: none"> <li>Bank group support—supports different timing parameters for between bank groups.</li> <li>Command/Address parity—command and address bus parity check.</li> <li>Support Direct Dual CS Mode and Direct QuadCS Mode for DDR4 LRDIMM devices.</li> <li>Support Encoded Quad CSMode for single CS assertion memory mapping for DDR4 LRDIMM devices.</li> </ul>
User ZQ calibration	Long or short ZQ calibration request for 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 10. Main Control Path Components**

Component	Description
Input interface	<ul style="list-style-type: none"> <li>• Accepts memory access commands from the core logic at half or quarter rate.</li> <li>• Uses the Avalon memory-mapped interface protocol.</li> <li>• You can connect the Avalon memory-mapped interface to an AXI bus master in Platform Designer. To connect the Avalon memory-mapped interface, implement the AXI bus master as a Platform Designer component and connect the AXI bus master to the Avalon memory-mapped slave. The Platform Designer interconnect performs the bus translation between the AXI and Avalon memory-mapped interface.</li> </ul>
Command generator and burst adapter	<ul style="list-style-type: none"> <li>• Drains your commands from the input interface and feeds them to the timing bank pool.</li> <li>• If read-modify-write is required, inserts the necessary read-modify-write read and write commands into the stream.</li> <li>• The burst adapter chops your arbitrary burst length to the number specified by the memory types.</li> </ul>
Timing Bank Pool	<ul style="list-style-type: none"> <li>• Key component in the memory controller.</li> <li>• Sets parallel queues to track command dependencies.</li> <li>• Signals the ready status of each command being tracked to the arbiter for the final dispatch.</li> <li>• Big scoreboard structure. The number of entries is currently sized to 16 where it monitors up to 16 commands at the same time.</li> <li>• 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.</li> <li>• Assist the arbiter in reordering row commands and column commands.</li> <li>• When the pool is full, a flow control signal is sent back upstream to stall the traffic.</li> </ul>
Arbiter	<ul style="list-style-type: none"> <li>• Enforces the arbitration rules.</li> <li>• Performs the final arbitration to select a command from all ready commands, and issues the selected command to the memory.</li> <li>• Supports Quasi-1T mode for half rate mode.</li> <li>• For the quasi modes, a row command must be paired with a column command.</li> </ul> <p><i>Note:</i> For quasi modes, a row command must be paired with a column command. Quasi-1T mode indicates that only one command is executed within one cycle; the command can be a row command or a column command. Quasi-2T allows two commands to be executed within one cycle and the two commands must be a row command paired with a column command.</p>
Global Timer	<p>Tracks the global timing constraints including:</p> <ul style="list-style-type: none"> <li>• <math>t_{FAW}</math>—the Four Activates Window parameter that specifies the time period in which only four activate commands are allowed.</li> <li>• <math>t_{RRD}</math>—the delay between back-to-back activate commands to different banks.</li> <li>• Some of the bus turnaround time parameters.</li> </ul>
MMR/IOCSR	<ul style="list-style-type: none"> <li>• The host of all the configuration registers.</li> <li>• Uses Avalon memory-mapped interface bus to talk to the core.</li> <li>• Core logic can read and write all the configuration bits.</li> </ul>
Sideband	Executes the refresh and power down features.
ECC controller	Although ECC encoding and decoding is performed in soft logic <sup>(1)</sup> , the ECC controller maintains the read-modify-write state machine in the hard solution. Because ECC is implemented in soft logic, it uses additional resources.
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:

- 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.

### 3.4.2. Intel Agilex 7 F-Series and I-Series 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 improve efficiency and help reduce overall latency, the hard memory controller and PHY run at half rate when the rate conversion feature is enabled. User logic runs 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.
- 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. User-requested Reset in Intel Agilex 7 F-Series and I-Series EMIF IP

The following table summarizes information about the user-requested reset mechanism in the Intel Agilex 7 F-Series and I-Series EMIF IP.

---

<sup>(1)</sup> 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.

**Table 11. User-requested Reset**

Item	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 upon completion of the first calibration, whether the calibration is successful or not. In subsequent calibration in user mode, local_reset_done transitions high once the calibration is completed. The local_reset_done signal takes longer to transition high in first calibration after device power-on as more operations are required to 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 necessary 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?	Each EMIF IP instance has its own local reset request port which it must assert in order to be recalibrated. The I/O SSM continually scans the reset requests of all the EMIF interfaces that it controls, and recalibrates them when it is able to do so. The exact timing of the recalibration cannot be predicted.
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 row).	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.

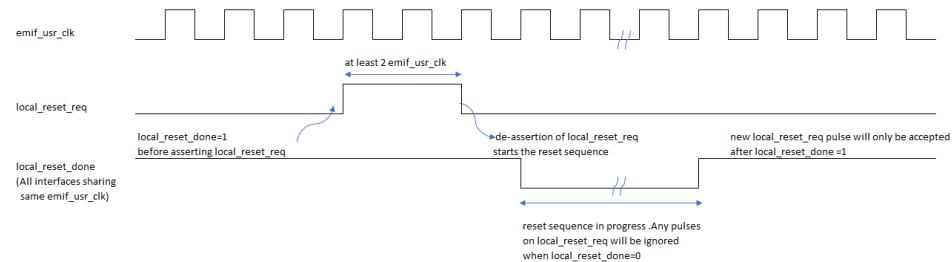
As part of the device power-on sequence, the local\_reset\_done signal automatically transitions to high upon the completion of the interface calibration sequence, regardless of whether calibration is successful or not.

## 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 asynchronously; 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.

**Figure 86. User-requested Reset Timing Diagram**



## 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`, `local_cal_success`, `local_cal_fail`, and `amm_ready`.

## 3.6. Intel Agilex 7 F-Series and I-Series EMIF for Hard Processor Subsystem

The Intel Agilex 7 F-Series and I-Series EMIF IP can enable the Intel Agilex 7 F-Series and I-Series Hard Processor Subsystem (HPS) to access external DRAM memory devices.

To enable connectivity between the Intel Agilex 7 HPS and the Intel Agilex 7 EMIF IP, you must create and configure an instance of the Intel Agilex 7 External Memory Interface for HPS IP core, and use Platform Designer to connect it to the Intel Agilex 7 Hard Processor Subsystem instance in your system.

### Supported Modes

The Intel Agilex 7 Hard Processor Subsystem is compatible with the following external memory configurations:

**Table 12. Intel Agilex 7 F-Series and I-Series Hard Processor Subsystem Compatibility**

Protocol	DDR4
Maximum memory clock frequency	1600MHz
Configuration	Hard PHY with hard memory controller
Clock rate of PHY and hard memory controller	Half-rate, Quarter-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
Memory format	Supports up to 32GB of memory. <ul style="list-style-type: none"><li>• Discrete components with up to 2 chip selects *</li><li>• Non-3DS UDIMM or RDIMM with up to 2 chip selects *</li><li>• SODIMM with up to 2 ranks *</li></ul>

\* Only one differential memory clock output is provided; therefore, you must do one of the following:

- Use single-rank discrete components, UDIMMs, or SODIMMs.
- Use dual-rank components that require only one clock input (for example, dual-die packages).
- Use RDIMMs that rely only on one clock input.
- Use the single clock output to drive both clock inputs and confirm through simulation that the memory interface margins are not adversely affected by the double loading of the clock output.

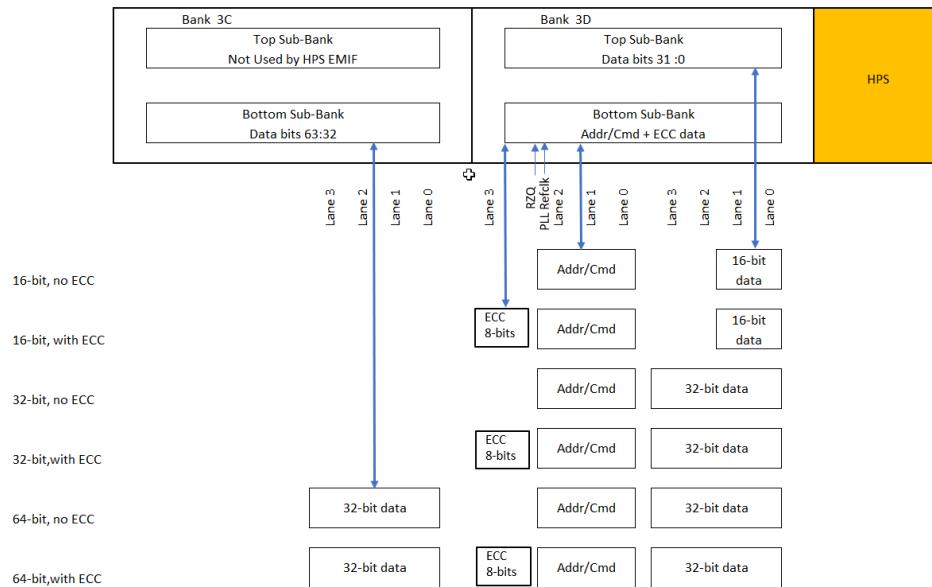
### 3.6.1. Restrictions on I/O Bank Usage for Intel Agilex 7 F-Series and I-Series EMIF IP with HPS

You can use only certain Intel Agilex 7 F-Series and I-Series I/O banks to implement Intel Agilex 7 F-Series and I-Series EMIF IP with the Intel Agilex 7 F-Series and I-Series Hard Processor Subsystem (HPS).

The restrictions on I/O bank usage result from the Intel Agilex 7 F-Series and I-Series HPS having hard-wired connections to the EMIF circuits in the I/O banks closest to the HPS. For any given EMIF configuration, the pin-out of the EMIF-to-HPS interface is fixed.

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

**Figure 87. Intel Agilex 7 F-Series and I-Series HPS - EMIF I/O Bank and Lanes Usage**



The HPS EMIF uses the closest located external memory interfaces I/O banks to connect to SDRAM. This arrangement of HPS EMIF address and command sub-banks relative to the data sub-banks is not supported for fabric EMIF in the current version of the Intel Quartus Prime Design Suite.

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 88. I/O Pin Usage Restrictions for Intel Agilex 7 F-Series and I-Series External Memory Interface with HPS (1 of 3)**

Bank 3C, Bottom Sub-Bank(Sub\_bank for Data bits 63:32)

Lane	Index		Pin Name	x16/x24		x32/x40		x64/x72	
	Within Sub-bank	Within Lane		ECC OFF	ECC ON	ECC OFF	ECC ON	ECC OFF	ECC ON
Lane 3	47	11	LVDSRX3C_13N						
	46	10	LVDSRX3C_13P						
	45	9	LVDSTX3C_13N						
	44	8	LVDSTX3C_13P						
	43	7	LVDSRX3C_14N						
	42	6	LVDSRX3C_14P						
	41	5	LVDSTX3C_14N						
	40	4	LVDSTX3C_14P						
	39	3	LVDSRX3C_15N						
	38	2	LVDSRX3C_15P						
	37	1	LVDSTX3C_15N						
	36	0	LVDSTX3C_15P						
	35	11	LVDSRX3C_16N						
	34	10	LVDSRX3C_16P						
	33	9	LVDSTX3C_16N						
Lane 2	32	8	LVDSTX3C_16P						
	31	7	LVDSRX3C_17N						
	30	6	LVDSRX3C_17P						
	29	5	LVDSTX3C_17N						
	28	4	LVDSTX3C_17P						
	27	3	LVDSRX3C_18N						
	26	2	LVDSRX3C_18P						
	25	1	LVDSTX3C_18N						
	24	0	LVDSTX3C_18P						
	23	11	LVDSRX3C_19N						
	22	10	LVDSRX3C_19P						
	21	9	LVDSTX3C_19N						
	20	8	LVDSTX3C_19P						
	19	7	LVDSRX3C_20N						
Lane 1	18	6	LVDSRX3C_20P						
	17	5	LVDSTX3C_20N						
	16	4	LVDSTX3C_20P						
	15	3	LVDSRX3C_21N						
	14	2	LVDSRX3C_21P						
	13	1	LVDSTX3C_21N						
	12	0	LVDSTX3C_21P						
	11	11	LVDSRX3C_22N						
	10	10	LVDSRX3C_22P						
	9	9	LVDSTX3C_22N						
	8	8	LVDSTX3C_22P						
	7	7	LVDSRX3C_23N						
	6	6	LVDSRX3C_23P						
	5	5	LVDSTX3C_23N						
	4	4	LVDSTX3C_23P						
	3	3	LVDSRX3C_24N						
	2	2	LVDSRX3C_24P						
	1	1	LVDSTX3C_24N						
	0	0	LVDSTX3C_24P						

Addr/Cmd pins only (Do not use unused pins)	RZQ only
ALERT_N Pin only	HPS REFCLK_P only
Do Not Use (No Connect)	HPS REFCLK_N only (LVDS reference clock mode, no connect if unused)
DQSp Pin Only	Free for FPGA Fabric Use
DQSn Pin Only	
DBIDM Pin Only	
DQ Pin Only	

**Figure 89. I/O Pin Usage Restrictions for Intel Agilex 7 F-Series and I-Series External Memory Interface with HPS (2 of 3)**

Bank 3D, Top Sub-Bank (Sub-bank for Data bits 31:0)

Lane	Index		Pin Name	x16/x24		x32/x40		x64/x72	
	Within Sub-bank	Within Lane		ECC OFF	ECC ON	ECC OFF	ECC ON	ECC OFF	ECC ON
Lane 3	95	11	LVDSRX3D_1N						
	94	10	LVDSRX3D_1P						
	93	9	LVDSTX3D_1N						
	92	8	LVDSTX3D_1P						
	91	7	LVDSPRX3D_2N						
	90	6	LVDSPRX3D_2P						
	89	5	LVDSTX3D_2N						
	88	4	LVDSTX3D_2P						
	87	3	LVDSRX3D_3N						
	86	2	LVDSRX3D_3P						
	85	1	LVDSTX3D_3N						
	84	0	LVDSTX3D_3P						
Lane 2	83	11	LVDSPRX3D_4N						
	82	10	LVDSPRX3D_4P						
	81	9	LVDSTX3D_4N						
	80	8	LVDSTX3D_4P						
	79	7	LVDSPRX3D_5N						
	78	6	LVDSPRX3D_5P						
	77	5	LVDSTX3D_5N						
	76	4	LVDSTX3D_5P						
	75	3	LVDSPRX3D_6N						
	74	2	LVDSPRX3D_6P						
	73	1	LVDSTX3D_6N						
	72	0	LVDSTX3D_6P						
Lane 1	71	11	LVDSPRX3D_7N						
	70	10	LVDSPRX3D_7P						
	69	9	LVDSTX3D_7N						
	68	8	LVDSTX3D_7P						
	67	7	LVDSPRX3D_8N						
	66	6	LVDSPRX3D_8P						
	65	5	LVDSTX3D_8N						
	64	4	LVDSTX3D_8P						
	63	3	LVDSPRX3D_9N						
	62	2	LVDSPRX3D_9P						
	61	1	LVDSTX3D_9N						
	60	0	LVDSTX3D_9P						
Lane 0	59	11	LVDSPRX3D_10N						
	58	10	LVDSPRX3D_10P						
	57	9	LVDSTX3D_10N						
	56	8	LVDSTX3D_10P						
	55	7	LVDSPRX3D_11N						
	54	6	LVDSPRX3D_11P						
	53	5	LVDSTX3D_11N						
	52	4	LVDSTX3D_11P						
	51	3	LVDSPRX3D_12N						
	50	2	LVDSPRX3D_12P						
	49	1	LVDSTX3D_12N						
	48	0	LVDSTX3D_12P						

	Addr/Cmd pins only (Do not use unused pins)		RZQ only
	ALERT_N Pin only		HPS REFCLK_P only
	Do Not Use (No Connect)		HPS REFCLK_N only (LVDS reference clock mode, no connect if unused)
	DQSp Pin Only		Free for FPGA Fabric Use
	DQSn Pin Only		
	DBI/IDM Pin Only		
	DQ Pin Only		

**Figure 90. I/O Pin Usage Restrictions for Intel Agilex 7 F-Series and I-Series External Memory Interface with HPS (3 of 3)**

Bank 3D, Bottom Sub-Bank (Sub-bank for Addr/Cmd + ECC Data)

Lane	Index		Pin Name	x16/x24		x32/x40		x64/x72	
	Within Sub-bank	Within Lane		ECC OFF	ECC ON	ECC OFF	ECC ON	ECC OFF	ECC ON
Lane 3	47	11	LVDSRX3D_13N						
	46	10	LVDSRX3D_13P						
	45	9	LVDSTX3D_13N						
	44	8	LVDSTX3D_13P						
	43	7	LVDSRX3D_14N						
	42	6	LVDSRX3D_14P						
	41	5	LVDSTX3D_14M						
	40	4	LVDSTX3D_14P						
	39	3	LVDSRX3D_15N						
	38	2	LVDSRX3D_15P						
	37	1	LVDSTX3D_15N						
	36	0	LVDSTX3D_15P						
	35	11	LVDSRX3D_16N						
	34	10	LVDSRX3D_16P						
	33	9	LVDSTX3D_16N						
	32	8	LVDSTX3D_16P				ALERT_N only		
	31	7	LVDSRX3D_17N						
	30	6	LVDSRX3D_17P						
	29	5	LVDSTX3D_17N						
	28	4	LVDSTX3D_17P						
	27	3	LVDSRX3D_18N						
	26	2	LVDSRX3D_18P				RZQ only		
	25	1	LVDSTX3D_18N						
	24	0	LVDSTX3D_18P						
Lane 2	23	11	LVDSRX3D_19N						
	22	10	LVDSRX3D_19P						
	21	9	LVDSTX3D_19N						
	20	8	LVDSTX3D_19P						
	19	7	LVDSRX3D_20N						
	18	6	LVDSRX3D_20P						
	17	5	LVDSTX3D_20N						
	16	4	LVDSTX3D_20P						
	15	3	LVDSRX3D_21N						
	14	2	LVDSRX3D_21P						
	13	1	LVDSTX3D_21N						
	12	0	LVDSTX3D_21P						
Lane 1	11	11	LVDSRX3D_22N						
	10	10	LVDSRX3D_22P						
	9	9	LVDSTX3D_22N						
	8	8	LVDSTX3D_22P						
	7	7	LVDSRX3D_23N						
	6	6	LVDSRX3D_23P						
	5	5	LVDSTX3D_23N						
	4	4	LVDSTX3D_23P						
	3	3	LVDSRX3D_24N						
	2	2	LVDSRX3D_24P						
	1	1	LVDSTX3D_24N						
	0	0	LVDSTX3D_24P						
Lane 0	11	11	LVDSRX3D_25N						
	10	10	LVDSRX3D_25P						
	9	9	LVDSTX3D_25N						
	8	8	LVDSTX3D_25P						
	7	7	LVDSRX3D_26N						
	6	6	LVDSRX3D_26P						
	5	5	LVDSTX3D_26N						
	4	4	LVDSTX3D_26P						
	3	3	LVDSRX3D_27N						
	2	2	LVDSRX3D_27P						
	1	1	LVDSTX3D_27N						
	0	0	LVDSTX3D_27P						

Addr/Cmd pins only (Do not use unused pins)	RZQ only
ALERT_N Pin only	HPS REFCLK_P only
Do Not Use (No Connect)	HPS REFCLK_N only (LVDS reference clock mode, no connect if unused)
DQSp Pin Only	Free for FPGA Fabric Use
DQSn Pin Only	
DBI/DM Pin Only	
DQ Pin Only	

The HPS EMIF IP must be used whenever the HPS is active. Thus, you should be aware that enabling the HPS necessarily means that an EMIF must be placed at this location in order to implement an FPGA design.

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 3D, Bottom Sub-bank (Sub-bank for Address/Command + ECC Data):
  - Lane 3 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 must not be used by the FPGA fabric.
  - ALERT\_N pin must be placed at pin index 8, lane 2. There is no flexibility on this,
- Bank 3D, Top Sub-bank (Sub-bank for data bits 31:0) :
  - Lanes 3, 2, 1, and 0 are used for data bits.
  - With 32-bit data widths, unused pins in this bank must not be used by the FPGA fabric.
  - With 16-bit data widths, lanes 0 and 1 are used as data lanes. Unused pins in lane 0 and lane 1 must not be used by FPGA fabric. Unused pins in lanes 2 and 3 must not be used by the FPGA fabric, even though lanes 2 and 3 are not used by HPS EMIF.
- Bank 3C, Bottom Sub-bank (Sub-bank for Data bits 63:32)
  - With 64-bit data widths, lanes 3, 2, 1, and 0 are used for data bits [63:32]. Unused pins in these lanes must not be used by the FPGA fabric.
  - With 32-bit data widths, the entire bottom sub-bank can be used by the FPGA fabric. There are no restrictions.
- Bank 3C, Top Sub-bank
  - Not used by HPS EMIF. Unused pins in this bank can be used by FPGA fabric when the bottom sub-bank in 3C is not used for 64-bit HPS EMIF.
  - The following restrictions apply on the top sub-bank when the bottom sub-bank in 3C is used for 64-bit HPS EMIF:
    - This sub-bank can be used to form a larger non-HPS EMIF, but you cannot place an address and command bank in this sub-bank.
    - 1.5V true differential signaling is not supported.
    - I/O PLL reconfiguration is not supported.

By default, the Intel Agilex 7 F-Series and I-Series 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 0, 1, 2, 3, 8, 9, 10, 11. 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 6. There is no flexibility.
  - DQS and DQS# must use pins at index 4 and 5, respectively. There is no flexibility.
  - Pin index 7 must have no fabric usage and cannot implement general purpose I/Os.
2. In all cases the DQS groups can be swapped around the I/O banks shown. There is no requirement for the ECC DQS group to be placed in the bottom sub-bank in bank 3D.
  3. In the bottom sub-bank in bank 3D (sub-bank for address and command + ECC data):
    - You must not change placement of the address and command pins from the default.
    - Place the alert# pin in lane 2, pin index 8.
    - Place the PLL reference clock in this sub-bank. Failure to place the PLL reference clock in this sub-bank causes device configuration problems. The PLL reference clock must be running at the correct frequency before device configuration occurs.
    - Place the RZQ pin in this sub-bank. Failure to place the RZQ pin in this sub-bank causes Fitter or device configuration problems.
  4. 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. Using a Custom Controller with the Hard PHY

If you want to use your own custom memory controller, you must integrate that controller with the hard PHY to achieve a complete memory solution.

Observe the following general guidelines:

- When you configure your external memory interface IP, ensure that you select **Configuration > Hard PHY Only** under the **Interface** group on the **General** tab in the parameter editor.
- The AFI interface is exposed at the top-level of the generated IP core; you can connect the AFI interface to your custom controller. Consult the AFI section for more detailed information on the AFI interface to the PHY.

#### Related Information

- [Intel Agilex 7 F-Series and I-Series EMIF Architecture: Introduction](#) on page 12
- [Intel Agilex 7 F-Series and I-Series EMIF IP AFI Signals](#) on page 87

## 4. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – End-User Signals

### 4.1. Intel Agilex 7 F-Series and I-Series EMIF IP Interface and Signal Descriptions

The following sections describe each of the interfaces and their signals, by protocol, for the Intel Agilex 7 F-Series and I-Series EMIF IP.

#### 4.1.1. Intel Agilex 7 EMIF IP Interfaces for DDR4

The interfaces in the Intel Agilex 7 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 13. 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
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
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
ctrl_amm_aux	Conduit	IF_CTRL_AMM_AUX_DESC

*continued...*

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

\*Other names and brands may be claimed as the property of others.

ISO  
9001:2015  
Registered

Interface Name	Interface Type	Description
ctrl_auto_preamble	Conduit	Controller auto-preamble 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
emif_calbus	Conduit	EMIF calibration component interface
emif_calbus_clk	Clock Output	EMIF calibration component clock input interface

#### 4.1.1.1. local\_reset\_req for DDR4

Local reset request. Output signal from local\_reset\_combiner

**Table 14. 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 DDR4

Local reset status. Input signal to the local\_reset\_combiner

**Table 15. 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 DDR4

PLL reference clock input

**Table 16. Interface: pll\_ref\_clk**

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

### Related Information

Intel Agilex 7 Device Data Sheet

Refer to *Differential I/O Standards Specifications for Intel Agilex 7 Devices*, in the *Intel Agilex 7 Device Data Sheet*.

#### 4.1.1.4. pll\_locked for DDR4

PLL locked signal

**Table 17.** **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. ac\_parity\_err for DDR4

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.

**Table 18.** **Interface: ac\_parity\_err**

Interface type: Conduit

Port Name	Direction	Description
ac_parity_err	Output	PORT_AC_PARITY_STATE_DESC

#### 4.1.1.6. oct for DDR4

On-Chip Termination (OCT) interface

**Table 19.** **Interface: oct**

Interface type: Conduit

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

#### 4.1.1.7. mem for DDR4

Interface between FPGA and external memory

**Table 20.** **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_bg	Output	Bank group

*continued...*

<b>Port Name</b>	<b>Direction</b>	<b>Description</b>
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_db1_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

#### 4.1.1.8. status for DDR4

PHY calibration status interface

**Table 21. Interface: status**

Interface type: Conduit

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

#### 4.1.1.9. afi\_reset\_n for DDR4

AFI reset interface

**Table 22. Interface: afi\_reset\_n**

Interface type: Reset Output

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

#### 4.1.1.10. afi\_clk for DDR4

AFI clock interface

**Table 23. Interface: afi\_clk**

Interface type: Clock Output

<b>Port Name</b>	<b>Direction</b>	<b>Description</b>
afi_clk	Output	Clock for the Altera PHY Interface (AFI)

#### 4.1.1.11. afi\_half\_clk for DDR4

AFI half-rate clock interface

**Table 24. 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

**4.1.1.12. afi for DDR4**

Altera PHY Interface (AFI)

**Table 25. 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
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

#### **4.1.1.13. emif\_usr\_reset\_n for DDR4**

User clock domain reset interface

**Table 26. 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

#### **4.1.1.14. emif\_usr\_clk for DDR4**

User clock interface

**Table 27. Interface: emif\_usr\_clk**

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

#### **4.1.1.15. ctrl\_amm for DDR4**

Controller Avalon Memory-Mapped interface

**Table 28. 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_bytelenable	Input	Byte-enable for write data
amm_beginbursttransfer	Input	Indicates when a burst is starting
amm_readdatavalid	Output	Indicates whether read data is valid

#### **4.1.1.16. ctrl\_amm\_aux for DDR4**

IF\_CTRL\_AMM\_AUX\_DESC

**Table 29. Interface: ctrl\_amm\_aux**

Interface type: Conduit

Port Name	Direction	Description
amm_early_ready	Output	1-cycle early version of Wait-request

#### 4.1.1.17. ctrl\_auto\_prelude for DDR4

Controller auto-precharge interface

**Table 30. Interface: ctrl\_auto\_prelude**

Interface type: Conduit

Port Name	Direction	Description
ctrl_auto_prelude_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.

#### 4.1.1.18. ctrl\_user\_priority for DDR4

Controller user-requested priority interface

**Table 31. 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.

#### 4.1.1.19. ctrl\_ecc\_user\_interrupt for DDR4

Controller ECC user interrupt interface

**Table 32. 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

#### 4.1.1.20. ctrl\_ecc\_readdataerror for DDR4

Controller ECC read data error indication interface

**Table 33. 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.

#### 4.1.1.21. ctrl\_ecc\_status for DDR4

Controller ECC status interface

**Table 34. Interface: ctrl\_ecc\_status**

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_sts_intr	Output	ECC interrupt status - '1' indicates interrupt occurred.
ctrl_ecc_sts_sbe_error	Output	'1' indicates SBE occurred.
ctrl_ecc_sts_dbe_error	Output	'1' indicates DBE occurred.
ctrl_ecc_sts_corr_dropped	Output	Correction command dropped status, '1' indicates correction command dropped.
ctrl_ecc_sts_sbe_count	Output	Number of times SBE error occurred.
ctrl_ecc_sts_dbe_count	Output	Number of times DBE error occurred.
ctrl_ecc_sts_corr_dropped_count	Output	Number of times correction command dropped.
ctrl_ecc_sts_err_addr	Output	Address of the most recent SBE or DBE.
ctrl_ecc_sts_corr_dropped_addr	Output	Address of the most recent correction command dropped.

#### 4.1.1.22. ctrl\_mmr\_slave for DDR4

Controller MMR slave interface

**Table 35. 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

#### 4.1.1.23. hps\_emif for DDR4

Conduit between Hard Processor Subsystem and memory interface

**Table 36. 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

**4.1.1.24. emif\_calbus for DDR4**

EMIF calibration component interface

**Table 37. Interface: emif\_calbus**

Interface type: Conduit

Port Name	Direction	Description
calbus_read	Output	EMIF Calibration component bus for read
calbus_write	Output	EMIF Calibration component bus for write
calbus_address	Output	EMIF Calibration component bus for address
calbus_wdata	Output	EMIF Calibration component bus for write data
calbus_rdata	Input	EMIF Calibration component bus for read data
calbus_seq_param_tbl	Input	EMIF Calibration component bus for parameter table data

**4.1.1.25. emif\_calbus\_clk for DDR4**

EMIF calibration component clock input interface

**Table 38. Interface: emif\_calbus\_clk**

Interface type: Clock Output

Port Name	Direction	Description
calbus_clk	Output	EMIF Calibration component bus for the clock

**4.1.2. Intel Agilex 7 EMIF IP Interfaces for QDR-IV**

The interfaces in the Intel Agilex 7 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 39. 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

*continued...*

Interface Name	Interface Type	Description
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
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
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
emif_calbus	Conduit	EMIF calibration component interface
emif_calbus_clk	Clock Output	EMIF calibration component clock input interface

#### 4.1.2.1. local\_reset\_req for QDR-IV

Local reset request. Output signal from local\_reset\_combiner

**Table 40. 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 QDR-IV

Local reset status. Input signal to the local\_reset\_combiner

**Table 41. 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 QDR-IV

PLL reference clock input

**Table 42. Interface: pll\_ref\_clk**

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

##### Related Information

[Intel Agilex 7 Device Data Sheet](#)

Refer to *Differential I/O Standards Specifications for Intel Agilex 7 Devices*, in the [Intel Agilex 7 Device Data Sheet](#).

#### 4.1.2.4. pll\_locked for QDR-IV

PLL locked signal

**Table 43. 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. oct for QDR-IV

On-Chip Termination (OCT) interface

**Table 44. Interface: oct**

Interface type: Conduit

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

#### 4.1.2.6. mem for QDR-IV

Interface between FPGA and external memory

**Table 45. 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

*continued...*

<b>Port Name</b>	<b>Direction</b>	<b>Description</b>
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
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

#### 4.1.2.7. status for QDR-IV

PHY calibration status interface

**Table 46. Interface: status**

Interface type: Conduit

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

#### 4.1.2.8. afi\_reset\_n for QDR-IV

AFI reset interface

**Table 47. Interface: afi\_reset\_n**

Interface type: Reset Output

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

#### 4.1.2.9. afi\_clk for QDR-IV

AFI clock interface

**Table 48. Interface: afi\_clk**

Interface type: Clock Output

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

#### 4.1.2.10. afi\_half\_clk for QDR-IV

AFI half-rate clock interface

**Table 49. 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

#### 4.1.2.11. afi for QDR-IV

Altera PHY Interface (AFI)

**Table 50. 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

#### 4.1.2.12. emif\_usr\_reset\_n for QDR-IV

User clock domain reset interface

**Table 51. 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

#### 4.1.2.13. emif\_usr\_clk for QDR-IV

User clock interface

**Table 52. Interface: emif\_usr\_clk**

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

#### 4.1.2.14. ctrl\_amm for QDR-IV

Controller Avalon Memory-Mapped interface

**Table 53. 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

#### 4.1.2.15. emif\_calbus for QDR-IV

EMIF calibration component interface

**Table 54. Interface: emif\_calbus**

Interface type: Conduit

Port Name	Direction	Description
calbus_read	Output	EMIF Calibration component bus for read
calbus_write	Output	EMIF Calibration component bus for write
calbus_address	Output	EMIF Calibration component bus for address
calbus_wdata	Output	EMIF Calibration component bus for write data
calbus_rdata	Input	EMIF Calibration component bus for read data
calbus_seq_param_tbl	Input	EMIF Calibration component bus for parameter table data

#### 4.1.2.16. emif\_calbus\_clk for QDR-IV

EMIF calibration component clock input interface

**Table 55. Interface: emif\_calbus\_clk**

Interface type: Clock Output

Port Name	Direction	Description
calbus_clk	Output	EMIF Calibration component bus for the clock

## 4.2. Intel Agilex 7 F-Series and I-Series EMIF IP 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.

### Related Information

[Using a Custom Controller with the Hard PHY](#) on page 72

### 4.2.1. AFI Clock and Reset Signals

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

**Table 56. 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 57. 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.

*continued...*

<b>Signal Name</b>	<b>Direction</b>	<b>Width</b>	<b>Description</b>
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_act_n	Input	AFI_CONTROL_WIDTH	ACT# (DDR4).
afi_RST_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR4 memory devices.)
afi_odt	Input	AFI_CLK_EN_WIDTH	On-die termination signal for 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)
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.

#### 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 58. Write Data Signals**

<b>Signal Name</b>	<b>Direction</b>	<b>Width</b>	<b>Description</b>
afi_dqs_burst	Input	AFI_RATE_RATIO	Controls the enable on the strobe (DQS) pins for 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. Also directly controls the PHY's mem_dbi signal for DDR4.

*continued...*

Signal Name	Direction	Width	Description
			The mem_dm and mem_db features share the same port on the memory device.

#### 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 59. 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 60. Calibration Status Signals**

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to <b>continued...</b>

Signal Name	Direction	Width	Description
			63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat <i>(1)</i>	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 Tracking Management Signals

When tracking management is enabled, the sequencer can take control over the AFI 4.0 interface at given intervals, and issue commands to the memory device to track the internal DQS Enable signal alignment to the DQS signal returning from the memory device. The tracking management portion of the AFI 4.0 interface provides a means for the sequencer and the controller to exchange handshake signals.

**Table 61. Tracking Management Signals**

Signal Name	Direction	Width	Description
afi_ctl_refresh_done	Input	4	Handshaking signal from controller to tracking manager, indicating that a refresh has occurred and waiting for a response.
afi_seq_busy	Output	4	Handshaking signal from sequencer to controller, indicating when DQS tracking is in progress.
afi_ctl_long_idle	Input	4	Handshaking signal from controller to tracking manager, indicating that it has exited low power state without a periodic refresh, and waiting for response.

#### 4.2.7. 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 62. Shadow Register Management Signals**

Signal Name	Direction	Width	Description
afi_wrrank	Input	AFI_WRANK_WIDTH	Signal from controller specifying which rank the write data is going to. The signal timing is identical to that of afi_dqs_burst. That is, afi_wrrank must be

*continued...*

Signal Name	Direction	Width	Description
			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 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup> rank respectively). The ordering within the bus is the same as other AFI signals. Specifically the bus is ordered by time slots, for example:

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

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

Where Tx is a number of rank-bit words that one-hot encodes the rank being accessed at the y<sup>th</sup> 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.

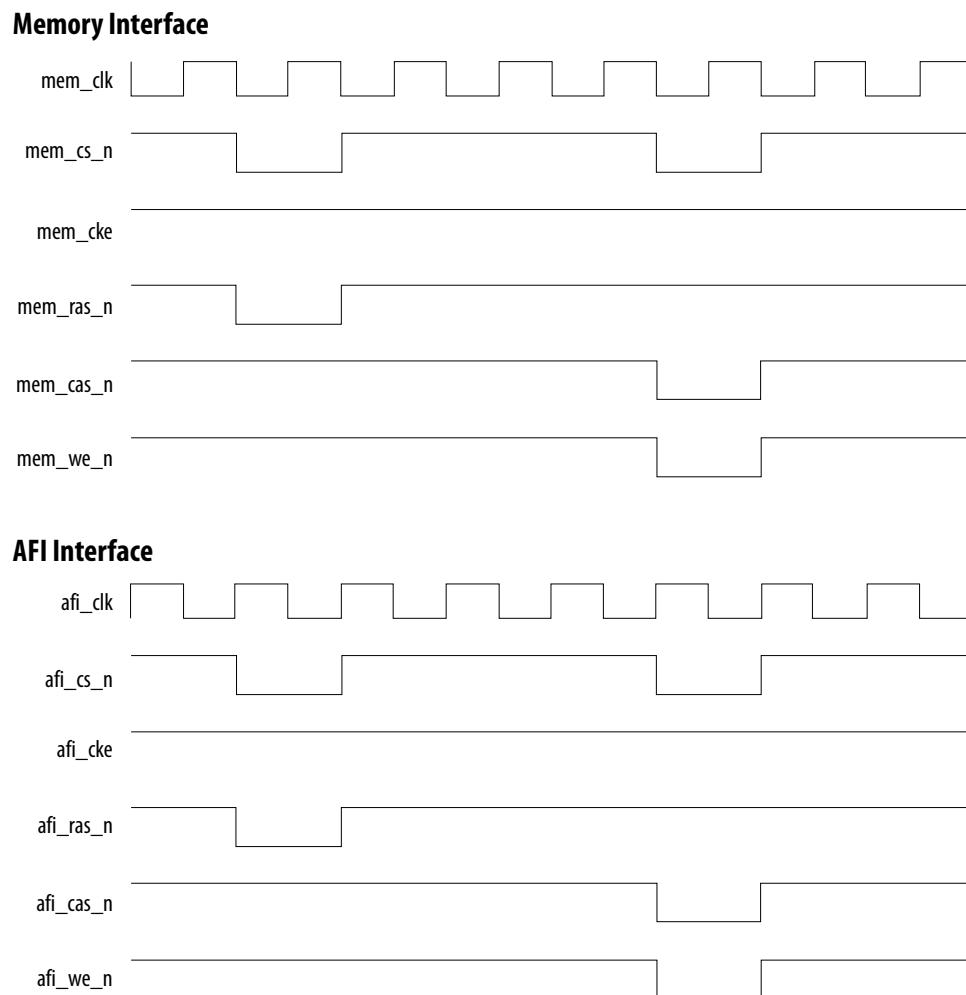
## 4.3. Intel Agilex 7 F-Series and I-Series EMIF IP AFI 4.0 Timing Diagrams

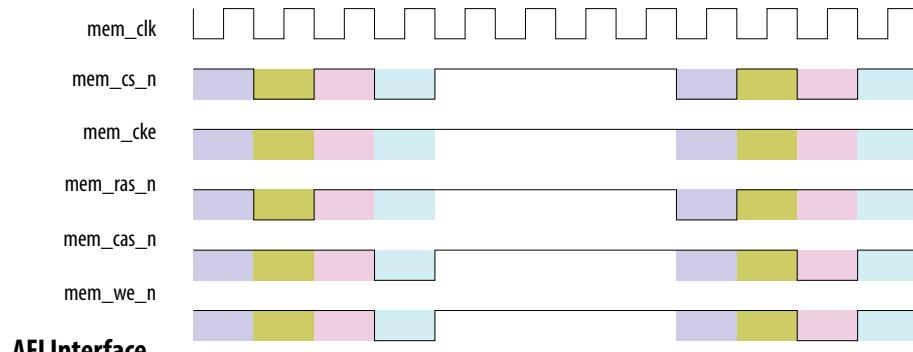
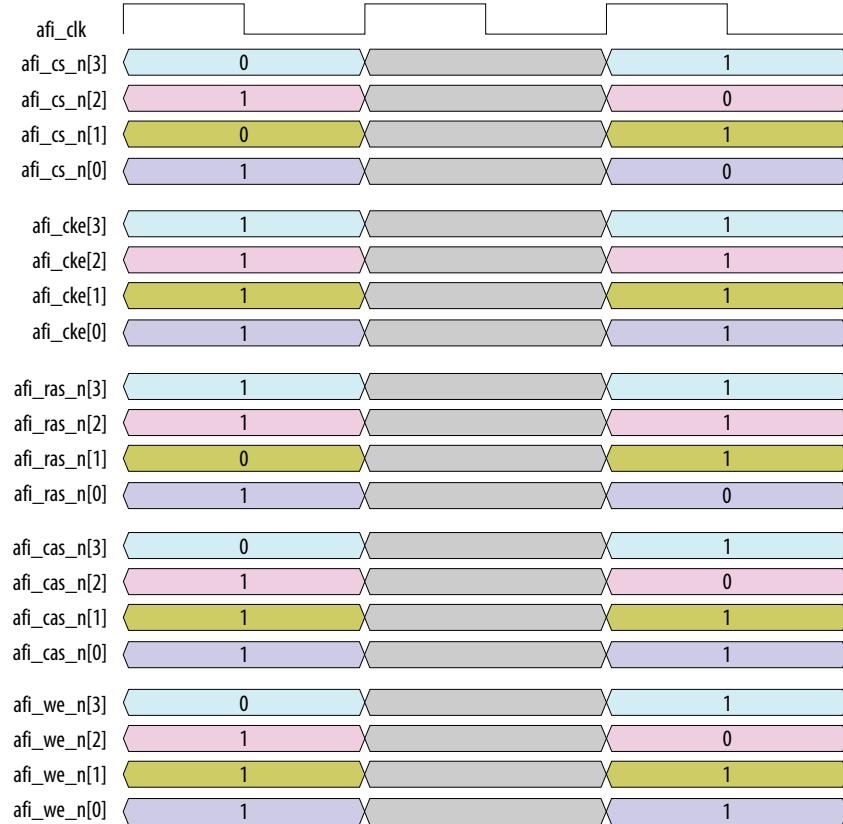
### 4.3.1. AFI Address and Command Timing Diagrams

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 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 91. AFI Address and Command Full-Rate**



**Figure 93. AFI Address and Command Quarter-Rate****Memory Interface****AFI Interface****4.3.2. AFI Write Sequence Timing Diagrams**

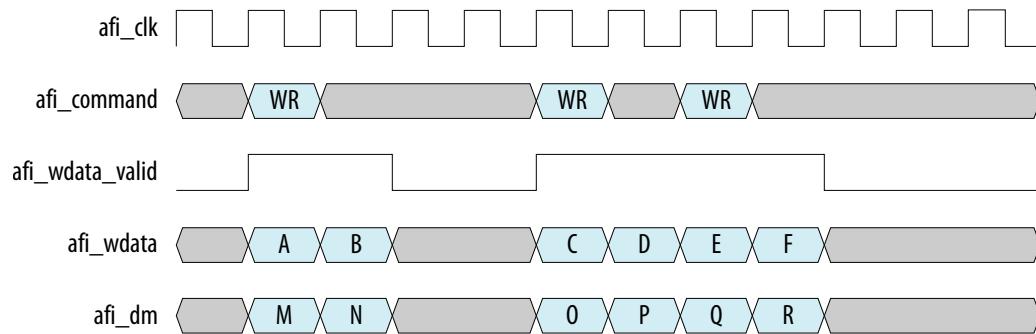
The following timing diagrams illustrate the relationships between the write command and corresponding write data and write enable signals, in full and quarter rate.

For quarter rate, when the write command is sent on the first memory clock in a PHY clock (for example,  $\text{afi\_cs\_n}[0] = 0$ ), 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

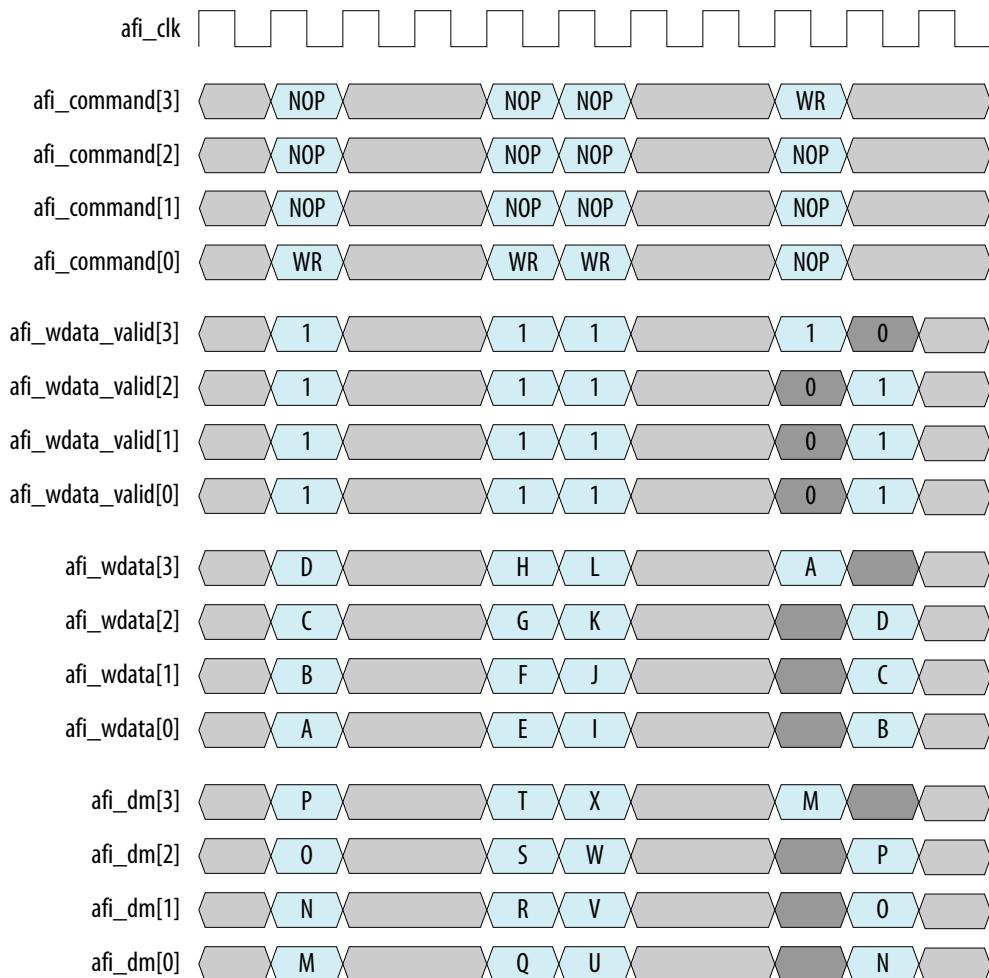
write command and the corresponding write data are constant on the AFI interface. For example, if a command is sent on the second memory clock in a PHY clock, the write data must also start at the second memory clock in a PHY clock.

### Write sequences with wlat=0

**Figure 94. AFI Write Data Full-Rate, wlat=0**

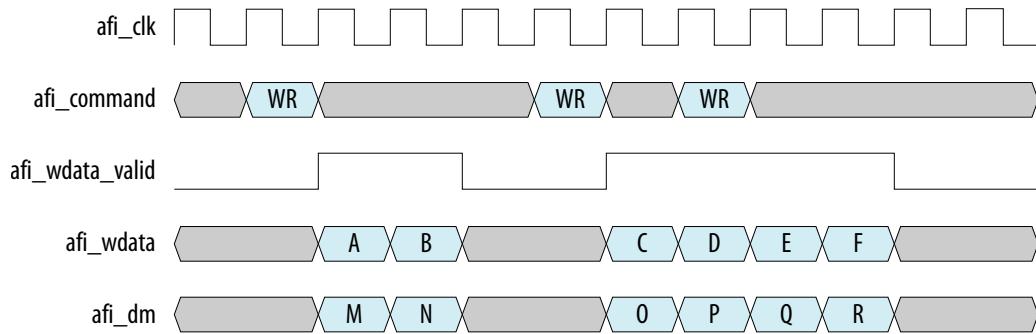
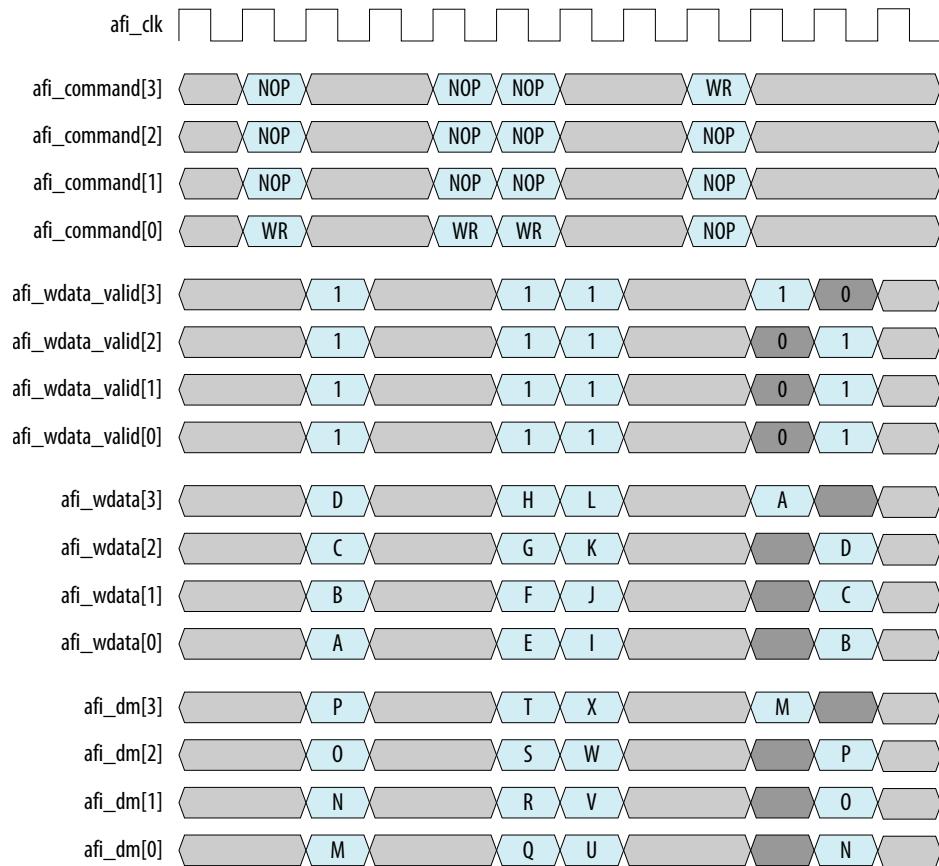


The following diagram illustrates both aligned and unaligned access. The first three write commands are aligned accesses where they were issued on LSB of `afi_command`. The fourth write command is unaligned access where it was issued on a different command slot. AFI signals must be shifted accordingly,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

**Figure 96. 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 97. AFI Write Data Full-Rate, wlat=1**

**Figure 99. AFI Write Data Quarter-Rate, wlat=1**


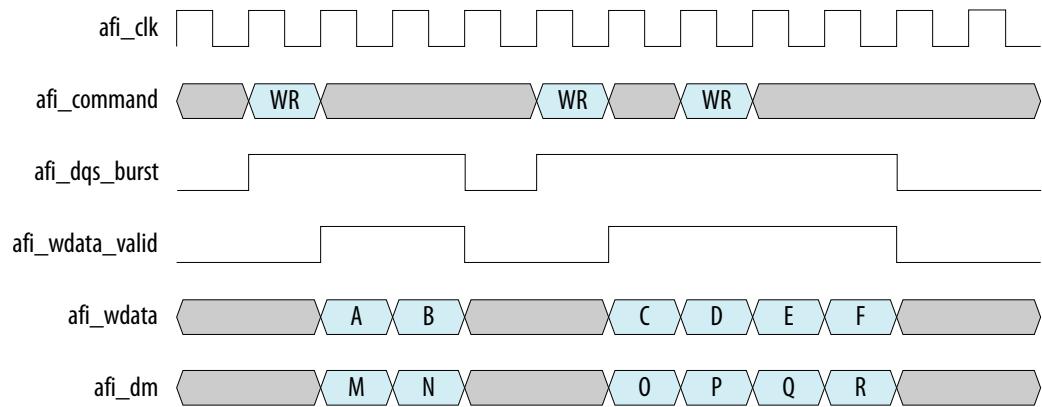
### DQS burst

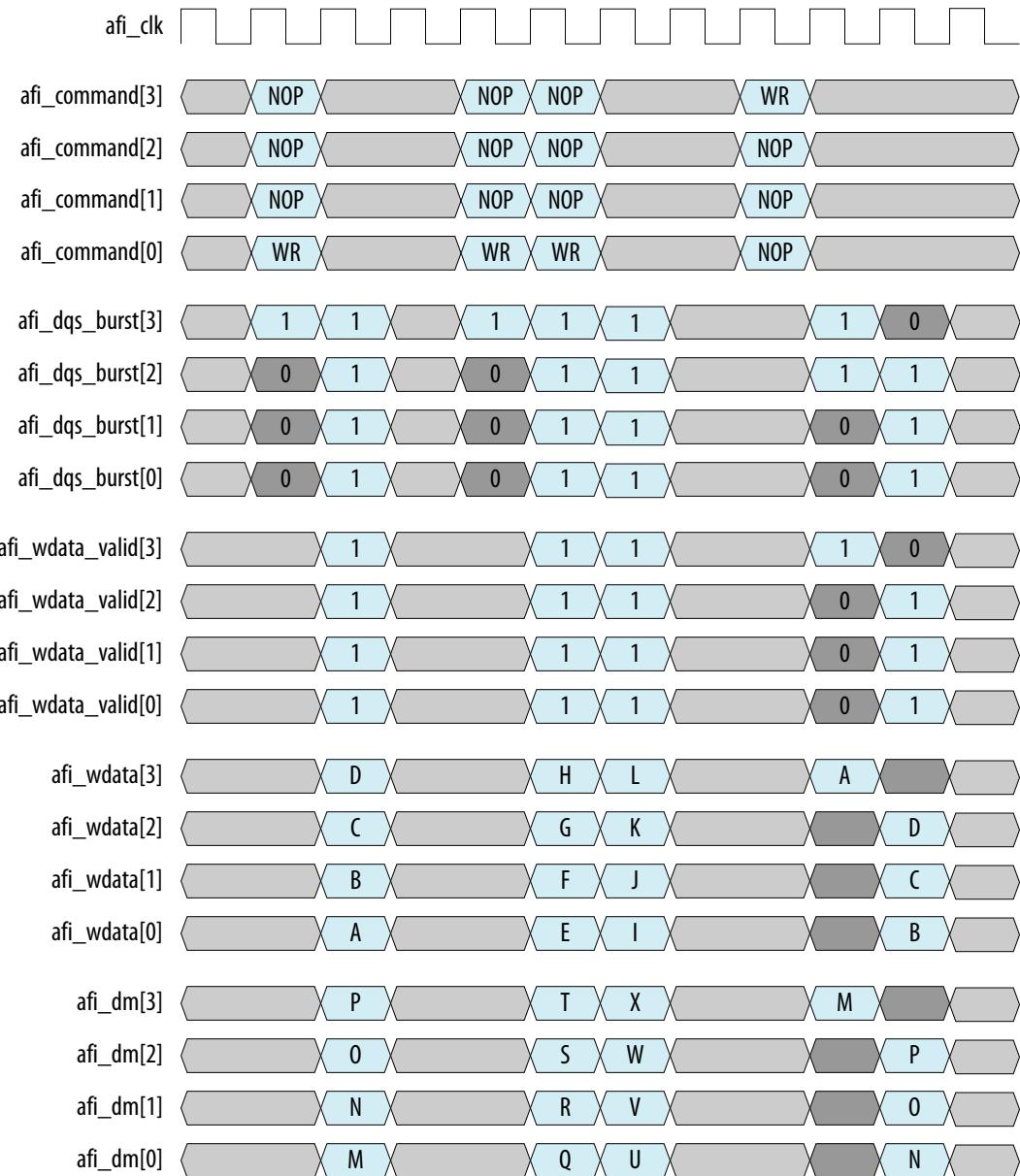
The **afi\_dqs\_burst** signal must be asserted one or two complete memory clock cycles earlier to generate DQS preamble. DQS preamble is equal to one-half and one-quarter AFI clock cycles in half rate.

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 and quarter-rate configurations.

**Figure 100. AFI DQS Burst Full-Rate, wlat=1**



**Figure 102. 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.

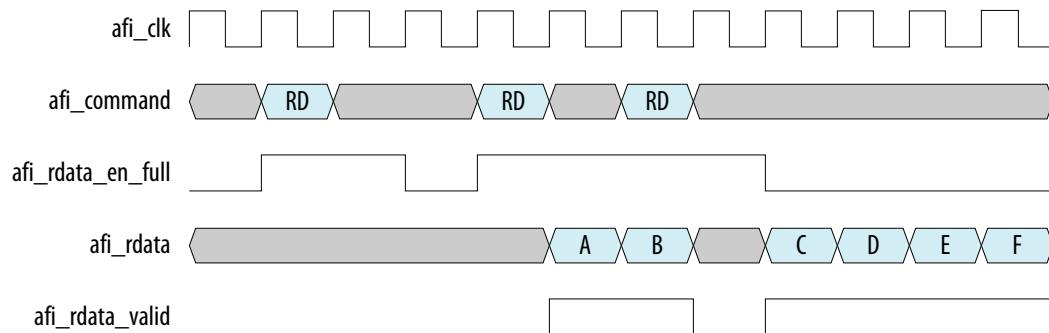
### **4.3.3. AFI Read Sequence Timing Diagrams**

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

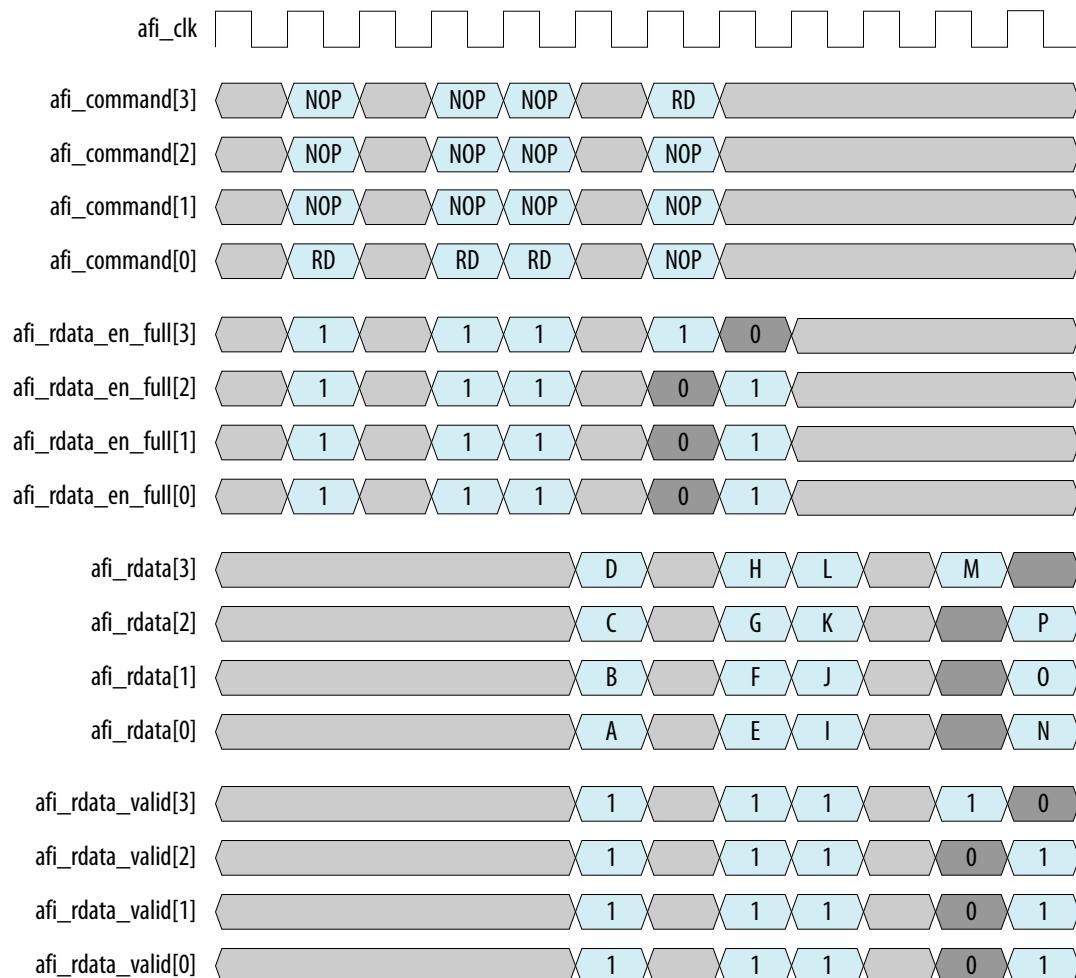
The `afi_rdata_en_full` signal must be asserted for the entire read burst operation. The `afi_rdata_en` signal need only be asserted for the intended read data.

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 105. AFI Read Data Full-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 107. AFI Read Data Quarter-Rate**


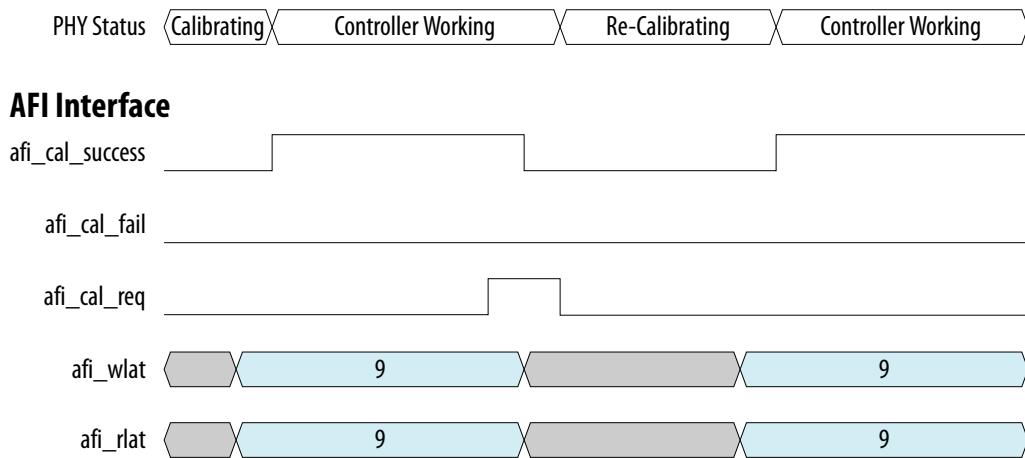
#### 4.3.4. AFI Calibration Status Timing Diagram

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 108. Calibration**



#### 4.4. Intel Agilex 7 F-Series and I-Series EMIF IP 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.

You should access the MMR interface only after calibration success is asserted.

##### Register Summary

Register	Address 32-bit Bus	Bits Register Link
ctrlcfg0	10	32
ctrlcfg1	11	32
dramtiming0	20	32
sbcfg1	24	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

*continued...*

Register	Address 32-bit Bus	Bits Register Link
sideband6	49	32
sideband7	50	32
sideband9	52	32
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.

#### Related Information

[Register Map IP-XACT Support for Intel Agilex 7 F-Series and I-Series EMIF DDR4 IP on page 148](#)

#### 4.4.1. ctrlcfg0

##### address=10(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_mem_type	3	0	Specifies memory type. Program this field with "0001" for DDR4 SDRAM.	Read
cfg_dimm_type	6	4	Specifies dimm type.	Read
cfg_ac_pos	8	7	Specifies 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. The value of this field yields different mappings between the AXI or Avalon-MM address and the SDRAM address. Program this field with the following binary values to select the ordering: "00" - chip, row, bank(BG, BA), column; "01" - chip, bank(BG, BA), row, column; "10" - row, chip, bank(BG, BA), column.	Read
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 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
Reserved	25	19	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

*continued...*

<b>Field</b>	<b>Bit High</b>	<b>Bit Low</b>	<b>Description</b>	<b>Access</b>
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)**

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

#### 4.4.4. sbcfg1

**address=24(32 bit)**

**Table 63.**

<b>Field</b>	<b>Bit High</b>	<b>Bit Low</b>	<b>Description</b>	<b>Access</b>
Reserved	0	0	Reserved.	Read
cfg_t_param_arf_to_valid	10	1	Auto Refresh to valid DRAM command window.	Read
Reserved	31	11	Reserved.	Read

#### 4.4.5. caltiming0

**address=31(32 bit)**

<b>Field</b>	<b>Bit High</b>	<b>Bit Low</b>	<b>Description</b>	<b>Access</b>
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

#### 4.4.6. 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 bank groups.	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

#### 4.4.7. 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

#### 4.4.8. 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

#### 4.4.9. 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_starve_limit	25	18	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
cfg_t_param_pdn_to_valid	31	26	Power down to valid bank command window.	Read

#### 4.4.10. 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

#### 4.4.11. 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

#### 4.4.12. sideband0

**address=43(32 bit)**

Field	Bit High	Bit Low	Description	Access
mr_cmd_trigger	0	0	When asserted, triggers the execution of the mode register command.	Read/Write

#### 4.4.13. 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

#### 4.4.14. 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

#### 4.4.15. 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

#### 4.4.16. 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

**4.4.17. 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

**4.4.18. 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

**4.4.19. 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 (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	

*continued...*

Field	Bit High	Bit Low	Description	Access
			0011 - Chip select 0 and chip select 1	
			1111 - all chip selects	
			Mode Register Set - Any combination of chip selects.	

#### 4.4.20. 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.  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] A17 [16] RAS# [15] CAS# [14] WE# [13:0] A13:A0  MRS: [19:18] is BA1:BA0, [13:0] is Opcode[13:0]	Read/Write

#### 4.4.21. 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 logical rank chip ID for 3DS Refresh.	Read

**4.4.22. 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

**4.4.23. niosreserve0**
**address=68(32 bit)**

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

**4.4.24. niosreserve1**
**address=69(32 bit)**

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

**4.4.25. 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.	

*continued...*

Field	Bit High	Bit Low	Description	Access
			[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.	

#### 4.4.26. 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 value of 1 to this self-clearing bit clears the interrupt signal, error status, and address.	Read/Write
Reserved	31	8	Reserved.	Read

#### 4.4.27. 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 overflows.	Read
sts_dbe_count	11	8	Indicates the number of times DBE error has occurred. The counter overflows.	Read
sts_corr_dropped_count	15	12	Indicates the number of times correction command has dropped. The counter overflows.	Read
Reserved	31	16	Reserved.	Read

**4.4.28. 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

**4.4.29. 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 reboot 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 Agilex 7 FPGAs have only 16 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.

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

**address=147(32 bit)**

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

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

**address=148(32 bit)**

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

## 5. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – 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:
  - Siemens EDA\* ModelSim
  - Siemens EDA QuestaSim\*
  - 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 Agilex 7 F-Series and I-Series 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.

### 5.1. Simulation Options

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

- Skip calibration—Loads memory configuration settings and enters user mode, providing the fastest simulation time.

Simulation represents accurate controller efficiency and does 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 *External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Design Example User Guide*.

### Abstract I/O SSM

To improve simulation time, the Intel Agilex 7 F-Series and I-Series EMIF IP for simulation implements an abstract I/O subsystem manager (Abstract I/O SSM). The Abstract I/O SSM is a behavioral implementation of the I/O SSM that executes the same calibration code while minimizing circuit complexity and processor delays. As a result of these optimizations, processor-generated events may occur at different times when compared with the true I/O SSM. The Abstract I/O SSM model is not currently supported in the QuestaSim simulator when two calibration IPs are instantiated. You can disable the Abstract I/O SSM by setting the *iossm\_use\_model* parameter to 0 in the simulation RTL or in your simulator; for the design example, the hierarchy for this parameter is:

```
ed_sim.emif_cal.emif_cal.emif_cal.arch_inst.io_ssm.iossm_use_mode  
1.
```

**Table 64. Typical Simulation Times Using Intel Agilex 7 F-Series and I-Series EMIF IP**

Calibration Mode/Run Time <sup>(1)</sup>	Estimated Simulation Time	
	Small Interface ( $\times 8$ Single Rank)	Large Interface ( $\times 72$ Quad Rank)
Skip <ul style="list-style-type: none"><li>• Skip calibration</li><li>• Preloads calculated settings</li></ul>	15 minutes	40 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.  
3. In Intel Quartus Prime version 22.1 and later, skip calibration simulations may realize significant speed improvement when the Abstract I/O SSM model is used.

### Related Information

[External Memory Interfaces Intel Agilex 7 FPGA IP Design Example User Guide](#)

## 5.2. 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.

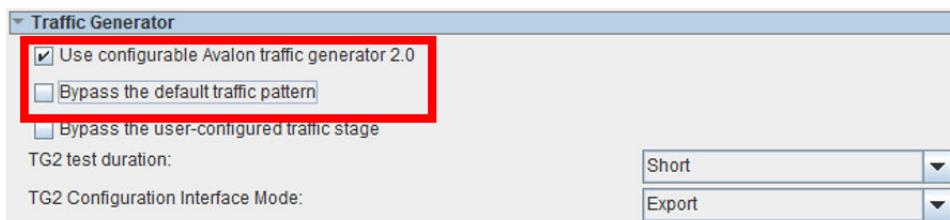
A given design may display different latency values on different boards, due to differences in board implementation.

The Intel Agilex 7 F-Series and I-Series 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 Agilex 7 F-Series and I-Series 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 Agilex 7 F-Series and I-Series EMIF IP supports functional simulation through the design example using the Traffic Generator (TG1) or the Traffic Generator 2.0 (TG2). TG2 is a configurable traffic generator. (For information on TG2, refer to [Configurable Traffic Generator \(TG2\) Description](#). Functional simulation using TG2 is allowed both in *default traffic pattern* and *user configured traffic* mode. In default traffic mode, TG2 runs a default traffic pattern after reset instead of waiting for user configuration for TG2; whereas user mode allows you to provide your own custom traffic. Do not select **Bypass the default traffic pattern** when creating a design example for functional simulation using TG2 *default traffic* mode. For more information on simulating with TG2 in user configured traffic mode, refer to [Configuration and Status Registers](#).

**Figure 109. Selecting Use configurable Avalon Traffic generator 2.0**



To perform functional simulation for an Intel Agilex 7 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 Agilex 7 F-Series and I-Series External Memory Interfaces IP Design Example User Guide*.

### 5.2.1. 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 QuestaSim - Intel FPGA Edition simulator, which is supplied with the Intel Quartus Prime software. Submodules normally provided as clear text SystemVerilog files are encrypted using IEEE Verilog HDL encryption for QuestaSim - Intel FPGA Edition.

## 5.2.2. Simulation Scripts

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

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

## 5.2.3. Functional Simulation with Verilog HDL

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

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`

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Questa - Intel FPGA Edition, ModelSim, and QuestaSim Simulator Support* chapter in the [Intel Quartus Prime Pro Edition User Guide, Third-party Simulation](#).

## 5.2.4. 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.

Only the top-level IP instance file is guaranteed to be written in VHDL; submodules can be deployed as Verilog/SystemVerilog (encrypted or plain text) files, or VHDL files. Note that the QuestaSim - Intel FPGA Edition is not restricted to a single HDL language, however, some files may 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 Cadence, Siemens EDA, and Synopsys simulators are provided for you to run the design example. Simulation scripts in the simulation folders are located as follows:

- sim\ed\_sim\mentor\msim\_setup.tcl
- sim\ed\_sim\synopsys\vcs\vcsmx\vcsmx\_setup.sh
- sim\ed\_sim\synopsys\vcs\vcs\_setup.sh
- sim\ed\_sim\xcelium\xcelium\_setup.sh

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Questa - Intel FPGA Edition, ModelSim, and QuestaSim Simulator Support* chapter in the [Intel Quartus Prime Pro Edition User Guide, Third-party Simulation](#).

## 5.2.5. Simulating the Design Example

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

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 Agilex 7 F-Series and I-Series 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.

For more information about simulation, refer to the *Intel Quartus Prime Pro Edition User Guide, Third-party Simulation*.

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.

## 5.3. Simulating the Design Example with Mentor Graphics\* AXI4 Master BFM (Intel FPGA Edition)

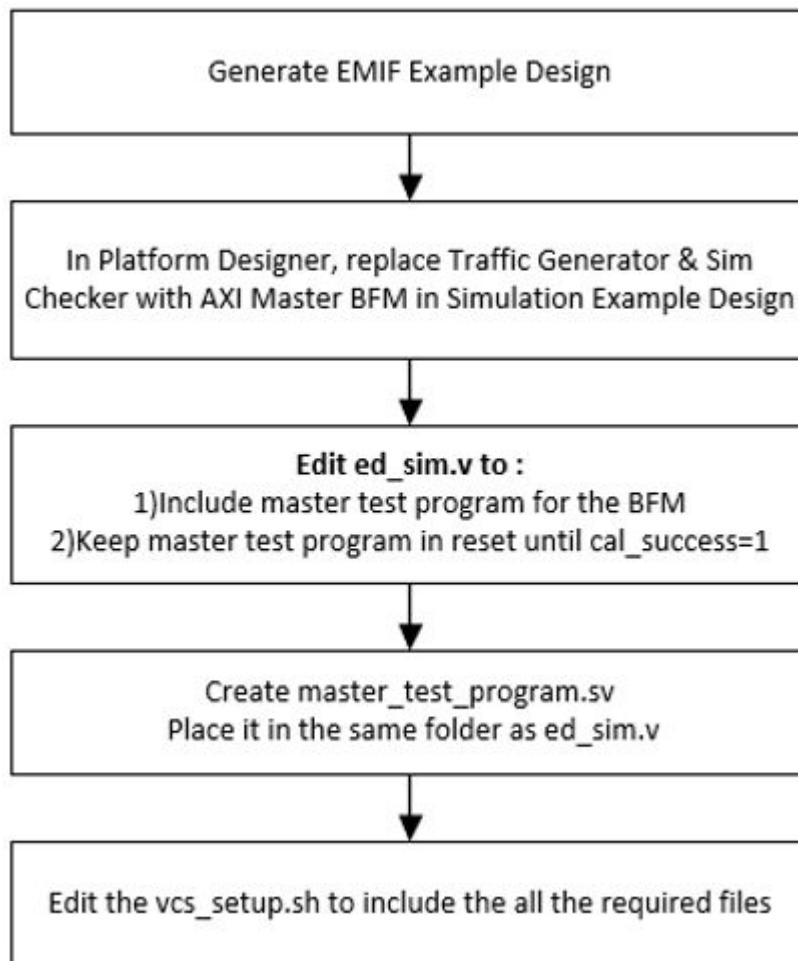
The traffic generator in the EMIF design example sends traffic using the Avalon memory-mapped interface.

This section describes how to modify the EMIF design example for a 64-bit DDR4 interface (with data mask enabled) to perform functional simulation using the Mentor Graphics\* AXI4 Master Bus Functional Model (BFM) (Intel FPGA Edition) to write data and read data from the EMIF IP.

### 5.3.1. Overview of Steps

The following figure shows the overall steps and changes required in the simulation design example, to perform functional simulation with the Mentor Graphics AXI4 Master BFM (Intel FPGA Edition).

**Figure 110. Steps to Simulate Design Example with the Mentor Graphics AXI4 Master BFM (Intel FPGA Edition)**



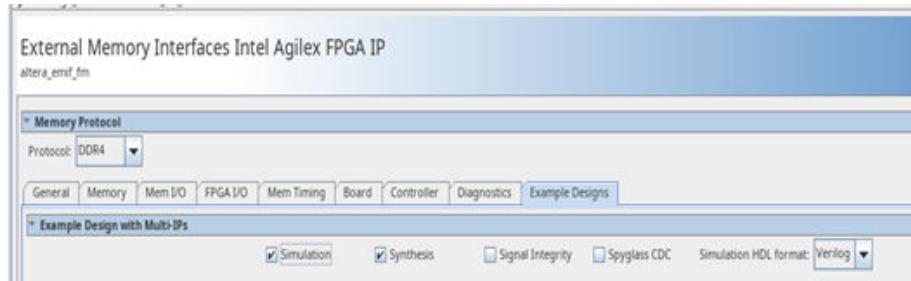
If you are creating the design with other DQ width, the DQ width of the EMIF IP must be one of 8,16,32 or 64. If this requirement is not met, the Platform Designer issues an error similar to the following, when you connect the AXI4 master BFM to the EMIF IP:

```
ed_sim.emif_fm_0.ctrl_amm_0          Date width must be of power of two and  
between 8 and 4096.
```

### 5.3.2. Generating the EMIF Design Example

1. Generate an EMIF design example with with the **Simulation** and **Synthesis** options selected.

**Figure 111. Generating a Design Example for Simulation and Synthesis**



The following discussion assumes that the design example targets a DDR4 RDIMM x64 interface with data mask enabled, implemented on the Intel Agilex 7 F-Series FPGA Development Kit.

2. With the .qpf file in the synthesizable design example, you can launch the project and open the ed\_sim.qsys file to edit the simulation design example with the Platform Designer.
3. After generating the simulation design example, delete the ed\_sim folder in the simulation design example. You regenerate the ed\_sim folder after incorporating the AXI4 Master BFM to the simulation design example.

### 5.3.3. Editing the Simulation Design Example with the Platform Designer

1. Open the synthesizable design example by clicking **Open Project**, and then selecting ed\_synth.qpf in the <example\_project>/qii folder:

**Figure 112. Opening ed\_synth.qpf**



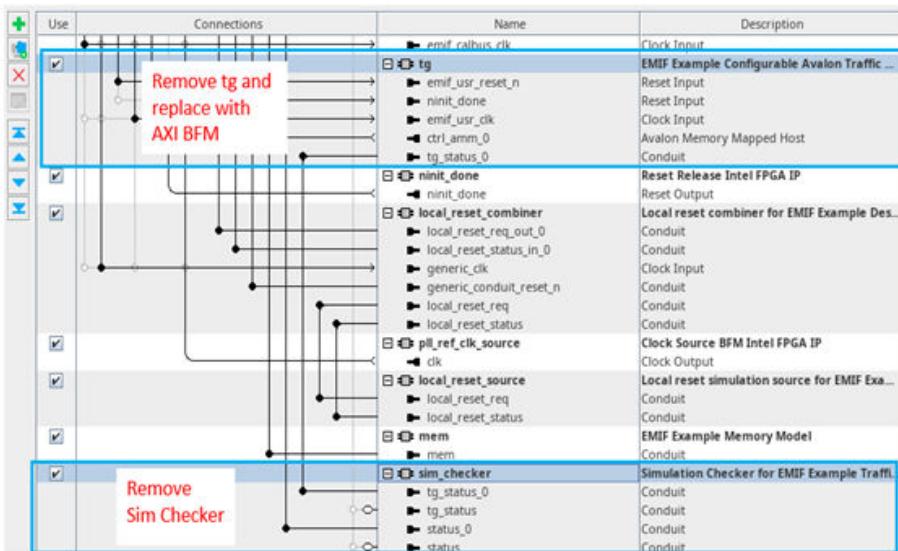
2. After the project has loaded, open ed\_sim.qsys in the <example\_design\_path>/sim folder:

**Figure 113.** Opening ed\_sim.qsys



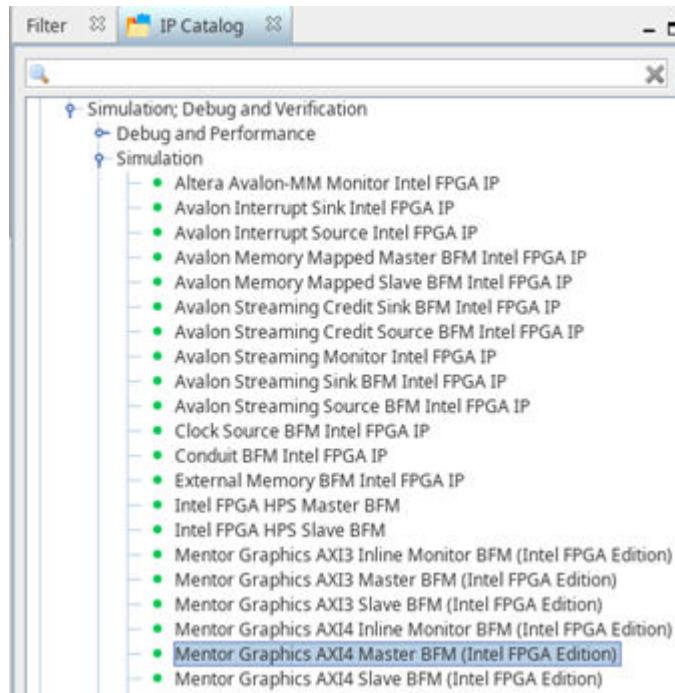
- In the Platform Designer, remove the Sim Checker and Traffic Generator:

**Figure 114.** Making Changes in the Simulation Example Design



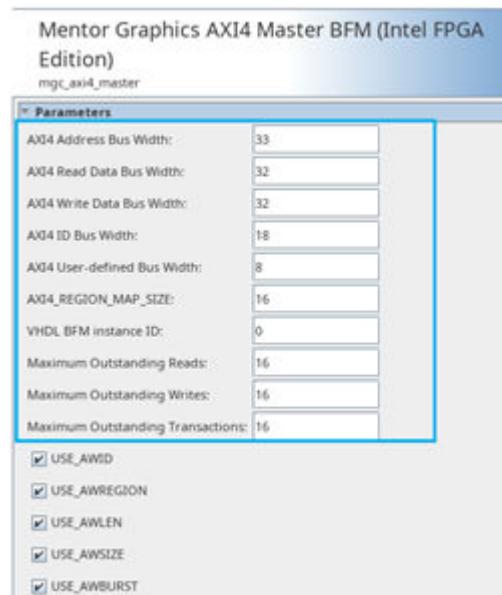
- Insert a Mentor Graphics AXI4 Master BFM (Intel FPGA Edition) to the simulation design example:
  - In the IP Catalog, click **Library > Basic Functions > Simulation; Debug and Verification > Mentor Graphic AXI4 Master BFM (Intel FPGA Edition)**.

**Figure 115. Inserting AXI4 master BFM**



- b. Parameterize the AXI4 Master BFM using the values shown in the figure below; retain the other default settings.

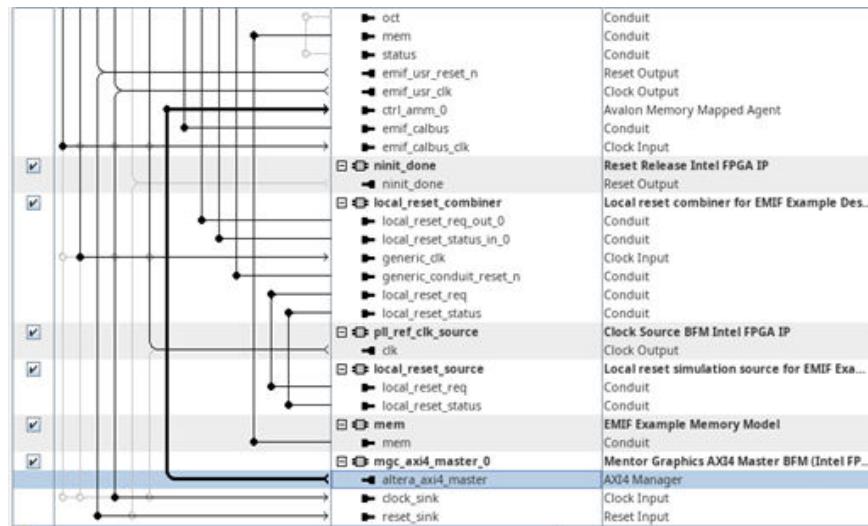
**Figure 116. AXI Master BFM Settings**



5. Make the following connections on the AXI4 master BFM as illustrated in the following figure:

- a. mgc\_axi\_master\_0.altera\_axi\_master to emif\_fm\_0.ctrl\_amm\_0
- b. mgc\_axi\_master\_0.clock\_sink to emif\_fm\_0.emif\_usr\_clk
- c. mgc\_axi\_master\_0.reset\_sink to emif\_fm\_0.emif\_usr\_reset\_n (This connection is changed manually in ed\_sim.v in a later stage, to keep the AXI4 Master BFM in reset until cal\_success=1'b1 ).

**Figure 117. Connectivity between EMIF IP and AXI4 Master BFM**

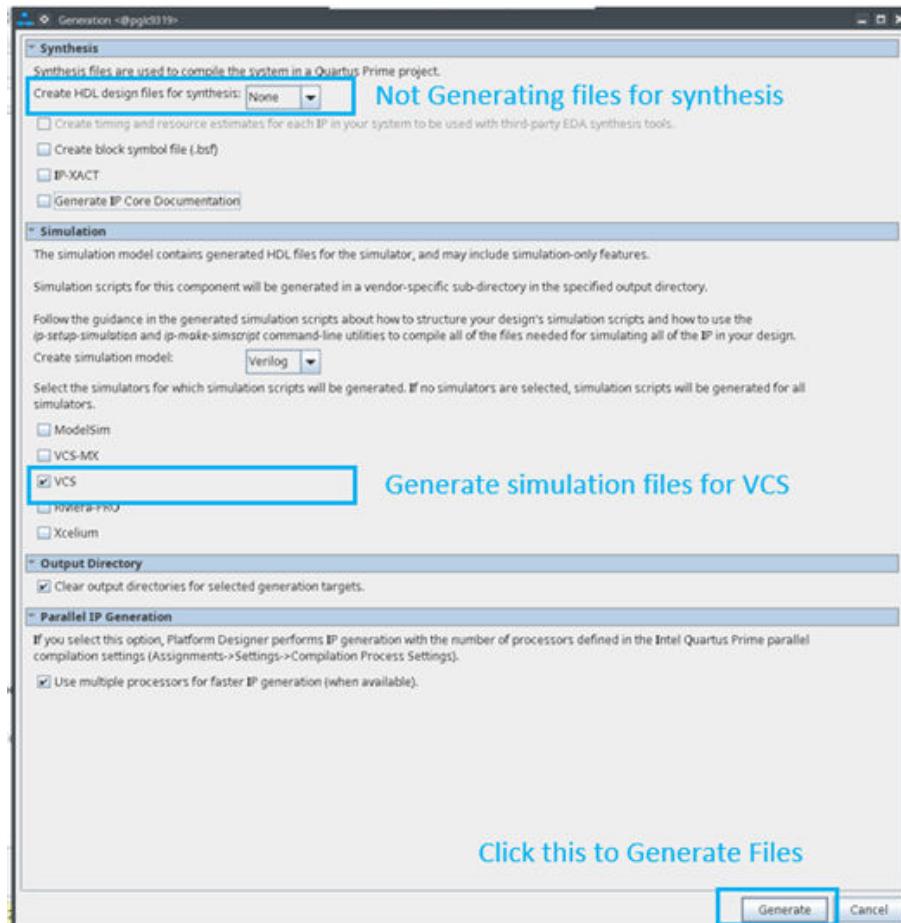


You can ignore the following warning message after connecting the AXI Master BFM:

```
ed_sim.emif_fm_0      emif_fm_0.oct must be exported or connected to a
matching conduit as it has unconnected inputs.
```

6. Save your edits and click **Generate HDL**.
7. In the generation window:
  - a. Select **None** for **Create HDL design files for synthesis**, because we are not generating a synthesizable design example.
  - b. Select **VCS** to generate the simulation scripts for VCS simulator.
  - c. Click **Generate** to generate the files.

Figure 118. Settings for Generating Files



#### 5.3.4. Editing the ed\_sim.v File

1. In a text editor, open the `ed_sim.v` file, located in the `<example_design_path>/sim/ed_sim/sim` folder.
2. Add the following lines to the `ed_sim.v` file, to include the master test program instance:

```
//Edit 1 : To connect local_cal_sucess to ARESETn of AXI4 Master BFM
wire local_cal_success;

//Edit 2 : To include master test program instance
master_test_program #(
    .AXI4_ADDRESS_WIDTH (33),
    .AXI4_RDATA_WIDTH (32),
    .AXI4_WDATA_WIDTH (32),
    .AXI4_ID_WIDTH (18),
    .AXI4_USER_WIDTH (8),
    .AXI4_REGION_MAP_SIZE (16))
u_master (mhc_axi4_master_0.mhc_axi4_master_0);
```

3. Connect the local\_cal\_success of emif\_fm\_0 to ARESETn of the AXI4 Master BFM. Doing this ensures that the BFM does not begin to issue transactions until after the EMIF calibration is successful.

**Figure 119. Edit at EMIF IP Connection**

```
ed_sim_emif_fm_0 emif_fm_0 (
    .local_reset_req      (local_reset_combiner_local_reset_req_out_0_local_reset_req),
    .local_reset_done     (emif_fm_0_local_reset_status_local_reset_done),
    .pll_ref_clk          (pll_ref_clk_source_clk_clk),
    .pll_ref_clk_out      (emif_fm_0_pll_ref_clk_out_clk),
    .pll_locked           (emif_fm_0_pll_locked_pll_locked),
    .oct_rzqin            (),
    .mem_ck                (emif_fm_0_mem_mem_ck),
    .mem_ck_n              (emif_fm_0_mem_mem_ck_n),
    .mem_a                 (emif_fm_0_mem_mem_a),
    .mem_act_n             (emif_fm_0_mem_mem_act_n),
    .mem_ba                (emif_fm_0_mem_mem_ba),
    .mem_bg                (emif_fm_0_mem_mem_bg),
    .mem_cke               (emif_fm_0_mem_mem_cke),
    .mem_cs_n              (emif_fm_0_mem_mem_cs_n),
    .mem_odt               (emif_fm_0_mem_mem_odt),
    .mem_reset_n            (emif_fm_0_mem_mem_reset_n),
    .mem_par               (emif_fm_0_mem_mem_par),
    .mem_alert_n            (mem_mem_mem_alert_n),
    .mem_dqs               (emif_fm_0_mem_mem_dqs),
    .mem_dqs_n              (emif_fm_0_mem_mem_dqs_n),
    .mem_dq                (emif_fm_0_mem_mem_dq),
    .mem_dbi_n              (emif_fm_0_mem_mem_dbi_n),
    .local_cal_success      (local_cal_success),
    .local_cal_fail         ()
);
```

**Figure 120. Edit at AXI4 Master BFM Connection**

```
215   .RDATA   (mgc_axi4_master_0_altera_axi4_master_rdata),
216   .RID     (mgc_axi4_master_0_altera_axi4_master_rid),
217   .WDATA   (mgc_axi4_master_0_altera_axi4_master_wdata),
218   .WSTRB   (mgc_axi4_master_0_altera_axi4_master_wstrb),
219   .BID     (mgc_axi4_master_0_altera_axi4_master_bid),
220   .ACLK    (emif_s10_0_emif_usr_clk_clk),
221   .ARESETn (local_cal_success)
222 );
```

### 5.3.5. Obtaining the master\_test\_program.sv File

You require the master\_test\_program.sv to initiate write/read transactions and to verify that the read data matches the write data.

As a start, you can re-use the master\_test\_program available in the qsys-example directory. Refer to the [Mentor® Verification IP Altera® Edition AMBA AXI3™/AXI4™ User Guide](#) for information on writing your own master\_test\_program.

The following steps outline how to copy the example master\_test\_program.sv from the qsys-example directory into your simulation directory on a UNIX platform:

1. Change to the directory where the ed\_sim.v is located:

```
cd <example_design_path>/sim/ed_sim/sim
```

2. Copy the master\_test\_program.sv to the same directory:

```
cp $QUARTUS_ROOTDIR/..../ip/altera/mentor_vip_ae/axi4/qsys-examples/ex1_back_to_back_sv/master_test_program.sv
```

## 5.3.6. Running the Simulation

### Modifying the vcs\_setup.sh File

In a text editor, modify the vcs\_setup.sh file to include the IP/files required for simulation. Referring to the following sample, add the lines shown in bold italics, to the vcs\_setup.sh file generated by the Intel Quartus Prime software:

```
# QIP stuff
MENTOR_VIP_AE=${QUARTUS_INSTALL_DIR}/../ip/altera/mentor_vip_ae
export QUESTA_MVC_GCC_LIB=${MENTOR_VIP_AE}/common/questa_mvc_core/\
linux_x86_64_gcc-6.2.0_vcs
export LDFLAGS="-L ${QUESTA_MVC_GCC_LIB} -Wl,-rpath ${QUESTA_MVC_GCC_LIB}\
-laxi4_IN_SystemVerilog_VCS_full_DVC"

vcs -lca -timescale=1ps/1ps -sverilog +verilog2001ext+.v $ELAB_OPTIONS
$USER_DEFINED_ELAB_OPTIONS \
+systemverilogext+.sv +vpi -debug_access+r+w+nomemcbk +vcs+lic+wait \
-cpp /usr/intel/pkg/gcc/6.2.0/bin/g++ \
-l vcs.log -kdb \
-v $QUARTUS_INSTALL_DIR/eda/sim_lib/altera_primitives.v \
-v $QUARTUS_INSTALL_DIR/eda/sim_lib/220model.v \
-v $QUARTUS_INSTALL_DIR/eda/sim_lib/sgate.v \
-v $QUARTUS_INSTALL_DIR/eda/sim_lib/altera_mf.v \
$MENTOR_VIP_AE/common/questa_mvc_svapi.svh \
$MENTOR_VIP_AE/axi4/bfm/mgc_common_axi4.sv \
$MENTOR_VIP_AE/axi4/bfm/mgc_axi4_monitor.sv \
$MENTOR_VIP_AE/axi4/bfm/mgc_axi4_inline_monitor.sv \
$MENTOR_VIP_AE/axi4/bfm/mgc_axi4_slave.sv \
$MENTOR_VIP_AE/axi4/bfm/mgc_axi4_master.sv \
../../../../master_test_program.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/altera_lnsim.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/fourteennm_atoms.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/synopsys/fourteennm_atoms_ncrypt.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/ctl_hssi_atoms.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/synopsys/ctl_hssi_atoms_ncrypt.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/ctl_hip_atoms.sv \
$QUARTUS_INSTALL_DIR/eda/sim_lib/synopsys/ctp_hssi_atoms_ncrypt.sv \
$common_design_files \
$design_files \
$USER_DEFINED_ELAB_OPTIONS_APPEND \
-top $TOP_LEVEL_NAME
```

### Running the Simulation

To run the simulation, follow these steps:

1. Change directory to the /vcs folder:

```
cd <example_design_path>/sim/ed_sim/sim/synopsys/vcs
```

2. Type the following command to run the simulation:

```
source ./vcs_setup.sh USER_DEFINED_ELAB_OPTIONS="+vcs+vcdpluson"  
USER_DEFINED_SIM_OPTIONS="" | tee out.txt
```

### Example Simulation Result

Upon successful completion, you should see simulation output similar to the following, at the terminal:

```
[300427058] [DWR=000]: Reading data ace6ace7ace4ace5 @ 10000001  
(CGBRC=0/1/0/0/0 ) burst 1  
[300427478] [DWR=000]: Reading data aceaaacebace8ace9 @ 10000002  
(CGBRC=0/1/0/0/0 ) burst 2  
[300427898] [DWR=000]: Reading data 0000000000000000 @ 10000003  
(CGBRC=0/1/0/0/0 ) burst 3  
[300428318] [DWR=000]: Reading data 0000000000000000 @ 10000004  
(CGBRC=0/1/0/0/0 ) burst 4  
[300428738] [DWR=000]: Reading data 0000000000000000 @ 10000005  
(CGBRC=0/1/0/0/0 ) burst 5  
[300429158] [DWR=000]: Reading data 0000000000000000 @ 10000006  
(CGBRC=0/1/0/0/0 ) burst 6  
[300429578] [DWR=000]: Reading data 0000000000000000 @ 10000007  
(CGBRC=0/1/0/0/0 ) burst 7  
@ 300498240, master_test_program: Read correct data (hACE0ACE1) at  
address (64)  
@ 300498240, master_test_program: Read correct data (hACE2ACE3) at  
address (68)  
@ 300498240, master_test_program: Read correct data (hACE4ACE5) at  
address (72)  
$finish called from file "./.../master_test_program.sv", line 375.  
$finish at simulation time 300508240  
V C S S i m u l a t i o n R e p o r t  
Time: 300508240 ps  
CPU Time: 400.970 seconds; Data structure size: 55.3 Mb
```

## 6. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – DDR4 Support

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Agilex 7 F-Series and I-Series FPGA external memory interface IP for DDR4.

### 6.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP 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.

**Note:** Also in this section are the parameters of the *External Memory Interfaces Intel Calibration IP*, which are included as part of the *Diagnostics* topic.

#### 6.1.1. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: General

**Table 66. Group: General / Interface**

Display Name	Description
<b>Configuration</b>	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: <code>PHY_DDR4_CONFIG_ENUM</code> )
<b>Use clamshell layout</b>	Specifies the use of a clamshell topology. When clamshell topology is used, the bottom memory chip should be wired with the address pins mirrored, in accordance with the JEDEC specification JESD21-C. <i>Each rank requires two CS pins</i> , such that the top and bottom memory chips can be configured separately. <b>For single-rank components:</b> For the top (non-mirrored) component, <code>FPGA_CS0</code> goes to <code>MEM_TOP_CS0</code> For the bottom (mirrored) component, <code>FPGA_CS1</code> goes to <code>MEM_BOT_CS0</code> <b>For dual-rank components:</b> For the top (non-mirrored) components, <code>FPGA_CS0</code> goes to <code>MEM_TOP_CS0</code> and <code>FPGA_CS1</code> goes to <code>MEM_TOP_CS1</code> For the bottom (mirrored) components, <code>FPGA_CS2</code> goes to <code>MEM_BOT_CS0</code> and <code>FPGA_CS3</code> goes to <code>MEM_BOT_CS1</code> (Identifier: <code>PHY_DDR4_USER_CLAMSHELL_EN</code> )

**Table 67. Group: General / Clocks**

Display Name	Description
<b>Memory clock frequency</b>	Specifies the <b>operating frequency</b> of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the <b>Memory</b> tab and the memory timing parameters on the <b>Mem Timing</b> tab. (Identifier: PHY_DDR4_MEM_CLK_FREQ_MHZ)
<b>Use recommended PLL reference clock frequency</b>	Specifies that the PLL reference clock frequency is automatically calculated for best performance. <i>If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.</i> (Identifier: PHY_DDR4_DEFAULT_REF_CLK_FREQ)
<b>PLL reference clock frequency</b>	This parameter tells the IP what PLL reference clock frequency you specify. You must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency or the clock rate of user logic changes. You should use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if you do not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_DDR4_USER_REF_CLK_FREQ_MHZ)
<b>PLL reference clock jitter</b>	Specifies the <b>peak-to-peak phase jitter</b> on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 20ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_DDR4_REF_CLK_JITTER_PS)
<b>Clock rate of user logic</b>	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)
<b>Specify additional core clocks based on existing PLL</b>	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter <b>provides an alternative clock-generation mechanism for when your design exhausts available PLL resources</b> . 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). <i>You must follow proper clock-domain-crossing techniques when transferring data between clock domains.</i> (Identifier: PLL_ADD_EXTRA_CLKS)

**Table 68. Group: General / Mimic Hard Processor System (HPS) EMIF**

Display Name	Description
<b>Mimic HPS EMIF</b>	This option generates an EMIF at the same tiles as HPS EMIF following the same rules as HPS EMIF. Use this option to generate a fabric EMIF that mimics HPS-EMIF restrictions. (Identifier: PHY_DDR4_MIMIC_HPS_EMIF)

**Table 69. Group: General / Clocks / Additional Core Clocks**

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

**Table 70. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_0**

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

**Table 71. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_1**

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

**Table 72. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_2**

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

**Table 73. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_3**

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

## 6.1.2. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Memory

**Table 74. Group: Memory / Topology**

Display Name	Description
<b>Memory format</b>	Specifies the format of the external memory device. The following formats are supported: <b>Component</b> - a Discrete memory device; <b>UDIMM</b> - Unregistered/Unbuffered DIMM where address/control, clock, and data are unbuffered; <b>RDIMM</b> - Registered DIMM where address/control and clock are buffered; <b>LRDIMM</b> - Load Reduction DIMM where address/control, clock, and data are buffered. <b>LRDIMM</b> reduces the load to increase memory speed and supports higher densities than RDIMM; <b>SODIMM</b> - 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)

*continued...*

<b>Display Name</b>	<b>Description</b>
	<p><b>Note:</b> 1. Intel Agilex 7 devices do not support custom DIMMs,      2. Intel does not provide support for customer-built custom DIMMs.      3. Intel does not provide support for memory down where components are laid out with a registered clock driver (RCD) chip for clock/address/command buffering or data buffering.      4. Intel continues to provide support for off-the-shelf JEDEC-compliant DIMMs, and memory down without RCD or data buffering, as well as clamshell topology, supported subject to Intel board design guidelines.</p>
<b>DQ width</b>	Specifies the total number of data pins in the interface. (Identifier: MEM_DDR4_DQ_WIDTH)
<b>DQ pins per DQS group</b>	Specifies the total number of DQ pins per DQS group. (Identifier: MEM_DDR4_DQ_PER_DQS)
<b>Number of DQS groups</b>	Specifies the number of DQS groups in the interface. This value is automatically calculated as the DQ width divided by the number of DQ pins per DQ group.
<b>Number of clocks</b>	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)
<b>Number of DIMMs</b>	Total number of DIMMs. (Identifier: MEM_DDR4_NUM_OF_DIMMS)
<b>Number of physical ranks per DIMM</b>	Number of ranks per DIMM. For LRDIMM, this represents the number of physical ranks on the DIMM behind the memory buffer (Identifier: MEM_DDR4_RANKS_PER_DIMM)
<b>Number of chip selects per DIMM</b>	Specifies the number of chip selects per DIMM.
<b>Number of Chip Select</b>	Specifies the number of chip select.
<b>Chip ID width</b>	Specifies the number of chip ID pins. Only applicable to <i>registered and load-reduced DIMMs</i> that use 3DS/TSV memory devices. (Identifier: MEM_DDR4_CHIP_ID_WIDTH)
<b>Row address width</b>	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)
<b>Column address width</b>	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)
<b>Bank address width</b>	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)
<b>Bank group width</b>	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)
<b>Data mask</b>	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). <b>One DM pin exists per DQS group.</b> (Identifier: MEM_DDR4_DM_EN)

*continued...*

Display Name	Description
<b>Write DBI</b>	Indicates whether the interface uses write data bus inversion (DBI). This feature provides <b>better signal integrity and write margin</b> . This feature is unavailable if Data Mask is enabled or in x4 mode. (Identifier: MEM_DDR4_WRITE_DBI)
<b>Read DBI</b>	Specifies whether the interface uses read data bus inversion (DBI). Enable this feature for <b>better signal integrity and read margin</b> . This feature is not available in x4 configurations. (Identifier: MEM_DDR4_READ_DBI)
<b>Enable address mirroring for odd chip-selects</b>	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)
<b>Enable address mirroring for odd ranks</b>	Enabling address mirroring for dual-rank or quad-rank DIMM. (Identifier: MEM_DDR4_MIRROR_ADDRESSING_EN)
<b>ALERT# pin placement</b>	Specifies placement for the mem_alert_n signal. If you select " <b>Automatically select a location</b> ", 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 results during compilation. For devices in the Intel Agilex 7 Family: You have the option of manually selecting either <b>Address/Command Lane 2, Pin 8</b> or <b>Address/Command Lane 3, Pin 8</b> only. For interfaces containing multiple memory devices, you should connect the ALERT# pins together to the ALERT# pin on the FPGA. (Identifier: MEM_DDR4_ALERT_N_PLACEMENT_ENUM)

**Table 75. Group: Memory / Latency and Burst**

Display Name	Description
<b>Memory CAS latency setting</b>	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)
<b>Memory write CAS latency setting</b>	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)
<b>Memory additive CAS latency setting</b>	Determines the posted CAS additive latency of the memory device. Enable this feature to <b>improve command and bus efficiency, and increase system bandwidth</b> . (Identifier: MEM_DDR4_ATCL_ENUM)

**Table 76. Group: Memory / Mode Register Settings**

<b>Addr/CMD parity latency</b>	Additional latency incurred by enabling address/command parity check after calibration. <b>Select a value</b> to enable address/command parity with the latency associated with the selected value. Select <b>Disable</b> to disable address/command parity. Address/command parity is enabled automatically during calibration regardless of the value of this setting.
<b>Fine granularity refresh</b>	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)

### 6.1.3. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Mem I/O

**Table 77. Group: Mem I/O / Memory I/O Settings**

Display Name	Description
<b>Use Default Memory I/O Settings</b>	Specifies to use the Intel default ODT settings. (Identifier: MEM_DDR4_INTEL_DEFAULT_TERM)
<b>Output drive strength setting</b>	Specifies the output driver impedance setting at the memory device. To obtain optimum signal integrity performance, <b>select option based on board simulation results.</b> (Identifier: MEM_DDR4_DRV_STR_ENUM)
<b>Dynamic ODT (Rtt_WR) value</b>	Specifies the mode of the dynamic on-die termination (ODT) during writes to the memory device (used for <b>multi-rank configurations</b> ). For optimum signal integrity performance, <b>select this option based on board simulation results.</b> (Identifier: MEM_DDR4_RTT_WR_ENUM)
<b>ODT Rtt nominal value</b>	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, <b>select your option based on board simulation results.</b> (Identifier: MEM_DDR4_RTT_NOM_ENUM)
<b>RTT PARK</b>	If set, the value is applied when the DRAM is not being written <b>AND</b> ODT is not asserted HIGH. (Identifier: MEM_DDR4_RTT_PARK)
<b>RCD CA Input Bus Termination</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_RCD_CA_IBT_ENUM)
<b>RCD DCS[3:0]_n Input Bus Termination</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_RCD_CS_IBT_ENUM)
<b>RCD DCKE Input Bus Termination</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_RCD_CKE_IBT_ENUM)
<b>RCD DODT Input Bus Termination</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_RCD_ODT_IBT_ENUM)
<b>DB Host Interface DQ RTT_NOM</b>	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)
<b>DB Host Interface DQ RTT_WR</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_DB_RTT_WR_ENUM)
<b>DB Host Interface DQ RTT_PARK</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_DB_RTT_PARK_ENUM)
<b>DB Host Interface DQ Driver</b>	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. <b>Perform board simulation to obtain the optimal value for this setting.</b> (Identifier: MEM_DDR4_DB_DQ_DRV_ENUM)

*continued...*

Display Name	Description
<b>Use recommended initial VrefDQ value</b>	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_VREFDQ)
<b>VrefDQ training value</b>	VrefDQ training value. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_VALUE)
<b>VrefDQ training range</b>	VrefDQ training range. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_RANGE)

**Table 78. Group: Mem I/O / RDIMM/LRDIMM Serial Presence Detect (SPD) Data**

Display Name	Description
<b>SPD Byte 137 - RCD Drive Strength for Command/Address</b>	Specifies the drive strength of the registering clock driver's control and command/address outputs to the DRAM. The value must come from <b>Byte 137 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_137_RCD_CA_DRV)
<b>SPD Byte 138 - RCD Drive Strength for CK</b>	Specifies the drive strength of the registering clock driver's clock outputs to the DRAM. The value must come from <b>Byte 138 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_138_RCD_CK_DRV)
<b>SPD Byte 140 - DRAM VrefDQ for Package Rank 0</b>	Specifies the VrefDQ setting for package rank 0 of an LRDIMM. The value must come from <b>Byte 140 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_140_DRAM_VREFDQ_R0)
<b>SPD Byte 141 - DRAM VrefDQ for Package Rank 1</b>	Specifies the VrefDQ setting for package rank 1 of an LRDIMM. The value must come from <b>Byte 141 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_141_DRAM_VREFDQ_R1)
<b>SPD Byte 142 - DRAM VrefDQ for Package Rank 2</b>	Specifies the VrefDQ setting for package rank 2 (if it exists) of an LRDIMM. The value must come from <b>Byte 142 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_142_DRAM_VREFDQ_R2)
<b>SPD Byte 143 - DRAM VrefDQ for Package Rank 3</b>	Specifies the VrefDQ setting for package rank 3 (if it exists) of an LRDIMM. The value must come from <b>Byte 143 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_143_DRAM_VREFDQ_R3)
<b>SPD Byte 144 - DB VrefDQ for DRAM Interface</b>	Specifies the VrefDQ setting of the data buffer's DRAM interface. The value must come from <b>Byte 144 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_144_DB_VREFDQ)
<b>SPD Byte 145-147 - DB MDQ Drive Strength and RTT</b>	Specifies the drive strength of the MDQ pins of the data buffer's DRAM interface. The value must come from <b>either Byte 145 (data rate = 1866), 146 (1866 data rate = 2400), or 147 (2400 data rate = 3200) of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_145_DB_MDQ_DRV)
<b>SPD Byte 148 - DRAM Drive Strength</b>	Specifies the drive strength of the DRAM. The value must come from <b>Byte 148 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_148_DRAM_DRV)
<b>SPD Byte 149-151 - DRAM ODT (RTT_WR and RTT_NOM)</b>	Specifies the RTT_WR and RTT_NOM setting of the DRAM. The value must come from <b>either Byte 149 (data rate = 1866), 150 (1866 data rate = 2400), or 151 (2400 data rate = 3200) of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_149_DRAM_RTT_WR_NOM)
<b>SPD Byte 152-154 - DRAM ODT (RTT_PARK)</b>	Specifies the RTT_PARK setting of the DRAM. The value must come from <b>either Byte 152 (data rate = 1866), 153 (1866 data rate = 2400), or 154 (2400 data rate = 3200) of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_152_DRAM_RTT_PARK)
<b>SPD Byte 155 - DB VrefDQ for DRAM Interface Range</b>	Specifies the RTT_PARK setting of the DRAM. The value must come from Byte 155 of the SPD from the DIMM vendor.

*continued...*

Display Name	Description
<b>RCD and DB Manufacturer (LSB)</b>	Specifies the LSB of the ID code of the registering clock driver and data buffer manufacturer. The value must come from <b>Byte 133 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_133_RCD_DB_VENDOR_LSB)
<b>RCD and DB Manufacturer (MSB)</b>	Specifies the MSB of the ID code of the registering clock driver and data buffer manufacturer. The value must come from <b>Byte 134 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_134_RCD_DB_VENDOR_MSB)
<b>RCD Revision Number</b>	Specifies the die revision of the registering clock driver. The value must come from <b>Byte 135 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_135_RCD_REV)
<b>DB Revision Number</b>	Specifies the die revision of the data buffer. The value must come from <b>Byte 139 of the SPD</b> from the DIMM vendor. (Identifier: MEM_DDR4_SPD_139_DB_REV)

**Table 79. Group: Mem I/O / ODT Activation**

Display Name	Description
<b>Use Default ODT Assertion Tables</b>	Enables the default ODT assertion pattern as determined from vendor guidelines. These settings are provided as a default only; <i>you should simulate your memory interface to determine the optimal ODT settings and assertion patterns.</i> (Identifier: MEM_DDR4_USE_DEFAULT_ODT)

#### 6.1.4. Intel Agilex 7 F-Series and I-Series 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 80. Group: FPGA I/O / FPGA I/O Settings**

Display Name	Description
<b>Voltage</b>	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)
<b>Use default I/O settings</b>	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. <i>To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results.</i> (Identifier: PHY_DDR4_DEFAULT_IO)

**Table 81. Group: FPGA I/O / FPGA I/O Settings / Address/Command**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_DDR4_USER_AC_MODE_ENUM)
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_DDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode of the address/command output pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the address and command signals.</i> (Identifier: PHY_DDR4_USER_AC_DEEMPHASIS_ENUM)

**Table 82. Group: FPGA I/O / FPGA I/O Settings / Memory Clock**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_DDR4_USER_CK_MODE_ENUM)
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_DDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode for the memory clock pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the memory clock signals.</i> (Identifier: PHY_DDR4_USER_CK_DEEMPHASIS_ENUM)

**Table 83. Group: FPGA I/O / FPGA I/O Settings / Data Bus**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_DDR4_USER_DATA_OUT_MODE_ENUM)
<i>continued...</i>	

Display Name	Description
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_DDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode for the memory bus data output pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_DDR4_USER_DATA_OUT_DEEMPHASIS_ENUM)
<b>Input mode</b>	This parameter allows you to change the input termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_DDR4_USER_DATA_IN_MODE_ENUM)
<b>Use recommended initial Vrefin</b>	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)
<b>Initial Vrefin</b>	Specifies the <b>initial value for the reference voltage on the data pins (Vrefin)</b> . 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 <b>skip Vref calibration (Diagnostics tab)</b> , this is the value that is used as the Vref for the interface. (Identifier: PHY_DDR4_USER_STARTING_VREFIN)

**Table 84. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs**

Display Name	Description
<b>PLL reference clock I/O standard</b>	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_DDR4_USER_PLL_REF_CLK_IO_STD_ENUM)
<b>RZQ I/O standard</b>	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_DDR4_USER_RZQ_IO_STD_ENUM)

### 6.1.5. Intel Agilex 7 F-Series and I-Series 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 85. Group: Mem Timing / Parameters dependent on Speed Bin**

Display Name	Description
<b>Speed bin</b>	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)
<b>tIS (base)</b>	tIS (base) refers to the <b>setup time for the Address/Command/Control (A) bus</b> to the rising edge of CK. (Identifier: MEM_DDR4_TIS_PS)

*continued...*

Display Name	Description
<b>tIS (base) AC level</b>	tIS (base) AC level refers to the <b>voltage level which the address/command signal must cross and remain above during the setup margin window</b> . 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)
<b>tIH (base)</b>	tIH (base) refers to the <b>hold time for the Address/Command (A) bus</b> after the rising edge of CK. Depending on what AC level you have chosen for a design, the hold margin can vary (this variance is determined automatically when you choose the " <b>tIH (base) AC level</b> "). (Identifier: MEM_DDR4_TIH_PS)
<b>tIH (base) DC level</b>	tIH (base) DC level refers to the <b>voltage level which the address/command signal must not cross during the hold window</b> . 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)
<b>TdiVW_total</b>	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)
<b>VdiVW_total</b>	VdiVW_total describes the <b>Rx Mask voltage</b> , 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)
<b>tDQSQ</b>	tDQSQ describes the <b>latest valid transition of the associated DQ pins for a READ</b> . 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)
<b>tQH</b>	tQH specifies the <b>output hold time for the DQ in relation to DQS, DQS#</b> . 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)
<b>tDVWP</b>	Data valid window per device per pin (Identifier: MEM_DDR4_TDWP_UI)
<b>tDQSK</b>	tDQSK describes the <b>skew between the memory clock (CK) and the input data strobes (DQS) used for reads</b> . It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSK_PS)
<b>tDQSS</b>	tDQSS describes the <b>skew between the memory clock (CK) and the output data strobes used for writes</b> . It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSS_CYC)
<b>tQSH</b>	tQSH refers to the differential High Pulse Width, which is measured as a percentage of tCK. It is the <b>time during which the DQS is high for a read</b> . (Identifier: MEM_DDR4_TQSH_CYC)
<b>tDSH</b>	tDSH specifies the <b>write DQS hold time</b> . 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)
<b>tDSS</b>	tDSS describes the <b>time between the falling edge of DQS to the rising edge of the next CK transition</b> . (Identifier: MEM_DDR4_TDSS_CYC)
<b>tWLS</b>	tWLS describes the <b>write leveling setup time</b> . It is measured from the rising edge of CK to the rising edge of DQS. (Identifier: MEM_DDR4_TWLS_CYC)
<b>tWLH</b>	tWLH describes the <b>write leveling hold time</b> . It is measured from the rising edge of DQS to the rising edge of CK. (Identifier: MEM_DDR4_TWHL_CYC)

*continued...*

Display Name	Description
<b>tINIT</b>	tINIT describes the <b>time duration of the memory initialization after a device power-up</b> . After RESET_n is de-asserted, wait for another 500us until CKE becomes active. During this time, the DRAM begins internal initialization; this occurs independently of external clocks. (Identifier: MEM_DDR4_TINIT_NS)
<b>tMRD</b>	The mode register set command cycle time, tMRD is the <b>minimum time period required between two MRS commands</b> . (Identifier: MEM_DDR4_TMRD_CK_CYC)
<b>tRAS</b>	tRAS describes the <b>activate to precharge duration</b> . 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)
<b>tRCD</b>	tRCD, <b>row command delay</b> , describes the <b>active to read/write time</b> . 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)
<b>tRP</b>	tRP refers to the <b>Precharge (PRE) command period</b> . 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)
<b>tWR</b>	tWR refers to the <b>Write Recovery time</b> . 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 86. Group: Mem Timing / Parameters dependent on Speed Bin, Operating Frequency, and Page Size**

Display Name	Description
<b>tRRD_S</b>	tRRD_S refers to the <b>Activate to Activate Command Period (short)</b> . It is the minimum time interval between two activate commands to the <b>different bank groups</b> . 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)
<b>tRRD_L</b>	tRRD_L refers to the <b>Activate to Activate Command Period (long)</b> . It is the minimum time interval (measured in memory clock cycles) between two activate commands to the <b>same bank group</b> . 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)
<b>tRRD_dlr</b>	tRRD_dlr refers to the <b>Activate to Activate Command Period to Different Logical Ranks</b> . 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)
<b>tFAW</b>	tFAW refers to the <b>four activate window time</b> . 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)
<b>tCCD_S</b>	tCCD_S refers to the <b>CAS_n-to-CAS_n delay (short)</b> . It is the minimum time interval between two read/write (CAS) commands to <b>different bank groups</b> . (Identifier: MEM_DDR4_TCCD_S_CYC)
<b>tCCD_L</b>	tCCD_L refers to the <b>CAS_n-to-CAS_n delay (long)</b> . It is the minimum time interval between two read/write (CAS) commands to the <b>same bank group</b> . (Identifier: MEM_DDR4_TCCD_L_CYC)
<b>tWTR_S</b>	tWTR_S or Write Timing Parameter refers to the <b>Write to Read period for different bank groups</b> . It describes the delay from start of internal write transaction to internal read command, for accesses to the different bank

*continued...*

Display Name	Description
	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)
<b>tWTR_L</b>	tWTR_L or Write Timing Parameter refers to the <b>Write to Read period for the same bank group</b> . 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 87. Group: Mem Timing / Parameters dependent on Density and Temperature**

Display Name	Description
<b>tRFC</b>	tRFC refers to the <b>Refresh Cycle Time</b> . 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)
<b>tREFI</b>	tREFI refers to the <b>average periodic refresh interval</b> . It is the maximum amount of time the memory can tolerate in between each refresh command (Identifier: MEM_DDR4_TREFI_US)

### 6.1.6. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Controller

**Table 88. Group: Controller / Low Power Mode**

Display Name	Description
<b>Enable Auto Power-Down</b>	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. <b>All ranks must be idle to enter auto power-down</b> . (Identifier: CTRL_DDR4_AUTO_POWER_DOWN_EN)
<b>Auto Power-Down Cycles</b>	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 89. Group: Controller / Efficiency**

Display Name	Description
<b>Enable User Refresh Control</b>	When enabled, user logic has complete control and is responsible for issuing adequate 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)
<b>Enable Auto-Precharge Control</b>	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)

*continued...*

Display Name	Description
<b>Address Ordering</b>	Controls the mapping between Avalon addresses and memory device addresses. <b>By changing the value of this parameter, you can change the mappings between the Avalon-MM address and the DRAM address.</b> ( <i>CS</i> = chip select, <i>CID</i> = chip ID in 3DS/TSV devices, <i>BG</i> = bank group address, <i>Bank</i> = bank address, <i>Row</i> = row address, <i>Col</i> = column address) (Identifier: CTRL_DDR4_ADDR_ORDER_ENUM)
<b>Enable Reordering</b>	Enable this parameter to allow the controller to perform command and data reordering. <b>Reordering can improve efficiency by reducing bus turnaround time and row/bank switching time.</b> 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. <i>For more information, refer to the Data Reordering topic in the EMIF Handbook.</i> (Identifier: CTRL_DDR4_REORDER_EN)
<b>Starvation limit for each command</b>	Specifies the <b>number of commands that can be served before a waiting command is served</b> . 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. <i>For more information, refer to the Starvation Control topic in the EMIF Handbook.</i> (Identifier: CTRL_DDR4_STARVE_LIMIT)
<b>Enable Command Priority Control</b>	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. <b>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.</b> (Identifier: CTRL_DDR4_USER_PRIORITY_EN)
<b>Enable controller major mode</b>	Enable read and write commands flow control in command arbiter to reduce turnaround time, thus improving efficiency of random traffic pattern. (Identifier: CRTL_DDR4_MAJOR_MODE_EN)
<b>Enable controller post-pay refresh</b>	This feature allows the controller to delay refreshes to give way for mainband activities, or issue multiple refreshes when traffic is idle to improve HMC efficiency. (Identifier: CRTL_DDR4_POST_REFRESH_EN)
<b>Post-pay refresh lower limit</b>	A low refresh threshold for controller to stop streaming refreshes to memory devices. (Identifier: CRTL_DDR4_POST_REFRESH_LOWER_LIMIT)
<b>Post-pay refresh upper limit</b>	A panic refresh threshold for the controller to start streaming accumulated refreshes to memory devices. (Identifier: CRTL_DDR4_POST_REFRESH_UPPER_LIMIT)
<b>Enable controller pre-pay refresh</b>	This feature allows the controller to pull in refreshes to give way for mainband activities, or issue multiple refreshes when traffic is idle to improve HMC efficiency. (Identifier: CRTL_DDR4_PRE_REFRESH_EN)
<b>Refresh pre-pay upper limit</b>	A refresh threshold for controller to stop streaming pre-pay refreshes to memory devices. (Identifier: CRTL_DDR4_PRE_REFRESH_UPPER_LIMIT)

**Table 90. Group: Controller / Configuration, Status and Error Handling**

Display Name	Description
<b>Enable Memory-Mapped Configuration and Status Register (MMR) Interface</b>	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)
<b>Enable Error Detection and Correction Logic with ECC</b>	Enables error-correction code (ECC) for <b>single-bit error correction and double-bit error detection</b> . <i>ECC is implemented as soft logic.</i> (Identifier: CTRL_DDR4_ECC_EN)

*continued...*

Display Name	Description
<b>Enable Auto Error Correction to External Memory</b>	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)
<b>Enable ctrl_ecc_readdataerror signal to indicate uncorrectable data errors</b>	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)
<b>Export error-correction code (ECC) status ports</b>	Enable this parameter to export ECC status ports. (Identifier: CTRL_DDR4_ECC_STATUS_EN)

**Table 91. Group: Controller / Data Bus Turnaround Time**

Display Name	Description
<b>Additional read-to-write turnaround time (same rank)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>read to a write within the same logical rank</b> . This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_RD_TO_WR_SAME_CHIP_DELTA_CYCS)
<b>Additional write-to-read turnaround time (same rank)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>write to a read within the same logical rank</b> . This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_WR_TO_RD_SAME_CHIP_DELTA_CYCS)
<b>Additional read-to-read turnaround time (different ranks)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>read of one logical rank to a read of another logical rank</b> . This can resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_RD_TO_RD_DIFF_CHIP_DELTA_CYCS)
<b>Additional read-to-write turnaround time (different ranks)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>read of one logical rank to a write of another logical rank</b> . This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_RD_TO_WR_DIFF_CHIP_DELTA_CYCS)
<b>Additional write-to-write turnaround time (different ranks)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>write of one logical rank to a write of another logical rank</b> . This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_WR_TO_WR_DIFF_CHIP_DELTA_CYCS)
<b>Additional write-to-read turnaround time (different ranks)</b>	Specifies additional number of <b>idle controller (not DRAM)</b> cycles when switching the data bus from a <b>write of one logical rank to a read of another logical rank</b> . This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. <i>Use the default setting unless you suspect a problem exists.</i> (Identifier: CTRL_DDR4_WR_TO_RD_DIFF_CHIP_DELTA_CYCS)

### **6.1.7. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Diagnostics**

**Table 92. Group: Diagnostics / Simulation Options**

Display Name	Description
<b>Calibration mode</b>	<p>Specifies whether to <b>skip memory interface calibration</b> during simulation, or to <b>simulate the full calibration</b> process.</p> <p>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.</p> <p><i>If you enable this parameter, the interface still performs some memory initialization before starting normal operations. Abstract PHY is supported with skip calibration.</i></p> <p>(Identifier: DIAG_DDR4_SIM_CAL_MODE_ENUM)</p>

**Table 93. Group: Diagnostics / Calibration Debug Options**

Display Name	Description
<b>Skip address/command parity check during calibration</b>	Allows you to skip the address/command parity check during calibration. This parity check comes from reading the alert0_n pin from the DDR4 interface. (Identifier: DIAG_DDR4_SKIP_AC_PARITY_CHECK)

**Table 94. Group: Diagnostics / Example Design**

Display Name	Description
<b>Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>	<p>Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic.</p> <p>If you set this parameter to "<b>Disabled</b>", no debug features are enabled. If you set this parameter to "<b>Export</b>", 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 "<b>Add EMIF Debug Interface</b>", 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.</p> <p><i>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 "<b>Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>" option for all cores in the chain, and selecting "<b>Export</b>" for the "<b>Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>" option on all cores after the first.</i></p> <p>(Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_SLAVE)</p>
<b>Enable In-System-Sources-and-Probes</b>	Enables In-System-Sources-and-Probes in the example design for <i>common debug signals, such as calibration status or example traffic generator per-bit status</i> . 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 95. Group: Diagnostics / Traffic Generator (settings only applicable for example design)**

Display Name	Description
<b>Use configurable Avalon traffic generator 2.0</b>	This option allows you to add the new configurable Avalon traffic generator to the example design. (Identifier: DIAG_DDR4_USE_TG_AVL_2)
<b>Enable default traffic pattern (pattern configured during compile-time)</b>	Specifies that the default traffic pattern is enabled. If this parameter is <b>enabled</b> , a default traffic pattern is run immediately every time the traffic generator comes out of reset. If this parameter is <b>disabled</b> , the traffic generator does not run any traffic until it is signaled to start by its Avalon configuration interface. (Identifier: DIAG_DDR4_ENABLE_DEFAULT_MODE)
<b>Enable user-configured traffic pattern (pattern configured during run-time)</b>	Specifies that the user-defined traffic pattern is enabled. If this parameter is <b>enabled</b> , the traffic generator responds to the configuration interface and launch a user-configured traffic pattern when signaled to. If this parameter is <b>disabled</b> , the traffic generator ignores commands on the configuration interface, and does not run any user-defined traffic. (Identifier: DIAG_DDR4_ENABLE_USER_MODE)
<b>TG2 default traffic duration</b>	This option allows adjusting the pattern length of default (compile-time) traffic (Identifier: DIAG_DDR4_TG2_TEST_DURATION)
<b>TG2 Configuration Interface Mode</b>	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 " <b>Export</b> ", an Avalon slave interface named "tg_cfg" is exported from the IP. If you select " <b>JTAG</b> ", a JTAG Avalon Master Endpoint is connected to the configuration interface, allowing the core to be accessed by the TG Configuration Toolkit. (Identifier: DIAG_DDR4_EXPORT_TG_CFG_AVALON_SLAVE)

**Table 96. Group: Diagnostics / Performance**

Display Name	Description
<b>Efficiency Monitor Mode</b>	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_DDR4_EFFICIENCY_MONITOR)

**Table 97. Group: Diagnostics / Miscellaneous**

Display Name	Description
<b>Export PLL lock signal</b>	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)
<b>Export Address/Command parity error indicator</b>	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. (Identifier: DIAG_DDR4_AC_PARITY_ERR)

## 6.1.8. Intel Agilex 7 F-Series and I-Series EMIF IP DDR4 Parameters: Example Designs

**Table 98. Group: Example Designs / Example Designs with Multi-IPs**

Display Name	Description
<b>Simulation</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, simulation file sets are not created.</i> Instead, the output directory contains the ed_sim.qsys file which

*continued...*

Display Name	Description
	holds Platform Designer 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 <b>stored in the /sim sub-directory</b> . (Identifier: EX_DESIGN_GUI_DDR4_GEN_SIM)
<b>Synthesis</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, synthesis file sets are not created.</i> Instead, the output directory contains the ed_synth.qsys file which holds Platform Designer 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 <b>stored in the /qii sub-directory</b> . (Identifier: EX_DESIGN_GUI_DDR4_GEN_SYNTH)
<b>Signal Integrity</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary collateral for performing Signal Integrity analysis with a third-party analog simulation tool. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, board simulation collateral is not created.</i> The generated collateral is <b>stored in the /bsi sub-directory</b> . Note that this option is only supported for selected memory protocols on the Intel Agilex 7 family. (Identifier: EX_DESIGN_GUI_DDR4_GEN_BSI)
<b>Spyglass CDC</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary files for performing CDC analysis with Spyglass. Expect a short additional delay as the file set is created. If you do not enable this parameter, CDC file sets simulation are not created. The generated collateral is stored in the /cdc sub-directory. Note that this option is only supported for selected memory protocols. (Identifier: EX_DESIGN_GUI_DDR4_GEN_CDC)
<b>Simulation HDL format</b>	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_DDR4_HDL_FORMAT)
<b>Number of IPs</b>	Specifies the <i>number of EMIF IPs to instantiate</i> in the example designs. All the IPs have individual TGs and can be connected to one of the available Cal-IPs.
<b>EMIF ID</b>	Specifies the number of EMIF IP that you want in the Multi-IP design. Depending on the size of the EMIF interface, you can have up to 16 EMIF interfaces.
<b>CAL-IP</b>	Specifies the Calibration IP to which a given EMIF should connect in the example design. All the EMIF IPs must be connected to one of the available Cal-IPs. There are two Calibration IPs on the device. Each of the EMIF IP must be connected to either of the two Cal-IPs.
<b>Capture</b>	The capture button allows you to take a capture of the current EMIF IP settings and apply to given EMIF ID interface.

**Table 99. Group: Example Designs / Target Development Kit**

Display Name	Description
<b>Select board</b>	Specifies that <i>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.</i> Any IP settings not applied directly from a development kit preset do not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select ' <b>none</b> ' from the ' <b>Select board</b> ' 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

Display Name	Description
	you apply the preset. You can save your settings under a different name using <b>File-&gt;Save as.</b> (Identifier: EX_DESIGN_GUI_DDR4_TARGET_DEV_KIT)

## 6.2. Intel Agilex 7 F-Series and I-Series External Memory Interfaces Intel Calibration IP Parameters

The following parameters are found in the *Intel Agilex 7 External Memory Interfaces Intel Calibration IP*.

**Table 101. Group: Calibration and Debug**

Display Name	Description
<b>Number of Calibration Interfaces</b>	Specifies the number of calbus interfaces to connect to the EMIF calibration IP (Identifier: NUM_CALBUS_INTERFACE)
<b>Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>	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_EXPORT_SEQ_AVALON_SLAVE)

**Table 102. Group: Simulation**

Display Name	Description
<b>Calibration mode for simulation</b>	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_SIM_CAL_MODE_ENUM)
<b>Show verbose simulation debug messages</b>	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_SIM_VERBOSE)

## 6.3. Register Map IP-XACT Support for Intel Agilex 7 F-Series and I-Series EMIF DDR4 IP

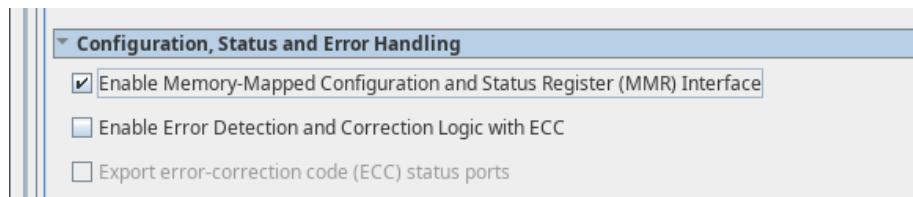
IP-XACT is an XML format that describes reusable intellectual property (IP).

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 Agilex 7 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:

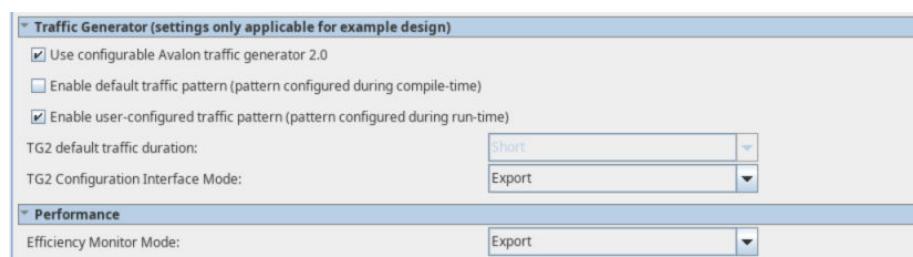
1. To enable generation of the IP-XACT information for Intel Agilex 7 IP MMR, check the **Enable Memory-Mapped Configuration and Status Register (MMR) Interface** box on the **Controller** tab of the parameter editor.

**Figure 121. 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 122. Enabling IP-XACT Generation for TG2 and Efficiency Monitor**



For information on the registers available for the Intel Agilex 7 EMIF IP, refer to *Intel Agilex 7 EMIF IP Memory Mapped Register (MMR) Tables* in the *Architecture* 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.

### Related Information

- Intel Agilex 7 F-Series and I-Series EMIF IP Memory Mapped Register (MMR) Tables on page 101
- Configuration and Status Registers on page 307
- Control and Status Registers on page 377

## 6.4. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP 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
- RZQ pins
- Other FPGA resources—for example, core fabric logic, and debug interfaces

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

### 6.4.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Interface Pins

Any I/O banks that do not support transceiver operations in Intel Agilex 7 FPGAs support external memory interfaces. However, DQS (data strobe or data clock) and DQ (data) pins are listed 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 pin table for the actual locations of the DQS and DQ pins.

You can find the pin tables at the following location: <https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html>.

**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.intel.com](http://www.intel.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 96 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.

#### 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. Many registered DIMMs use only one pair of clocks; however, this is not a universal rule, so you should check your memory vendor's data sheet to be sure. DDR4 registered DIMMs require a minimum of one chip-select signal.

**Table 103. UDIMM, RDIMM, and LRDIMM Pin Options for DDR4**

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Data	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}
Data Mask	DM#/DBI#[8:0] <sup>(1)</sup>	DM#/DBI#[8:0] <sup>(1)</sup>	DM#/DBI#[8:0] <sup>(1)</sup>	DM#/DBI#[8:0] <sup>(1)</sup>	—	—
Data Strobe	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]
Address	BA[1:0], BG[1:0], A[16:0] - 4GB: A[14:0] 8GB: A[15:0]	BA[1:0], BG[1:0], A[16:0] - 8GB: A[14:0] 16GB: A[15:0]	BA[1:0], BG[1:0], x8: A[16:0] - A[17:0] x4: A[17:0] - 8GB: A[14:0] 16GB: A[15:0]	BA[1:0], BG[1:0], A[17:0] - 16GB: A[15:0] 32GB: A[15:0] 64GB: A[16:0] <sup>(2)</sup>	BA[1:0], BG[1:0], A[17:0] - 16GB: A[15:0] 32GB: A[15:0] 64GB: A[16:0] <sup>(2)</sup>	BA[1:0], BG[1:0], A[17:0] - 32GB: A[15:0] 64GB: A[16:0] <sup>(2)</sup>

*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)
	16GB: A[16:0] (2)	32GB: A[16:0] (2)	16GB: A[16:0] (2) 32GB: A[17:0] (3)	32GB: A[16:0] (2) 64GB: A[17:0] (3)	64GB: A[17:0] (3)	128GB: A[17:0] (3)
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#, CK1/CK1#	CK0/CK0#, CK1/CK1#
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:						
<ol style="list-style-type: none"> <li>1. DM/DBI pins are available only for DIMMs constructed using x8 or greater components.</li> <li>2. This density requires 4Gb x4 or 2Gb x8 DRAM components.</li> <li>3. This density requires 8Gb x4 DRAM components.</li> <li>4. This table assumes a single slot configuration. The Intel Agilex 7 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.</li> </ol>						

#### 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 Agilex 7 devices, consult the EMIF Device Selector on [www.intel.com](http://www.intel.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*.

#### 6.4.2. Intel Agilex 7 FPGA EMIF IP Resources

The Intel Agilex 7 F-Series and I-Series FPGA memory interface IP uses several FPGA resources to implement the memory interface.

#### 6.4.2.1. OCT

You require an OCT calibration block if you are using an Intel Agilex 7 F-Series and I-Series FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. There are two OCT blocks in an I/O bank, one for each sub-bank.

You must observe the following requirements when using OCT blocks:

- The I/O bank where you place the OCT calibration block must use the same  $V_{CCIO\_PIO}$  voltage as the memory interface.
- The OCT calibration block uses a single fixed  $R_{ZQ}$ . You must ensure that an external termination resistor is connected to the correct pin for a given OCT block.

For specific pin connection requirements, refer to [Specific Pin Connection 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.

For specific pin connection requirements, refer to [Specific Pin Connection Requirements](#).

#### 6.4.3. Pin Guidelines for Intel Agilex 7 F-Series and I-Series FPGA EMIF IP

The Intel Agilex 7 F-Series and I-Series FPGA contains I/O banks on the top and bottom edges of the device, which can be used by external memory interfaces.

Intel Agilex 7 F-Series and I-Series FPGA I/O banks contain 96 I/O pins. Each bank is divided into two sub-banks with 48 I/O pins in each. Sub-banks are further divided into four I/O lanes, where each I/O lane is a group of twelve I/O ports.

The I/O bank, I/O lane, and pairing pin for every physical I/O pin can be uniquely identified by the following naming convention in the device pin table:

- The I/O pins in a bank are represented as P#X#Y#, where:
  - P# represents the pin number in a bank. It ranges from P0 to P95, for 96 pins in a bank.
  - X# represents the bank number on a given edge of the device. X0 is the farthest bank from the zipper.
  - Y# represents the top or bottom edge of the device. Y0 and Y1 refer to the I/O banks on the bottom and top edge, respectively.
- Because an IO96 bank comprises two IO48 sub-banks, all pins with P# value less than 48 (P# <48) belong to the same I/O sub-bank. All other pins belong to the second IO48 sub-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 one of I/O lanes 0, 1, 2, or 3, respectively.
- To determine whether I/O banks are adjacent, you can refer to the sub-bank-ordering figures for your device family in the [Architecture: I/O Bank](#) topic, to the [Typical Intel Agilex 7 FPGA with all Banks Bonded Out](#) figure, and to the I/O Pin Count Tables located in the [Intel Agilex 7 General Purpose I/O User Guide](#).

In general, you can assume that I/O banks are adjacent within an I/O edge, unless the I/O bank is not bonded out on the package (indicated by the presence of the " - " symbol in the I/O table), or if the I/O bank does not contain 96 pins, indicating that it is only partially bonded out. If an I/O bank is not fully bonded out in a particular device, it cannot be included within the span of sub-banks for a larger external memory interface. In all cases, you should use the Intel Quartus Prime software to verify that your usage can be implemented.

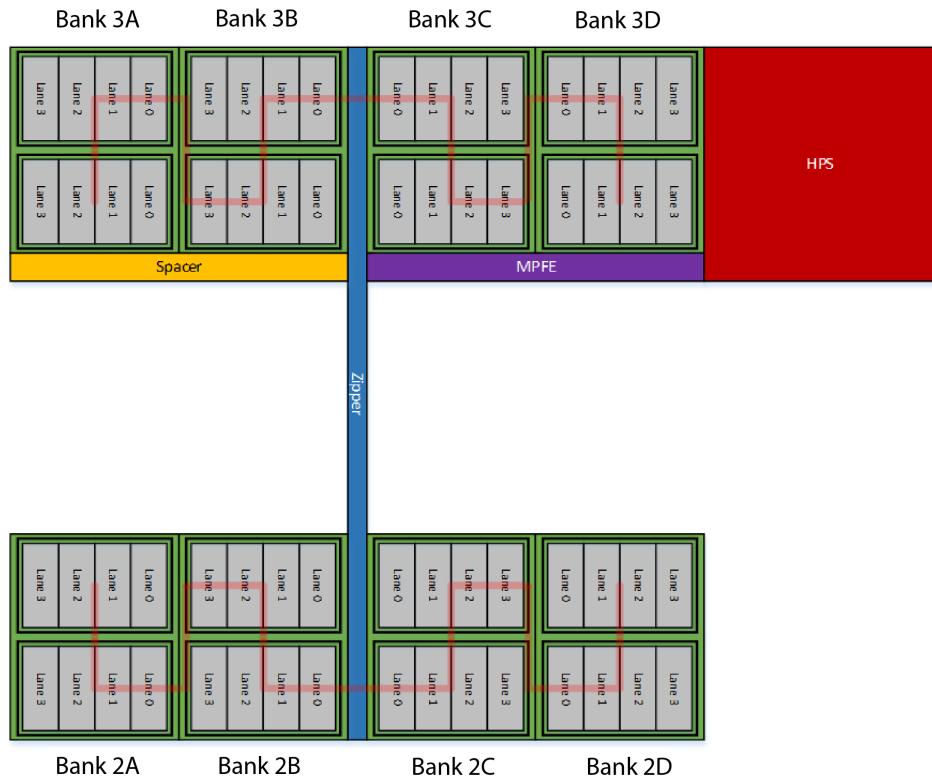
- The pairing pin for an I/O pin is in the same I/O bank. You can identify the pairing pin by adding 1 to its *Index Within I/O Bank* number (if it is an even number), or by subtracting 1 from its *Index Within I/O Bank* number (if it is an odd number).

#### 6.4.3.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Banks

Before you select pins for your Intel Agilex 7 F-Series and I-Series FPGA external memory interface, it is important that you understand how banks and sub-banks are grouped together to form a single interface.

The following diagram illustrates a typical Intel Agilex 7 F-Series and I-Series FPGA with all banks bonded out to pins.

**Figure 123. Typical Intel Agilex 7 FPGA with all Banks Bonded Out**



In the above diagram, the group of 4 lanes in the top-left corner (the top sub-bank in bank 3A), denotes the IO48 block that is used for test mode and AVST configuration. If all I/O lanes in this sub-bank are used for configuration, then this bank cannot be used for the external memory interface. Similarly, the bottom sub-bank in bank 3A could not be used for the external memory interface either, because all the I/O sub-banks in a given interface must be contiguous.

**The red line in the above diagram denotes the chaining order of the sub-banks to form an external memory interface. The chaining order flips when crossing the zipper.**

For additional details, refer to the [Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Product Architecture](#) chapter.

#### 6.4.3.2. General Guidelines

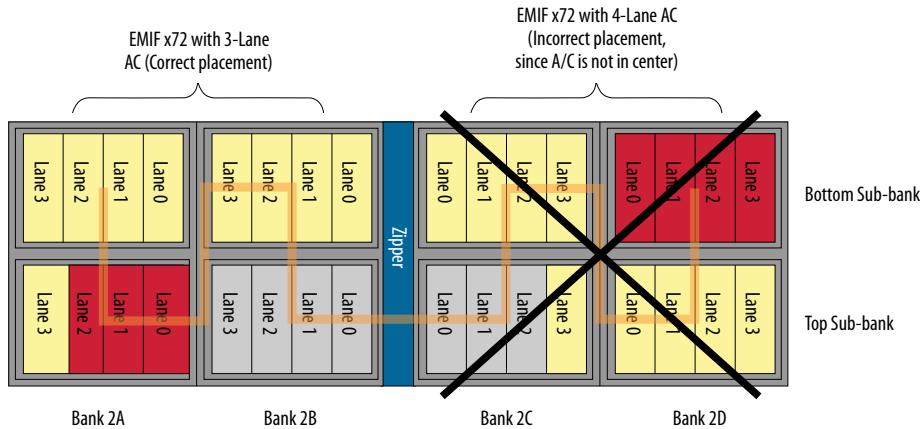
You should follow the recommended guidelines when performing pin placement for all external memory interface pins targeting Intel Agilex 7 F-Series and I-Series devices, whether you are using the hard memory controller or your own solution.

*Note:*

- EMIF IP pin-out requirements for the Intel Agilex 7 F-Series and I-Series 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 Pro Edition IP file (.qip), based on the IP configuration.
- PHY only, RLDRAMx, and QDRx are not supported with HPS.

Observe the following general guidelines when placing pins for your Intel Agilex 7 F-Series and I-Series external memory interface:

1. Ensure that the pins of a single external memory interface reside on the same edge I/O.
2. An external memory interface can occupy one or more banks on the same edge. When an interface must occupy multiple banks, ensure that those banks are adjacent to one another.
  - If an I/O bank is shared between two interfaces—meaning that two sub-banks belong to two different EMIF interfaces—then both the interfaces must share the same voltage.
  - Sharing of I/O lanes within a sub-bank for two different EMIF interfaces is not permitted; I/O lanes within a sub-bank can be assigned to one EMIF interface only.
3. Any pin in the same bank that is not used by an external memory interface may not be available for use as a general purpose I/O pin:
  - For fabric EMIF, unused pins in an I/O lane assigned to an EMIF interface cannot be used as general-purpose I/O pins. In the same sub-bank, pins in an I/O lane that is not assigned to an EMIF interface, can be used as general-purpose I/O pins.
  - For HPS EMIF, unused pins in an I/O lane assigned to an EMIF interface cannot be used as general-purpose I/O pins. In the same sub-bank, pins in an I/O lane that is not assigned to an EMIF interface cannot be used as general-purpose I/O pins either. Refer to [Restrictions on I/O Bank Usage for Intel Agilex 7 EMIF IP with HPS](#) for more information.
4. All address and command pins and their associated clock pins (CK and CK#) must reside within a single sub-bank. The sub-bank containing the address and command pins is identified as the address and command sub-bank.
5. To minimize latency, when the interface uses more than two sub-banks, you must select the center sub-bank as the address and command sub-bank. For example, the following image shows placement of two DDR4 x72 interfaces:



Legend: red = address/command, yellow = data.

- In the above illustration, the placement on the left is correct. If you follow the sub-bank chaining order, the address and command sub-bank is in the center.
  - The placement on the right is incorrect, because the address and command sub-bank is the first sub-bank in the chain. Correct placement in this case, would be to place the address and command pin in the top sub-bank of tile 2D, and place data pins in the bottom sub-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 Agilex 7 External Memory Interface Pin Information* file, which is available here: [Pin-Out Files for Intel FPGA Devices](#).
  7. An unused I/O lane in the address and command sub-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:* For DDR4 interfaces with x4 components, you can use the strobe pins with either of the upper or lower DQ nibbles that are placed within a x8 DQS group in an I/O lane. Intel recommends placing the DQ pins and associated strobes entirely in either the upper or lower half of a 12-bit bank sub-group. Consult the pin table for your device to identify the association between DQ pins and DQS pins for x4 mode operation. Additional restrictions apply for x4/x8 DIMM interoperability.
  10. One of the sub-banks in the device (typically the sub-bank within corner bank 3A) may not be available if you use certain device configuration schemes. For some schemes, there may be an I/O lane available for EMIF data group.
    - AVST-8 – This is contained entirely within the SDM, therefore all lanes of sub-bank 3A can be used by the external memory interface.
    - AVST-32 – Lanes 0, 1, 2, and 3 are all effectively occupied and are not usable by the external memory interface.
    - AVST-16 – Lanes 0, 1, and 3 are not usable by the external memory interface. However, lane 2 contains SDM\_MISSION\_DATA[25:16]. If SDM\_MISSION\_DATA[25:16] is not required for AVSTx16, then Lane 2 is available for use by the external memory interface.
  11. Two memory interfaces cannot share an I/O 48 sub-bank.

#### 6.4.3.3. 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 104. 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]
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<sub>n</sub> 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. Intel strongly recommends that trace delays for two nibbles packed within the IO12 lanes are matched using the same guidelines as a single x8 byte 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.
  - b. Check that DQ[3:0] are in the x4 group with DQS0, DQ[7:4] are in the x4 group with DQS9, and so forth. This is the *DIMM* numbering convention column shown in the table at the beginning of this topic.
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 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.
- b. Check that DQ[3:0] are in the x4 group with DQS0, DQ[7:4] are in the x4 group with DQS1 and so forth. This is the *Intel Quartus Prime EMIF IP* mapping shown in the table at the beginning of this topic.
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 DQy on the DIMM maps to the DQy on the FPGA pinout. Note that there is scope for swapping pins within the x4 DQS group to optimize the PCB layout.

#### 6.4.3.4. Specific Pin Connection Requirements

##### PLL

You must constrain the PLL reference clock to the address and command sub-bank only.

- You must constrain the single-ended reference clock to pin index 0 in lane 2.
- When pin index 0 in lane 2 is used for a single-ended reference clock, you cannot use pin index 1 in lane 2 as a general purpose I/O pin.
- You must constrain differential reference clocks to pin indices 0 and 1 in lane 2.
- The sharing of PLL reference clocks across multiple external memory interfaces is permitted; however, pin indices 0 and 1 of Lane 2 of the address and command sub-bank for all slave EMIF interfaces can be used only for supplying reference clocks. Intel recommends that you consider connecting these clocks input pins to a reference clock source to facilitate greater system implementation flexibility.

##### OCT

You must constrain the RZQ pin to pin index 2 in lane 2 of the address and command sub-bank only.

- Every EMIF instance requires its own dedicated RZQ pin.
- The sharing of RZQ pins is not permitted.

##### Address and Command

For DDR4, you must constrain the ALERT\_N pin to the address and command lane only.

- In three-lane address and command schemes, you can place the ALERT\_N pin at pin index 8 in lane 2 only.
- In four-lane address and command schemes, you can place the ALERT\_N pin at pin index 8 in lane 2 or at pin index 8 in lane 3. When you generate the IP, the resulting RTL specifies which connection to use.

### DQS/DQ/DBI#

For DDR4 x8 DQS grouping, the following rules apply:

- You may use pin indices 0, 1, 2, 3, 8, 9, 10, and 11 within a lane for DQ mode pins only.
- You must use pin index 4 for the DQS\_p pin only.
- You must use pin index 5 for the DQS\_n pin only.
- You must ensure that pin index 7 remains unused. Pin index 7 is not available for use as a general purpose I/O.
- You must use pin index 6 for the DM/DBI\_N pin only.

For DDR4 x4 DQS grouping, the following rules apply:

- You may use pin indices 0, 1, 2, and 3 within a lane for DQ mode pins for the lower nibble only. Pin rotation within this group is permitted.
- You must use pin index 4 for the DQS\_p pin only of the lower nibble.
- You must use pin index 5 for the DQS\_n pin only of the lower nibble.
- You may use pin indices 8, 9, 10, and 11 within a lane for the DQ mode pins only for the upper nibble. Pin rotation within this group is permitted.
- You must use pin index 6 for the DQS\_p pin only of the upper nibble.
- You must use pin index 7 for the DQS\_n pin only of the upper nibble.

#### 6.4.3.5. 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.

Although DDR4 operates in fundamentally the same way as other SDRAM, there are no dedicated pins for RAS#, CAS#, and WE#, as those are shared with higher-order address pins. DDR4 has CS#, CKE, ODT, and RESET# pins, similar to DDR3. DDR4 also has some additional pins, including the ACT# (activate) pin and BG (bank group) pins.

#### 6.4.3.6. Clock Signals

DDR4 SDRAM devices use CK and CK# signals to clock the address and command signals into the memory. 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 devices have a write requirement ( $t_{DQSS}$ ) that states the positive edge of the DQS signal on writes must be within  $\pm 25\%$  ( $\pm 90^\circ$ ) of the positive edge of the SDRAM clock input. Therefore, you should generate the CK and CK# signals using the DDR registers in the IOE to match with the DQS signal and reduce any variations across process, voltage, and temperature. The positive edge of the SDRAM clock, CK, is aligned with the DQS write to satisfy  $t_{DQSS}$ .

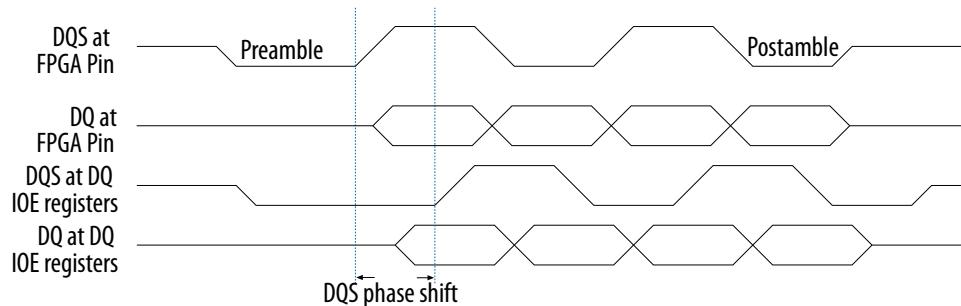
#### **6.4.3.7. Data, Data Strobes, DM/DBI, and Optional ECC Signals**

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

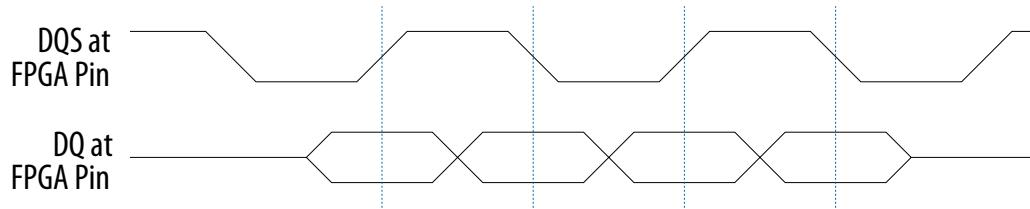
The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by -90 degrees during a write operation to center align the DQ and DQS signals. The PHY IP delays the 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 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 SDRAM.

**Figure 124. 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 125. 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.

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 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 Agilex 7 interfaces, the DM/DBI pins 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.

## 6.5. 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:

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

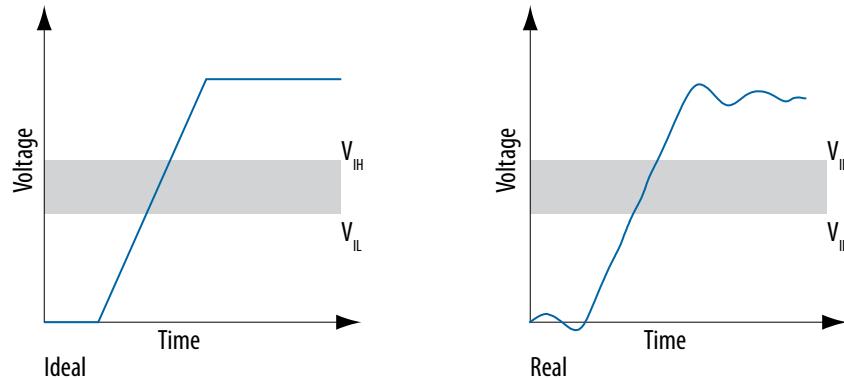
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

The following key factors affect signal quality at the receiver:

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

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

**Figure 126. Ideal and Real Signal at the Receiver**



## 6.5.1. Terminations for DDR4 with Intel Agilex 7 F-Series and I-Series Devices

The following topics describe considerations specific to DDR4 external memory interface protocols on Intel Agilex 7 F-Series and I-Series devices.

### 6.5.1.1. Dynamic On-Chip Termination (OCT)

Depending upon the  $R_s$  (series) and  $R_t$  (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.

Refer to the *External Memory Interfaces Intel Agilex 7 FPGA IP* parameter editor to determine the supported termination values.

#### 6.5.1.2. 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.

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.

#### 6.5.1.3. Choosing Terminations on Intel Agilex 7 F-Series and I-Series FPGA 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, using a simulation model extracted from the PCB of your memory interface channel.

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.

#### 6.5.1.4. On-Chip Termination Recommendations for Intel Agilex 7 F-Series and I-Series FPGA Devices

In the EMIF IP parameter editor you can select values from drop-down lists for each of the following:

- output mode drive strength for the address/command bus.
- output mode drive strength for the memory clock.
- output mode drive strength for the data bus.
- input mode termination strength for the data bus.

The range of available values may vary, depending on your memory protocol and silicon revision.

You can use the default values as starting points; however, for best results, you should sweep the entire range of legal values and generate multiple hardware designs to determine the optimal settings for your board and memory device. The optimal settings are those that yield the largest margin as measured by the Driver Margining tool.

Once you have found the optimal settings for your design, uncheck the **Use Default I/O settings** checkbox and use your optimal settings for all future compilations, even if those settings align with the default settings. This ensures that your settings are preserved if the IP is upgraded to a future version.

## 6.5.2. 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 105. 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 <sup>(1)</sup>	BG0	BG1
BG1 <sup>(1)</sup>	BG1	BG0

<sup>(1)</sup> 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 106. 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 107. 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.

### 6.5.3. General Layout Routing Guidelines

Follow the guidelines in this section for routing from the FPGA to memory for Intel Agilex 7 F-Series and I-Series devices.

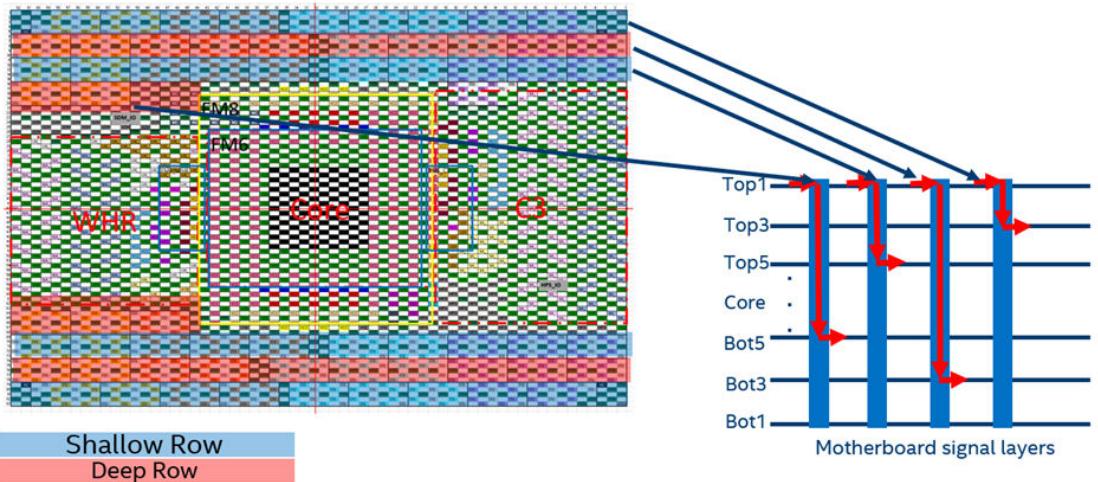
For maximum channel margin, you should consider the following general routing optimizations during the layout design phase:

- When routing the memory interface, ensure that there are solid ground reference planes without any plane splits or voids, to ensure an uninterrupted current return path.
- For signal vias in layer transitions, you must place ground stitching vias close by, within 80 mil in distance (closer is better), and in between signal vias, to minimize crosstalk among signal vias. Avoid any unnecessary signal layer transitions to minimize crosstalk, loss, and skews.
- Trace impedance plays an important role in signal integrity; board designers must follow impedance recommendations for each signal group and configuration according to the guidelines in this document. If you use a different stackup than the reference stackup in the PCB design, you must tune the trace width and geometries to achieve the impedance recommendations.
- Intel recommends using 45-degree angles (not sharp 90-degree corners) when routing signal turns. Use  $3 \times h$  spacing for serpentine routing, where  $h$  is the height or distance from the trace to the nearest GND reference plane.

- Avoid referencing a signal to both power and ground planes at the same time (dual referencing), for signal return paths. When this cannot be avoided, ensure that the closer reference plane is solid ground, and the far side power plane is not noisy.
- Avoid routing two internal signal layers adjacent to each other (dual stripline routing). When this cannot be avoided, use angled routing between two signal layers to minimize crosstalk and coupling between the layers.
- Follow time-domain length and skew matching rules to ensure that your interface meets timing requirements. You should route signals from the same byte or group together on the same layer to avoid any out-of-phase crosstalk caused by varying layer transition lengths.
- To optimize memory interface margins, Intel recommends the following routing strategies:
  - For DIMM configurations, route DQ and DQS signals on shallow layers with short via transition lengths, because they have tighter timing margins than address, command, and control signals. (Shallow layers are those above the PCB core where via transition lengths are short.)
  - For discrete device configurations, route address, command, and control signals on shallow layers.

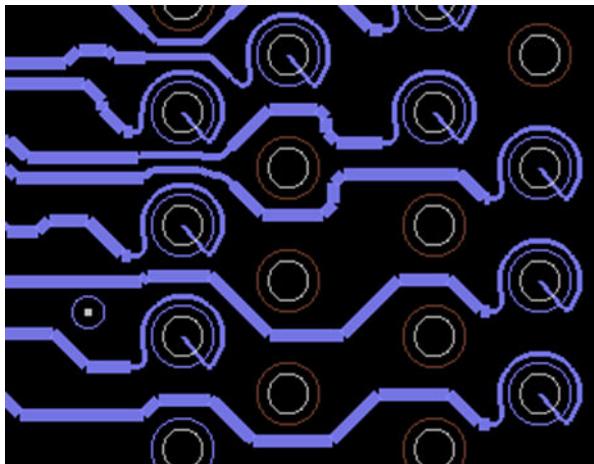
- For boards thicker than 65 mil, Intel recommends alternating adjacent FPGA EMIF BGA/ball rows with deep and shallow board via transitions to minimize crosstalk between adjacent bytes. This method is illustrated in the following figure:

**Figure 127. Recommended alternate adjacent via transitions to avoid crosstalk between adjacent bytes**



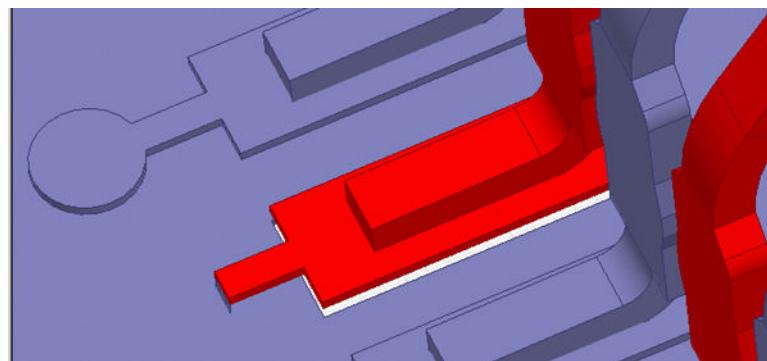
- For boards thicker than 65 mil, using the pin-through-hole (PTH) type of DIMM connector, Intel recommends implementing a loop-routing-around-DIMM-pin structure (Lcomp) to improve impedance matching between signal routing and the DIMM connector. Refer to the following figure.

**Figure 128. Recommended Lcomp structure for better impedance matching**

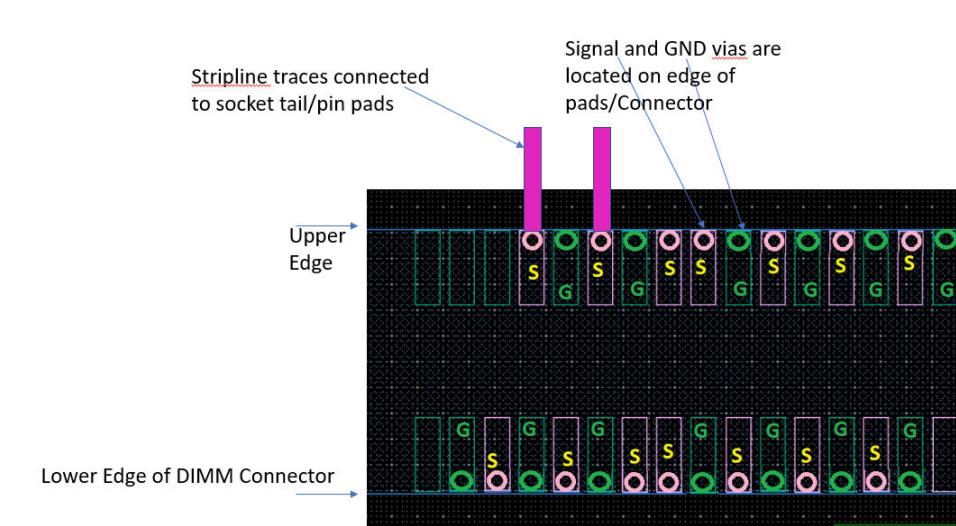


- For PCB designs using a surface mount technology (SMT) type of DIMM connector, Intel recommends placing a cutout (void) in the ground reference plane underneath the connector pads for DDR4 signals to minimize connector pad capacitance. Refer to the following figure for the recommended cutout on ground reference plane underneath the connector pad on surface layer.

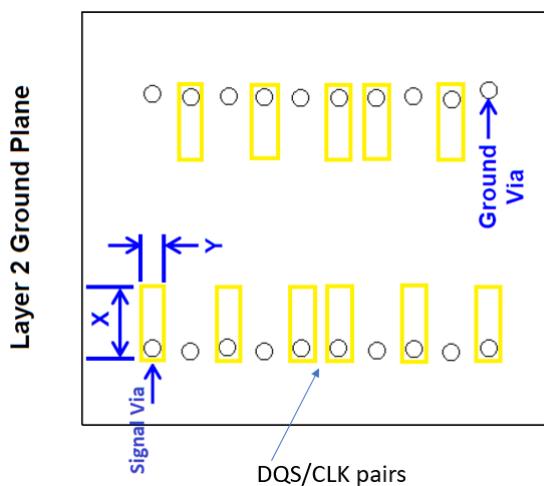
**Figure 129. Recommended Cutout on Ground Reference Plane**



**Figure 130. Closeup View of Connections**

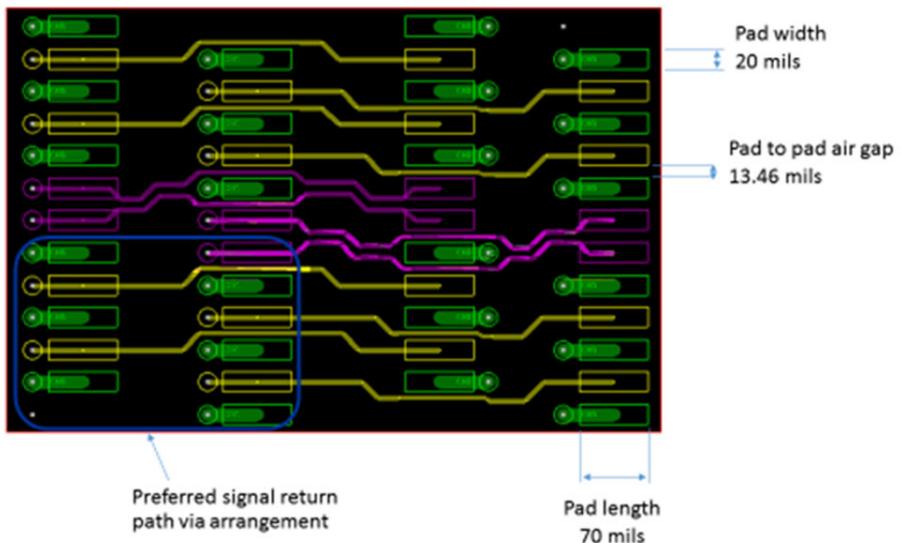


**Figure 131. Closeup View of Connections**



- For two-DIMMs-per-channel (2DPC) in UDIMM, RDIMM, or LRDIMM configurations with an SMT type of DIMM connector, Intel recommends have signal transition via to the near DIMM connector, then routing the trace on the surface layer (microstrip line routing) through connector pads to the far DIMM connector. The following figure depicts the recommended routing guidelines between two DIMMs in one channel.

**Figure 132. Recommended routing guidelines between two DIMMs in 2DPC EMIF topology**



#### 6.5.4. Reference Stackup

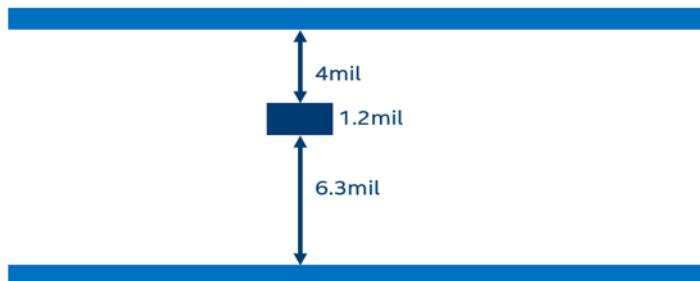
This topic illustrates the reference stackup on which EMIF routing design guidelines are based.

It is important to understand that trace geometry such as width, thickness, and edge-to-edge spacing, and the distance to reference planes, all impact trace impedance and crosstalk levels.

**Table 108. Reference stackup details**

Layer	Type	Thickness
<b>SM TOP</b>		0.5
<b>L1</b>	signal	1.8
<b>D1</b>	prepreg	2.7
<b>L2</b>	gnd/power	1.2
<b>D2</b>	core	4.0
<b>L3</b>	signal	1.2
<b>D3</b>	prepreg	6.3
<b>L4</b>	gnd/power	1.2
<b>D4</b>	core	4.0
<i>continued...</i>		

Layer	Type	Thickness
<b>L5</b>	signal	1.2
<b>D5</b>	prepreg	6.3
<b>L6</b>	gnd/power	1.2
<b>D6</b>	core	4.0
<b>L7</b>	signal	1.2
<b>D7</b>	prepreg	6.3
<b>L8</b>	gnd	1.2
<b>D8</b>	core	4
	Power	1.2
	prepreg	6.3
	power	1.2
	core	4
	gnd	1.2
	prepreg	6.3
	power	1.2
	core	4
<b>L9</b>	gnd	1.2
<b>D9</b>	prepreg	6.3
<b>L10</b>	signal	1.2
<b>D10</b>	core	4.0
<b>L11</b>	gnd/power	1.2
<b>D11</b>	prepreg	6.3
<b>L12</b>	signal	1.2
<b>D12</b>	core	4.0
<b>L13</b>	gnd/power	1.2
<b>D13</b>	prepreg	6.3
<b>L14</b>	signal	1.2
<b>D14</b>	core	4.0
<b>L15</b>	gnd/power	1.2
<b>D15</b>	prepreg	2.7
<b>L16</b>	signal	1.8
<b>SM BOT</b>		0.5
	Total	120.1

**Figure 133. Reference trace geometries**

The reference stackup height is selected to be 120 mil to cover maximum signal via coupling (110mil) in simulation while extracting EMIF design guideline. Intel recommends that board designers do not exceed 110mil signal via coupling (stripline routing on inner layers) in the EMIF layout PCB design for DDR4 interfaces.

If the PCB stackup exceeds 120 mil in height, Intel recommends routing EMIF signals on upper layers, not to exceed more than 110 mil of signal via coupling.

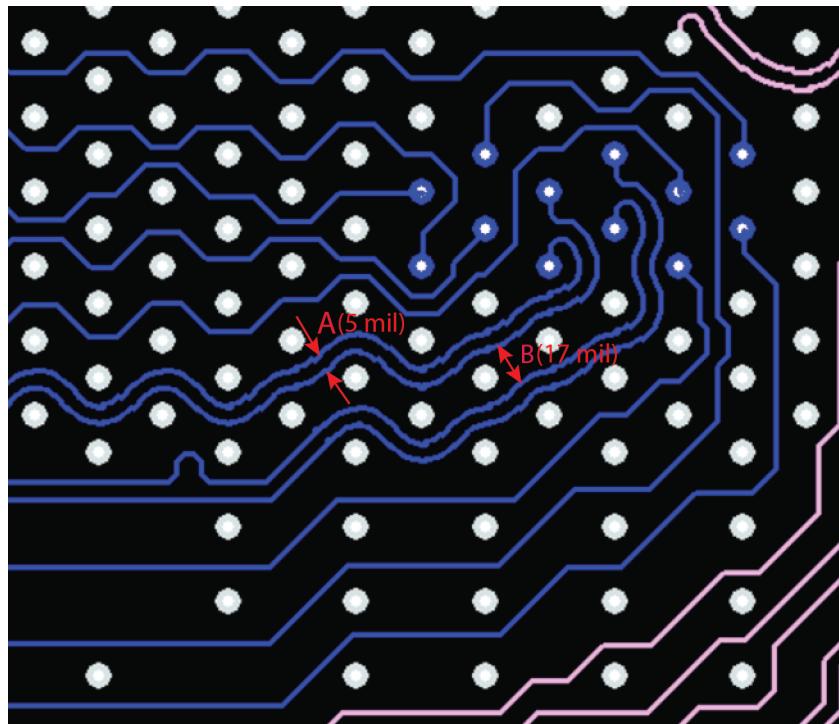
The reference stackup materials in the above figure are selected as FR4, to represent worst-case signal loss in design phase simulation. In case of low-loss materials, the maximum end-to-end routing length shall be larger than the recommended end-to-end routing length in the design guidelines; however, you must perform time-domain channel simulation to ensure that timing requirements are met.

### 6.5.5. Intel Agilex 7 F-Series and I-Series EMIF-Specific Routing Guidelines for Various DDR4 Topologies

This section discusses EMIF-related layout guidelines for Intel Agilex 7 devices.

The Intel Agilex 7 family pin floorplan is a HEX pattern with 1mm pitch. The following figure shows an example of DDR routing for an IO12 (one-byte data) on PCB within FPGA fan-out region.

**Figure 134. Intel Agilex 7 1mm HEX pin pattern/floorplan and recommended routing for one byte of data (IO12)**



The following general notes apply to the EMIF routing guidelines tables in subsequent topics:

- All spacing requirements are the minimum requirement to be met on PCB in EMIF routing guideline table.
- Breakout (BO1/BO2) spacings have two different values in guideline tables. The first value represents minimum spacing between two signals routed as a pair (tightly coupled signals); this value is marked as A (5 mil) in the above figure. The second value represents minimum spacing between two pairs, and is marked as B (17 mil) in the above figure.
- Main route (M) spacings have both value in mil and formula. In formula,  $h$  represents the trace-to-nearest-reference-plane height or distance. In cases using a stackup different than the reference stackup, board designers shall use formula to calculate the correct spacing requirements.
- There is no differential impedance target for CLK nor DQS. Board designers shall follow single-ended impedance target and keep the signals within the pair closely coupled, within 3-4 mil spacing. For information on DQS/DQSB and CLK/CLKB, refer to the [Skew Matching Guidelines for DDR4 DIMM Topologies](#) and [Skew Matching Guidelines for DDR4 Discrete Topologies](#) tables, for DIMM and discrete device implementations, respectively.

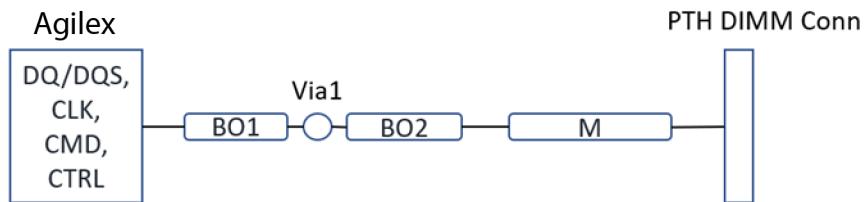
- In guideline tables, *SL* stands for stripline routing recommendation and *US* stands for upper surface (Microstrip) routing recommendation.
- The trace width value/geometry in guideline tables stands for trace designed for target impedance based on the reference stackup. This trace geometry shall be designed based on actual stackup and target impedance in guideline table.
- In guideline tables, *BO1* and *BO2* represent fan-out routing lengths. *M* stands for out of fan-out (PCB main) routing lengths

#### **6.5.5.1. One DIMM per Channel (1DPC) for UDIMM, RDIMM, LRDIMM, and SODIMM DDR4 Topologies**

The interface covers data bytes (DQ/DQS), address signals, command signals (BA, BG, RAS, CAS, WE, ACT, PAR), control signals (CKE, CS, ODT) and clocks (CLK).

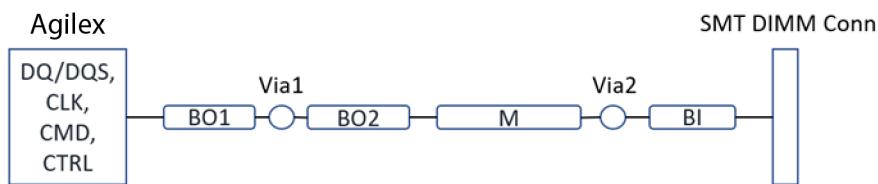
The following figure illustrates the signal connection topology for a PTH type of connector for UDIMM, RDIMM, and LRDIMM topologies.

**Figure 135. Signal connections for DDR4 1DPC DIMM configuration using PTH DIMM connector**



The following figure illustrates the signal connection topology for an SMT type of connector for UDIMM, RDIMM, LRDIMM, and SODIMM topologies.

**Figure 136. Signal connections for DDR4 1DPC DIMM configuration using SMT DIMM connector**



The following table provides specific routing guidelines for one DIMM per channel in UDIMM, RDIMM, LRDIMM, and SODIMM topologies for all supported signals in the interface.

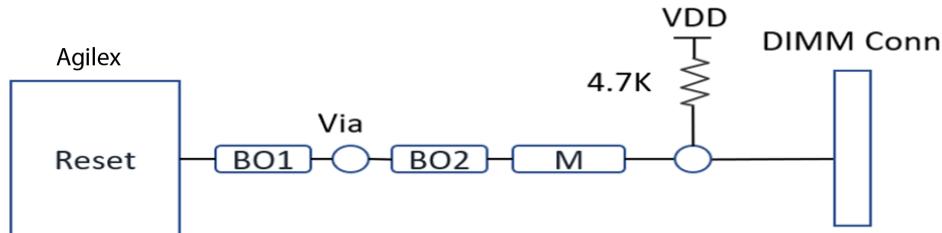
**Table 109. Specific DDR4 IDPC routing guidelines for UDIMM, RDIMM, LRDIMM, and SODIMM configurations**

Signal Group	Segment	Routing Layer	Max Length (mil)	Target Width, W (mil)	Target Zse (ohms)	Total MB Segment	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), Within DIFF pair	Trace Spacing (mil), CLK pair to CMD/CTRL/CK E	Trace Spacing (mil), DQ pair to DQ	Trace Spacing (mil), CLK pair to CMD/CTRL/CLK E	Channel to Channel Spacing (DQ to DQ, between two channels )
CLK	BO1	US	50	4500			4	5, 17	5, 17	4				17
	BO2	SL	1000				4	5, 17	5, 17	4				17
M	SL				45					4				12 (3)
BI	US	50					4			4				12 (3h)
CMD, CTRL, ALERT	BO1	US	50	4500			4	5, 17	5, 17					
	BO2	SL	1000				4	5, 17	5, 17					
M	SL				45									
BI	US	100					4	8 (2h)	12 (3h)					
DQ	BO1	US	50	4500			3	5, 17		17				17
	BO2	SL	1000				3	5, 17		17				17
M	SL				50					12 (3h)				16 (4h)
BI	US	50												16 (4h)
DQS	BO1	US	50	4500			3	5, 17			4			17
	BO2	SL	1000				3	5, 17			4			17
M	SL				50									
BI	US	50												12 (3h)

For an explanation of the guidelines represented in this table, refer to the bullet points immediately following Figure 134 on page 173.

The following figure shows the RESET signal scheme and routing guideline for one DIMM per channel topologies.

**Figure 137. Reset scheme for 1DPC DIMM topologies**



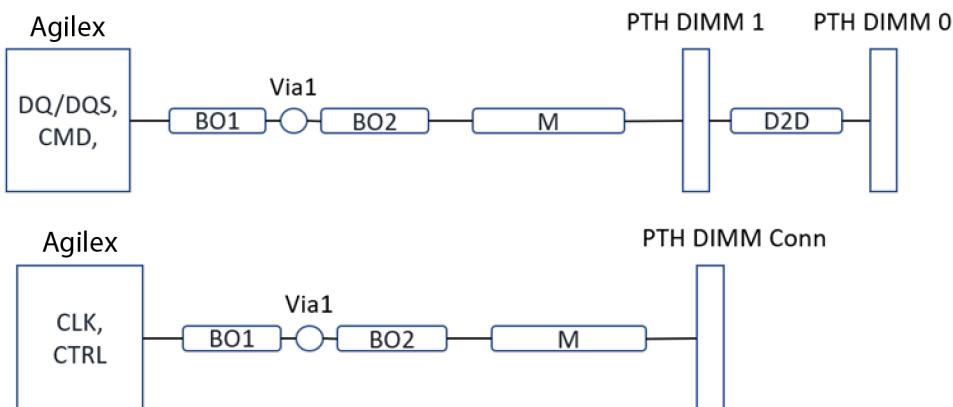
The target impedance for the RESET signal is 50 ohms. The RESET signal shall have at least  $3 \times h$  (where  $h$  stands for trace to nearest reference plane height or distance) spacing to other nearby signals on the same layer. The end-to-end RESET trace length is not limited but shall not exceed 5 inches.

#### 6.5.5.2. Two DIMMs per Channel (2DPC) for UDIMM, RDIMM, and LRDIMM DDR4 Topologies

The interface covers data bytes (DQ/DQS), address signals, command signals (BA, BG, RAS, CAS, WE, ACT, PAR), control signals (CKE, CS, ODT), and clocks (CLK).

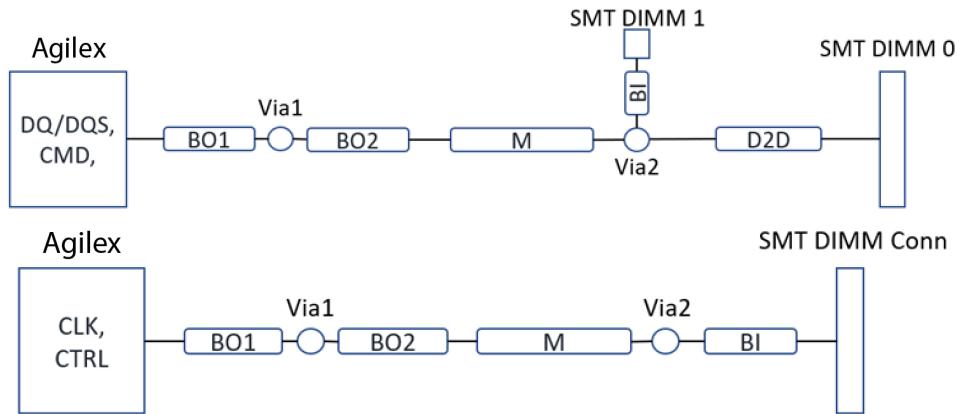
The following figure illustrates the signal connection topology for a PTH type of connector for UDIMM, RDIMM, and LRDIMM topologies.

**Figure 138. Signal connections for DDR4 2DPC DIMM configuration using PTH DIMM connector**



The following figure illustrates the signal connection topology for an SMT type of connector for UDIMM, RDIMM, and LRDIMM topologies.

**Figure 139. Signal connections for DDR4 2DPC DIMM configuration using SMT DIMM connector**



The following table provides specific routing guidelines for two DIMMs per channel in UDIMM, RDIMM, LRDIMM, and SODIMM topologies for all supported signals in the interface.

**Table 110. Specific DDR4 2DPC routing guidelines for UDIMM, RDIMM, LRDIMM, and SODIMM configurations**

Signal Group	Segment	Routing Layer	Max Length (mil)	Total MB Segment	Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), DQS pair to DQ Within DIFF Pair	Trace Spacing (mil), CLK pair to CMD/CTRL/CK E	Channel to Channel Spacing (DQ to DQ between two channels)
CLK	B01	US	50	5000		4	5, 17	5, 17	4			17
	B02	SL	1000			4	5, 17	5, 17	4			17
M	SL				45	4.5		12 (3h)	4			
	BI	US	50			4		12 (3h)	4			
CMD, ALERT	B01	US	50	5000		4	5, 17	5, 17				
	B02	SL	1000			4	5, 17	5, 17				
M	SL				40	5.5	8 (2h)	12 (3h)				
	BI	US	100			4	8 (2h)	12 (3h)				
D2D	SL				PTH: 340 SMT: 400		5.5	8 (2h)	12 (3h)			
	BI	US	50	5000		4	5, 17	5, 17				
CTRL	B01	US	50			4	5, 17	5, 17				
	B02	SL	100			4	5, 17	5, 17				
M	SL				45	4.5	8 (2h)	12 (3h)				
	BI	US	100			4		12 (3h)				
DQ	B01	US	50	5000		3	5, 17		17			17
	B02	SL	1000			3	5, 17		17			17
M	SL				40	5.5	8 (2h)		12 (3h)			16 (4h)
	BI	US	50			4	8 (2h)		12 (3h)			16 (4h)
D2D	SL				PTH: 500 SMT: 400		4	8 (2h)	12 (3h)			16 (4h)
	BI	US	50	5000		3	5, 17		4	17		
DQS	B01	US	50			3	5, 17		4	17		
	B02	SL	1000			3						

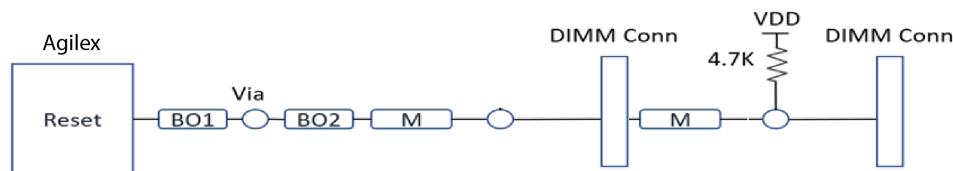
*continued...*

Signal Group	Segment	Routing Layer	Max Length (mil)	Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), CLK pair to DQ	Trace Spacing (mil), Within DIFF Pair	Trace Spacing (mil), CLK pair to CMD/CTRL/CK E	Channel to Channel Spacing (DQ to DQ between two channels )
	M	SL			40	5.5				4	12 (3h)	
	BI	US	50			4						
	D2D	SL	PTH: 500 SMT: 400			4						

For an explanation of the guidelines represented in this table, refer to the bullet points immediately following Figure 134 on page 173.

The following figure shows the RESET signal scheme and routing guideline for two DIMMs per channel topologies.

**Figure 140. Reset scheme for 2DPC DIMM topologies**



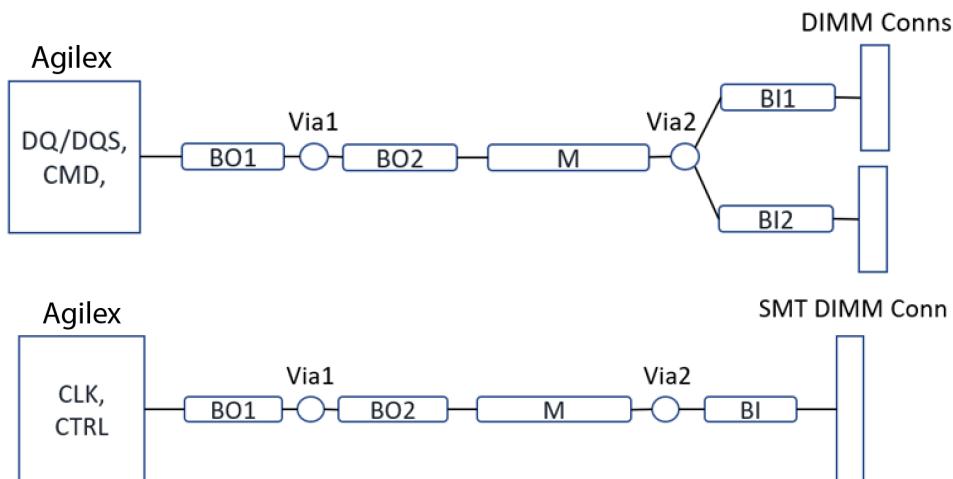
The target impedance for the RESET signal is 50 ohms. The RESET signal shall have at least  $3 \times h$  (where  $h$  stands for trace to nearest reference plane height or distance) spacing to other nearby signals on the same layer. The end-to-end RESET trace length is not limited but shall not exceed 5 inches.

#### 6.5.5.3. Two DIMMs per Channel (2DPC) for SODIMM Topology

The interface covers data bytes (DQ/DQS), address signals, command signals (BA, BG, RAS, CAS, WE, ACT, PAR), control signals (CKE, CS, ODT) and clocks (CLK).

For DDR4 two DIMM per channel (2DPC) designs, Intel recommends placing the two DIMM connectors for a channel on both the top and bottom sides of the board, for best performance.

**Figure 141. Signal connections for DDR4 2DPC SODIMM configuration using SMT DIMM connector**



(In the above figure, CLK, CTRL refers to the per-DIMM signals (mem\_ck, mem\_cke, mem\_odt) and remain as point-to-point connections.)

The following table provides specific routing guidelines for two DIMMs per channel in SODIMM topologies for all supported signals in the interface.

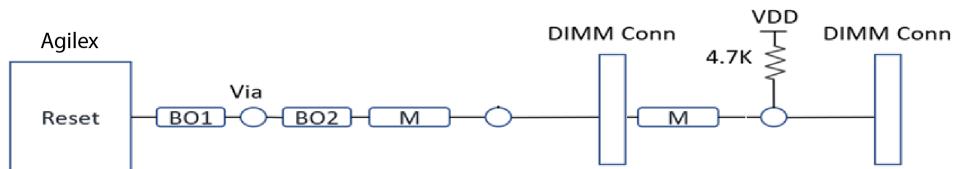
**Table 111. Specific DDR4 2DPC routing guidelines for SODIMM configurations**

Signal Group	Segment	Routing Layer	Max Length (mil)	Target Zse (ohms)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CLK to DQ/DQS	Trace Spacing, S3 (mil): DQ to Nibble	Trace Spacing (mil), DQ Within DIFF Pair	Trace Spacing (mil), DQS pair to DQ	Channel to Channel spacing (DQ to DQ between two channels )
CLK	BO1	US	50	5000	4	5, 17	5, 17		4	17	
	BO2	SL	1000		4	5, 17	5, 17		4	17	
M		SL			45	4.5		12 (3h)	4	12 (3h)	
BI	US	300				4		12, (3h)	4	12 (3h)	
CMD, ALERT	BO1	US	50	5000	4	5, 17	5, 17				
	BO2	SL	1000		4	5, 17	5, 17				
M		SL			40	5.5	8 (2h)	12 (3h)			
BI	US	300				4	8 (2h)	12, (3h)			
CTRL	BO1	US	50	5000		4	5, 17	5, 17			
	BO2	SL	1000			4	5, 17	5, 17			
M		SL			45	4.5	8 (2h)	12 (3h)			
BI	US	300				4		12, (3h)			
DQ	BO1	US	50	5000	3	5, 17			17	17	
	BO2	SL	1000		3	5, 17			17	17	
M		SL			40	5.5	8 (2h)		12 (3h)		16 (4h)
BI	US	300				4	8 (2h)		12 (3h)		16 (4h)
DQS	BO1	US	50	5000		3	5, 17			4	17
	BO2	SL	1000			3	5, 17			4	17
M		SL			40	5.5				4	12 (3h)
BI	US	300					4				

For an explanation of the guidelines represented in this table, refer to the bullet points immediately following Figure 134 on page 173.

The following figure shows the RESET signal scheme and routing guideline for two DIMMs per channel topologies.

**Figure 142. Reset scheme for 2DPC DIMM topologies**



The target impedance for RESET signal is 50 ohms. The RESET signal shall have at least  $3 \times h$  (where  $h$  stands for trace to nearest reference plane height or distance) spacing to other nearby signals on the same layer. The end-to-end RESET trace length is not limited but shall not exceed 5 inches.

#### 6.5.5.4. Skew Matching Guidelines for DIMM Configurations

The guidelines in this topic apply to any DIMM topology, regardless of DIMM type or number of ranks.

Board designers must observe the following guidelines for DDR4 DIMM skew matching:

- Perform skew matching in time (picoseconds) rather than in actual trace length, to better account for via delays when signals are routed on different layers.
- Include both package per-pin skew and PCB delay when performing skew matching.
- Skew (length) matching for the alert signal is not required.

The following table provides skew matching guidelines for DDR4 DIMM topologies.

**Table 112. Skew Matching Guidelines for DDR4 DIMM Topologies**

DIMM Skew Matching Rule	Length in Time (ps)
Length matching between DQS and CLK	-255ps < CLK - DQS < 425ps
Length matching between DQ and DQS within byte	-3.5ps < DQ - DQS < 3.5ps
Length matching between DQS and DQS#	< 1ps
Length matching between CLK and CLK#	< 1ps
Length matching between CLK0 and CLK1	< 8ps
Length matching between CMD/ADDR/CTRL and CLK	-20ps < CLK - CMD/ADDR/CTRL < 20ps
Length matching among CMD/ADDR/CTRL within each channel	< 20ps
Include package length in skew matching for FPGA device with no migration	Required
Include package length in skew matching for FPGA device with migration when all package net length are available	It is recommended to use the final migrated package net length
Include package length in skew matching for FPGA device with migration when all package net length are not available	Not recommended

### 6.5.5.5. Power Delivery Recommendations for the Memory / DIMM Side

This topic describes power distribution network (PDN) design guidelines for one DIMM per channel (1DPC) and two DIMMs per channel (2DPC) memory interfaces at the DIMM/memory side.

**Note:** For information on power distribution network design at the FPGA to meet timing margins, refer to the AG014 PDN design guideline.

In the following table, the number of decoupling capacitors is based on a single channel. If multiple channels are sharing the same power rail at the DIMM, the number of decoupling capacitors at the DIMM must be scaled accordingly.

Physically small decoupling capacitors are recommended to minimize area, inductance, and resistance on the PDN path on the printed circuit board.

**Table 113. Required Decoupling Capacitors on the PCB for the Memory/DIMM Side**

Memory Configuration	Power Domain	Decoupling Location	Quantity × Value (size)
DDR4 1DPC	VDDQ	4 near each side of DIMM connector close to VDDQ pins	8 × 47uF (0805)
		4 near each side of DIMM connector close to VDDQ pins	8 × 1uF (0402)
	VTT	Place capacitor on VTT plane close to DIMM	1 × 47uF (0805)
		Place capacitor on VTT plane close to DIMM	2 × 1uF (0402)
	VPP	Place capacitor close to DIMM	1 × 47uF (0805)
		Place capacitor close to DIMM	1 × 1uF (0402)
	VDDSPD	Place capacitor close to DIMM	1 × 0.1uF (0402)
		Place capacitor close to DIMM	1 × 2.2uF (0402)
DDR4 2DPC	VDDQ	4 near each side of DIMM connector close to VDDQ pins	16 × 47uF (0805)
		4 near each side of DIMM connector close to VDDQ pins	16 × 1uF (0402)
	VTT	Place capacitor on VTT plane close to DIMM	1 × 47uF (0805)
		Place capacitor on VTT plane close to DIMM	4 × 1uF (0402)
	VPP	Place capacitor close to DIMM	2 × 47uF (0805)

*continued...*

Memory Configuration	Power Domain	Decoupling Location	Quantity x Value (size)
	VDDSPD	Place capacitor close to DIMM	2 × 1uF (0402)
		Place capacitor close to DIMM	1 × 0.1uF (0402)
		Place capacitor close to DIMM	1 × 2.2uF (0402)

## 6.5.6. DDR4 Routing Guidelines: Discrete (Component) Topologies

This section discusses two topologies for down-memory configurations: DDR4 single rank  $\times$  8 and DDR4 single rank  $\times$  16 for a 72 bit interface.

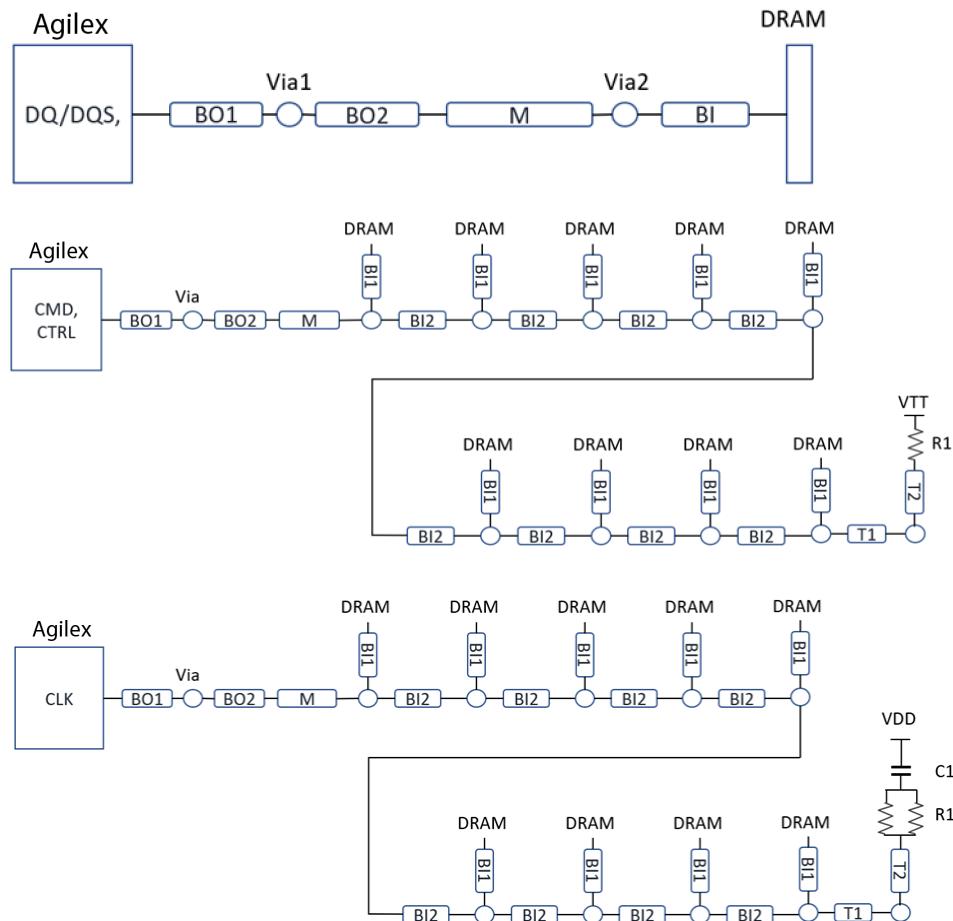
Intel strongly recommends that you perform simulations using extracted PCB models to ensure that component topologies remain robust under all PCB manufacturing tolerances. Also, carefully consider the number of components on the flyby chain, because every additional component on the flyby chain reduces timing margin on the address/command bus. Take care to provide a proper VTT termination voltage network with a reference voltage that feeds back to the VREFCA input of every component on the flyby chain. Intel Agilex 7 F-Series and I-Series FPGA circuitry cannot compensate for discontinuities or trace length mismatches along the flyby chain, or for crosstalk between address/command or DQ signals.

### 6.5.6.1. Single Rank $\times$ 8 Discrete (Component) Topology

Nine memory devices are required to cover 72 bits of data in a single channel, with one rank and  $\times$ 8 memory devices.

The interface covers data bytes (DQ/DQS), address signals, command signals (BA, BG, RAS, CAS, WE, ACT, PAR), control signals (CKE, CS, ODT) and clocks (CLK).

**Figure 143. Signal connections for DDR4 Single Rank × 8 Discrete Topology (9 memory devices to cover 72 bits)**



**Table 114. Specific Routing Guidelines for Single Rank x8 Discrete Memory Topology for All Supported Signals in the Interface**

Signal Group	Segment	Routing Layer	Max Length (mil)	Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), DQS pair to DQ pair	Trace Spacing (mil), CLK pair to CMD/CTRL/CKE	Rtt / Ctt
CLK	B01	US	50	To first DRAM; 4000 To last DRAM: 9600	4	5, 17	17	4	4	17	R1=36Ω C1=10nF
	B02	SL	1000		4	5, 17	17	4	4	17	
M	SL				40	5.5	12 (3h)	4	4	12 (3h)	
	B11	US	50		3		12 (3h)	4	4	12 (3h)	
	B12	SL	700		50	3		12 (3h)	4	12 (3h)	
T1	SL		300			3		12 (3h)	4	12 (3h)	
	T2	US	50			3		12 (3h)	4	12 (3h)	
CMD, CTRL, Alert	B01	US	50	To first DRAM; 4000 To last DRAM: 9600	4	5, 17	17				R1=36Ω
	B02	SL	1000		4	5, 17	17				alert_n requires an external pullup resistor to VDD (1.2V) of approximately 10kΩ.
M	SL				40	5.5	8(2h)	12 (3h)			
	B11	US	50			3	8(2h)	12 (3h)			
	B12	SL	700		50	3	8(2h)	12 (3h)			
T1	SL		300			3	8(2h)	12 (3h)			
	T2	US	50			3	8(2h)	12 (3h)			
DQ	B01	US	50	5000	4	5, 17	17				
	B02	SL	1000		4	5, 17	17				
M	SL				45	4.5	8(2h)	12 (3h)			
	B1	US	50			4	8(2h)	12 (3h)			
DQS	B01	US	50	5000	4			4	17		
	B02	SL	1000			4		4	17		

*continued...*

Signal Group	Segment	Routing Layer	Max Length (mil)	Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CL K to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), DQS pair to DQ	Trace Spacing (mil), CLK pair to CMD/CTRL/CK E	Rtt / Ctt
M	SL			45	4.5				4	12 (3h)	
BT	US	US	50		4				4	12 (3h)	

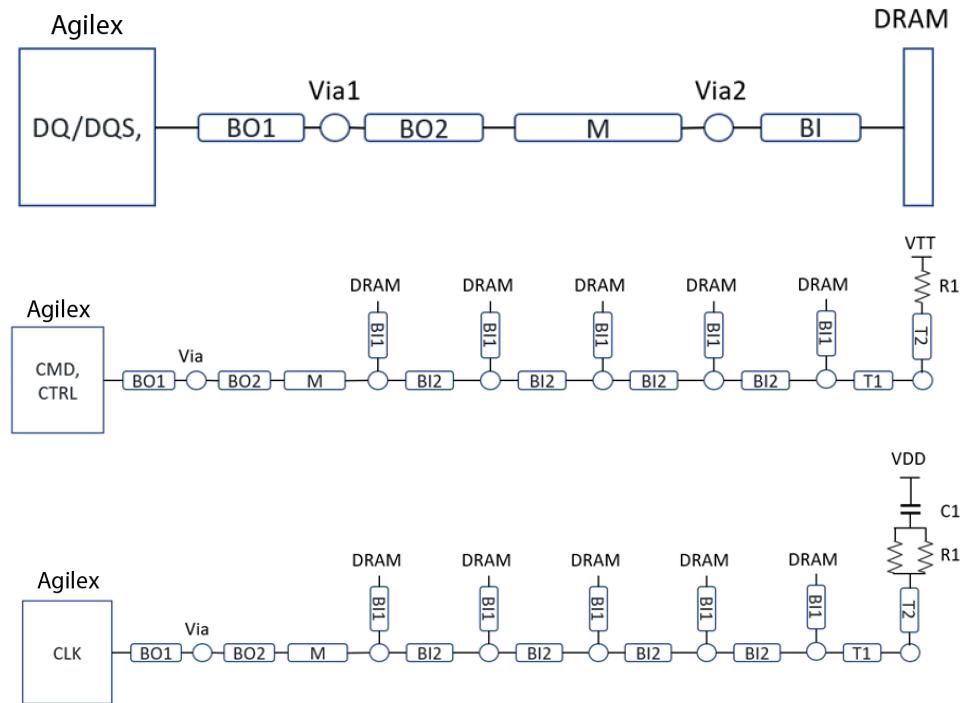
For an explanation of the guidelines represented in this table, refer to the bullet points immediately following Figure 134 on page 173.

### 6.5.6.2. Single Rank x 16 Discrete (Component) Topology

Five memory devices are required to cover 72 bits of data in a single channel, with one rank and  $\times 16$  memory devices.

The interface covers data bytes (DQ/DQS), address signals, command signals (BA, BG, RAS, CAS, WE, ACT, PAR), control signals (CKE, CS, ODT) and clocks (CLK).

**Figure 144. Signal connections for DDR4 Single Rank x 16 Discrete Topology (5 memory devices to cover 72 bits)**



**Table 115. Specific Routing Guidelines for Single Rank x16 Discrete Memory Topology for All Supported Signals in the Interface**

Signal Group	Segment	Routing Layer	Max Length (mil)	Total MB Segment	Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CL K to DQ/DQS	Trace Spacing, S3 (mil): DQ Nibble to Nibble	Trace Spacing (mil), DQS pair to DQ	Trace Spacing (mil), CLK pair to CMD/CTRL/CK E	Rtt / Ctt
CLK	B01	US	50	To first DRAM: 4000. To last DRAM: 6800.	40	4	5, 17	17	4	4	17	R1=36Ω. C1=10nF
	B02	SL	1000		40	4	5, 17	17	4	4	17	
	M	SL			50	5.5		12 (3h)	4		12 (3h)	
	B11	US	50		50	3		12 (3h)	4		12 (3h)	
	B12	SL	700		50	3		12 (3h)	4		12 (3h)	
	T1	SL	300		50	3		12 (3h)	4		12 (3h)	
CMD, CTRL, Alert	T2	US	50		50	3		12 (3h)	4		12 (3h)	
	B01	US	50	To first DRAM: 4000. To last DRAM: 6800.	40	4	5, 17	17				R1=36Ω
	B02	SL	1000		40	5.5	8 (2H)	12 (3h)				alert_n requires an external pullup resistor to VDD (1.2V) of approximately 10kΩ.
	M	SL			50	3	8 (2H)	12 (3h)				
	B11	US	50		50	3	8 (2H)	12 (3h)				
	B12	SL	700		50	3	8 (2H)	12 (3h)				
DQ	T1	SL	300		50	3	8 (2H)	12 (3h)				
	T2	US	50		50	3	8 (2H)	12 (3h)				
	B01	US	50	5000	5000	4	5, 17	17				
	B02	SL	1000		5000	4	5, 17	17				
	M	SL			5000	45	4.5	8 (2H)	12 (3h)			
	B1	US	50		5000	4	8 (2H)	12 (3h)				
DQS	B01	US	50		5000	4			4	17		
	B02	SL	1000		5000	4			4	17		

*continued...*

<b>Signal Group</b>	<b>Segment</b>	<b>Routing Layer</b>	<b>Max Length (mil)</b>	<b>Target Zse (ohm)</b>	<b>Trace Width, W (mil)</b>	<b>Trace Spacing, S1 (mil): Within Group</b>	<b>Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DQS</b>	<b>Trace Spacing, S3 (mil): DQ Nibble to Nibble</b>	<b>Trace Spacing (mil), DQS pair to DQ</b>	<b>Trace Spacing (mil), CLK pair to CMD/CTRL/CK E</b>	<b>Rtt / Ctt</b>
Segment	Total MB	Segment									
M	SL				45	4.5					
BI	US	50				4					

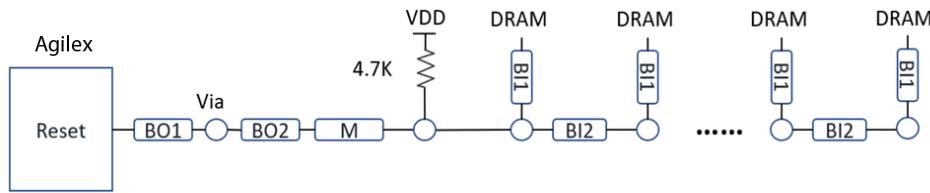
For an explanation of the guidelines represented in this table, refer to the bullet points immediately following Figure 134 on page 173.

### 6.5.6.3. ADDR/CMD Reference Voltage/RESET Signal Routing Guidelines for Single Rank x 8 and R Rank x 16 Discrete (Component) Topologies

The target impedance for the RESET signal is 50 ohms. The RESET signal must have at least  $3 \times h$  (where  $h$  is the distance from the trace to the nearest reference plane) spacing to other nearby signals on the same layer. The end-to-end RESET trace length is not limited but must not exceed 5 inches to the first DRAM.

The following figure shows the RESET routing scheme, which you can apply to both single rank x 8 and single rank x16 topologies.

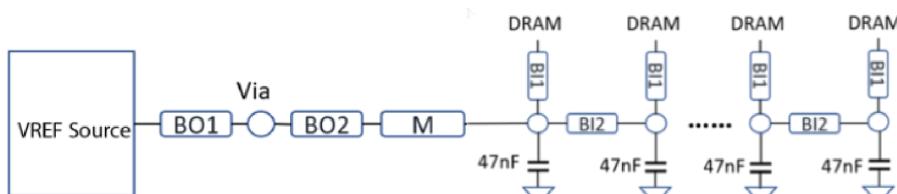
**Figure 145. RESET Scheme for Single Rank x8 and Single Rank x16 Topologies**



The Address/Command reference voltage input (VREF\_CA) must track the VTT regulator output as closely as possible. There are two methods to achieve this:

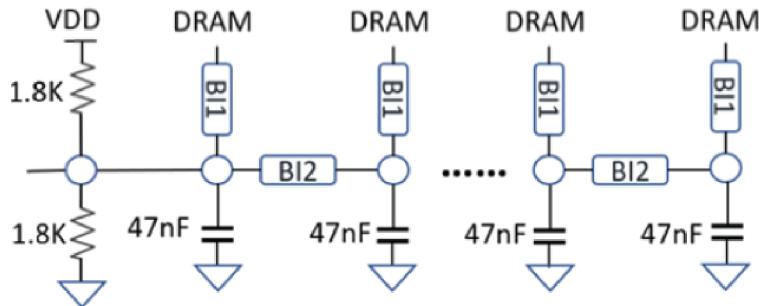
One method is to use a regulator that provides a dedicated tracking voltage reference output that can be connected directly to memory component VREF\_CA inputs, as shown in the figure below.

**Figure 147. VTT Regulator Supplies VREF\_CA Output**



A second method is to create a voltage divider using precision resistors. Place the resistor network in a location that is likely to track IR losses on the VDD supply due to memory loading (that is, close to the VTT regulator or memory components, rather than next to the VDD regulator output). The following figure illustrates this configuration.

**Figure 148. Resistor Divider Provides the VREF\_CA Signal**



Intel recommends using a PCB trace width of at least 10 mils for VREF\_CA routing. The VREF\_CA signal must have at least  $3 \times h$  spacing (where  $h$  is the distance or height from the trace to the nearest reference plane) to other nearby signals on the same layer.

#### 6.5.6.4. Skew Matching Guidelines for DDR4 Discrete Configurations

This topic describes skew matching guidelines for single rank x 8 and single rank x 16 topologies.

Observe the following rules when skew matching DDR4 discrete configurations:

- Perform skew matching in time (picoseconds) rather than in actual trace length, to better account for via delays when signals are routed on different layers.
- Include both package per-pin skew and PCB delay when performing skew matching.
- Skew (length) matching for the alert signal is not required.

The following table provides skew matching guidelines for DDR4 down-memory topologies.

**Table 116. Skew Matching Guidelines for DDR4 Discrete Topologies**

DDR4 Device-down Length Matching Rules	Length Matching in Time (ps)
Skew matching between DQS and CLK	-85ps < CLK - DQS < 935ps
Skew matching between DQ and DQS within byte	-3.5ps < DQ - DQS < 3.5ps
Skew matching between DQS and DQS#	< 1ps
Skew matching between CLK and CLK#	< 1ps
Skew matching between CMD/ADDR/CTRL and Clock	-20ps < CLK - CMD/ADDR/CTRL < 20ps
Skew matching among CMD/ADDR/CTRL within each channel	< 20ps
Include package length in skew matching for FPGA device with no migration	Required
Include package length in skew matching for FPGA device with migration when all package net length are available	It is recommended to use the final migrated package net length
Include package length in skew matching for FPGA device with migration when all package net length are not available	Not recommended

### 6.5.6.5. Power Delivery Recommendations for DDR4 Discrete Configurations

This topic describes power distribution network (PDN) design guidelines for the memory side in discrete topologies.

**Note:** For information on power distribution network design at the FPGA to meet timing margins, refer to the Intel Agilex 7 F-Series and I-Series PDN design guidelines.

In the following table, the number of decoupling capacitors is based on a single channel. If multiple channels are sharing the same power rail, the number of decoupling capacitors at the memories must be scaled accordingly for all channels.

Physically small decoupling capacitors are recommended to minimize area, inductance, and resistance on the PDN path on the printed circuit board.

**Table 117. Required Decoupling Capacitors on the PCB for the Memory Side**

Memory Configuration	Power Domain	Decoupling Location	Quantity × Value (size)
Discrete (Component) Single Rank x8	VDDQ/VDD shorted	4 near each x8 DRAM device	36 x 1uF (0402)
		Distribute around DRAM devices	9 x 10uF (0603)
	VPP	2 near each x8 DRAM device	18 x 1uF (0402)
		Distribute around DRAM devices	5 x 10uF (0603)
	VTT	Place near Rtt (termination resistors)	16 x 1uF (0402)
		Place near Rtt (termination resistors)	4 x 10uF (0603)
	VDDQ/VDD shorted	4 near each x16 DRAM device	18 x 1uF (0402)
		Distribute around DRAM devices	5 x 10uF (0603)
Discrete (Component) Single Rank x16	VPP	2 near each x16 DRAM device	10 x 1uF (0402)
		Distribute around DRAM devices	3 x 10uF (0603)
	VTT	Place near Rtt (termination resistors)	8 x 1uF (0402)
		Place near Rtt (termination resistors)	2 x 10uF (0603)

### 6.5.7. Intel Agilex 7 F-Series and I-Series EMIF Pin Swapping Guidelines

In Intel Agilex 7 F-Series and I-Series devices, EMIF pin swapping is allowed under certain conditions.

An IO12 lane in an EMIF data byte includes 12 signal pins (pins 0,1,2,3,4,5,6,7,8,9,10,11) at the package level. These 12 x I/O pins are arranged into 6 groups of 2 pins each, called *pairs* (pair 0 for pins 0/1, pair 1 for pins 2/3, pair 2 for pins 4/5, pair 3 for pins 6/7, pair 4 for pins 8/9, and pair 5 for pins 10/11).

### DDR4 interface x 8 data lane

The following are EMIF I/O pin swapping restrictions applicable to a DDR4 interface  $\times 8$  data lane:

- One-byte data lane must be assigned for each IO12 lane, where the byte lane covers DQ [0:7], DQSp/DQSn and DBIn.
- DQSp must go to pin 4 in IO12 pins.
- DQSn must go to pin 5 in IO12 pins.
- DBIn must go to pin 6 in IO12 pins. If the interface does not use the DBIn pin, this pin 6 in IO12 lane must remain unconnected.
- Pin 7 in IO12 lane remains unconnected. Intel recommends that you connect this pin 7 to the TDQS dummy load of the memory component and route it as a differential trace along with DBIn (pin 6). This facilitates  $\times 4$  or  $\times 8$  data interoperability in DIMMs configuration.
- You can connect data byte (DQ [0:7]) to any pins [0,1,2,3,8,9,10,11] in IO12 lane. Any permutation within selected pins is permitted.

### DDR4 interface x 4 data lane

Intel Agilex 7 F-Series and I-Series devices allow you to swap data pins within memory interfaces under certain conditions to simplify PCB routing. This table lists the swapping rules for DDR4 x4 and x8 interfaces. (Note that the rules for pin swapping for Intel Agilex 7 F-Series and I-Series devices are stricter than the rules for previous 10-series device families.)

An IO12 lane in an external memory interface consists of 12 signal pins, denoted 0-11. For DDR4 x8 interfaces, two pins are reserved for DQS-P and DQS-N signals, one pin is reserved for the optional DM/DBI signal, one pin must be reserved, and the remaining eight pins are for DQ signals. The following table lists the supported pin functionality and the groups of pins that may be swapped amongst each other. Pins belonging to the same swap group may be freely interchanged with each other.

**Table 118.**

Pin Index within IO12	DDR4 x8 Data Lane Function	Swap Considerations
0	DQ Pin	Swap Group "A"
1	DQ Pin	Swap Group "A"
2	DQ Pin	Swap Group "A"
3	DQ Pin	Swap Group "A"
4	DQS-P Pin	Fixed Location (not swappable)
5	DQS-N Pin	Fixed Location (not swappable)
6	DM/DBI Pin	Fixed Location (not swappable)
7	Unused	Fixed Location (not swappable)
8	DQ Pin	Swap Group "A"
9	DQ Pin	Swap Group "A"
10	DQ Pin	Swap Group "A"
11	DQ Pin	Swap Group "A"

For DDR4 x4 interfaces, two nibbles must be packed into the same IO12 lane. Four pins are reserved for DQS-P and DQS-N signals and the remaining eight pins are used to implement the DQ signals. The IO12 lane is divided into upper and lower halves to accommodate each nibble. Signals belonging to one nibble cannot be swapped with signals belonging to the other nibble. DQ signals within a nibble swap group may be swapped with each other. Entire nibbles—that is, nibble 0 and nibble 1—may also be swapped with each other provided the DQS pin functionality transfers to the correct pin locations. However, this process is not recommended for JEDEC-compliant DIMM interfaces, as it prohibits the interoperability between DIMMs constructed with x4 components and DIMMs constructed with x8 components as described in [x4 DIMM Implementation](#).

The following table lists the supported pin functionality in x4 mode and the pins that may be swapped with each other.

**Table 119.**

Pin Index within IO12	DDR4 x4 Data Lane Function	Swap Considerations	
0	DQ Pin (lower nibble)	Swap Group "A"	Nibble 0
1	DQ Pin (lower nibble)	Swap Group "A"	
2	DQ Pin (lower nibble)	Swap Group "A"	
3	DQ Pin (lower nibble)	Swap Group "A"	
4	DQS-P Pin (lower nibble)	Fixed Location (not swappable)	
5	DQS-N Pin (lower nibble)	Fixed Location (not swappable)	
6	DQS-P Pin (upper nibble)	Fixed Location (not swappable)	Nibble 1
7	DQS-N Pin (upper nibble)	Fixed Location (not swappable)	
8	DQ Pin (upper nibble)	Swap Group "B"	
9	DQ Pin (upper nibble)	Swap Group "B"	
10	DQ Pin (upper nibble)	Swap Group "B"	
11	DQ Pin (upper nibble)	Swap Group "B"	

- Nibble 1 must correspond to DQS[17:9] on a physical JEDEC-compliant DIMM for x4/x8 interoperability.
- Nibbles 0 and 1 must follow the same skew matching rules among all 12 signals in the IO12 lane as are specified for a x8-based DQS group.

**Note:**

- Although the current version of the Intel Quartus Prime software may not enforce all of the rules listed in the above table, be aware that all of these rules may be enforced in later versions of the software.
  - At present, the Intel Quartus Prime software checks the following:
    - Address and command pin placement, per the *Intel Agilex 7 F-Series and I-Series External Memory Interface Pin Information* file, which is available here: [Pin-Out Files for Intel FPGA Devices](#).
    - For x8, the Intel Quartus Prime software checks the following:
      - DQS p/n are on pin index 4 and pin index 5 in an I/O lane.
      - DM/DBI is on pin index 6.
      - DQ[x] are on pin indices [11:8] and [3:0].
    - For x4, the Intel Quartus Prime software checks the following:
      - DQS p/n on pin index 4 and pin index 5 and associated DQs are within the corresponding IO12 lane.
      - DQS p/n on pin index 6 and pin index 7 and associated DQs are within the corresponding IO12 lane.
- You are responsible for ensuring that these conditions are met.
- The Intel Quartus Prime software does not currently check whether DQ pins associated with the lower nibble DQS are actually placed in pin[3:0] or whether DQ pins associated with the upper nibble DQS are actually placed in pin[11:8].

### **DDR4 interface x 4 or x8 A/C and CLK lane**

Address and command and control signals in a bank cannot be swapped.

CLKp/n bits (the p and n lanes) and DQS bits cannot be swapped with each other.

## 7. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – QDR-IV Support

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Agilex 7 F-Series and I-Series FPGA external memory interface IP for QDR-IV.

### 7.1. Intel Agilex 7 FPGA EMIF IP 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.

**Note:** Also in this section are the parameters of the *External Memory Interfaces Intel Calibration IP*, which are included as part of the *Diagnostics* topic.

#### 7.1.1. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: General

**Table 121. Group: General / Interface**

Display Name	Description
<b>Configuration</b>	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 122. Group: General / Clocks**

Display Name	Description
<b>Memory clock frequency</b>	Specifies the <b>operating frequency</b> of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the <b>Memory</b> tab and the memory timing parameters on the <b>Mem Timing</b> tab. (Identifier: PHY_QDR4_MEM_CLK_FREQ_MHZ)
<b>Use recommended PLL reference clock frequency</b>	Specifies that the PLL reference clock frequency is automatically calculated for best performance. <i>If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.</i> (Identifier: PHY_QDR4_DEFAULT_REF_CLK_FREQ)
<b>PLL reference clock frequency</b>	This parameter tells the IP what PLL reference clock frequency the user is supplying. 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 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)

*continued...*

Display Name	Description
<b>PLL reference clock jitter</b>	Specifies the <b>peak-to-peak phase jitter</b> on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 20ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_QDR4_REF_CLK_JITTER_PS)
<b>Clock rate of user logic</b>	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)
<b>Specify additional core clocks based on existing PLL</b>	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter <b>provides an alternative clock-generation mechanism for when your design exhausts available PLL resources</b> . 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). <i>You must follow proper clock-domain-crossing techniques when transferring data between clock domains.</i> (Identifier: PLL_ADD_EXTRA_CLKS)

**Table 123. Group: General / Clocks / Additional Core Clocks**

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

**Table 124. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_0**

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

**Table 125. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_1**

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

**Table 126. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_2**

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

**Table 127. Group: General / Clocks / Additional Core Clocks / pll\_extra\_clk\_3**

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

### 7.1.2. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Memory

**Table 128. Group: Memory / Topology**

Display Name	Description
<b>DQ width per device</b>	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)
<b>Address width</b>	Number of address pins. (Identifier: MEM_QDR4_ADDR_WIDTH)
<b>Memory Type</b>	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 129. Group: Memory / Configuration Register Settings**

Display Name	Description
<b>Address bus inversion</b>	Enable address bus inversion. AINV are all active high at memory device. (Identifier: MEM_QDR4_ADDR_INV_ENA)
<b>Data bus inversion</b>	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 are always driven to 0. (Identifier: MEM_QDR4_DATA_INV_ENA)
<b>ODT (Clock)</b>	Determines the configuration register setting that controls the clock ODT setting. (Identifier: MEM_QDR4_CK_ODT_MODE_ENUM)
<b>ODT (Address/Command)</b>	Determines the configuration register setting that controls the address/command ODT setting. (Identifier: MEM_QDR4_AC_ODT_MODE_ENUM)
<b>ODT (Data)</b>	Determines the configuration register setting that controls the data ODT setting. (Identifier: MEM_QDR4_DATA_ODT_MODE_ENUM)
<b>Output drive (pull-up)</b>	Determines the configuration register setting that controls the pull-up output drive setting. (Identifier: MEM_QDR4_PU_OUTPUT_DRIVE_MODE_ENUM)
<b>Output drive (pull-down)</b>	Determines the configuration register setting that controls the pull-down output drive setting. (Identifier: MEM_QDR4_PD_OUTPUT_DRIVE_MODE_ENUM)

### 7.1.3. Intel Agilex 7 F-Series and I-Series 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 130. Group: FPGA I/O / FPGA I/O Settings**

Display Name	Description
<b>Voltage</b>	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)
<b>Use default I/O settings</b>	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. <i>To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results.</i> (Identifier: PHY_QDR4_DEFAULT_IO)

**Table 131. Group: FPGA I/O / FPGA I/O Settings / Address/Command**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_QDR4_USER_AC_MODE_ENUM)
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_QDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode of the address/command output pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the address and command signals.</i> (Identifier: PHY_QDR4_USER_AC_DEEMPHASIS_ENUM)

**Table 132. Group: FPGA I/O / FPGA I/O Settings / Memory Clock**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_QDR4_USER_CK_MODE_ENUM)
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data</i>

*continued...*

Display Name	Description
	<i>output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_QDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode for the memory clock pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the memory clock signals.</i> (Identifier: PHY_QDR4_USER_CK_DEEMPHASIS_ENUM)

**Table 133. Group: FPGA I/O / FPGA I/O Settings / Data Bus**

Display Name	Description
<b>I/O standard</b>	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)
<b>Output mode</b>	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_QDR4_USER_DATA_OUT_MODE_ENUM)
<b>Slew rate</b>	Specifies the slew rate of the memory bus data output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. <i>Perform board simulations to determine the data output slew rate that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_QDR4_USER_DATA_OUT_SLEW_RATE_ENUM)
<b>Deemphasis mode</b>	Specifies the deemphasis mode for the memory bus data output pins. The deemphasis mode controls how quickly individual driver stages of the output buffer are enabled. Adjusting this setting can help control voltage overshoot at the receiver. <i>Perform board simulations to determine the deemphasis setting that provides the best eye opening for the data bus pins measured at the memory receiver.</i> (Identifier: PHY_QDR4_USER_DATA_OUT_DEEMPHASIS_ENUM)
<b>Input mode</b>	This parameter allows you to change the input termination settings for the selected I/O standard. <i>Perform board simulation with IBIS models to determine the best settings for your design.</i> (Identifier: PHY_QDR4_USER_DATA_IN_MODE_ENUM)
<b>Initial Vrefin</b>	Specifies the <b>initial value for the reference voltage on the data pins (Vrefin)</b> . 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 <b>skip Vref calibration (Diagnostics tab)</b> , this is the value that is used as the Vref for the interface. (Identifier: PHY_QDR4_USER_STARTING_VREFIN)

**Table 134. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs**

Display Name	Description
<b>PLL reference clock I/O standard</b>	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_QDR4_USER_PLL_REF_CLK_IO_STD_ENUM)
<b>RZQ I/O standard</b>	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_QDR4_USER_RZQ_IO_STD_ENUM)

### 7.1.4. Intel Agilex 7 F-Series and I-Series 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 135. Group: Mem Timing**

Display Name	Description
<b>Speed bin</b>	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)
<b>tISH</b>	tISH provides the <b>setup/hold window requirement for the entire data bus (DK or DINV) in all the data groups with respect to the DK clock</b> . After deskew calibration, this parameter describes the intersection window for all the individual data bus signals setup/hold margins. (Identifier: MEM_QDR4_TISH_PS)
<b>tQKQ_max</b>	tQKQ_max describes the <b>maximum skew</b> between the <b>read strobe (QK)</b> clock edge to the <b>data bus (DQ/DINV)</b> edge. (Identifier: MEM_QDR4_TQKQ_MAX_PS)
<b>tQH</b>	tQH specifies the <b>output hold time for the DQ/DINV in relation to QK</b> . (Identifier: MEM_QDR4_TQH_CYC)
<b>tCKDK_max</b>	tCKDK_max refers to the <b>maximum skew</b> from the <b>memory clock (CK)</b> to the <b>write strobe (DK)</b> . (Identifier: MEM_QDR4_TCKDK_MAX_PS)
<b>tCKDK_min</b>	tCKDK_min refers to the <b>minimum skew</b> from the <b>memory clock (CK)</b> to the <b>write strobe (DK)</b> . (Identifier: MEM_QDR4_TCKDK_MIN_PS)
<b>tCKQK_max</b>	tCKQK_max refers to the <b>maximum skew</b> from the <b>memory clock (CK)</b> to the <b>read strobe (QK)</b> . (Identifier: MEM_QDR4_TCKQK_MAX_PS)
<b>tASH</b>	tASH provides the <b>setup/hold window requirement for the address bus in relation to the CK clock</b> . 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)
<b>tCSH</b>	tCSH provides the <b>setup/hold window requirement for the control bus (LD#, RW#) in relation to the CK clock</b> . 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)

### 7.1.5. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Controller

**Table 136. Group: Controller**

Display Name	Description
<b>Maximum Avalon-MM burst length</b>	Specifies the maximum burst length on the Avalon-MM bus. This is used to configure the FIFOs to be able to manage the maximum data burst. <b>More core logic is required for an increase in FIFO length.</b> (Identifier: CTRL_QDR4_AVL_MAX_BURST_COUNT)
<b>Generate power-of-2 data bus widths for Qsys</b>	<b>If enabled, the Avalon data bus width is rounded down to the nearest power-of-2.</b> 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

*continued...*

Display Name	Description
	<p>if you plan to connect the memory interface to Platform Designer interconnect components that require the data bus and symbol width to be a power-of-2. <b>If this option is enabled, you cannot utilize the full density of the memory device.</b></p> <p>For example, in x36 data width upon selecting this parameter, defines the Avalon data bus to 256-bit. This ignores the upper 4-bit of data width. (Identifier: CTRL_QDR4_AVL_ENABLE_POWER_OF_TWO_BUS)</p>
<b>Additional read-after-write turnaround time</b>	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)
<b>Additional write-after-read turnaround time</b>	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)

### 7.1.6. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Diagnostics

**Table 137. Group: Diagnostics / Simulation Options**

Display Name	Description
<b>Calibration mode</b>	<p>Specifies whether to <b>skip memory interface calibration</b> during simulation, or to <b>simulate the full calibration</b> process.</p> <p>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.</p> <p><i>If you enable this parameter, the interface still performs some memory initialization before starting normal operations. Abstract PHY is supported with skip calibration.</i></p> <p>(Identifier: DIAG_QDR4_SIM_CAL_MODE_ENUM)</p>

**Table 138. Group: Diagnostics / Calibration Debug Options**

Display Name	Description
<b>Skip VREF_in calibration</b>	Specifies to skip the VREF stage of calibration. <b>Enable this parameter for debug purposes only</b> ; generally, you should include the VREF calibration stage during normal operation. (Identifier: DIAG_QDR4_SKIP_VREF_CAL)

**Table 139. Group: Diagnostics / Example Design**

Display Name	Description
<b>Enable In-System-Sources-and-Probes</b>	Enables In-System-Sources-and-Probes in the example design for <i>common debug signals, such as calibration status or example traffic generator per-bit status</i> . 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 140. Group: Diagnostics / Performance**

Display Name	Description
<b>Efficiency Monitor Mode</b>	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 141. Group: Diagnostics / Miscellaneous**

Display Name	Description
<b>Export PLL lock signal</b>	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

### 7.1.7. Intel Agilex 7 F-Series and I-Series EMIF IP QDR-IV Parameters: Example Designs

**Table 143. Group: Example Designs / Example Design with Multi-IPs**

Display Name	Description
<b>Simulation</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, simulation file sets are not created.</i> Instead, the output directory contains the ed_sim.qsys file which holds Platform Designer 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 <b>stored in the /sim sub-directory</b> . (Identifier: EX_DESIGN_GUI_QDR4_GEN_SIM)
<b>Synthesis</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, synthesis file sets are not created.</i> Instead, the output directory contains the ed_synth.qsys file which holds Platform Designer 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 <b>stored in the /qii sub-directory</b> . (Identifier: EX_DESIGN_GUI_QDR4_GEN_SYNTH)
<b>Signal Integrity</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary collateral for performing Signal Integrity analysis with a third-party analog simulation tool. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, board simulation collateral is not created.</i> The generated collateral is <b>stored in the /bsi sub-directory</b> . Note that this option is only supported for selected memory protocols on the Intel Agilex 7 family. (Identifier: EX_DESIGN_GUI_QDR4_GEN_BSI)
<b>Spyglass CDC</b>	Specifies that the ' <b>Generate Example Design</b> ' button create all necessary files for performing CDC analysis with Spyglass. Expect a short additional delay as the file set is created. <i>If you do not enable this parameter, CDC file sets simulation are not created.</i> The generated collateral is <b>stored in the /cdc sub-directory</b> . Note that this option is only supported for selected memory protocols on the Stratix 10 family. (Identifier: EX_DESIGN_GUI_QDR4_GEN_CDC)
<b>Simulation HDL format</b>	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_QDR4_HDL_FORMAT)
<b>Number of IPs</b>	Specifies the number of EMIF IPs to instantiate in the example designs. All the IPs have individual TGs and can be connected to one of the available Cal-IPs.

*continued...*

Display Name	Description
<b>EMIF ID</b>	Specifies the number of EMIF IP that you want in the Multi-IP design. Depending on the size of the EMIF interface, you can have up to 16 EMIF interfaces.
<b>CAL-IP</b>	Specifies the Calibration IP to which a given EMIF should connect in the example design. All the EMIF IPs must be connected to one of the available Cal-IPs. There are two Calibration IPs on the device. Each of the EMIF IP must be connected to either of the two Cal-IPs.
<b>Capture</b>	The capture button allows you to take a capture of the current EMIF IP settings and apply to given EMIF ID interface.

**Table 144. Group: Example Designs / Target Development Kit**

Display Name	Description
<b>Select board</b>	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 do 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 <b>File-&gt;Save as</b> . (Identifier: EX_DESIGN_GUI_QDR4_TARGET_DEV_KIT)

## 7.2. Intel Agilex 7 F-Series and I-Series External Memory Interfaces

### Intel Calibration IP Parameters

The following parameters are found in the *Intel Agilex 7 External Memory Interfaces Intel Calibration IP*.

**Table 145. Group: Calibration and Debug**

Display Name	Description
<b>Number of Calibration Interfaces</b>	Specifies the number of calbus interfaces to connect to the EMIF calibration IP (Identifier: NUM_CALBUS_INTERFACE)
<b>Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>	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_EXPORT_SEQ_AVALON_SLAVE)

**Table 146. Group: Simulation**

Display Name	Description
<b>Calibration mode for simulation</b>	<p>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.</p> <p>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_SIM_CAL_MODE_ENUM)</p>
<b>Show verbose simulation debug messages</b>	<p>This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_SIM_VERBOSE)</p>

## 7.3. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP 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
- RZQ pins
- Other FPGA resources—for example, core fabric logic, and debug interfaces

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

### 7.3.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Interface Pins

Any I/O banks that do not support transceiver operations in Intel Agilex 7 F-Series and I-Series FPGAs support external memory interfaces. However, DQS (data strobe or data clock) and DQ (data) pins are listed 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 pin table for the actual locations of the DQS and DQ pins.

You can find the pin tables at the following location: <https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html>.

*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.

### 7.3.1.1. 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.intel.com](http://www.intel.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 96 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.

### 7.3.1.2. 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 Agilex 7 F-Series and I-Series devices, consult the EMIF Device Selector on [www.intel.com](http://www.intel.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*.

## 7.3.2. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Resources

The Intel Agilex 7 F-Series and I-Series FPGA memory interface IP uses several FPGA resources to implement the memory interface.

### 7.3.2.1. OCT

You require an OCT calibration block if you are using an Intel Agilex 7 F-Series and I-Series FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. There are two OCT blocks in an I/O bank, one for each sub-bank.

You must observe the following requirements when using OCT blocks:

- The I/O bank where you place the OCT calibration block must use the same  $V_{CCIO\_PIO}$  voltage as the memory interface.
- The OCT calibration block uses a single fixed  $R_{ZQ}$ . You must ensure that an external termination resistor is connected to the correct pin for a given OCT block.

### 7.3.2.2. 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.

## 7.3.3. Pin Guidelines for Intel Agilex 7 F-Series and I-Series FPGA EMIF IP

The Intel Agilex 7 F-Series and I-Series FPGA contains I/O banks on the top and bottom edges of the device, which can be used by external memory interfaces.

Intel Agilex 7 F-Series and I-Series FPGA I/O banks contain 96 I/O pins. Each bank is divided into two sub-banks with 48 I/O pins in each. Sub-banks are further divided into four I/O lanes, where each I/O lane is a group of twelve I/O ports.

The I/O bank, I/O lane, and pairing pin for every physical I/O pin can be uniquely identified by the following naming convention in the device pin table:

- The I/O pins in a bank are represented as P#X#Y#, where:
  - P# represents the pin number in a bank. It ranges from P0 to P95, for 96 pins in a bank.
  - X# represents the bank number on a given edge of the device. X0 is the farthest bank from the zipper.
  - Y# represents the top or bottom edge of the device. Y0 and Y1 refer to the I/O banks on the bottom and top edge, respectively.
- Because an IO96 bank comprises two IO48 sub-banks, all pins with P# value less than 48 (P# <48) belong to the same I/O sub-bank. All other pins belong to the second IO48 sub-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 one of I/O lanes 1, 2, 3, or 4, respectively.
- To determine whether I/O banks are adjacent, you can refer to the sub-bank-ordering figures for your device family in the [Architecture: I/O Bank](#) topic, to the [Typical Intel Agilex 7 F-Series and I-Series FPGA with all Banks Bonded Out](#) figure, and to the I/O Pin Count Tables located in the [Intel Agilex 7 F-Series and I-Series General Purpose I/O User Guide](#).

In general, you can assume that I/O banks are adjacent within an I/O edge, unless the I/O bank is not bonded out on the package (indicated by the presence of the " - " symbol in the I/O table), or if the I/O bank does not contain 96 pins, indicating that it is only partially bonded out. If an I/O bank is not fully bonded out in a particular device, it cannot be included within the span of sub-banks for a larger external memory interface. In all cases, you should use the Intel Quartus Prime software to verify that your usage can be implemented.

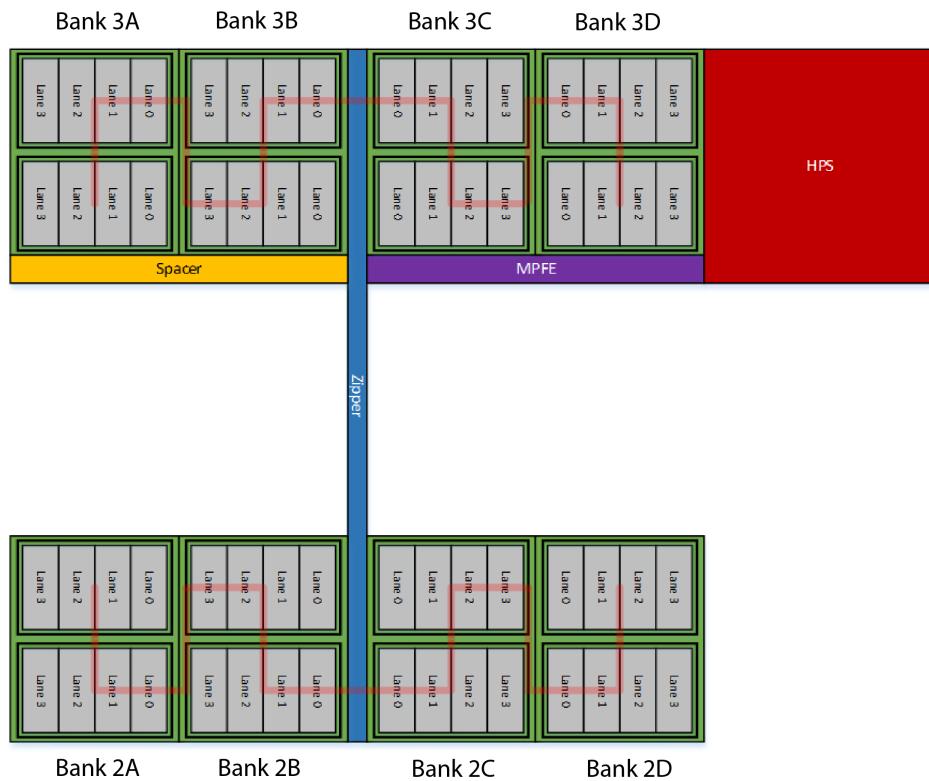
- The pairing pin for an I/O pin is in the same I/O bank. You can identify the pairing pin by adding 1 to its *Index Within I/O Bank* number (if it is an even number), or by subtracting 1 from its *Index Within I/O Bank* number (if it is an odd number).

### 7.3.3.1. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP Banks

Before you select pins for your Intel Agilex 7 F-Series and I-Series FPGA external memory interface, it is important that you understand how banks and sub-banks are grouped together to form a single interface.

The following diagram illustrates a typical Intel Agilex 7 F-Series and I-Series FPGA with all banks bonded out to pins.

**Figure 149. Typical Intel Agilex 7 F-Series and I-Series FPGA with all Banks Bonded Out**



In the above diagram, the group of 4 lanes in the top-left corner (the top sub-bank in bank 3A), denotes the IO48 block that is used for test mode and AVST configuration. If all I/O lanes in this sub-bank are used for configuration, then this bank cannot be used for the external memory interface. Similarly, the bottom sub-bank in bank 3A could not be used for the external memory interface either, because all the I/O sub-banks in a given interface must be contiguous.

**The red line in the above diagram denotes the chaining order of the sub-banks to form an external memory interface. The chaining order flips when crossing the zipper.**

For additional details, refer to the [Intel Agilex 7 FPGA EMIF IP Product Architecture](#) chapter.

### 7.3.3.2. General Guidelines

You should follow the recommended guidelines when performing pin placement for all external memory interface pins targeting Intel Agilex 7 F-Series and I-Series devices, whether you are using the hard memory controller or your own solution.

*Note:*

- EMIF IP pin-out requirements for the Intel Agilex 7 F-Series and I-Series 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 Pro Edition IP file (.qip), based on the IP configuration.
- QDR-IV is not supported with HPS.

Observe the following general guidelines when placing pins for your Intel Agilex 7 F-Series and I-Series external memory interface:

1. Ensure that the pins of a single external memory interface reside on the same edge I/O.
2. An external memory interface can occupy one or more banks on the same edge. When an interface must occupy multiple banks, ensure that those banks are adjacent to one another.
  - If an I/O bank is shared between two interfaces—meaning that two sub-banks belong to two different EMIF interfaces—then both the interfaces must share the same voltage.
  - Sharing of I/O lanes within a sub-bank for two different EMIF interfaces is not permitted; I/O lanes within a sub-bank can be assigned to one EMIF interface only.
3. Any pin in the same bank that is not used by an external memory interface may not be available for use as a general purpose I/O pin:
  - For fabric EMIF, unused pins in an I/O lane assigned to an EMIF interface cannot be used as general-purpose I/O pins. In the same sub-bank, pins in an I/O lane that is not assigned to an EMIF interface, can be used as general-purpose I/O pins.
  - For HPS EMIF, unused pins in an I/O lane assigned to an EMIF interface cannot be used as general-purpose I/O pins. In the same sub-bank, pins in an I/O lane that is not assigned to an EMIF interface cannot be used as general-purpose I/O pins either. Refer to [Restrictions on I/O Bank Usage for Intel Agilex 7 EMIF IP with HPS](#) for more information.
4. All address and command pins and their associated clock pins (CK and CK#) must reside within a single sub-bank. The sub-bank containing the address and command pins is identified as the address and command sub-bank.
5. To minimize latency, when the interface uses more than two sub-banks, you must select the center sub-bank as the address and command sub-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 Agilex 7 External Memory Interface Pin Information* file, which is available here: [Pin-Out Files for Intel FPGA Devices](#).
7. An unused I/O lane in the address and command sub-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.
10. One of the sub-banks in the device (typically the sub-bank within corner bank 3A) may not be available if you use certain device configuration schemes. For some schemes, there may be an I/O lane available for EMIF data group.

- AVST-8 – This is contained entirely within the SDM, therefore all lanes of sub-bank 3A can be used by the external memory interface.
- AVST-32 – Lanes 0, 1, 2, and 3 are all effectively occupied and are not usable by the external memory interface.
- AVST-16 – Lanes 0, 1, and 3 are not usable by the external memory interface. However, lane 2 contains SDM\_MISSION\_DATA[25:16]. If SDM\_MISSION\_DATA[25:16] is not required for AVSTx16, then Lane 2 is available for use by the external memory interface.

11. Sub-banks and I/O banks cannot be shared by two different memory interfaces.

### 7.3.3.3. 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** parameters in the **Configuration Register Settings** section of the **Memory** tab in the parameter editor. The commands and addresses must meet the memory address and command setup (tAS, tCS) and hold (tAH, tCH) time requirements.

### 7.3.3.4. 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 DK<sub>x</sub> and DK<sub>x</sub>#, where x can be A or B, referring to the respective ports
- Data Output Clocks, QK<sub>x</sub> and QK<sub>x</sub>#, 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.

DK<sub>x</sub> and DK<sub>x</sub># samples the DQ<sub>x</sub> inputs on both rising and falling edges. Similarly, QK<sub>x</sub> and QK<sub>x</sub># samples the DQ<sub>x</sub> outputs on both rising and falling edges.

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

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

QDR IV SRAM devices are available in  $\times 18$  and  $\times 36$  bus width configurations. The exact clock-data relationships are as follows:

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

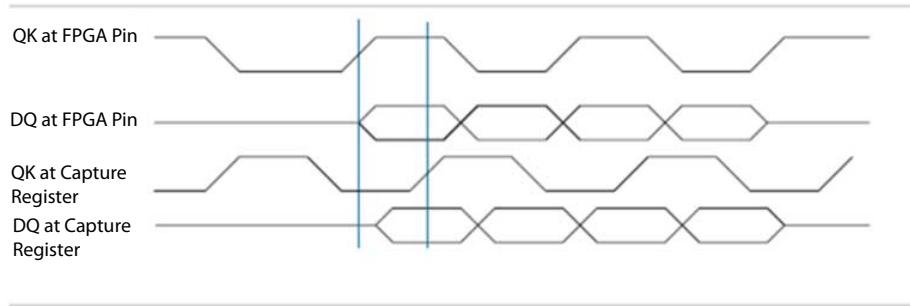
There are  $t_{CKDK}$  timing requirements for skew between  $\text{CK}$  and  $\text{DK}_x$  or  $\text{CK}\#$  and  $\text{DK}_x\#$ . Similarly, there are  $t_{CKQK}$  timing requirements for skew between  $\text{CK}$  and  $\text{QK}_x$  or  $\text{CK}\#$  and  $\text{QK}_x\#$ .

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

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

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

**Figure 150. Edge-Aligned DQ and QK Relationship During Read**



**Figure 151. Center-Aligned DQ and DK Relationship During Write**

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, go to the **Option Control** parameters in the **Configuration Register Settings** section of the **Memory** tab 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.

### 7.3.3.6. Specific Pin Connection Requirements

#### PLL

You must constrain the PLL reference clock to the address and command sub-bank only.

- You must constrain the single-ended reference clock to pin index 0 in lane 2.
- When pin index 0 in lane 2 is used for a single-ended reference clock, you cannot use pin index 1 in lane 2 as a general purpose I/O pin.
- You must constrain differential reference clocks to pin indices 0 and 1 in lane 2.
- The sharing of PLL reference clocks across multiple external memory interfaces is permitted; however, pin indices 0 and 1 of Lane 2 of the address and command sub-bank for all slave EMIF interfaces can be used only for supplying reference clocks. Intel recommends that you consider connecting these clocks input pins to a reference clock source to facilitate greater system implementation flexibility.

#### OCT

You must constrain the `RZQ` pin to pin index 2 in lane 2 of the address and command sub-bank only.

- Every EMIF instance requires its own dedicated `RZQ` pin.
- The sharing of `RZQ` pins is not permitted.

### 7.3.3.7. 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.

#### 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. Also, any pin in the same bank that is not used by an external memory interface may not be available for use as a general purpose I/O pin.
4. You cannot share sub-banks between external memory interfaces.

## 7.4. QDR-IV Board Design Guidelines

The following topics provide guidelines for improving the signal integrity of your system and for successfully implementing a QDR-IV SRAM interface on your system.

The following areas are discussed:

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

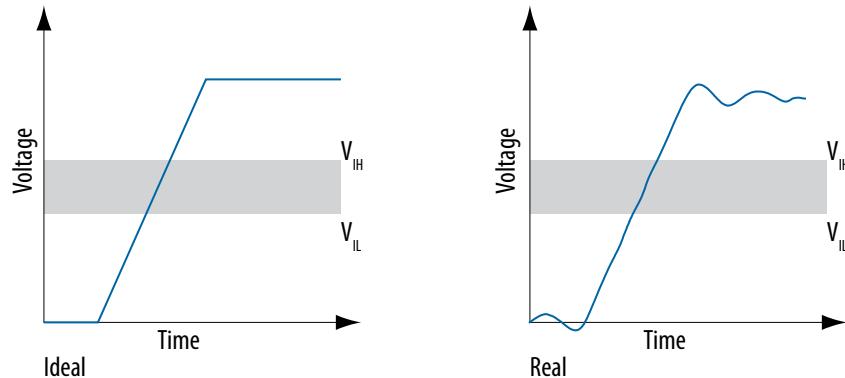
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

The following key factors affect signal quality at the receiver:

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

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

**Figure 152. Ideal and Real Signal at the Receiver**



#### 7.4.1. General Layout Routing Guidelines

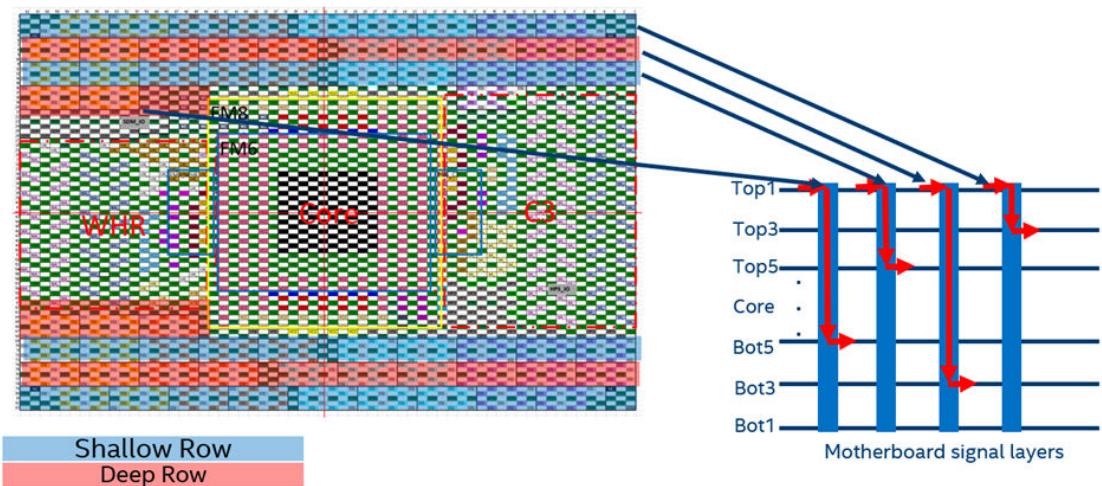
Follow the guidelines in this section for routing from the FPGA to memory for Intel Agilex 7 F-Series and I-Series devices.

For maximum channel margin, you should consider the following general routing optimizations during the layout design phase:

- When routing the memory interface, ensure that there are solid ground reference planes without any plane splits or voids, to ensure an uninterrupted current return path.
- For signal vias in layer transitions, you must place ground stitching vias close by, within 80 mil in distance (closer is better), and in between signal vias, to minimize crosstalk among signal vias. Avoid any unnecessary signal layer transitions to minimize crosstalk, loss, and skews.
- Trace impedance plays an important role in signal integrity; board designers must follow impedance recommendations for each signal group and configuration according to the guidelines in this document. If you use a different stackup than the reference stackup in the PCB design, you must tune the trace width and geometries to achieve the impedance recommendations.
- Intel recommends using 45-degree angles (not sharp 90-degree corners) when routing signal turns. Use  $3 \times h$  spacing for serpentine routing, where  $h$  is the height or distance from the trace to the nearest GND reference plane.
- Avoid referencing a signal to both power and ground planes at the same time (dual referencing), for signal return paths. When this cannot be avoided, ensure that the closer reference plane is solid ground, and the far side power plane is not noisy.
- Avoid routing two internal signal layers adjacent to each other (dual stripline routing). When this cannot be avoided, use angled routing between two signal layers to minimize crosstalk and coupling between the layers.

- Follow time-domain length and skew matching rules to ensure that your interface meets timing requirements. You should route signals from the same byte or group together on the same layer to avoid any out-of-phase crosstalk caused by varying layer transition lengths.
- To optimize memory interface margins, Intel recommends routing address, command, and control signals on shallow layers. (Shallow layers are those above the PCB core where via transition lengths are short.)
- For boards thicker than 65 mil, Intel recommends alternating adjacent FPGA EMIF BGA/ball rows with deep and shallow board via transitions to minimize crosstalk between adjacent bytes. This method is illustrated in the following figure:

**Figure 153. Recommended alternate adjacent via transitions to avoid crosstalk between adjacent bytes**



#### 7.4.2. Reference Stackup

This topic illustrates the reference stackup on which EMIF routing design guidelines are based.

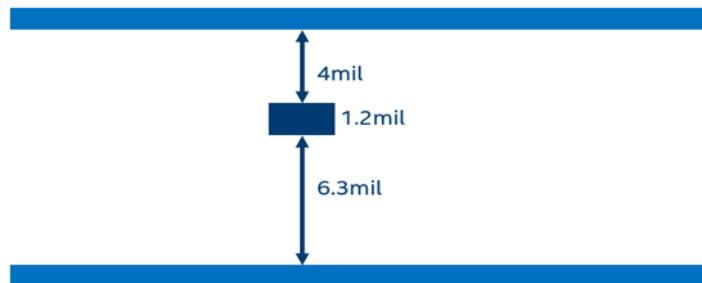
It is important to understand that trace geometry such as width, thickness, and edge-to-edge spacing, and the distance to reference planes, all impact trace impedance and crosstalk levels.

**Table 147. Reference stackup details**

Layer	Type	Thickness
<b>SM TOP</b>		0.5
<b>L1</b>	signal	1.8
<b>D1</b>	prepreg	2.7
<b>L2</b>	gnd/power	1.2
<b>D2</b>	core	4.0
<b>L3</b>	signal	1.2
<b>D3</b>	prepreg	6.3

*continued...*

Layer	Type	Thickness
<b>L4</b>	gnd/power	1.2
<b>D4</b>	core	4.0
<b>L5</b>	signal	1.2
<b>D5</b>	prepreg	6.3
<b>L6</b>	gnd/power	1.2
<b>D6</b>	core	4.0
<b>L7</b>	signal	1.2
<b>D7</b>	prepreg	6.3
<b>L8</b>	gnd	1.2
<b>D8</b>	core	4
	Power	1.2
	prepreg	6.3
	power	1.2
	core	4
	gnd	1.2
	prepreg	6.3
	power	1.2
	core	4
<b>L9</b>	gnd	1.2
<b>D9</b>	prepreg	6.3
<b>L10</b>	signal	1.2
<b>D10</b>	core	4.0
<b>L11</b>	gnd/power	1.2
<b>D11</b>	prepreg	6.3
<b>L12</b>	signal	1.2
<b>D12</b>	core	4.0
<b>L13</b>	gnd/power	1.2
<b>D13</b>	prepreg	6.3
<b>L14</b>	signal	1.2
<b>D14</b>	core	4.0
<b>L15</b>	gnd/power	1.2
<b>D15</b>	prepreg	2.7
<b>L16</b>	signal	1.8
<b>SM BOT</b>		0.5
	Total	120.1

**Figure 154. Reference trace geometries**

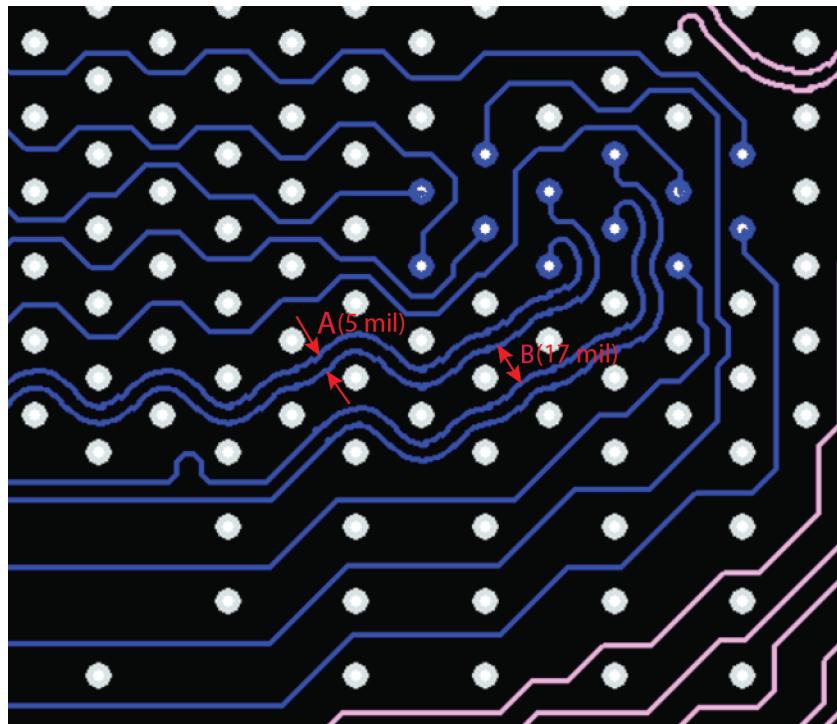
The reference stackup height is selected to be 120 mil to cover maximum signal via coupling (110mil) in simulation while extracting EMIF design guideline. Intel recommends that board designers do not exceed 110mil signal via coupling (stripline routing on inner layers) in the EMIF layout PCB design.

If the PCB stackup exceeds 120 mil in height, Intel recommends routing EMIF signals on upper layers, not to exceed more than 110 mil of signal via coupling.

The reference stackup materials in the above figure are selected as FR4, to represent worse-case signal loss in design phase simulation. In case of low-loss materials, the maximum end-to-end routing length shall be larger than the recommended end-to-end routing length in the design guidelines; however, you must perform time-domain channel simulation to ensure that timing requirements are met.

The Intel Agilex 7 F-Series and I-Series family pin floorplan is a HEX pattern with 1mm pitch. The following figure shows an example of routing for an IO12 (one-byte data) on PCB within FPGA fan-out region.

**Figure 155. Intel Agilex 7 F-Series and I-Series 1mm HEX pin pattern/floorplan and recommended routing for one byte of data (IO12)**



The following general notes apply to the EMIF routing guidelines for QDR-IV topologies:

- All spacing requirements are the minimum requirement to be met on PCB in EMIF routing guideline table.
- Breakout (BO1/BO2) spacings have two different values in guideline tables. The first value represents minimum spacing between two signals routed as a pair (tightly coupled signals); this value is marked as A (5 mil) in the above figure. The second value represents minimum spacing between two pairs, and is marked as B (17 mil) in the above figure.
- Main route (M) spacings have both value in mil and formula. In formula,  $h$  represents the trace-to-nearest-reference-plane height or distance. In cases using a stackup different than the reference stackup, board designers shall use formula to calculate the correct spacing requirements.
- There is no differential impedance target for CLK nor QK/DK. Board designers shall follow single-ended impedance target and keep the signals within the pair closely coupled, within 3-4 mil spacing. For information on QK/DK and CLK/CLKB, refer to [Skew Matching Guidelines for QDR-IV Configurations](#).
- In guideline tables, *SL* stands for stripline routing recommendation and *US* stands for upper surface (Microstrip) routing recommendation.
- The trace width value/geometry in guideline tables stands for trace designed for target impedance based on the reference stackup. This trace geometry shall be designed based on actual stackup and target impedance in guideline table.
- In guideline tables, *BO1* and *BO2* represent fan-out routing lengths. *M* stands for out of fan-out (PCB main) routing lengths

### 7.4.3. Routing Guidelines for QDR-IV Topology

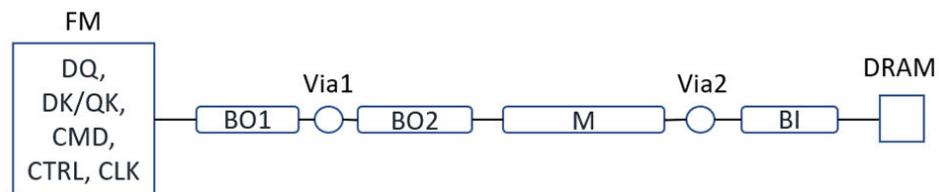
This section discusses QDR-IV single-memory-device-per-channel topology.

#### 7.4.3.1. QDR-IV Single Device Memory Topology

This topic provides routing guidelines for a QDR-IV single-device memory topology.

The following figure illustrates the signal connection topology for a QDR-IV single-device memory configuration.

**Figure 156. Signal Connections for QDR-IV Single Device Configuration**



**Table 148. Specific Routing Guidelines for QDR-IV Single Device Memory Topology for Supported Signals in the Interface**

Signal Group	Segment	Routing Layer	Max Length (mil)		Target Zse (ohm)	Trace Width, W (mil)	Trace Spacing, S1 (mil): Within Group	Trace Spacing, S2 (mil): CMD/CTRL/CLK to DQ/DK/QK	Trace Spacing, S3 (mil): Byte to Byte	Trace Spacing, (mil), Within DIFF pair	Trace Spacing, (mil), DK/QK pair to DQ	Trace Spacing, (mil), CLK pair to CMD/CTRL/CKE
			Segment	Total MB								
DQ, CMD, CTRL	BO1	US	50	4000	45	4	5, 17	5, 17	17			
	BO2	SL	1000			4	5, 17	5, 17	17			
	M	SL				4.5	8 (2H)	12 (3H)	12 (3H)			
	BI	US	150			4	8 (2H)	12 (3H)	12 (3H)			
DK/QK, CLK	BO1	US	50	4000	45	4		5, 17		4	17	17
	BO2	SL	1000			4		5, 17		4	17	17
	M	SL				4.5		12 (3H)		4	12 (3H)	12 (3H)
	BI	US	150			4		12 (3H)		4	12 (3H)	12 (3H)

For related information, refer to the figures in the [Reference Stackup](#) topic in this chapter.

#### 7.4.3.2. Skew Matching Guidelines for QDR-IV Configurations

The guidelines in this topic apply to QDR-IV memory configurations.

Board designers must observe the following guidelines for QDR-IV skew matching:

- Perform skew matching in time (picoseconds) rather than in actual trace length, to better account for via delays when signals are routed on different layers.
- Include both package per-pin skew and PCB delay when performing skew matching.
- Skew (length) matching for the alert signal is not required.

The following table provides skew matching guidelines for QDR-IV topologies.

**Table 149. QDR-IV Skew Matching Guidelines**

QDR-IV Length Matching Rules	Length Matching in Time (ps)
Length Matching between DK/QK and CLK	-50 < CLK - DK/QK < 50ps
Length Matching between DQ and DK/QK within byte	-5ps < DQ - DK/QK < 5ps
Length matching between DK/QK and DK/QK#	< 1 ps
Length matching between DK and QK pairs	< 5 ps
Length matching between CLK and CLK#	< 1 ps
Length Matching between CMD and CTRL and Clock	-20 ps < CLK - CMD < 20 ps
Length matching among CMD and CTRL within each channel	< 20 ps
Include package length in skew matching for FPGA device with no migration	Required
Include package length in skew matching for FPGA device with migration when all package net length are available	It is recommended to use the final migrated package net length
Include package length in skew matching for FPGA device with migration when all package net length are not available	Not recommended

#### 7.4.3.3. Power Delivery Recommendation for QDR-IV Configurations

This topic describes power distribution network (PDN) design guidelines for QDR-IV configurations.

**Note:** For information on power distribution network design at the FPGA to meet timing margins, refer to the AG014 PDN design guideline.

The number of decoupling capacitors is based on a single channel. If multiple channels are sharing the same power rail, the number of decoupling capacitors at the memories must be scaled accordingly for all channels.

Physically small decoupling capacitors are recommended to minimize area, inductance, and resistance on the PDN path on the printed circuit board.

The following are recommended guidelines for designing power delivery for QDR-IV memory:

- Use 0.1uF in 0402 size to minimize inductance.
- Make VTT voltage decoupling close to termination resistors.
- Connect decoupling capacitors between VTT and ground at the memory device.
- Use a 0.1uF capacitor for every other VTT pin and a 0.01uF cap for every VDD and VDDQ pin.

#### 7.4.4. Intel Agilex 7 EMIF Design Guideline Summary

At a high level, the Intel Agilex 7 F-Series and I-Series device EMIF design guidelines are summarized below.

1. Define the stackup and topology for the PCB.
2. Follow the general and specific routing guidelines in this document for target impedance, the maximum trace length on the mother board, and the spacing between traces based on the provided table for EMIF specific topology on the PCB. For EMIF topologies not covered by this document, contact your Intel support representative.
3. Follow the skew and length (package plus PCB) guidelines for your specific topology.
4. Follow the recommended power delivery guidelines for memory components and specific PCB EMIF topology.
5. For post-layout simulation, follow the guidelines in the [I/O Timing Closure](#) chapter.

## 8. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Timing Closure

This chapter describes timing analysis and optimization techniques that you can use to achieve timing closure within the FPGA.

### 8.1. 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 constraints created and are analyzed by the Timing Analyzer. Core timing does not include user logic timing within core or to and from the EMIF block. The EMIF IP provides the constrained clock to the customer logic.
- Core to periphery (C2P) transfers have timing constraints created and are timing analyzed by the Timing Analyzer.
- Periphery to core (P2C) transfers have timing constraints created and are timing analyzed by the Timing Analyzer.
- Periphery to periphery (P2P) transfers are modeled entirely by a minimum pulse width violation on the hard block, and have no internal timing arc.

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.

#### 8.1.1. Timing Analysis

Timing analysis of Intel Agilex 7 F-Series and I-Series EMIF IP is somewhat simpler than that of earlier device families, because Intel Agilex 7 F-Series and I-Series devices have more hardened blocks and fewer soft logic registers to be analyzed, because most are user logic registers.

Your Intel Agilex 7 F-Series and I-Series 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.

### 8.1.1.1. 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), or the path from the last set of registers in the periphery to the first of registers in the core (P2C) and the ECC related path if it is enabled.

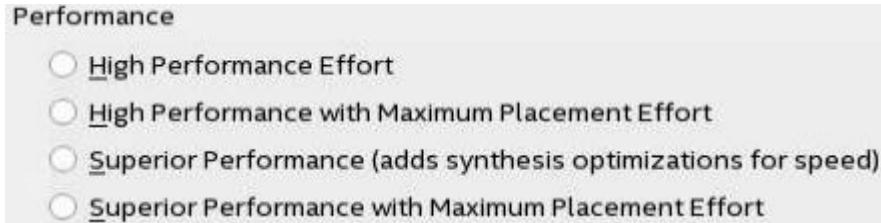
Core timing analysis excludes user logic timing to or from EMIF blocks. The EMIF IP provides a constrained clock (for example: ddr4\_usr\_clk) with which to clock customer logic; pll\_afi\_clk serves this purpose.

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`.

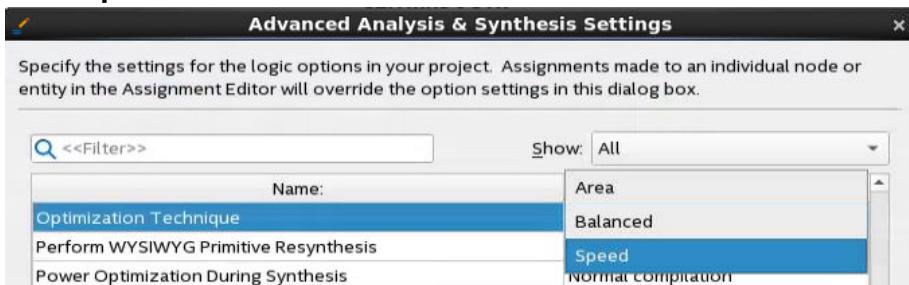
## 8.2. Optimizing Timing

The Intel Quartus Prime software offers several advanced features that you can use to assist in meeting core timing requirements.

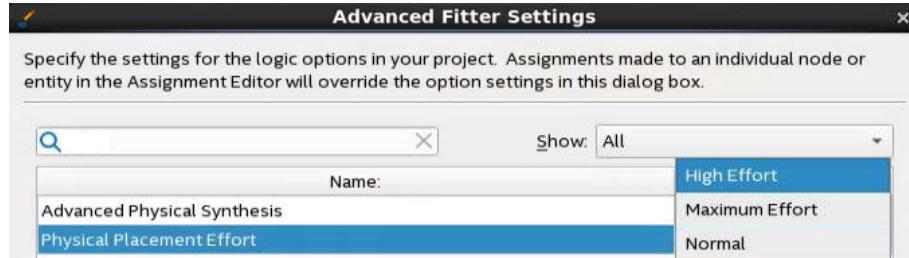
1. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings**. Under **Optimization mode**, select one of the **Performance** options.



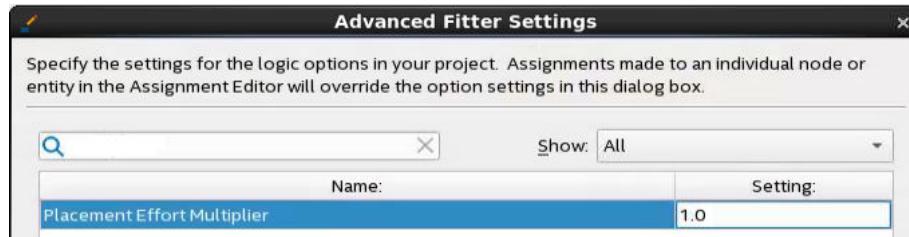
2. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings > Advanced Settings (Synthesis)**. For **Optimization Technique**, select **Speed**.



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.



## 9. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – I/O Timing Closure

This chapter describes how to evaluate I/O timing outside of the FPGA, by determining whether the memory interface channel of your printed circuit board (PCB) meets the necessary signal integrity requirements for your design.

This approach is based on SPICE analog simulations using extracted models of the PCB channel, IBIS models of the FPGA buffers, and IBIS models of the memory devices to determine if the external I/O channel is adequate for the target design.

### 9.1. I/O Timing Closure Overview

When designing printed circuit boards for high-speed external memory interfaces, you must route many sensitive signals, often with limited board resources. The drive to reduce bill-of-material cost also necessitates the use of cheaper materials, leading to a larger range of PCB tolerances that must be supported. The new I/O timing closure methodology provides a simulation-based method for determining whether a board layout is likely to meet the external channel signal integrity requirements for your memory interface IP.

This simulation-based approach for evaluating board signal integrity is new for Intel Agilex 7 F-Series and I-Series devices. With earlier Intel FPGA families, you had to perform your own simulations to determine signal integrity figures of merit such as crosstalk and ISI, and enter them into the EMIF parameter editor; the channel budget was then calculated as part of the timing analysis performed by the Intel Quartus Prime Timing Analyzer.

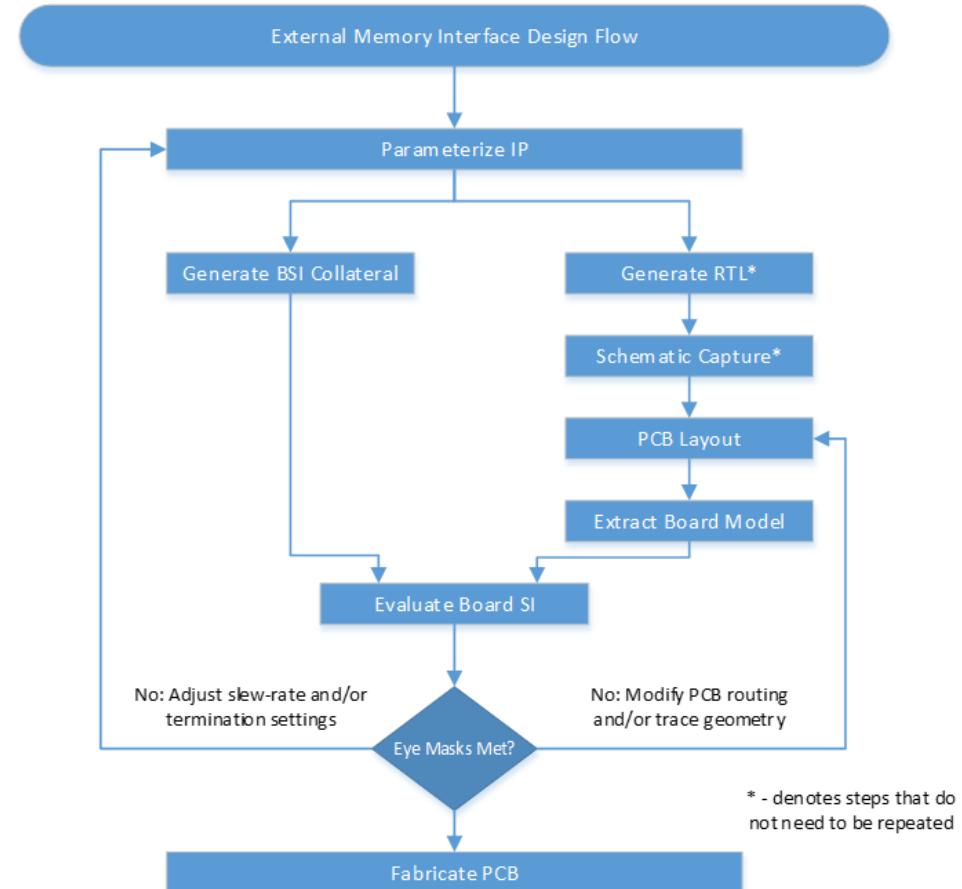
With the Intel Agilex 7 F-Series and I-Series FPGA EMIF IP, the timing analysis of the external I/O channel is no longer part of the signoff timing analysis flow. The EMIF IP now generates a set of SPICE simulation collateral for evaluating board signal integrity, which you can use in conjunction with extractions from your PCB to generate eye diagrams. The collateral also provides eye masks appropriate to the IP settings and FPGA family that are used to determine if the receiver eye has adequate voltage and timing margin.

### Board Signal Integrity Flow

The following steps summarize the high-level tasks comprising the board signal integrity flow:

1. Parameterize the EMIF IP to suit your memory interface, and generate the design example to create the board signal integrity (BSI) and synthesis RTL file sets.
2. Use the synthesis design example to determine a suitable location for the interface, enabling schematic capture and memory interface PCB layout.
3. After you have created an initial layout of the memory interface, PCB extractions are generated in the form of a Touchstone file that captures the multi-port network behavior of the PCB.
4. The generated board signal integrity collateral uses these extractions to generate eye diagrams for the address/command path, as well as both data bus write and read operations.
5. The SPICE simulation parameters for configuring memory and FPGA IBIS models, as well as appropriate input stimuli to match the IP, are automatically generated for you.

**Figure 157. Board Signal Integrity Flow**



The board signal integrity flow allows you to iterate your design quickly if necessary. You can make adjustments to signal integrity settings such as slew rate, deemphasis, drive strength, or read termination in the parameter editor, and rerun the simulations with the new IP-specific SPICE parameter file. If you cannot achieve mask compliance by modifying the IP, then you can re-route the critical areas of the PCB and repeat the simulation using the updated extracted model.

The following topics in this chapter provide detailed information about the generated SPICE simulation collateral, changes and customizations that you may have to make, as well as guidance for evaluating the generated eye diagrams.

## 9.2. Collateral Generated with Your EMIF IP

The SPICE simulation files that you need to evaluate I/O timing closure are generated automatically as part of the EMIF IP file set.

IP options that affect electrical behavior are translated into parameters that are consumed by SPICE simulation decks to ensure that the analog simulation matches the IP configuration; these parameters include the following:

- FPGA
  - On-chip termination settings
  - Slew rate and deemphasis settings
  - Multi-rank ODT settings
  - Memory clock frequency
  - Clock and strobe settings and phases
  - IBIS model configuration
- Memory
  - Number of components and topology
  - IBIS model configuration
  - Clock and strobe settings and phases

The generated SPICE simulation decks allow for evaluation of the address/command channel, the FPGA-to-Memory write channel, and the Memory-to-FPGA read channel. The resulting data eye for each of these simulations is evaluated against a compliance mask supplied by the IP to determine whether the channel crosstalk, ISI and loss characteristics are adequate for the desired operating frequency.

The simulation decks are constructed in a way that allows you to replace default transmission-line models with S-parameter touchstone extractions of the PCB for channel evaluation. The system assumes that you have followed the PCB design guidelines for trace length, geometry, and spacing as closely as possible, and have determined the dielectric material and stackup of your PCB.

## 9.3. SPICE Decks

The generated EMIF IP provides three SPICE decks for channel evaluation: address/command channel, FPGA write operation, and FPGA read operation.

Each simulation deck uses a 12-line channel model that maps to a 12-pin lane within a sub-bank. In all decks, two pins are designated as clock or strobe pins, one pin is designated as a victim pin and driven with a PRBS-10 pattern, and the remaining pins are designated as aggressor pins and driven with identical PRBS-15 patterns.

### 9.3.1. Address/Command Simulation Deck

The address/command simulation deck is configured to mimic Lane 0 of the External Memory Pin Interface table.

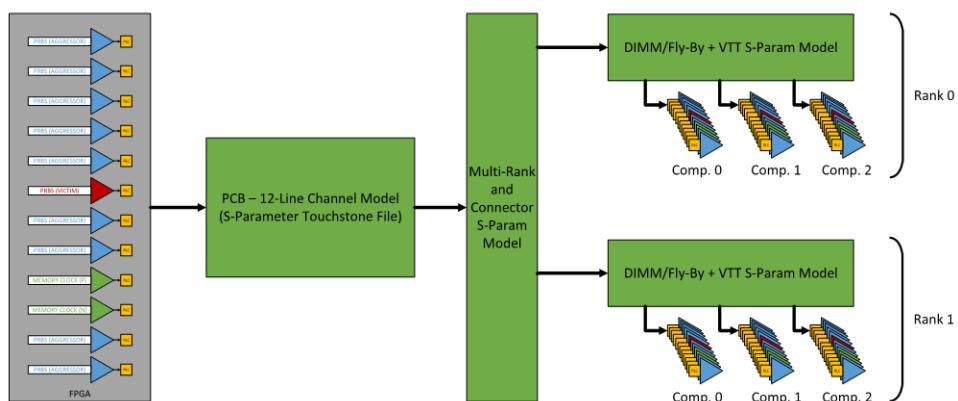
**Note:** All pin indices are 0-based throughout this section.

The configuration has the following characteristics:

- Two pins within the 12-pin lane are driven with a differential clock pattern to map to the CK0 and CK0\_N clock pairs. These pins are driven with a repeating clock pattern, phase-adjusted such that the rising/falling edge of CK0/CK0\_N are centered within the valid data window.
  - For DDR4, CK0/CK0\_N map to pins 8/9 within the lane.
  - For QDR4, CK0/CK0\_N map to pins 4/5 within the lane.
- The symbol rate of the address/command pins matches the characteristics of the memory technology.
  - For DDR4, the address/command channel is single-data-rate.
  - For QDR4, the address/command channel is double-data-rate.
- One pin is designated as a victim pin, driven with a PRBS-10 pattern (repeated three times).
  - For DDR4, pin 5 is designated as the victim pin.
  - For QDR4, pin 8 is designated as the victim pin.
- The remaining pins are driven with the same aggressor PRBS-15 data pattern.

The structure of the address/command simulation deck is illustrated below.

**Figure 158. Address/Command Simulation Deck Structure**



The simulation deck assumes a 12-line channel model; the purpose of each block is described in the sections below.

## FPGA

At the far left of the above diagram, 12 FPGA IBIS models are instantiated and configured to match the IP-specified memory-clock and address/command electrical settings. The pattern generators are embedded within this sub-circuit and automatically configured via the IP-generated parameter file.

## PCB — 12-Line Channel Model

This block connects to a 12-line (24-port plus ground) channel model that matches the signals implementing Lane 0 of the address/command bus for that interface. By default, this subcircuit instantiates 12 independent ideal 50-ohm transmission lines, which can be replaced by specifying the following options in the IP-generated parameter file:

Parameter Name	Default Value	Description
USE_AC_PCB_EXTRACTION	False	Specifies if a 24-port Touchstone extraction is to be used in the Address/Command simulation. If set to true, then the AC_PCB_EXTRACTION_FILE parameter must specify a valid Touchstone file location. If set to false, ideal transmission line models are used.
AC_PCB_EXTRACTION_FILE	<empty>	Specifies the file name for the 24-port Touchstone extraction file (.s24p) that represents the address/command channel. The first 12 pins map to Lane 0, Pins 0 through 11 connected to the balls of the FPGA. The next 12 pins map to the far-end of the channel, which is either the memory balls (for direct point-to-point connections) or to the bifurcation point for a multi-rank topology.

**Note:** Refer to the `ac_pcb_wrapper.sp` file for details on how the PCB extraction file is integrated into the SPICE simulation deck.

## Multi-Rank and Connector Model

Use this block in cases where you want to do the following:

- Model the multi-rank topology of your interface to explore the effect of parameters such as DIMM spacing, channel loading, or component stacking.
- Model the effect of the DIMM connector by inserting a vendor-supplied simulation model.

This block is situated between the end of the address/command PCB extraction model and the start of the fly-by topology for each rank. If you don't specify an extraction model, then the default model of the block is as follows:

- For single rank DDR4 systems, the default model for this block is a direct pass-through.
- For dual-rank DDR4 systems, the default model for this block is to connect directly to rank 0, and insert 12 independent 50-ohm, 50ps transmission lines between rank 0 and rank 1 to model a slight delay difference between the two ranks.
- For quad-rank DDR4 systems, the default model for this block is to connect directly to ranks 0 and 1, and insert 12 independent 50-ohm, 50ps transmission lines between ranks 0/1 and ranks 2/3.

You can override this default behavior by specifying an S-Parameter model of your own, using the following options. (Note that the number of ports of the model should align with the IP-calculated value of the number of A/C ranks in the system, MEMAC\_RANKS. A value of 1, 2 or 4 correspond to single, dual- or quad-rank systems and require 24, 36 or 60-port models, respectively.)

Parameter Name	Default Value	Description
USE_AC_MULTIRANK_CONNECTOR_EXTRACTION	False	Specifies if a N-port Touchstone extraction is to be used in the address/command simulation to model the multi-rank bifurcation topology or the DIMM connector. If set to true, then the AC_MULTIRANK_CONNECTOR_EXTRACTION_FILE parameter must specify a valid Touchstone file location. If set to false, ideal transmission line models are used.
AC_MULTIRANK_CONNECTOR_EXTRACTION_FILE	<empty>	Specifies the file name for the N-port Touchstone extraction file (.sNp) that represents the address/command channel. The first 12 pins map to Lane 0, Pins 0 through 11 connected to the endpoint of the 12-line A/C channel model. The next 12 pins connect to the start of the A/C Fly-by topology for Rank 0. If MEMAC_RANKS is greater than 1, the next 12 pins connect to the address/command fly-by topology for Rank 1, and Rank 2 and Rank 3 for quad-rank systems.

**Note:** Refer to the ac\_mr\_conn\_wrapper.sp file for details on how the multi-rank/connector model integrates into the SPICE simulation deck.

### DIMM/Fly-By + VTT Model

This block models the fly-by topology of the address/command bus for a multi-component memory interface. You can use this file to adjust factors that can affect the signal integrity of the address/command interface, including the following:

- Component loading
- Component spacing
- Trace geometry/impedance between components
- Effect of using DDP components vs planar components
- Effect of placement/value of termination resistor network
- Effect of trace spacing on signal integrity
- Effect of vias, back-drilling, or placement of components on both sides of board

By default, this block generates MEMAC\_COMP\_PER\_RANKS components in the fly-by chain. The first component is connected directly to the first 12 ports of the subcircuit. Subsequent components are connected to the previous component using 12 independent 50-ohm, 50ps ideal transmission lines. The final component in the flyby chain has each of its signals terminated to the V<sub>TT</sub> source via 50-ohm resistors. Note that the differential clock is AC-coupled to the V<sub>TT</sub> source through a 0.01uF capacitor.

You can override this default behavior by specifying an S-Parameter model of your own, using the following options. (Note that the number of ports of the model should align with the IP-calculated value of the number of A/C components in the system, MEMAC\_COMP\_PER\_RANKS. The minimum number of ports for a value of 1 is 24 ports, increasing by 12 ports for every additional component in the flyby chain.)

Parameter Name	Default Value	Description
USE_AC_FLYBY_EXTRACTION	False	Specifies if an N-port Touchstone extraction is to be used in the address/command simulation to model the fly-by topology. If set to true, then the AC_FLYBY_EXTRACTION_FILE parameter must specify a valid Touchstone file location. If set to false, ideal transmission line models are used.
AC_FLYBY_EXTRACTION_FILE	<empty>	Specifies the file name for the N-port Touchstone extraction file (.sNp) that represents the address/command channel. The first 12 pins map to Lane 0, Pins 0 through 11 connected to the endpoint of the 12-line A/C multirank/connector model. The next 12 pins connect to the first component of the address/command Fly-by topology. If MEMAC_COMP_PER_RANKS is greater than 1, the next 12 pins connect to each additional component in the chain up to a maximum of 9 components (a 12-port model).

**Note:** Refer to the ac\_dimm\_flyby\_wrapper.sp file for details on how the multi-rank/connector model integrates into the SPICE simulation deck.

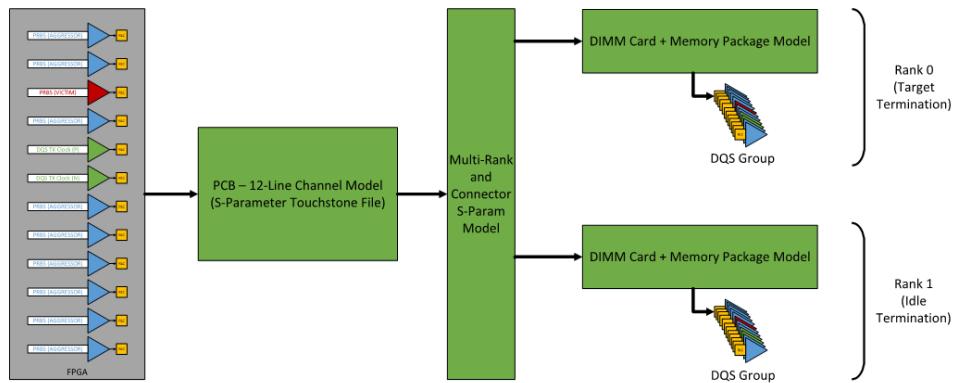
### 9.3.2. FPGA Write Operation Simulation Deck

The FPGA Write simulation deck allows you to assess the channel characteristics for write operations.

For DDR4 interfaces, the 12-line extraction maps either to one x8 DQS group (8 DQ pins, a complementary DQS pair, and a write DBI/DM pin if specified in the IP), or two x4 DQS groups (8 DQ pins and two complementary DQS pairs). The following figure illustrates the composition of the FPGA write SPICE simulation deck.

The simulation deck generates a strobe pattern to model the behavior of the DQS pair in a conventional write operation. Pin 2 within the lane is designated as a victim pin and driven with a repeating PRBS-10 pattern, and all other DQ and DM/DBI pins are designated as aggressors and driven with a PRBS-15 pattern.

**Figure 159. FPGA Write Simulation Deck Structure**



## FPGA

At the left side of the above diagram, 12 FPGA IBIS models are instantiated and configured to match the IP-specified DQ and DQS electrical settings. The pattern generators are embedded within this sub-circuit and automatically configured by the IP-generated parameter file.

## PCB – 12-Line Channel Model

This block models a 12-line (24-port plus ground) channel that matches the signals implementing any of the DQS groups of the memory interface. For DDR4, this can be either the signals comprising a DQSx8 group (data, DM/DBI and strobes) or two DQSx4 nibbles packed within a single 12-pin lane. Typically, the lane expected to have the worst-case signal integrity should be used for these simulations, however you can run this simulation with models for every lane in the system if you want.

By default, this subcircuit instantiates 12 independent ideal 50-ohm transmission lines, which you can replace with a Touchstone extraction by specifying the following options in the IP-generated parameter file:

Parameter Name	Default Value	Description
USE_DQ_PCB_EXTRACTION	False	Specifies whether a 24-port Touchstone extraction is to be used in the FPGA read/write simulation. If set to true, then the DQ_PCB_EXTRACTION_FILE parameter must specify a valid Touchstone file location. If set to false, ideal transmission line models are used
DQ_PCB_EXTRACTION_FILE	<empty>	Specifies the file name for the 24-port Touchstone extraction file (.s24p) that represents the data channel. The first 12 pins map to Pins 0 through 11 connected to the balls of the FPGA for the selected DQS group. The next 12 pins map to the far-end of the channel, which is either the memory balls (for direct point-to-point connections) or to the bifurcation point for a multi-rank topology.

**Note:** Refer to the `dq_pcb_wrapper.sp` file for details on how the PCB extraction file is integrated into the SPICE simulation deck.

## Multi-Rank and Connector Model

Similar to the address/command channel multi-rank/connector model block, you can use this block in cases where you want to do either of the following:

- Model the multi-rank topology of the interface to explore the effect of parameters such as DIMM spacing, channel loading, or component stacking.
- Model the effect of the DIMM connector by inserting a vendor-supplied simulation model.

This block is situated between the end of the DQ PCB extraction model and the start of memory IBIS model receiver bank for each rank. If you do not specify an extraction model, the default model of the block is as follows:

- For single-rank DDR4 systems, the default model for this block is a direct pass-through.
- For dual-rank DDR4 systems, the default model for this block is to connect directly to rank 0, and insert 12 independent 50-ohm, 50ps transmission lines between rank 0 and rank 1 to model a slight delay difference between the two ranks.
- For quad-rank DDR4 systems, the default model for this block is to connect directly to ranks 0 and 1, and insert 12 independent 50-ohm, 50ps transmission lines between ranks 0/1 and ranks 2/3.

You can override this default behavior by specifying an S-Parameter model of your own, using the following options. (Note that the number of ports of the model should align with the IP-calculated value of the number of DQ Ranks in the system, **MEM\_DQ\_RANKS**. A value of 1, 2 or 4 correspond to single, dual- or quad-rank systems and require 24, 36 or 60-port models respectively.)

Parameter Name	Default Value	Description
USE_DQ_MULTIRANK_CONNECTOR_EXTRACTION	False	Specifies whether a N-port Touchstone extraction is to be used in the FPGA write/read simulation to model the multi-rank bifurcation topology or the DIMM connector. If set to true, then the <b>DQ_MULTIRANK_CONNECTOR_EXTRACTION_FILE</b> parameter must specify a valid Touchstone file location. If set to false, ideal transmission line models are used.
DQ_MULTIRANK_CONNECTOR_EXTRACTION_FILE	<empty>	Specifies the file name for the N-port Touchstone extraction file (.sNp) that represents the data channel. The first 12 pins map to Lane 0, Pins 0 through 11 connected to the endpoint of the 12-line DQ channel model. The next 12 pins connect to the start of the memory IBIS model receiver bank for Rank 0. If <b>MEM_DQ_RANKS</b> is greater than 1, the next 12 pins connect to the DIMM card/memory package model for Rank 1, and Rank 2 and Rank 3 for quad-rank systems.

**Note:** Refer to the `dq_mr_conn_wrapper.sp` file for details on how the multi-rank/connector model is integrated into the SPICE simulation deck.

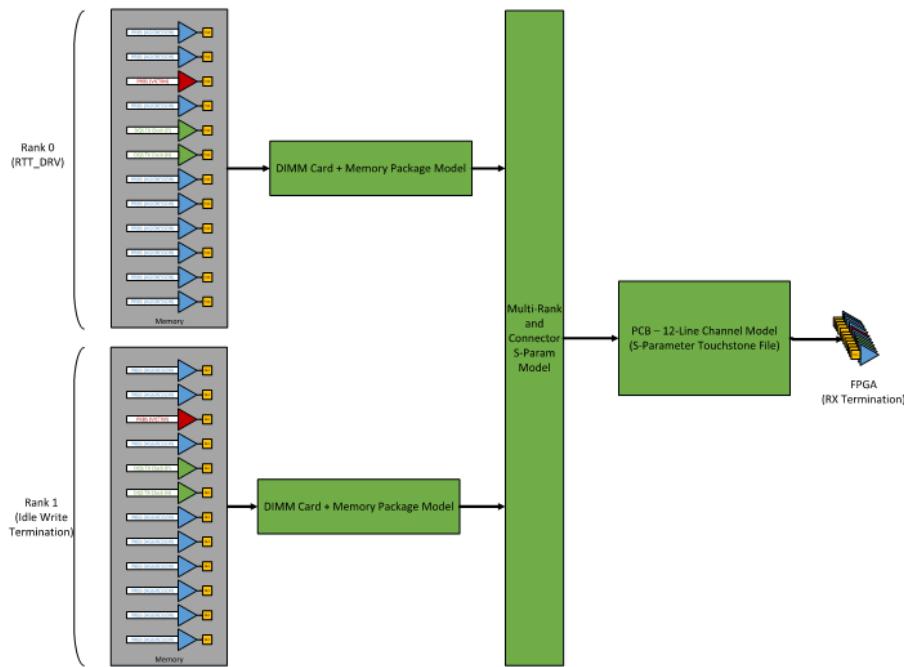
### 9.3.3. FPGA Read Operation Simulation Deck

The FPGA Read simulation deck lets you evaluate the channel characteristics for read operations.

The figure below illustrates the composition of the FPGA Read SPICE simulation deck. The composition of the FPGA Read simulation deck is identical to that of the FPGA Write simulation deck, except that the active memory rank serves as the driver and the FPGA is a receiver. The top-level parameterization file configures all electrical settings to match the options for RTT\_WR and OCT\_IN chosen during IP configuration.

The simulation deck generates an edge-aligned strobe pattern to model the behavior of the DQS pair in a conventional read operation. Similar to the FPGA Write simulation deck, Pin 2 within the lane is a victim pin and driven with a repeating PRBS-10 pattern, and all other DQ and Read DBI pins (if applicable) are aggressors and driven with a PRBS-15 pattern.

**Figure 160. FPGA Read Simulation Deck Structure**



## 9.4. File Organization

This topic lists the SPICE simulation files supplied with the EMIF IP.

The top-level parameterization file accompanies the IP as derived from the settings of that IP instance. The remaining files are common to all IP instances and come as a .ZIP file in the synthesis file set with a name that is unique to the IP.

In the Intel Quartus Prime software version 20.3 and later, the top-level SPICE parameterization file has the following location and structure:

```
altera_emif_arch_fm_<ip_version>/synth/
<instance_name>_altera_emif_arch_fm_<ip_version>_<uniquification_
code>_ip_parameter.dat
```

You can find the .ZIP file containing the simulation collateral in the same directory:

```
altera_emif_arch_fm_<ip_version>/synth/
<instance_name>_altera_emif_arch_fm_<ip_version>_<uniquification_
code>_spice_files.zip
```

**Table 150. SPICE Deck File Organization**

File Name	Function
User-Editable Collateral	
membsi_ip_parameters.dat	Include file for IP Parameters. You should copy into this file the contents of the IP-specific parameter file that the IP generates.
<i>continued...</i>	

File Name	Function
pin_parasitics.dat	Include file containing FPGA and memory package pin parasitic information. You should modify the contents of this file.
finesim_options.inc	Include file to specify SPICE simulator options.
Top-Level Simulation Decks	
ac_top.sp	Top-level SPICE deck for Address/Command simulations.
dq_wr_top.sp dq_2rank_wr_top.sp dq_4rank_wr_top.sp	Top-level SPICE deck for FPGA write operations. Use only the file corresponding to the number of DQ ranks in your IP.
dq_rd_top.sp dq_2rank_rd_top.sp dq_4rank_rd_top.sp	Top-level SPICE deck for FPGA read operations. Use only the file corresponding to the number of DQ ranks in your IP.
Extracted Models	
ac_pcb_wrapper.sp	12-line extraction model for the address/command channel (Lane 0 of the memory interface).
dq_pcb_wrapper.sp	12-line extraction model for the data channel (worst-case DQS group).
ac_mr_conn_wrapper.sp	Extraction model for the multi-rank bifurcation point or DIMM connector of the address/command channel.
dq_mr_conn_wrapper.sp	Extraction model for the multi-rank bifurcation point or DIMM connector of the data channel.
ac_dimm_flyby_wrapper.sp	Extraction model for the fly-by channel of a component interface, including $V_{TT}$ termination resistors, or the extraction model of the DIMM raw card including $V_{TT}$ termination resistors.
dq_dimm_pkg_wrapper.sp	Currently unused.
Buffer Model Wrappers	
tx_buffer.sp	Wrapper for transmit buffer IBIS models and data generator.
rx_buffer.sp	Wrapper for receive buffer IBIS model.
lane_tx12.sp	12-line bundle wrapper for transmit buffers.
lane_rx12.sp	12-line bundle wrapper for receive buffers.
Pattern Generators	
dqs_wave.sp	Free-running clock pattern generator.
prbs_15.sp	PRBS-15 pattern generator. Produces 32,768 pseudo-random bits.
prbs_10_x3.sp	PRBS-10 pattern generator. Produces three complete cycles of 1024 pseudo-random bits each for a total of 3,072 bits.

## 9.5. Top-level Parameterization File

The top-level parameterization file contains all the necessary IP information to perform the three SPICE simulations to obtain the address and command, DQ-Write, and DQ-Read data eyes.

To enable your simulations, follow these steps:

1. Modify the required parameters to supply file name locations for IBIS models and Touchstone models.
2. Modify the IP-supplied parameters to correct any discrepancies between the generated simulation parameters and your design (optional).
3. Import the contents of the IP-generated `<instance_name>_altera_emif_arch_fm_<ip_version>_<unique_id>_ip_parameter.dat` file into the `membsi_ip_parameter.dat` file.

## 9.6. IP-Supplied Parameters that You Might Need to Override

The IP generation framework automatically creates parameters to designate the number of ranks and the number of components in the address/command channel. However, there are situations where the IP framework is not aware of modifications that are made on your PCB — for example, repeater or buffer devices, additional external termination resistors, or non-standard memory components.

In these cases, it may be necessary for you to modify these generated parameters. The table below highlights some of the parameters that you might need to change to create a customized simulation deck.

**Table 151.**

Parameter Name	Description	Reasons for Modification
MEM_VCC	Specifies the memory voltage. This is set to 1.2V for DDR4 and QDR4 applications by default.	This value may be raised or lowered to account for regulator tolerances or PCB PDN IR droop.
MEMCLK_COMP_IBIS	Specifies the IBIS model name for the clock input buffer on the memory model.	This value may need to change if a different memory model or a buffer device is used.
MEMAC_IBIS	Specifies the IBIS model name for the address/command input buffers on the memory model.	This value may need to change if a different memory model or a buffer device is used.
MEMAC_RANKS	Specifies the number of address/command ranks in the system.	This value may need to change if the number of address/command loads differs from this value due to the use of buffer devices or multi-die components.
MEMAC_COMPS_PER_RANK	Specifies the number of address/command components on the flyby chain.	This value may need to change if non-standard memory devices are used and the number of components in the chain differs from this value, or if a buffer chip is used.
MEM_DQ_RANKS	Specifies the number of data ranks in the system.	This value may need to change if the number of databus ranks differs from this value.
WR_MEM_*_IBIS	Specifies the name of the IBIS model on the memory to be used during write operations.	These values may need to be changed if the naming of the IBIS models supplied by the vendor does not align to this format.
RD_MEM_*_IBIS	Specifies the name of the IBIS model on the memory to be used during read operations.	These values may need to be changed if the naming of the IBIS models supplied by the vendor does not align to this format.

*continued...*

Parameter Name	Description	Reasons for Modification
AC_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for address/command pins.	Address/command pins usually use the IBIS model type of "input" (buffer type = 1), but this may differ across vendor models.
DQ_WR_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for DQ pins in read mode (FPGA write operations).	DQ pins in read mode usually use the IBIS model type of "input" (buffer type = 1), but this may differ across vendor models.
DQ_RD_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for DQ pins in write mode (FPGA read operations).	DQ pins in write mode usually use the IBIS model type of "input_output" (buffer type = 3), but this may differ across vendor models.

**Table 152. Customizable IP-Generated SPICE Parameters**

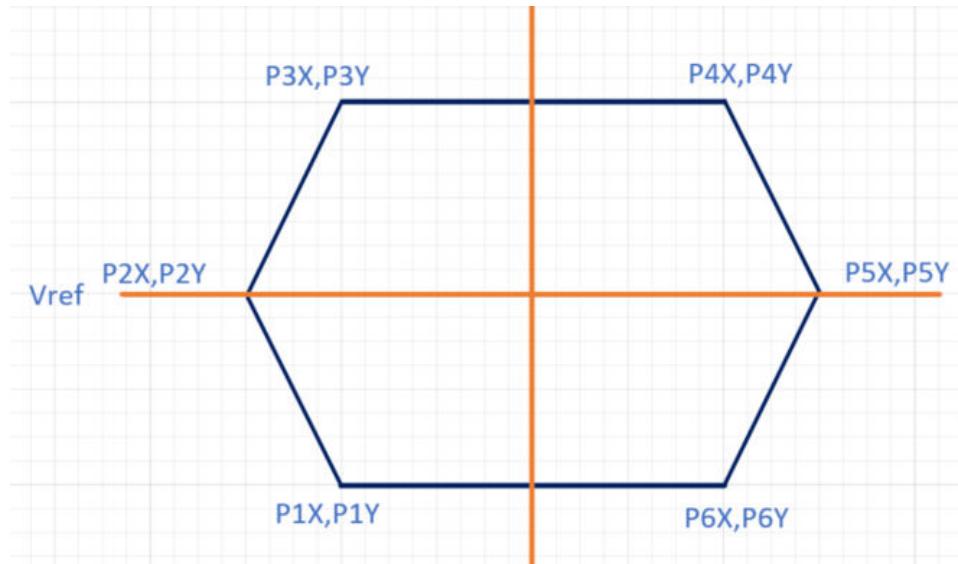
Parameter Name	Description
MEM_VCC	Specifies the memory voltage. This is set to 1.2V for DDR4 and QDR4 applications.
MEMCLK_COMP_IBIS	Specifies the IBIS model name for the clock input buffer on the memory model. You may need to change this value if a different memory model or a buffer device is used.
MEMAC_IBIS	Specifies the IBIS model name for the address/command input buffers on the memory model. You may need to change this value if a different memory model or a buffer device is used.
MEMAC_RANKS	Specifies the number of address/command ranks in the system. You may need to change this value if the number of address/command loads differs from this value due to the use of buffer devices.
MEMAC_COMPS_PER_RANK	Specifies the number of address/command components on the fly-by chain. You may need to change this value if non-standard memory devices are used and the number of components in the chain differs from this value, or if a buffer chip is used.
MEM_DQ_RANKS	Specifies the number of data ranks in the system. You may need to change this value if the number of databus ranks differs from this value.
WR_MEM_*_IBIS	Specifies the name of the IBIS model on the memory to be used during write operations. You may need to change these values if the naming of the IBIS models supplied by the vendor does not align to this format.
RD_MEM_*_IBIS	Specifies the name of the IBIS model on the memory to be used during read operations. You may need to change these values if the naming of the IBIS models supplied by the vendor does not align to this format.
AC_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for address/command pins. Normally, address/command pins use the IBIS model type of input (buffer type = 1), but this may differ across vendor models.
DQ_WR_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for DQ pins in read mode (FPGA write operations). Normally, DQ pins in read mode use the IBIS model type of input (buffer type = 1), but this may differ across vendor models.
DQ_RD_M_*_IBISTYPE	Specifies the type of IBIS buffer used by the memory model for DQ pins in write mode (FPGA read operations). Normally, DQ pins in write mode use the IBIS model type of input_output (buffer type = 3), but this may differ across vendor models.

## 9.7. Understanding the \*\_ip\_parameters.dat File and Making a Mask Polygon

The generated \*\_ip\_parameters.dat file contains information that you can use to create masks for eye diagram evaluation.

Each mask requires six pairs of X,Y coordinates that define a polygon establishing the shape of the mask. In each coordinate pair, the X value specifies the eye width as a percentage of a unit interval (UI), and the Y value specifies the eye height, in millivolts (mV).

**Figure 161. Six Coordinate Pairs Establishing an Eye Mask Polygon**



The following sections describe the content of the \*\_ip\_parameters.dat file, and provide examples for eye mask values for WR and RD masks.

### Sample \*\_ip\_parameters.dat Content

The following sample illustrates the top part of a \*\_ip\_parameters.dat file. The following information is included:

- Memory type
- Memory parameters
- Number of ranks
- MEM\_AC\_FREQ\_PS (in picoseconds, not megahertz)
- MEM\_DQ\_FREQ\_PS (in picoseconds, not megahertz)

**Figure 162. Top Part of an \*\_ip\_parameters.dat File**

```
* Auto-generated contents start below...
.param emif_corename=str('rdimm_3ds_x4_3h_altera_emif_arch_fm_19l_ycjjzvwy')

.param PROROCOL                         = str('DDR4') - - - - - Memory Type
.param NUM_RANKS                          = 2
.param SLEW_RATE_DRAM                    = 4.0
.param SLEW_RATE_DRAM_CLOCK              = 4.0
.param VIN_Ms                            = 0.136
.param VIN_Mh                            = 0.068
.param SLEW_RATE_PHY                     = 2.0
.param SLEW_RATE_PHY_CLOCK               = 2.0
.param SLEW_RATE_CA                      = 2.0
.param SLEW_RATE_CLOCK                  = 4.0
.param UI                                = 0.625
.param tCK                                = 0.625
.param tDQSQ                             = 0.10624999999999998
.param tQH                                = 0.41500000000000004
.param tDS                                = 0.0625
.param tDH                                = 0.0625
.param tIS                                = 0.115
.param tIH                                = 0.14
.param tDQSCK                            = 0.225
.param tDQSS                             = 0.27
.param tWLS                               = 0.162
.param tWLH                               = 0.162
.param tDSS                               = 0.0625
.param tDSH                               = 0.0625
.param BD_PKG_SKEW                       = 0.02
.param CA_BD_PKG_SKEW                   = 0.18
.param CA_TO_CK_BD_PKG_SKEW             = 0.0
.param DQE_BOARD_SKEW                   = 0.02
.param DQS_TO_CK_BOARD_SKEW             = 0.02
.param RD_ISI                            = 0.155
.param WR_ISI                            = 0.16
.param CA_ISI                            = 0.15
.param DQSG_ISI                          = 0.15
.param WL_ISI                            = 0.078
.param X4                                = 1
.param RDBI                               = 0
.param WDBI                               = 0
.param MEM_VCC                           = 1.2
.param MEM_AC_FREQ_PS                  = 1250.0 - - - - - MEM_AC_FREQ_PS
.param MEM_DQ_FREQ_PS                  = 1250.0 - - - - - MEM_DQ_FREQ_PS
                                            (in ps, not MHz)
```

**Figure 163. Write (WR) IBIS Models and ODT Settings in an \*\_ip\_parameters.dat File**

```
.param MEM_CLK_IBIS                      = str('dsstl12_io_s2r40c_doff') - - - Mem CK
.param MEMCLK_COMP_IBIS                  = str('CLKIN')
.param MEMCLK_DIFF_SE                   = str('true')
.param MEMCLK_PHASE                     = 90
.param MEMAC_IBIS                        = str('sstl12_io_s2r40c_doff') - - - IBIS model
.param MEMAC_COMP_IBIS                  = str('INPUT')
.param MEMAC_RANKS                       = 1
.param MEMAC_COMPS_PER_RANK            = 4
.param MEM_DQ_RANKS                      = 2
.param FPGA_DQ_WR_IBIS                 = str('pod12_io_s2r40c_dh') - - - CA IBIS model
.param FPGA_DBI_WR_IBIS                = str('pod12_io_s2r40c_dh')
.param FPGA_DQS_WR_IBIS                = str('dpod12_io_s2r40c_dh')
.param FPGA_DQS_WR_PHASE               = 90
.param WR_MEM_TO_R0_DQ_IBIS             = str('DQ_IN_ODT240_3200') - - - DQ WR IBIS model
.param WR_MEM_TO_R1_DQ_IBIS             = str('DQ_IN_ODT40_3200')
.param WR_MEM_TO_R2_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_TO_R3_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T1_R0_DQ_IBIS             = str('DQ_IN_ODT40_3200')
.param WR_MEM_T1_R1_DQ_IBIS             = str('DQ_IN_ODT240_3200')
.param WR_MEM_T1_R2_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T1_R3_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T2_R0_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T2_R1_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T2_R2_DQ_IBIS             = str('NF_INPUT')
.param WR_MEM_T3_R3_DQ_IBIS             = str('NF_INPUT')
```

**Figure 164. Read (RD) IBIS Models and ODT Settings in an \*\_ip\_parameters.dat File**

```

.param FPGA_DQ_RD_IBIS = str('pod12_in_g60c') - - - - DQ RD IBIS models
.param FPGA_DBI_RD_IBIS = str('pod12_in_g60c') (DQ, DBI, DQS)
.param FPGA_DQS_RD_IBIS = str('dpod12_in_g60c')
.param FPGA_DQS_RD_PHASE = 0
.param RD_MEM_TO_R0_RD_DQ_IBIS = str('DQ_34_3200')
.param RD_MEM_TO_R1_RD_DQ_IBIS = str('DQ_IN_ODT120_3200') - - - - TDPC 2R ODT settings
.param RD_MEM_TO_R2_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_TO_R3_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T1_R0_DQ_IBIS = str('DQ_IN_ODT120_3200')
.param RD_MEM_T1_R1_DQ_IBIS = str('DQ_34_3200')
.param RD_MEM_T1_R2_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T1_R3_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T2_R0_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T2_R1_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T2_R2_DQ_IBIS = str('NF_INPUT')
.param RD_MEM_T3_R3_DQ_IBIS = str('NF_INPUT')

```

**Figure 165.** CA, WR, and RD Mask settings in an `*_ip_parameters.dat` File

```
.param AC_MASK_P1X = 0.16899999999999998
.param AC_MASK_P1Y = 0.51
.param AC_MASK_P2X = 0.16899999999999998
.param AC_MASK_P2Y = 0.6
.param AC_MASK_P3X = 0.16899999999999998
.param AC_MASK_P3Y = 0.69
.param AC_MASK_P4X = 0.831
.param AC_MASK_P4Y = 0.69
.param AC_MASK_P5X = 0.831
.param AC_MASK_P5Y = 0.6
.param AC_MASK_P6X = 0.831
.param AC_MASK_P6Y = 0.51
.param DQ_WR_MASK_P1X = 0.1795
.param DQ_WR_MASK_P1Y = 0.775
.param DQ_WR_MASK_P2X = 0.1795
.param DQ_WR_MASK_P2Y = 0.875
.param DQ_WR_MASK_P3X = 0.1795
.param DQ_WR_MASK_P3Y = 0.975
.param DQ_WR_MASK_P4X = 0.8205
.param DQ_WR_MASK_P4Y = 0.975
.param DQ_WR_MASK_P5X = 0.8275
.param DQ_WR_MASK_P5Y = 0.875
.param DQ_WR_MASK_P6X = 0.8205
.param DQ_WR_MASK_P6Y = 0.775
.param DQ_RD_MASK_P1X = 0.1919999999999995
.param DQ_RD_MASK_P1Y = 0.885
.param DQ_RD_MASK_P2X = 0.1919999999999995
.param DQ_RD_MASK_P2Y = 0.985
.param DQ_RD_MASK_P3X = 0.1919999999999995
.param DQ_RD_MASK_P3Y = 1.085
.param DQ_RD_MASK_P4X = 0.808
.param DQ_RD_MASK_P4Y = 1.085
.param DQ_RD_MASK_P5X = 0.808
.param DQ_RD_MASK_P5Y = 0.985
.param DQ_RD_MASK_P6X = 0.808
.param DQ_RD_MASK_P6Y = 0.885
```

## Example of WR Eye Mask Values

Eye height (EH) requirement:

- Vref  $\pm 100$  mV
  - Vref = P2Y/P5Y = 0.875 V (calculated from DC termination)
  - P2Y - P1Y = 100 mV

Eye width (EW) requirement:

- $P1X - P6X$  or  $P3X - P4X = 0.641 \text{ UI}$



**Figure 166. WR Mask Coordinate Values in an \*\_ip\_parameters.dat File**

```
.param DQ_WR_MASK_P1X      = 0.1795
.param DQ_WR_MASK_P1Y      = 0.775
.param DQ_WR_MASK_P2X      = 0.1795
.param DQ_WR_MASK_P2Y      = 0.875
.param DQ_WR_MASK_P3X      = 0.1795
.param DQ_WR_MASK_P3Y      = 0.975
.param DQ_WR_MASK_P4X      = 0.8205
.param DQ_WR_MASK_P4Y      = 0.975
.param DQ_WR_MASK_P5X      = 0.8275
.param DQ_WR_MASK_P5Y      = 0.875
.param DQ_WR_MASK_P6X      = 0.8205
.param DQ_WR_MASK_P6Y      = 0.775
```

### Example of RD Eye Mask Values

Eye height (EH) requirement:

- $V_{ref} \pm 100$  mV
- $V_{ref} = P2Y/P5Y = 0.985$  V (calculated from DC termination)
- $P2Y - P1Y = 100$  mV

Eye width (EW) requirement:

- $P1X - P6X$  or  $P3X - P4X = 0.616$  UI

**Figure 167. RD Mask Coordinate Values in an \*\_ip\_parameters.dat File**

```
.param DQ_RD_MASK_P1X      = 0.19199999999999995
.param DQ_RD_MASK_P1Y      = 0.885
.param DQ_RD_MASK_P2X      = 0.19199999999999995
.param DQ_RD_MASK_P2Y      = 0.985
.param DQ_RD_MASK_P3X      = 0.19199999999999995
.param DQ_RD_MASK_P3Y      = 1.085
.param DQ_RD_MASK_P4X      = 0.808
.param DQ_RD_MASK_P4Y      = 1.085
.param DQ_RD_MASK_P5X      = 0.808
.param DQ_RD_MASK_P5Y      = 0.985
.param DQ_RD_MASK_P6X      = 0.808
.param DQ_RD_MASK_P6Y      = 0.885
```

## 9.8. Multi-Rank Topology

The Intel Agilex 7 F-Series and I-Series FPGA EMIF I/O timing collateral supplies simulation deck variants that allow you to evaluate the termination settings for multi-rank memory interfaces. For DDR4 multi-rank interfaces, simulation scenarios are provided to target every rank in the system. Non-target ranks provide idle termination as specified by the RTT\_NOM and RTT\_PARK ODT values for in the EMIF IP as well as the ODT activation matrices.

The simulation decks for multi-rank designs use different top-level files:

- For write operations, the files `dq_2rank_wr_top.sp`, `dq_4rank_wr_top.sp` implement the simulation deck for 2-rank and 4-rank interfaces, respectively.
- For read operations, the files `dq_2rank_rd_top.sp` and `dq_4rank_rd_top.sp` implement the simulation decks.

The `ALTER` construct is used to modify the SPICE deck to exercise different target ranks, which facilitates parallel simulations in most SPICE simulators. You must evaluate the data eyes for compliance at each target rank.

## 9.9. Pin Parasitics

The IBIS support in SPICE does not permit using the RLC parasitic data embedded within the IBIS model, therefore a separate include file is provided for annotating pin parasitic data for both the memory-side and FPGA-side package pins. This include file is named `pin_parasitics.dat` and is initially set to 0 for all values of parasitic pin resistance, inductance and capacitance.

You can annotate the values in the `pin_parasitics.dat` file with the appropriate package parasitic information once the exact placement and package types are determined. (Note that this file assumes that the same memory component is used throughout the interface, meaning that only one set of RLC values per memory package pin is supported – 12 values for each pin in the Address/Command channel, and 12 values for each pin in the DQ channel.)

You can obtain pin parasitic information for Intel Agilex 7 F-Series and I-Series FPGAs from the Intel website. You can extract parasitic information for the memory models from the *Component* section of the IBIS model file. Note that the pins corresponding to the signals of the 12-line extractions for the AC and DQ paths should be used for both the memory and the FPGA.

## 9.10. Mask Evaluation

After you have completed each of the three simulations, eye diagrams must be generated at the receiver IBIS model for each of the victim pins; you must check these eye diagrams against compliance masks supplied by the IP. This topic explains how to capture the eye diagram.

**Note:** Final masks will be available in a later version of the Intel Quartus Prime software. To obtain compliance masks for ES revisions of Intel Agilex 7 F-Series and I-Series devices, you should contact Intel.

For each of the simulation decks listed below, the probe point for evaluation is located at the IBIS receiver node. This node can be found by navigating to the `lane_rx12` instance of the desired component/rank, expanding to find the victim `rx_buffer` instance, and locating the `ibis` node. The `data_out` nodes of the IBIS models are not used for eye diagram evaluation.

For SSTL and other center-tap-terminated I/O standards, the mask is centered about the termination voltage; however for POD I/O standards, you may need to shift the mask vertically to account for dynamically-calibrated reference voltages.

### Address/Command Eyes

You must evaluate the address/command data eye at every victim pin for every component in the interface, including all components in the fly-by chain, and every rank in the system. You should capture the eye for each component using a trigger of the crossing point of the rising and falling edges of the memory clock pin measured at the IBIS model nodes in the same component.

### Data Write Eyes

You must evaluate the write data eye at every victim pin for all possible target DQ ranks in the interface. You should capture the eye at the memory for each component, using a trigger of the crossing point of the rising and falling edge of the DQS write strobes measured at the IBIS model nodes in the same component.

### Data Read Eyes

You must evaluate the read data eyes using the victim pin at the FPGA, for all possible target DQ ranks in the interface. You should capture the eye using the crossing point of the rising and falling edges of the complementary DQS strobe as measured at the IBIS model nodes at the FPGA.

### Mask Properties

The masks supplied by the IP for address/command and FPGA write operations are constructed using the JEDEC specifications as a starting point. The masks are then enlarged to account for the following effects from the FPGA:

- Memory Clock / Transmit strobe jitter
- Worst-case FPGA package cross-talk effects
- Worst-case temperature drift variations
- Calibrated termination uncertainty
- Output delay chain variations
- Calibrated voltage reference uncertainty
- Package PDN effects
- Process variation
- Volume data collection

For FPGA read operations, the following additional effects are incorporated into the receiver mask:

- Input delay chain jitter
- Input delay chain INL/DNL error
- Calibration algorithm uncertainty

## 10. Intel Agilex 7 F-Series and I-Series FPGA EMIF IP – Controller Optimization

---

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.

### 10.1. 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 memory device has its own read latency, and the higher the CAS latency, the less efficient an individual access. 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.

## 10.2. Bank Management Efficiency

Bank management operation affects controller efficiency.

When a read operation reads changes from a row in a bank, it has an impact on efficiency, relative to the row in the bank remaining unchanged.

When a row in the bank is unchanged, the controller does not need to issue precharge and activate commands; by not issuing precharge and activate commands, the speed of the read operation is increased, resulting in better efficiency.

Similarly, if you do not switch between read and write frequently, the efficiency of your controller improves significantly.

## 10.3. 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 reduces 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.

## 10.4. Improving Controller Efficiency

You can use the following tools and methods to improve the efficiency of your controller.

- Auto-Precharge Commands
- Additive Latency
- Bank Interleaving
- User-Controlled Refresh
- Frequency of Operation
- Series of Reads or Writes
- Data Reordering
- Command Reordering

- Bandwidth
- Enable Command Priority Control
- Controller pre-pay and post-pay refresh

The following sections discuss these methods in detail.

#### 10.4.1. Auto-Precache 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-Precache 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, autoprecharge would keep a page open or closed. In a closed-page policy, a page is always closed after it is accessed with an 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-Precache Control**, you can instruct the controller to issue an auto-precharge 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 auto-precharge logic evaluates the pending commands in the command buffer and determines the most efficient auto-precharge operation to perform. The auto-precharge logic can reorder commands if necessary. When the TBP is occupied due to tracking an open page, the 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 171. 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	Activate required
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 the 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 153. 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%.

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.

#### 10.4.1.1. Using Auto-precharge to Achieve Highest Memory Bandwidth for DDR4 Interfaces

The following two examples illustrate how to achieve the best performance using the auto-precharge feature.

Regardless of how many DDR4 bank groups are open for a series of Avalon memory-mapped interface accesses to the controller, the auto-precharge takes effect only on the last beat of the Avalon memory-mapped interface burst.

In each of the following cases, you would use long bursts of sequentially addressed read or write traffic data patterns and the auto-precharge when an access is the last to a memory page. (A memory page is defined as a bank group, bank address and row address combination that is opened by a DDR4 activate command). Long bursts at the DDR4 memory can originate from either an Avalon access with a large burst size or sequentially addressed Avalon accesses of smaller burst sizes. The controller open page policy keeps the memory page open so it can sustain back-to-back accesses at the DDR4 memory.

##### Example A

The DDR4 IP is configured with the **Efficiency > Address** ordering parameter on the **Controller** tab. You can set this value to *CS-CID-Row-Bank-Col-BG* or *CID-Row-CS-Bank-Col-BG*.

Break your Avalon accesses to the DDR4 hard controller into sequentially addressed accesses with a burst size of 1. Four bank groups are used, and for the final four accesses, assert the auto-precharge signal so that all of the bank groups receive read or write with auto-precharge commands. DDR4 devices with x4 and x8 configurations have four bank groups. (DDR4 x16 devices have only two bank groups.)

##### Example B

The DDR4 IP is configured with the **Efficiency > Address** ordering parameter on the **Controller** tab set to *CS-BG-Bank-CID-Row-Col*. With this address ordering, only one memory page is opened and you can use Avalon burst accesses with burst sizes greater than one. For the last access in the burst, assert the auto-precharge signal.

#### 10.4.2. Additive Latency

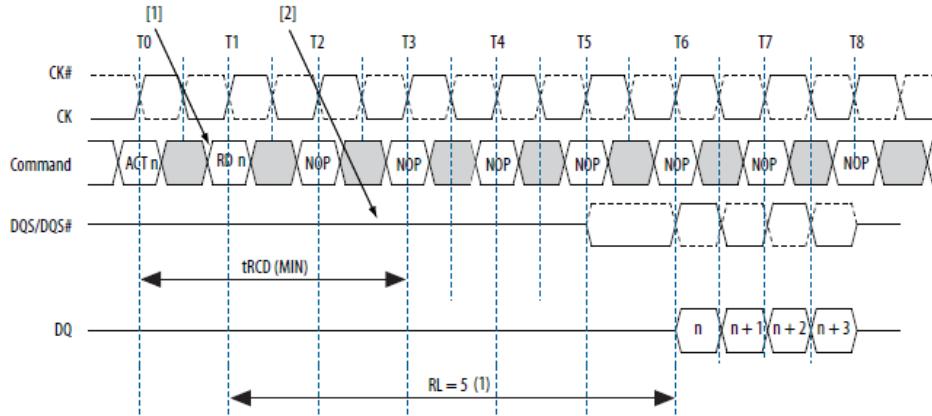
Additive latency increases the efficiency of the command and data bus for sustainable bandwidths.

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 173. 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 -  $t_{CK}$

#### 10.4.3. 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 DDR4, bank refers to a bank address and a bank group.

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.

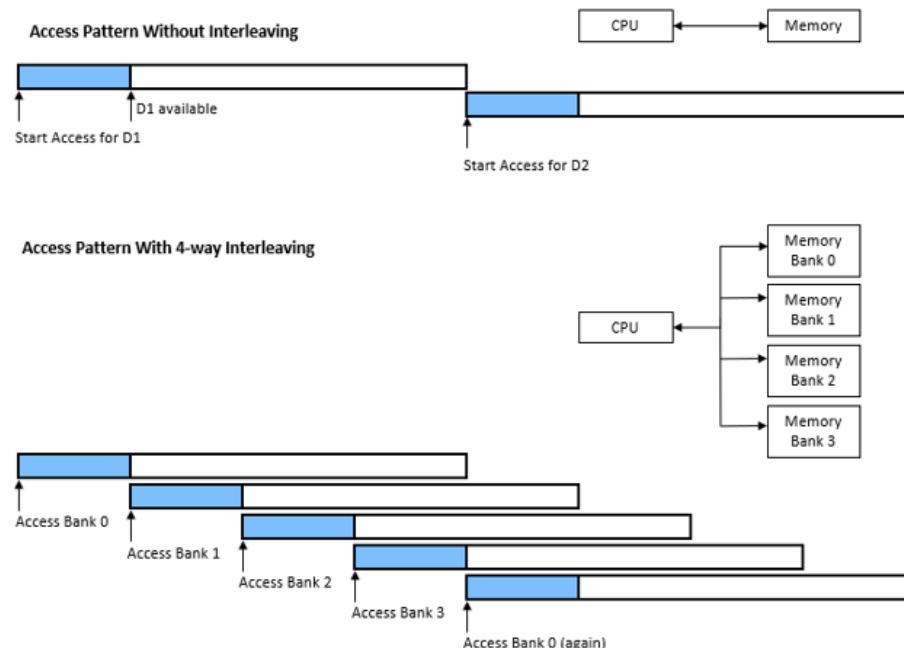
Interleaving allows banks of the SDRAM device to alternate their background operations and access cycles. One bank undergoes its precharge/activate cycle while another is being accessed. By alternating banks, the controller improves its performance by masking the precharge/activate time of each bank. 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 174. Using Interleaving to Increase Bandwidth**



The controller supports three interleaving options:

**CS-BG-Bank-CID-Row-Col** – This is a non-interleaved option. Select this option to improve efficiency with random traffic

**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.

**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  $t_{RC}$ . An optimal system can achieve 100% efficiency for bank interleave transactions with 8 banks. A system with less than 8 banks is unlikely to achieve 100%.

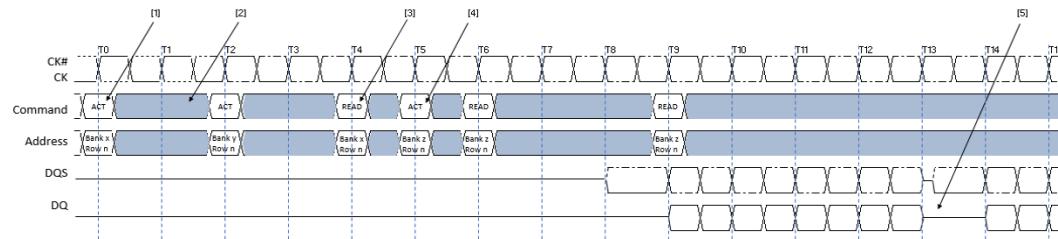
**Note:** Additive latency improves the efficiency of back-to-back interleaved reads or writes, but not individual random reads or writes.

#### 10.4.4. 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 175. Bank Interleaving—Without Additive Latency**



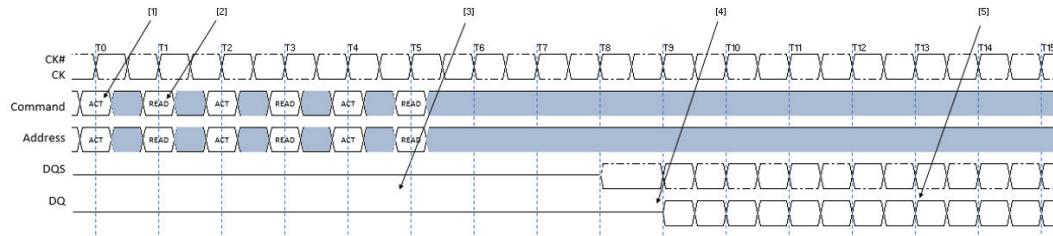
The following sequence of events describes the above figure:

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 176. 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. Within 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.

#### 10.4.5. 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.

#### 10.4.6. 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 are 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.

#### 10.4.7. Series of Reads or Writes

Performing a series of reads or writes from the same bank and row increases controller efficiency.

For best performance, minimize random reads and 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.

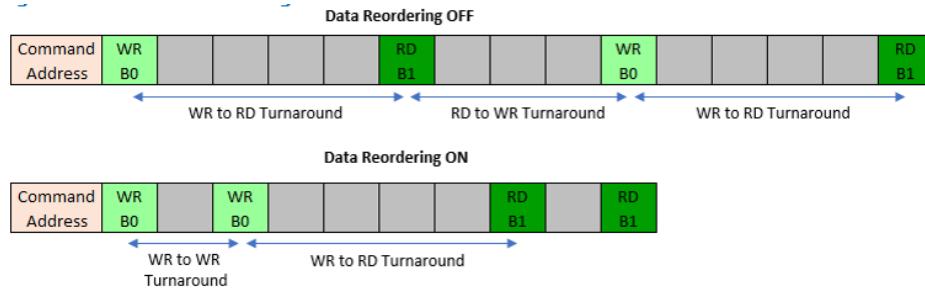
#### 10.4.8. Data Reordering

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.

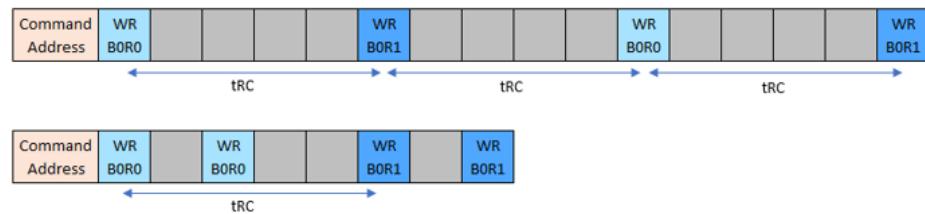
In the soft memory controller, inter-bank data reordering serves to minimize bus turnaround time by optimizing the ordering of read and write commands going to different banks; commands going to the same bank address are not reordered.

**Figure 177. 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; commands 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 178. Data Reordering for Minimum  $t_{RC}$**



#### 10.4.9. 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.

#### 10.4.10. 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 179. Comparison With and Without Command Reordering Feature**

Without Lookahead			With Lookahead		
Cycle	Command	Data	Cycle	Command	Data
1	ACT		1	ACT	
2	NOP		2		
3	NOP		3		
4	READ		4	READ	
5		DATA0 (Burst 0, Burst 1)	5	ACT	DATA0 (Burst 0, Burst 1)
6	NOP	DATA0 (Burst 2, Burst 3)	6	NOP	DATA0 (Burst 2, Burst 3)
7	ACT	DATA0 (Burst 4, Burst 5)	7	ACT	DATA0 (Burst 4, Burst 5)
8	NOP	DATA0 (Burst 6, Burst 7)	8	READ	DATA0 (Burst 6, Burst 7)
9	NOP	Wasted Cycle	9	NOP	DATA1 (Burst 0, Burst 1)
10	READ	Wasted Cycle	10	NOP	DATA1 (Burst 2, Burst 3)
11		DATA1 (Burst 0, Burst 1)	11	NOP	DATA1 (Burst 4, Burst 5)
12	NOP	DATA1 (Burst 2, Burst 3)	12	READ	DATA1 (Burst 6, Burst 7)
13	ACT	DATA1 (Burst 4, Burst 5)	13	NOP	DATA2 (Burst 0, Burst 1)
14	NOP	DATA1 (Burst 6, Burst 7)	14	NOP	DATA2 (Burst 2, Burst 3)
15	NOP	Wasted Cycle	15	NOP	DATA2 (Burst 4, Burst 5)
16	READ	Wasted Cycle	16	NOP	DATA2 (Burst 6, Burst 7)
17	NOP	DATA2 (Burst 0, Burst 1)			
18	NOP	DATA2 (Burst 2, Burst 3)			
19	NOP	DATA2 (Burst 4, Burst 5)			
20	NOP	DATA2 (Burst 6, Burst 7)			

Command	Address	Condition
Read	Bank 0	Activate required
Read	Bank 1	Precharge required
Read	Bank 2	Precharge required

Compare the following efficiency results for the above figure:

**Table 154. 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.

#### 10.4.11. 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).

DRAM typically has an efficiency of around 70%, but when you use the memory controller, efficiency can vary from 10 to 92%.

The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory.

#### 10.4.12. 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.

#### 10.4.13. Controller Pre-pay and Post-pay Refresh (DDR4 Only)

There is some flexibility in the refresh interval for postponing or hastening refresh commands; you can use this flexibility to achieve improved controller efficiency in scheduling and switching between tasks. A maximum of 8 refresh commands can be postponed or hastened at one time for DDR4; the maximum interval between the surrounding refresh commands is  $9 \times t_{REFI}$ .

You can postpone or hasten a refresh command using the **Enable controller post-pay refresh** and **Enable controller pre-pay refresh** parameters on the **Controller** tab in the parameter editor. You can select a lower limit and upper limit for the post-pay refresh, and an upper limit for the pre-pay refresh. The combined total of post-pay refresh and pre-pay refresh upper limit cannot exceed the refresh command limit of 8.

For example, a maximum of 8 refreshes in a row may be postponed for a design with the post-pay refresh upper limit set to 8. The accumulated refresh commands are issued continuously when the upper limit is reached. For the pre-pay refresh, the refresh command can be issued in advance or hastened depending on the specified refresh pre-pay upper limit. The refresh command is issued opportunistically when there is no traffic. When you have enabled both pre-pay and post-pay refreshes, the post-pay policy takes priority, and the refreshes are always accumulated. When post-pay refreshes have been opportunistically drained and there is no traffic, pre-pay refresh commands are then issued.

For traffic pattern in blocks, you must calculate the block time and schedule the refresh in the gaps between the blocks. You should select the pre-pay or pre-pay refresh limits (or both) that suit the design. For designs implementing non-stop access or random traffic, you should simulate the traffic pattern and check for the efficiency improvement using these settings. If you prefer to enable this feature while implementing random traffic, the controller efficiency might increase with a post-pay refresh lower limit of 1 and pre-pay refresh plus post-pay refresh upper limit combination of 8.

## 11. Intel Agilex 7 F-Series and I-Series FPGA 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](#).

### 11.1. 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)

#### 11.1.1. 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.

---

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

\*Other names and brands may be claimed as the property of others.

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 vendor's 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.

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.

## 11.2. 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.

### 11.2.1. 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.

### 11.2.2. 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.

### 11.2.3. 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 sheets 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 indicates only 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.

#### 11.2.4. 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 logic 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.

## 11.3. 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.

### 11.3.1. Evaluating FPGA Timing Issues

Usually, you should not encounter timing issues with Intel-provided IP unless your design exceeds Intel's 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. Configure the Settings as follows, to help close timing in the design:

- a. On the **Assignments** menu click **Settings**.  
b. In the **Category** list, click **Compiler Settings**.  
c. Select **Optimization mode > Performance > High Performance Effort**.
  - a. On the **Assignments** menu click **Settings**.  
b. In the **Category** list, click **Compiler Settings > Advanced Settings (Synthesis)**..  
c. For **Optimization Technique**, select **Speed**.
  - a. On the **Assignments** menu click **Settings**.  
b. In the **Category** list, click **Compiler Settings > Advanced Settings (Fitter)**.  
c. For **Physical Placement Effort**, select **High Effort/Maximum Effort**.
3. Use **Timing Analyzer Report Ignored Constraints**, to ensure that .sdc files are successfully applied.
4. 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 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.

### 11.3.2. 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, to reduce the loading.

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.

## 11.4. Verifying Memory IP Using the Signal Tap Logic Analyzer

The Signal Tap logic analyzer shows read and write activity in the system.

For more information about using the Signal Tap logic analyzer, refer to the *Intel Quartus Prime Pro Edition User Guide: Debug Tools*.

To add the Signal Tap logic analyzer, follow these steps:

1. On the Tools menu click **Signal Tap 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 \* emif\_usr\_clk) in the **Named** box, for **Filter** select **Signal Tap: 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**

#### 11.4.1. Signals to Monitor with the Signal Tap 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:

- status\_local\_cal\_success
- status\_local\_cal\_fail
- local\_reset\_done
- local\_reset\_req
- pll\_locked
- pnf\_per\_bit\_persist
- pnf\_per\_bit
- amm\_write
- amm\_writedata
- amm\_read
- amm\_readdata
- amm\_readdatavalid
- amm\_address

### 11.5. Hardware Debugging Guidelines

Before debugging your design, confirm that it follows the recommended design flow. Refer to the *Intel Agilex 7 F-Series and I-Series EMIF IP Design Flow* section in chapter 1 of this user guide.

Always keep a record of tests, to avoid repeating the same tests later. To start debugging the design, perform the following initial steps.

#### 11.5.1. 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.

#### 11.5.2. 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.

Confirm that reference voltages ( $V_{REF\_CA}$ ) and termination voltages are active and within specification.

### 11.5.3. 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.

### 11.5.4. 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.

### 11.5.5. 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, DDR4 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

### 11.5.6. 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.

### 11.5.7. 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.

### 11.5.8. 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.

### 11.5.9. 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.

### 11.5.10. Debugging Checklist

The following checklist is a good starting point when debugging an external memory interface.

**Table 155. Checklist**

Check	Item
<input type="checkbox"/>	Try a different fit.
<input type="checkbox"/>	Check IP parameters at the operating frequency ( $t_{MRD}$ , $t_{WTR}$ for example).
<input type="checkbox"/>	Ensure you have constrained your design with proper timing derivation and have closed timing.
<input type="checkbox"/>	Simulate the design. If it fails in simulation, it will fail in hardware.
<input type="checkbox"/>	Analyze timing.
<input type="checkbox"/>	Place and assign $R_{ZQ}$ (OCT).
<input type="checkbox"/>	Measure the power distribution network (PDN).
<input type="checkbox"/>	Measure signal integrity.
<input type="checkbox"/>	Measure setup and hold timing.
<input type="checkbox"/>	Measure FPGA voltages.
<input type="checkbox"/>	Vary voltages.
<input type="checkbox"/>	Heat and cool the PCB.
<input type="checkbox"/>	Operate at a lower or higher frequency.
<input type="checkbox"/>	Check board timing and trace Information.
<input type="checkbox"/>	Check LVDS and clock sources, I/O voltages and termination.
<input type="checkbox"/>	Check PLL clock source, specification, and jitter.
<input type="checkbox"/>	Retarget to a smaller interface width or a single bank.

### 11.6. Categorizing Hardware Issues

The following topics divide issues into categories. By determining the category (or categories) in which an issue belongs, 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

## 11.6.1. 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.

### 11.6.1.1. 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.

### 11.6.1.2. Evaluating Signal Integrity Issues

Signal integrity problems can only really be evaluated in two ways:

- direct measurement using suitable test equipment like an oscilloscope and probe
- simulation using a tool like HyperLynx or Allegro PCB SI

Compare signals to the respective electrical specification. You should look for overshoot and undershoot, non-monotonicity, eye height and width, and crosstalk.

#### 11.6.1.2.1. 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, DDR4-3200 at 1600 MHz has a data valid window that is smaller than 313 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.

#### 11.6.1.2.2. 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.

#### 11.6.1.2.3. 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.

#### 11.6.1.2.4. 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.

#### 11.6.1.2.5. Address and Command Signals

Confirm that address and command signals are reaching the memory devices correctly.

After the memory interface has been successfully calibrated, you can probe the ALERT\_N pin to determine if any memory component has encountered an address and command parity error.

#### 11.6.1.2.6. 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.

#### 11.6.1.2.7. 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.

#### 11.6.1.2.8. 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.

For the FPGA, ensure that you perform the following:

- Connect the R<sub>ZQ</sub> pin to the correct resistors (either a 240 Ω or 100 Ω resistor, depending on the desired OCT impedance) and pull-down to ground in the schematic or PCB.
- Contain the R<sub>ZQ</sub> 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<sub>ZQ</sub> pins to ensure that they are on the correct pins, and that only the correct number of calibration blocks exists in your design.
- Check in the fitter report that the input, output, and bidirectional signals with calibrated OCT all have the termination control block applicable to the associated R<sub>ZQ</sub> pins.

For the memory components, ensure that you perform the following:

- Connect the required resistor to the correct pin on each and every component, and ensure that it is pulled to the correct voltage.
- Place the required resistor close to the memory component.
- Correctly configure the IP to enable the desired termination at initialization time.
- Check that the speed grade of memory component supports the selected ODT setting.
- Check that the second source part that may have been fitted to the PCB, supports the same ODT settings as the original.

## 11.6.2. Hardware and Calibration Issues

Hardware and calibration issues have the following definitions:

- Calibration issues result in calibration failure, which usually causes the emif\_fm\_0\_status\_local\_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.

### 11.6.2.1. 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.

### 11.6.2.2. 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 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 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.

#### 11.6.2.3. Memory Timing Parameter Evaluation

Review and update the memory timing parameters, 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 parameters, CAS, or Write CAS latency can cause data corruption in the memory component.

#### 11.6.2.4. Verify that the Board Has the Correct Memory Component or DIMM Installed

Verify that the correct memory component or DIMM is installed on the circuit board.

If an incorrect memory part is used, the memory part may not support the memory timing parameters, CAS, or Write CAS latency used in parameterizing the IP; this situation can result in data corruption. If a memory device with smaller memory capacity is installed incorrectly, data written to the higher address space overwrites the data at the lower address space.

### 11.7. Debugging with the External Memory Interface Debug Toolkit

The External Memory Interface Debug Toolkit for Intel Agilex 7 F-Series and I-Series FPGAs provides access to data collected by the Nios II sequencer during memory calibration, as well as analysis tools to evaluate the stability of the calibrated interface and assess hardware conditions.

The toolkit provides the following reports:

- Interface and memory configuration, such as external memory protocol and interface width.
- Calibration results including calibration status (pass or fail), calibration failure stage (if applicable), delay settings and margins, as well as  $V_{REF}$  settings and margins.

The available task and analysis capabilities include 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 design example.
- Running  $V_{REF}$  Margining on the interface.
- Running Driver Margining on the interface.
- Calibrating or updating the termination settings (or both).

### 11.7.1. Prerequisites for Using the EMIF Debug Toolkit

Certain prerequisites — including generating a design example — must be completed before you can use the EMIF Debug Toolkit.

Complete the following steps to enable the toolkit:

1. Configure your design example to use the EMIF Debug Toolkit, as described in the following topics.
2. Compile your design.
3. Program the target device with the resulting SRAM Object File (.sof).

After completing the above steps, you are ready to run the EMIF Debug toolkit.

### 11.7.2. Configuring a Design to Use the Toolkit

Perform the following tasks to configure your design for use with the toolkit.

#### 11.7.2.1. Generating a Design Example with the Debug Toolkit

To enable the Debug Toolkit in the design example, follow these steps:

1. Navigate to the **Diagnostics** tab of the parameter editor for your EMIF IP.
2. Click **Intel Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port > Add EMIF Debug Interface**.
3. Select **Enable In-System Sources and Probes**.
4. After you have fully parameterized the interface, click **Generate Example Design**.

The generated design example has the debug toolkit enabled and all the necessary components wired up, as required for a single interface.

### Related Information

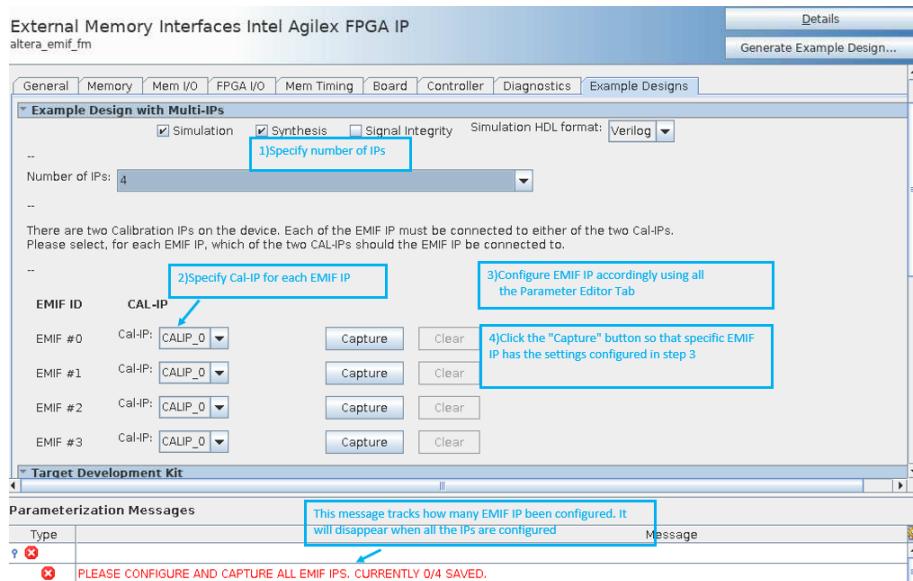
[Enabling the EMIF Toolkit in an Existing Design](#) on page 279

#### 11.7.2.2. Creating a Design Example with Multiple External Memory Interfaces

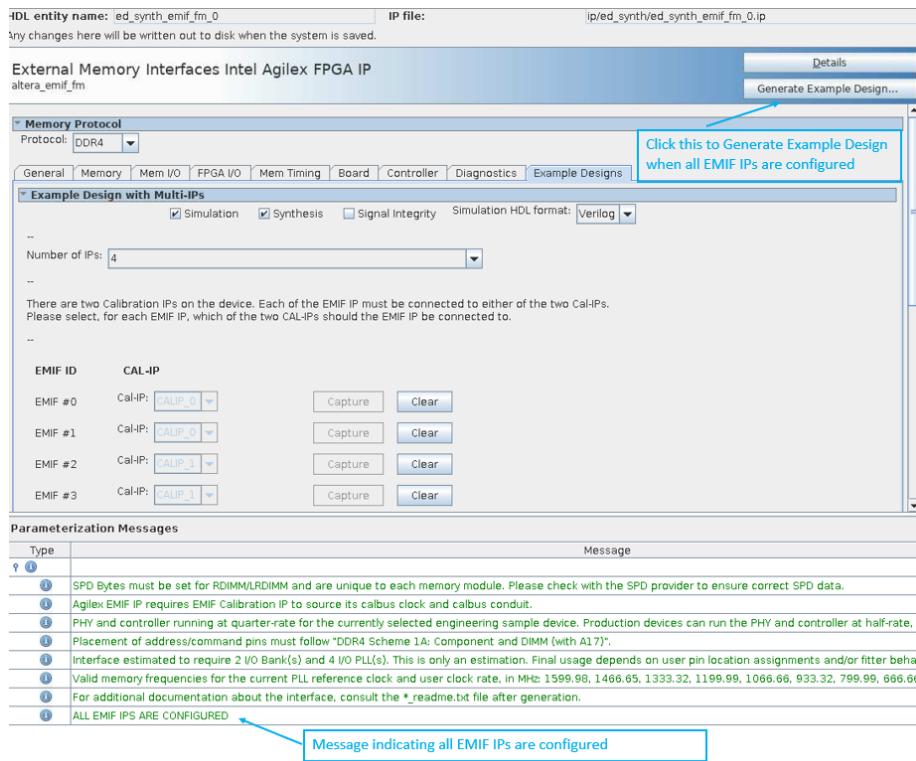
To create a design example with two or more external memory interfaces, follow these steps:

1. On the **Example Designs** tab, specify the number of external memory interfaces in the design example..
2. Select the **Cal-IP** to specify the connection of each interface to the Calibration IP.  
*Note:* Ensure that any interfaces located in the same I/O row are connected to the same Cal-IP.
3. Parameterize the interface as you normally would.
4. After you have fully parameterized the interface, return to the **Example Designs** tab and click **Capture** on the desired **EMIF ID**, so that the EMIF IP has the parameter settings established in step 3.
5. Repeat steps 2 through 4 for all EMIF IDs. (If you want to make changes to the EMIF IP, you can click the **Clear** button to remove the captured parameters, and then repeat steps 2 to 4.)

**Figure 180. Generating a Design Example with Multiple EMIFs**



**Figure 181. Generating a Design Example After All EMIFs are Configured**



6. After you have configured all the EMIF IPs, generate the design example by clicking **Generate Example Design** in the upper-right corner of the window as shown in the figure above.
7. Add pin assignments for all the EMIF IPs.
8. Compile the design.

**Note:** If you choose to manually add multiple EMIF instances to your design using Platform Designer, ensure that you do the following:

- Use the ed\_synth.qsys file generated for the design example as a starting point to make the necessary edits.
- Use a single .qsys file to create the connections for all the external memory interfaces in the design. (The system must have a flat hierarchy to work with the Calibration Debug Toolkit.)

#### Related Information

[Enabling the EMIF Toolkit in an Existing Design](#) on page 279

### 11.7.2.3. Enabling the EMIF Toolkit in an Existing Design

To enable 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 instance of EMIF in the design, set **Intel Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port > Add EMIF Debug Interface**.
3. In the Calibration IP, ensure that the value for **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** is set to *Add EMIF Debug Interface*.
4. Generate HDL and compile your design.

**Note:**

If your design was not generated based on the design example, you must regenerate the design beginning with a design example. Refer to the instructions in *Generating a Design Example with the Debug Toolkit*.

**Related Information**

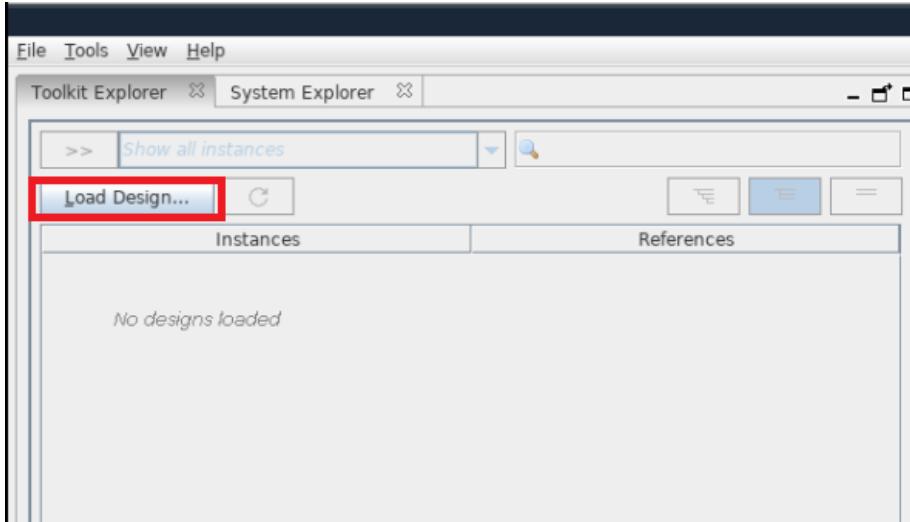
- [Generating a Design Example with the Debug Toolkit](#) on page 277
- [Creating a Design Example with Multiple External Memory Interfaces](#) on page 278

### 11.7.3. Launching the EMIF Debug Toolkit

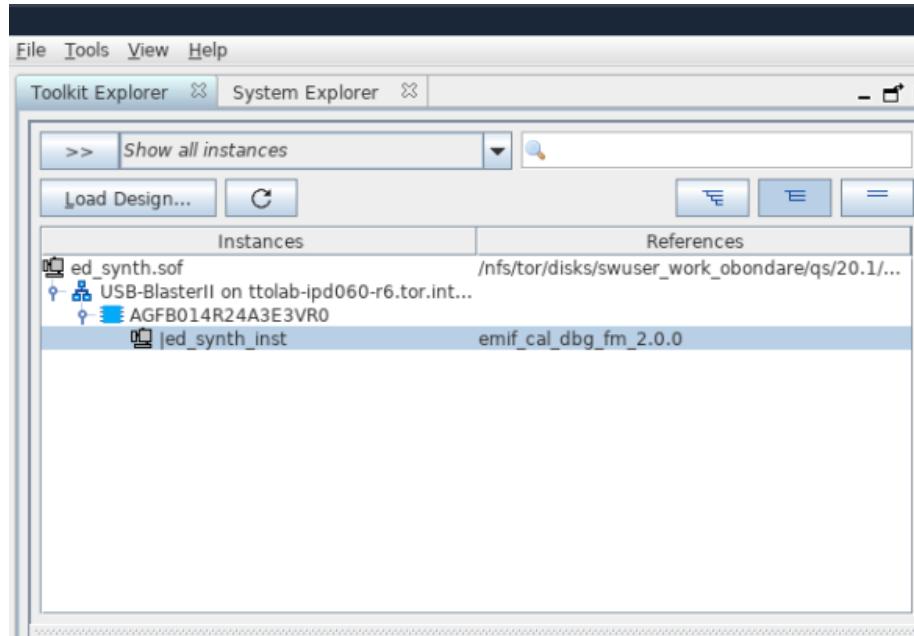
Before launching the EMIF Debug Toolkit, ensure that you have configured your device with a programming file that has the EMIF Debug Toolkit enabled. Refer to [Configuring a Design to Use the Debug Toolkit](#).

To launch the EMIF Debug Toolkit, follow these steps:

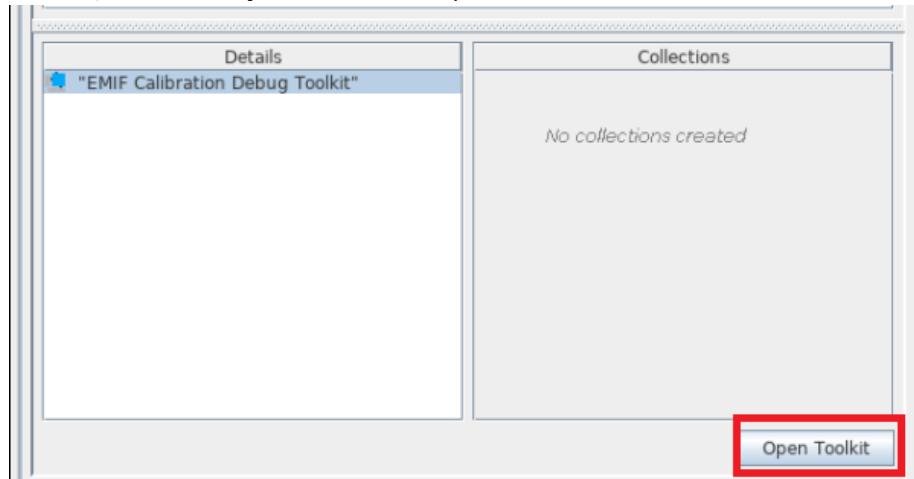
1. In the Intel Quartus Prime software, open the System Console by clicking **Tools > System Debugging Tools > System Console**.
2. In the System Console, load the SRAM Object File (.sof) with which you programmed the board in [Prerequisites for Using the EMIF Debug Toolkit](#).



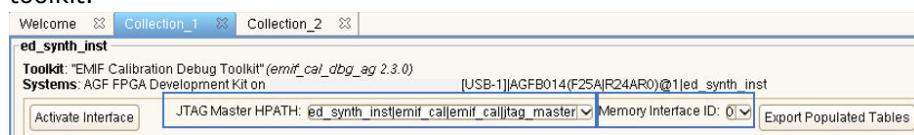
3. Select the correct instance to debug:



4. Select the **EMIF Calibration Debug Toolkit**, which appears in the **Details** section, and click **Open Toolkit** to open the main view of the toolkit.



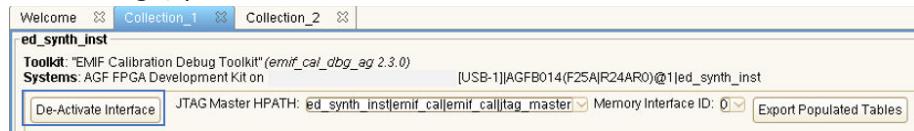
5. If there are multiple EMIF instances in the programmed design, select the column (path to JTAG master) and ID of the EMIF instance for which to activate the toolkit.



6. Click **Activate Interface** to allow the toolkit to read the parameters and status for the selected interface, and to perform analysis tasks.



7. You must debug one interface at a time; therefore, to connect to another interface in the design, you must first deactivate the current interface.



## 11.7.4. Using the EMIF Debug Toolkit

The Main View of the EMIF Debug Toolkit contains the **Memory Configuration**, **Calibration**, **Calibration Report**, **Calibrate Termination**, **Vref Margining**, **Driver Margining**, **ISSP**, and **Pin Delay Settings** tabs.

### 11.7.4.1. Memory Configuration Tab

The **Memory Configuration** tab shows the IP settings, which were defined when you parameterized the EMIF IP.

**Figure 182. Memory Configuration Tab**

Parameter	Value
Memory Type	DDR4
Dimm Type	SODIMM
Controller Type	Hard Memory Controller
PHY Clock Frequency (MHz)	300.0
IP Rate	Quarter Rate
Number of Ranks	1
Number of DIMMs	1
Number of DQ Pins (Data Width)	16
Number of Write DQS Groups	2
Number of Read DQS Groups	2
Number of DM/DBI Pins	2
Burst Length	8
Read Latency	20
Write Latency	16
Address Width	17
Bank Address Width	2
Bank Group Width	2
Chip Select Width	1
Clock Enable Width	1
ODT Width	1
CK Width	1

### 11.7.4.2. Calibration Tab

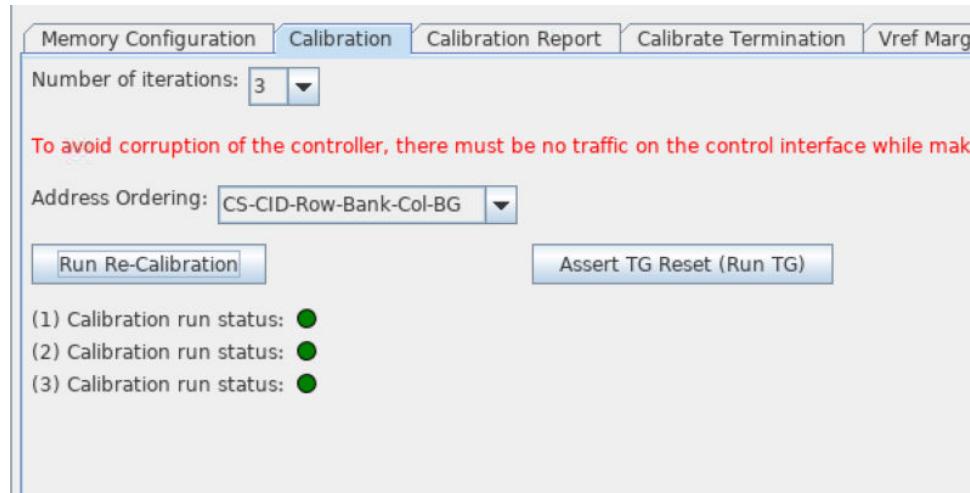
The Calibration tab allows you to rerun calibration and the traffic generator.

#### 11.7.4.2.1. Rerunning Calibration

The **Calibration** tab lets you specify a number of iterations by which to rerun calibration.

To rerun calibration, follow these steps:

1. Select the desired number of iterations from the **Number of iterations** pull-down menu.
2. Click **Run Re-Calibration** to repeat calibration the specified number of times.

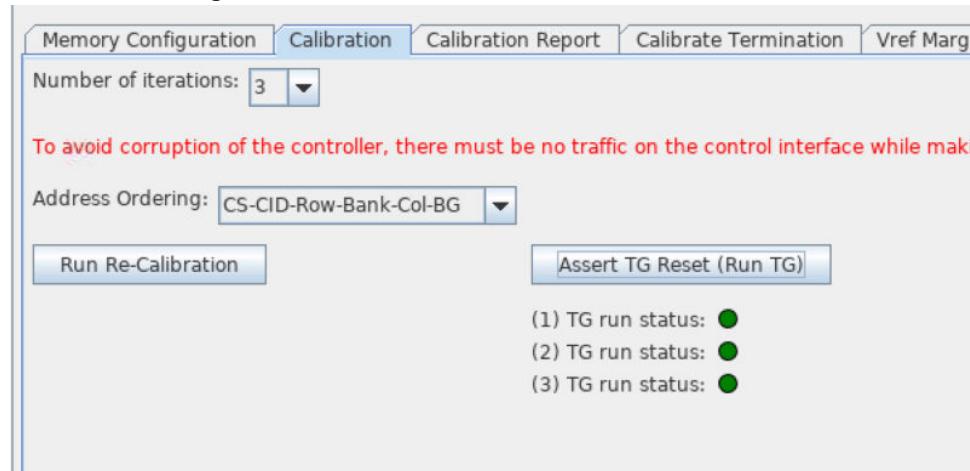


The system reports the *Calibration run status* for each iteration of calibration. A green dot indicates a pass, while a red dot indicates a calibration failure.

#### 11.7.4.2.2. Rerunning the Traffic Generator

You can rerun the traffic generator.

To rerun the Traffic Generator, select the desired number of iterations from the **Number of Iterations** pulldown menu, and click **Assert TG Reset (Run TG)** to rerun the traffic generator with each iteration.



The system reports the *TG run status* for each iteration of the traffic generator. A green dot indicates a pass, while a red dot indicates a failure.

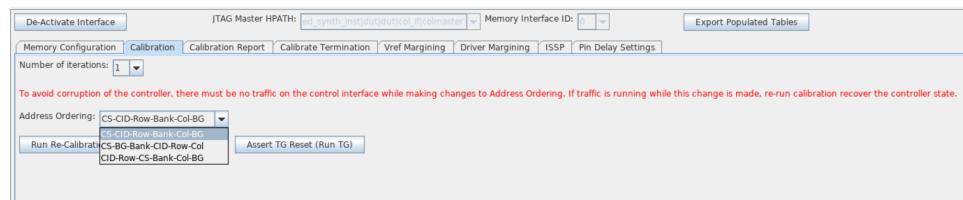
**Note:**

The **Assert TG Reset (Run TG)** button is available only when the default traffic generator is instantiated in the design example. If the EMIF control interface is driven by TG2 or by custom user logic, the **Assert TG Reset (Run TG)** button is not available. If you use TG2 in the design example, you may run it through the EMIF TG Configuration Toolkit.

### Changing Address Ordering

To change the address ordering, you can select one of the settings from the **Address Ordering** dropdown menu.

**Figure 183. Address Ordering Menu**



#### 11.7.4.3. Guidelines for Debugging Calibration Issues

The following topics provide general guidelines for debugging calibration-related issues.

##### General Hardware Debugging for Calibration Issues

1. Verify that the correct memory component or DIMM is installed on the circuit board.
2. 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.
3. 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.
4. Ensure that the PCB has correct termination resistors on the address and command signals. Refer to the Board Layout Guidelines section of this user guide for more information on suggested termination values.
5. If you implement the EMIF interface on I/O lanes in Bank 3A that are utilized for AVST x16 or AVST x32 configuration mode, ensure that the MSEL pins are not set to AVST x16 or AVST x32 configuration. These configuration modes utilize the I/O lane in the top sub-bank 3A and may cause EMIF calibration issues if the EMIF interface uses the same I/O lane as the AVST x16 or AVST x32 configuration mode.
6. 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.
7. 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.

8. For DDR4 discrete memory components, the TEN pin on the memory component must not be left floating. If the TEN feature is not used, connect the TEN pin directly to GND. If the TEN feature is used, connect the pin to GND through a  $1\text{K}\Omega$  resistor.
9. 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.
10. 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 DDR4 memory device.
11. Ensure that the reset signal to the DDR4 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.
12. Check to determine whether the calibration problem exists on more than one board.
13. 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. Refer to [Intel Agilex 7 F-Series and I-Series F-Series and I-Series FPGA EMIF IP – I/O Timing Closure](#) for more information.
14. Repeat the calibration multiple times without reconfiguring the device, to see whether the calibration can recover by recalibrating the interface.
15. Rerun the calibration by reconfiguring the device to see whether the calibration can recover after reconfiguring the device.

#### **11.7.4.3.1. 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.

#### **11.7.4.3.2. 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.

#### **11.7.4.3.3. 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 of 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.

#### 11.7.4.3.4. 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).

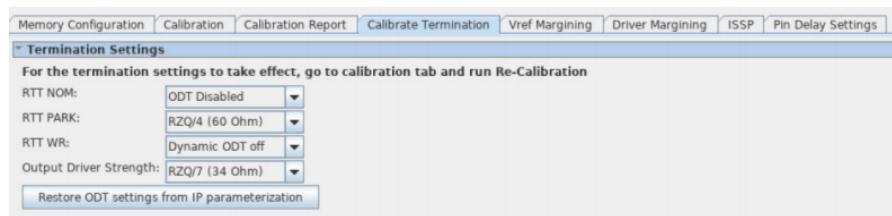
#### 11.7.4.3.5. Debugging Read Deskew Calibration Failure

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<sub>tt</sub> nominal value = ODT Disable.
  - Output drive strength setting = RZQ/7 (34 ohm)
  - R<sub>tt</sub> Park = RZQ /3 (80 Ohm)

If you have enabled the Debug Toolkit in your design, you can change the above settings on the *Calibrate Terminations* tab, without recompiling your design.

**Figure 184. Changing Termination Settings with the Debug Toolkit**

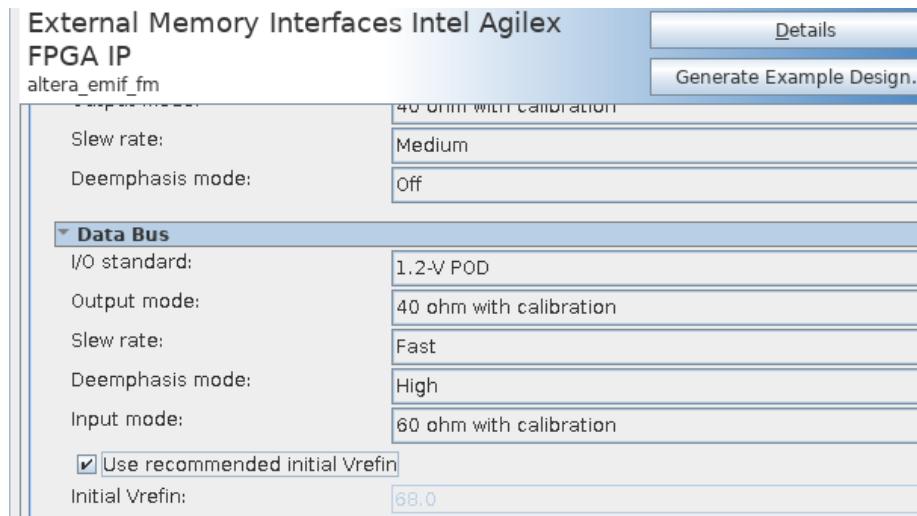


**Figure 185. Changing Termination Setting when Regenerating EMIF IP – Recompilation Required**



#### 11.7.4.3.6. 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 68% when using the default FPGA I/O settings.

**Figure 186. Changing the Initial V<sub>REFIN</sub> Value**

#### 11.7.4.3.7. 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.

#### 11.7.4.3.8. 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.

#### 11.7.4.3.9. Debugging Write Deskew Calibration Failure

If write deskew calibration is failing, perform the same checks as for [Debugging Read Deskew Calibration Failure](#).

#### 11.7.4.3.10. Debugging V<sub>REFOUT</sub> Calibration Failure

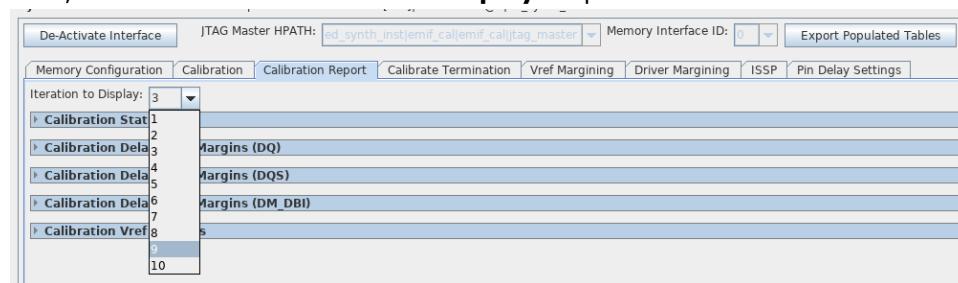
1. Ensure the address and command pins are connected correctly and that every calibrated pin has sufficient margin.
2. Ensure that the V<sub>REFCA</sub> pins on the DDR memory component are powered up to 0.6V.

#### 11.7.4.4. Calibration Report Tab

The **Calibration Report** tab shows calibration status.

##### Choosing the Iteration to View

You may choose to view the Status, Delay Settings, or Margins reports for any of the most recent calibration iterations initiated through the toolkit. To select the iteration to view, select from the **Iteration to Display** dropdown menu.



##### ODT Settings in Effect

This report shows the ODT settings for the latest calibration.

**Figure 187. ODT Settings**

RTT NOM	RTT PARK	RTT WR	Output Driver Strength
RZQ/4 (60 Ohm)	Park ODT off	Dynamic ODT off	RZQ/7 (34 Ohm)

##### Calibration Status Report

The **Calibration Status Report** displays 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 the data groups that failed at this stage. Memory parity status observed during calibration is shown for DDR4 interfaces if ISSPs are enabled in the design. The calibration status report window contains a status indicator for memory parity and a button that allows you to reread memory parity status.

Parameter	Value
error_code	SUCCESS
error_stage	N/A
error_group	0x00000000

## Calibration Delays and Margins Reports

These 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\_DB, V<sub>ref</sub>, or Address/Command.

**Figure 188. DQ Margins**

The screenshot shows the 'Calibration Status' interface with the 'Calibration Delays and Margins (DQ)' section expanded. The table lists DQ values from 0 to 15 along with their corresponding Read Margin (ps), Read Delay (taps), Write Margin (ps), and Write Delay (taps). The data is as follows:

DQ	Read Margin (ps)	Read Delay (taps)	Write Margin (ps)	Write Delay (taps)
0	-162 to 166	16	-182 to 182	1081
1	-162 to 162	17	-169 to 175	1085
2	-162 to 162	19	-175 to 175	1083
3	-159 to 162	16	-175 to 182	1083
4	-166 to 166	22	-175 to 175	1081
5	-162 to 166	19	-175 to 175	1081
6	-166 to 166	21	-175 to 182	1081
7	-166 to 169	22	-169 to 169	1086
8	-166 to 166	21	-175 to 175	1083
9	-169 to 172	17	-182 to 188	1081
10	-162 to 166	21	-175 to 175	1082
11	-169 to 169	17	-182 to 188	1084
12	-166 to 166	16	-175 to 182	1081
13	-169 to 172	16	-188 to 195	1082
14	-166 to 169	19	-175 to 182	1080
15	-169 to 169	16	-182 to 188	1082

**Figure 189. DQS Margins**

The screenshot shows the 'Calibration Status' interface with the 'Calibration Delays and Margins (DQS)' section expanded. The table lists DQS values from 0 to 1 along with their corresponding Read Margin (ps), Read Delay (taps), Write Margin (ps), and Write Delay (taps). The data is as follows:

DQS	Read Margin (ps)	Read Delay (taps)	Write Margin (ps)	Write Delay (taps)
0	-175 to 179	72	-169 to 169	1113
1	-179 to 179	69	-175 to 175	1112

**Figure 190. DM\_DB Margins**

The screenshot shows the 'Calibration Status' interface with the 'Calibration Delays and Margins (DM\_DB)' section expanded. The table lists DM\_DB values from 0 to 1 along with their corresponding Read Margin (ps), Read Delay (taps), Write Margin (ps), and Write Delay (taps). The data is as follows:

DM/DBI	Read Margin (ps)	Read Delay (taps)	Write Margin (ps)	Write Delay (taps)
0	-175 to 179	17	-195 to 195	1081
1	-179 to 179	14	-195 to 188	1079

**Figure 191. V<sub>ref</sub> Settings**

Group	VREFIN margin	VREFIN setting (V)	VREFOUT margin	VREFOUT setting (V)
0	0.540V to 0.930V	0.813	0.736V to 1.110V	0.962
1	0.540V to 0.930V	0.805	0.720V to 1.110V	0.985

**Figure 192. Address/Command Margins**

Pin Name	AC Margin (ps)	Delay (taps)
CKE_0	Not explicitly calibrated	1951
ODT_0	Not explicitly calibrated	1951
RESET	Not explicitly calibrated	1951
ACT	-410 to 410	1954
CS_0	-384 to 384	1951
CS_1	Not explicitly calibrated	1951
BA_0	-390 to 390	1951
BA_1	-390 to 390	1949
BG_0	-384 to 384	1950
ADD_0	-377 to 377	1951
ADD_1	-377 to 377	1951
ADD_2	-390 to 390	1951
ADD_3	-390 to 390	1951
ADD_4	-390 to 390	1951
ADD_5	-384 to 384	1950
ADD_6	-384 to 384	1950
ADD_7	-397 to 397	1950
ADD_8	-377 to 377	1951
ADD_9	-390 to 390	1951
ADD_10	-397 to 397	1952

These reports can also be viewed in a graphical format. Refer to [Viewing Reports Graphically in the Eye Viewer](#).

#### 11.7.4.5. Calibrate Termination Tab

The **Calibrate Termination** tab allows you to update the **RTT\_NOM**, **RTT\_PARK**, **RTT\_WR**, and **Output Drive Strength** termination settings without having to recompile or reprogram the design.

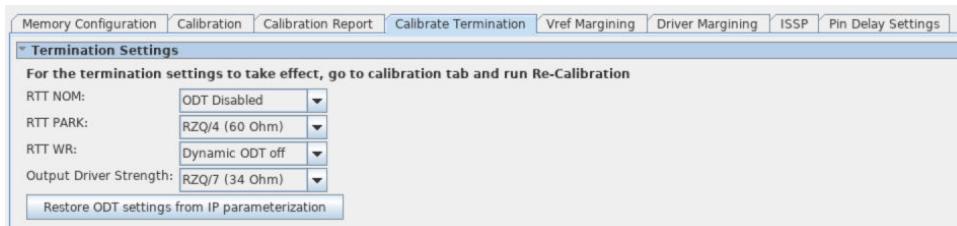
In addition, the **Calibrate ODT** feature lets you determine the optimal **On-Die Termination** and **Output Drive Strength** settings for your memory interface.

- Press the **Calibrate ODT** button. The system runs calibration with the cross product of the termination settings you want to sweep, and displays 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. (This option is disabled when the traffic generator is enabled in the Design Example.)

You can review the report and apply the optimal termination settings—that is, those with the largest margins—using the dropdown menu for each setting.

- To reset the ODT settings from the IP parameterization, press **Restore ODT settings from IP parameterization**.

**Figure 193. Termination Settings**



**Figure 194. Calibrate ODT**

Calibrate ODT						
RTT NOM	RTT PARK	RTT WR	Output Drive Strength	Worst Write Margin (ps)	Worst Read Margin (ps)	TG Status
RZQ/4 (60 Ohm)	Park ODT off	Dynamic ODT off	RZQ/7 (34 Ohm)	384	373	pass
RZQ/4 (60 Ohm)	Park ODT off	Dynamic ODT off	RZQ/5 (48 Ohm)	380	373	pass
RZQ/4 (60 Ohm)	Park ODT off	Dynamic ODT off	RZQ/6 (40 Ohm)	376	380	pass

The **ODT Activation** section displays the ODT assertion patterns in use and the ODT settings in effect during read and write.

**Figure 195. Assertion Table**

ODT Activation				
ODT Assertion Table During Read Accesses (READ-ONLY)				
Read Target	ODT0	ODT1	ODT2	ODT3
Rank 0	off	-	-	-

ODT Assertion Table During Write Accesses (READ-ONLY)				
Write Target	ODT0	ODT1	ODT2	ODT3
Rank 0	off	-	-	-

Derived ODT Matrix for Read Accesses				
The matrix is derived based on settings for RTT_DRV, RTT_NOM, RTT_PARK and the read ODT assertion table				
Read Target	ODT0 Value	ODT1 Value	ODT2 Value	ODT3 Value
Rank 0	(Drive) RZQ/7 (34 Ohm)	-	-	-

Derived ODT Matrix for Write Accesses				
The matrix is derived based on settings for RTT_WR, RTT_NOM, RTT_PARK and the write ODT assertion table				
Write Target	ODT0 Value	ODT1 Value	ODT2 Value	ODT3 Value
Rank 0	(Park) RZQ/4 (60 Ohm)	-	-	-

#### 11.7.4.6. V<sub>ref</sub> Margining Tab

The V<sub>ref</sub> Margining tab sweeps different V<sub>ref-in</sub> and V<sub>ref-out</sub> settings. At each V<sub>ref</sub> 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<sub>ref-in</sub> or V<sub>ref-out</sub>—using the checkboxes near the **Run Vref Margining** button. The tool reports the passing delay margins for each pin, at each V<sub>ref</sub> 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 196.** V<sub>ref</sub> Margining Tab

Memory Configuration		Calibration	Calibration Report	Calibrate Termination	Vref Margining	Driver Margining	ISSP	Pin Delay Settings
		Run Vref Margining	<input checked="" type="checkbox"/> Margin Vref-in	<input checked="" type="checkbox"/> Margin Vref-out				
Pin ID:	0	Direction:	In					
		Voltage (V)	Margins (ps)					
		0.540	.58 to 58					
		0.548	.58 to 61					
		0.556	.68 to 68					
		0.563	.71 to 74					
		0.571	.78 to 81					
		0.579	.81 to 81					
		0.587	.84 to 87					
		0.595	.91 to 94					
		0.602	.94 to 94					
		0.610	.97 to 100					
		0.618	.100 to 100					
		0.626	.107 to 110					
		0.634	.110 to 110					
		0.641	.110 to 113					
		0.649	.113 to 117					
		0.657	.117 to 120					
		0.665	.120 to 123					
		0.673	.123 to 126					
		0.680	.126 to 130					
		0.688	.130 to 130					
		0.696	.133 to 133					
		0.704	.136 to 139					
		0.712	.136 to 139					
		0.719	.139 to 139					
		0.727	.139 to 143					

The V<sub>ref</sub> Margining report can also be viewed in a graphical format. Refer to [Viewing Reports Graphically in the Eye Viewer](#).

#### 11.7.4.7. 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 margins measured during calibration, because this traffic pattern is longer than those 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:

- Is connected to a TG IP (altera\_emif\_tg\_avl). (That is, it is not using the TG2 configurable traffic generator.)
- Does not have ECC enabled.
- Has ISSPs enabled in the project’s QSF file.

To use Driver Margining, press the **Run Driver Margining** button at the top-left 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 system displays the test results in the table when the test has completed.

**Figure 197. Driver Margining Tab**

The screenshot shows a software interface with a navigation bar at the top containing tabs: Memory Configuration, Calibration, Calibration Report, Calibrate Termination, Vref Margining, Driver Margining, ISSP, Pin Delay Settings. The 'Driver Margining' tab is selected. Below the tabs is a sub-header 'Run Driver Margining'. A table follows, with columns: DQ Pin, Read Right Margin, Read Left Margin, Write Right Margin, and Write Left Margin. The table contains 16 rows, indexed from 0 to 15, showing various margin values.

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

The Driver Margining report can also be viewed in a graphical format. Refer to [Viewing Reports Graphically in the Eye Viewer](#).

#### 11.7.4.8. ISSPs Tab

The ISSP tab lets you read probe data and set source values for the In-System Sources and Probes in the design.

To reread the probe data from the ISSPs in the design, expand the **In-System Probes** section and click the **Update Probe Info** button.

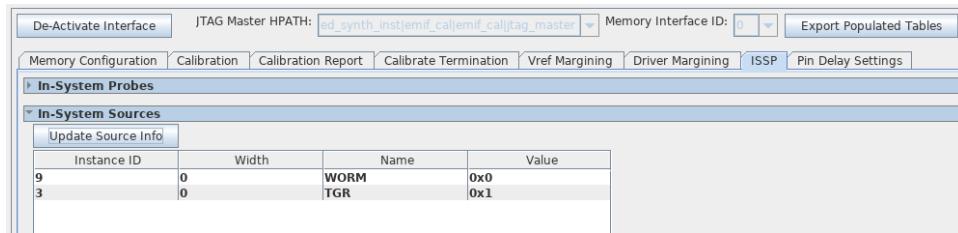
**Figure 198. Displayed Probe Data**

The screenshot shows a software interface with a navigation bar at the top containing tabs: De-Activate Interface, JTAG Master HPATH: jd\_synth\_instjmemif\_caljmemif\_calljtag\_master, Memory Interface ID: 0, Export Populated Tables. Below the tabs is a sub-header 'In-System Probes'. A table follows, with columns: Instance ID, Width, Name, and Value. The table contains 25 rows, indexed from 0 to 24, showing various probe configurations.

Instance ID	Width	Name	Value
24	128	ACP0	0x0
11	1	TGF	0x0
17	1	RAVP	0x0
27	96	AVSC	0xcfc35ba80032c80039ac360c
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	FEPO	0x0
10	1	TGP	0x1
26	128	LRD0	0x0
13	128	PNF0	0xffffffffffffffffffff
23	128	ACT0	0x0
22	128	FEN0	0x0
15	32	FCNT	0x0
19	128	FPN0	0xffffffffffffffffffff
6	33	CALC	0x10145bee9
1	1	CALF	0x0
0	1	CALP	0x1
2	1	PLL	0x1
4	1	PALP	0x1
5	1	PALS	0x0

To reread the source data from the ISSPs in the design, expand the **In-System Sources** section and click the **Update Source Info** button.

**Figure 199. Displayed Source Data**



To overwrite the source data, select the *Instance Name* and change the *Written data* value. The new source value is written when you click **Write Source Info**.

#### Descriptions of ISSPs in the EMIF Design Example

Instance name	Description
PLL	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): <ul style="list-style-type: none"> <li>• Count of read/write requests on the ctrl_amm interface.</li> <li>• Count of only read requests on the ctrl_amm interface.</li> <li>• Number of clocks counted since receiving the first read/write request.</li> </ul>
PALP	Clock phase alignment lock status.
PALS	Clock phase alignment lock (secondary). <i>Note:</i> This is not used in Intel Agilex 7 F-Series and I-Series FPGAs.
CALC	Calibration counter. Highest bit is a <i>done</i> 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.

*continued...*

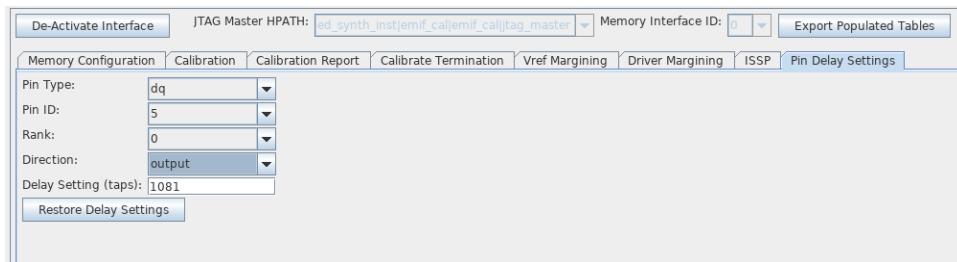
Instance name	Description
TGF	Traffic Generator Fail Flag. Fail=1.
TGT	Traffic Generator Timeout. Timeout=1.
TGR	Traffic Generator Reset. Active High. Toggle TGR to rerun the traffic generator.
RSTN	Global Reset for the design example. Active Low. Toggle RSTN to reset and recalibrate the interface.
WORM	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 parameter editor.

#### 11.7.4.9. Pin Delay Settings Tab

The **Pin Delay Settings** tab lets you view and change delay values on specific pins.

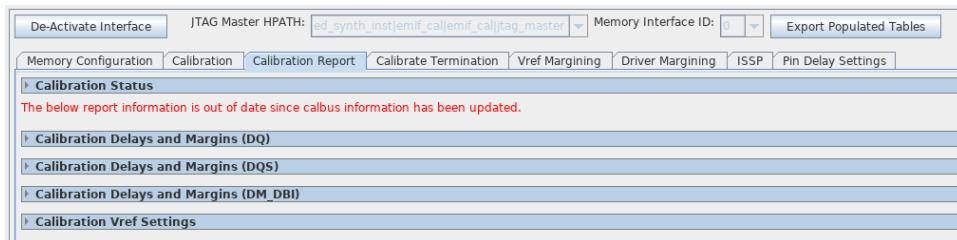
You can select the **Pin Type**, **Pin ID**, **Rank**, and **Direction** values of a delay that you are interested in, and the toolkit displays the delay value in the **Delay Setting (taps)** field.

**Figure 200. Pin Delay Settings Tab**



If you make changes to **Delay Setting (taps)** the delay value is updated in hardware. After the value changes in hardware and no longer matches the setting found in calibration, you receive a warning message indicating that the margins in the **Calibration** tab are out-of-date.

**Figure 201. Margins Out-of-Date Warning Message**

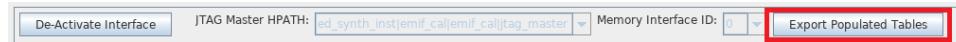


To restore the delay settings to the values found during the most recent calibration, click **Pin Delay Settings > Restore Delay Settings**.

### 11.7.5. 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 202. Export Populated Tables button**



The system saves all the .csv files to the directory where the System Console was originally launched.

**Table 156. Generated .csv Files**

Name of *.csv File	Corresponding Tab (Table) in the Toolkit GUI
mem_config_table.csv	Memory Configuration
vref_margins_table.csv	Vref 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_db_table.csv	Calibration Report [Calibration Delays and Margins (DM_DB)]
cal_results_vref_table.csv	Calibration Report [Calibration Vref 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

### 11.7.6. Viewing Reports Graphically in the Eye Viewer

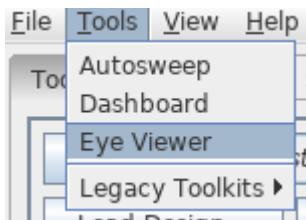
Some of the tables displayed in the toolkit can be viewed in a graphical format.

The tabs and reports that can be displayed are as follows:

- Calibration report: Report types are DQ, DQS, or DM/DBI, and are split between input and output delays.
- V<sub>ref</sub> 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:

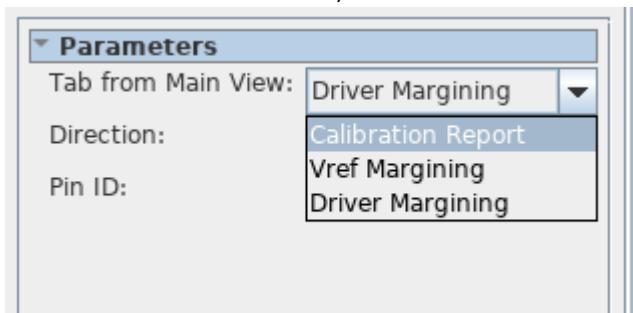
1. Click **Tools > Eye Viewer >**.



2. Select the toolkit instance that you want:



3. In the **Parameters** section, selected the desired tab from the dropdown menu.



4. Click the Start button at the bottom-right to create the graphical report based on the selected parameters. (The diagrams below demonstrate different reports taken from various designs.)

**Figure 203. Graphic Representation of DQ Input Margins from Calibration**

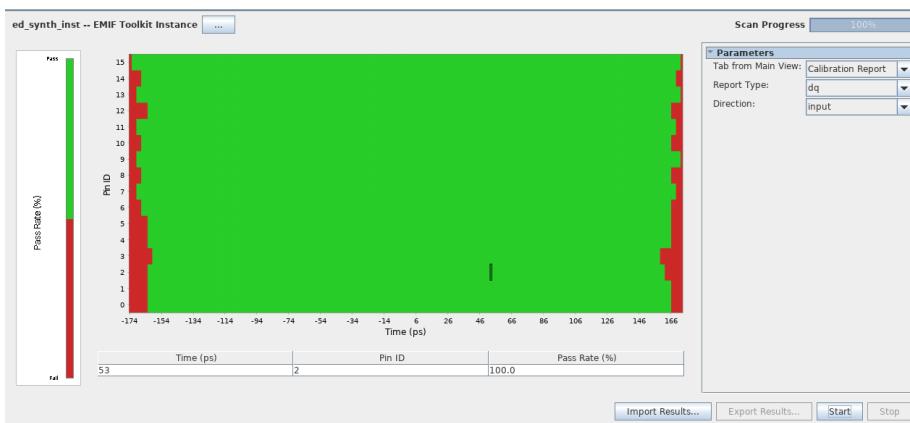


Figure 204. Eye Diagram – Vref-in Margins on DQ0

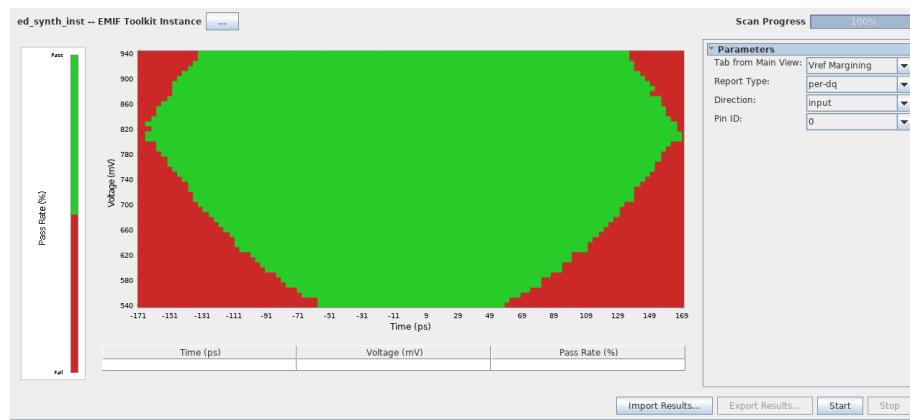


Figure 205. Eye Diagram – Vref-in Margins, Averaged Across All DQ Pins

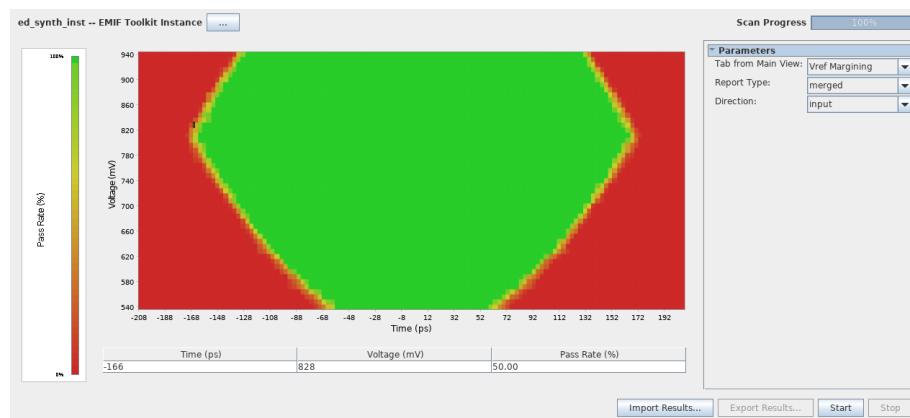
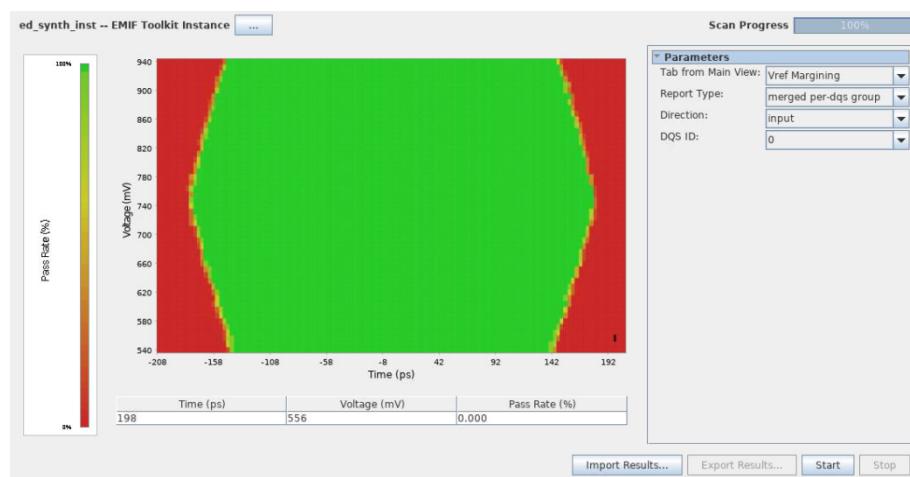


Figure 206. Eye Diagram – Vref-in Margins, Averaged Across the DQ Pins in Group 0



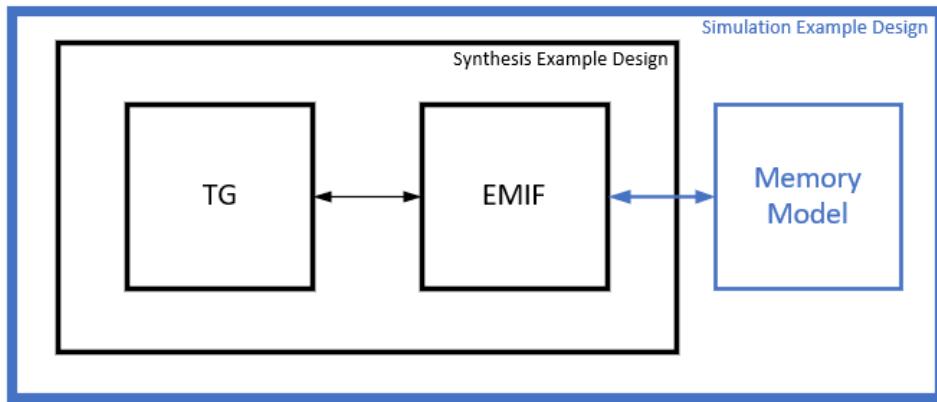
## 11.8. Using the Default Traffic Generator

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 207. EMIF Design Example Overview**

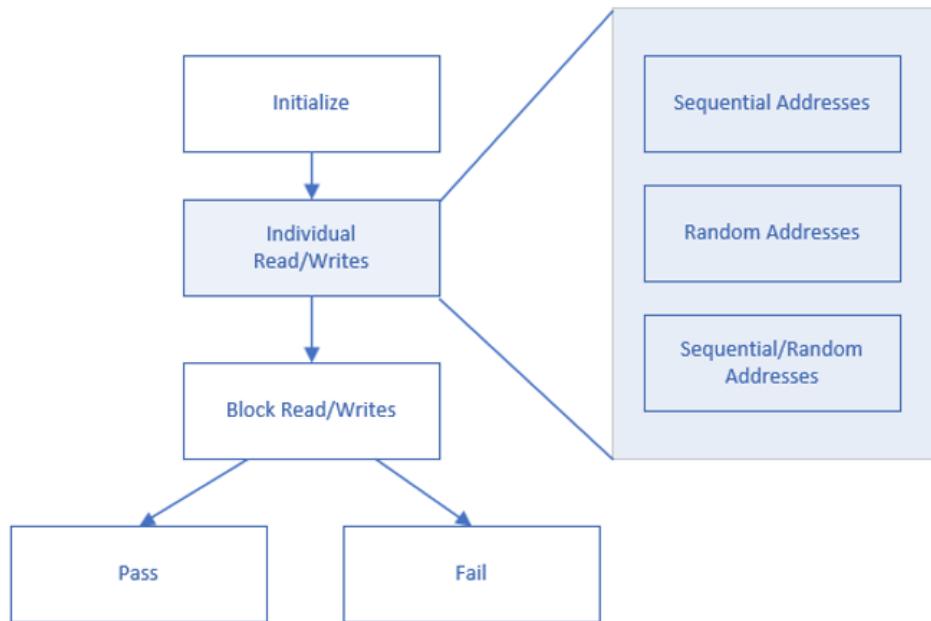


For general information about the generated EMIF design example, refer to the [External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA IP Design Example User Guide](#).

Each operation generated by the traffic generator is a single write or block of writes followed by a single read or block of reads to the same addresses, which allows the driver to precisely determine the data that should be expected when the memory interface returns the read data.

The traffic generator comprises a traffic generation block, the Avalon memory-mapped interface and a read comparison block. The traffic generation block generates addresses and write data, which are then sent out over the Avalon memory-mapped interface. The read comparison block compares the read data received from the Avalon memory-mapped interface to the write data from the traffic generator. If at any time the data received is not the expected data, the read comparison block records the failure, finishes reading all the data, and then signals that there is a failure and the traffic generator enters a fail state. If all patterns are generated and compared successfully, the traffic generator enters a pass state.

**Figure 208. Default Traffic Generator Operations**



Within the traffic generator, there are the following main states:

- Generation of individual read and writes.
- Generation of block read and writes.
- The pass state.
- The fail state.

Within each of the generation states, there are the following substates:

- Sequential address generation.
- Random address generation.
- Mixed sequential and random address generation.

### Read and Write Generation

The traffic generator can generate individual or block reads and writes. Individual read and write generation during the traffic generator's individual read and write generation state, the traffic generation block generates individual write followed by individual read Avalon memory-mapped interface transactions, where the address for the transactions is chosen according to the specific substate

### Block read and Write Generation

During the traffic generator's block read and write generation state, the traffic generator block generates a parameterizable number of write operations followed by the same number of read operations. The specific addresses generated for the blocks are chosen by the specific substates.

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.

### 11.8.1. 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 as long as 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 (immediately after the preamble) of a BL8 memory read operation
- `pnf[1]` maps to `dq[1]` for the first beat (immediately after the preamble) of a BL8 memory read operation
- ...
- `pnf[31]` maps to `dq[31]` for the first beat (immediately after the preamble) of a BL8 memory read operation
- `pnf[32]` maps to `dq[0]` for the second beat of a BL8 memory read operation
- ...
- `pnf[64]` maps to `dq[0]` for the third beat of a BL8 memory read operation
- ...
- `pnf[96]` maps to `dq[0]` for the fourth beat of a BL8 memory read operation

- ...
- pnf[128] maps to dq[0] for the fifth beat of a BL8 memory read operation
- pnf[160] maps to dq[0] for the sixth beat of a BL8 memory read operation
- ...
- pnf[192] maps to dq[0] for the seventh beat of a BL8 memory read operation
- ...
- pnf[224] maps to dq[0] for the final beat (immediately before the postamble) of a BL8 memory read operation

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.

### 11.8.2. 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.

### 11.8.3. 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.

### 11.8.4. 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 157. 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
<i>continued...</i>	

Avalon bus signals	amm_read_0 amm_readdatavalid_0 amm_ready_0 amm_write_0 amm_address_0 amm_burstcount_0 amm_bytelenable_0 amm_readdata_0 amm_writedata_0
For the Signal Tap clock, Signal Tap : Pre-synthesis	emif_usr_clk

## 11.9. 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.

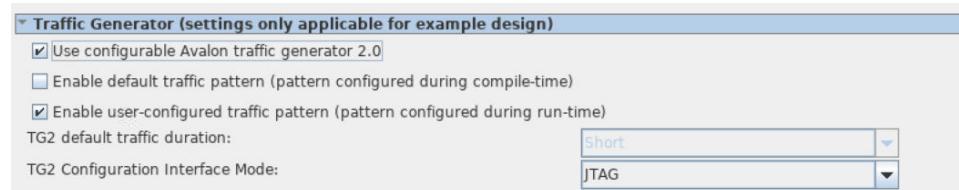
In the Configurable Traffic Generator (TG2) (altera\_tg\_avl\_2), you can configure the traffic pattern in real time through control registers—meaning that you do not have to recompile the design to change or relaunch the traffic pattern. This traffic generator provides fine control over the type of traffic that it sends on the EMIF control interface. Additionally, it provides status registers that contain detailed failure information.

### 11.9.1. 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 209.**

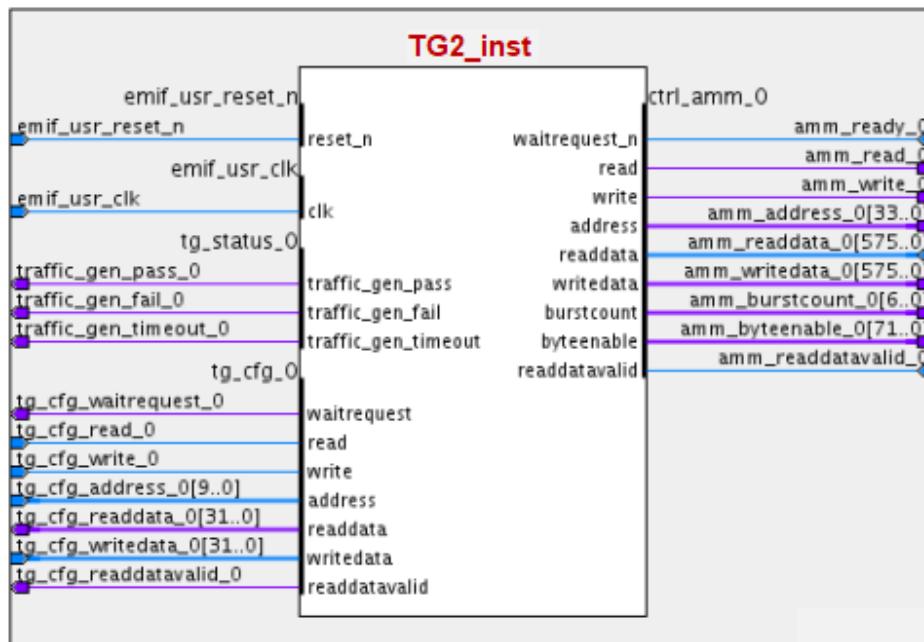


- You may choose to disable the **default traffic pattern** stage or the **user-configured traffic** stage, but you must have at least one stage enabled. For information on these stages, refer to [Default Traffic Pattern](#) and [User-Configured Traffic Pattern](#).
- The **TG2 test duration** parameter 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.

### 11.9.2. Traffic Generator Block Description

The traffic generator sends traffic through the external memory interface using the `ctrl_amm` interface.

**Figure 210. 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.

#### **11.9.3. 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 158.**

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 <i>loop</i> . 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 <b>TG2 test duration</b> parameter. The traffic generator executes this stage once for each of the three address modes (Sequential, Random, and Random-Sequential).
Byte-enable stage	<ul style="list-style-type: none"><li>The traffic generator randomly generates a byte-enable value and performs a block of writes to a start address, in Sequential Address Mode.</li><li>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.</li><li>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.</li></ul>

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](#) section.

#### **11.9.4. 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.

### Configuration and Status Registers

<b>Symbol Address</b>	<b>Register Name</b>	<b>Number of Registers</b>	<b>Access Type</b>	<b>Register Section</b>	<b>Register Description</b>
0x4	TG_START	1	Read/ Write	Start	Perform a write to this register to start the traffic generator.
0x8	TG_LOOP_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of read/write loops to be run. A loop is defined as one block of writes followed by a block of reads. Setting to 0 enables Infinite User Mode.
0xC	TG_WRITE_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of unique writes to be performed in each loop.
0x10	TG_READ_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of unique reads to be performed in each loop.
0x14	TG_WRITE_REPEAT_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of times to repeat each write operation (i.e. same address and same data).
0x18	TG_READ_REPEAT_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of times to repeat each read operation (i.e. same address).
0x1C	TG_BURST_LENGTH	1	Read/ Write	Test Duration/ Instruction Pattern	Avalon burst length.
0x20	TG_CLEAR	1	Read/ Write	Status	<p>Clears the failure status registers. Allows clearing these registers independently from one another by writing a 1 to the following bits:</p> <ul style="list-style-type: none"> <li>BIT0 - Clears the recorded PNF data and starts fresh.</li> <li>BIT1 - Clears the recorded number of avalon reads.</li> <li>BIT2 - Clears the recorded data of the first failure (address; expected data; actual data).</li> <li>BIT3 - Clears the burstlength overflow flag and burstlength overflow address.</li> <li>BIT4 - Clears the WORM mode failure info (i.e. targeted read data)</li> </ul>
0x38	TG_RW_GEN_IDLE_COUNT	1	Read/ Write	Test Duration/ Instruction Pattern	Number of cycles for which TG remains idle between a write block and the next read block.

*continued...*

<b>Symbol Address</b>	<b>Register Name</b>	<b>Number of Registers</b>	<b>Access Type</b>	<b>Register Section</b>	<b>Register Description</b>
0x3C	TG_RW_GEN_LOOP_IDLE_COUNT	1	Read/Write	Test Duration/Instruction Pattern	Number of cycles for which TG remains idle between a read block and the next write block.
0x40	TG_SEQ_START_ADDR_WR	6	Read/Write	Address Pattern	Start address for writes; used as seed address in Random Mode.
0x80	TG_ADDR_MODE_WR	6	Read/Write	Address Pattern	Array that selects address generator mode for each address field for writes TG_ADDR_MODE==0: Fixed TG_ADDR_MODE==1: Random TG_ADDR_MODE==2: Sequential TG_ADDR_MODE==3 Field Unused.
0xC0	TG_RETURN_TO_START_ADDR	1	Read/Write	Address Pattern	Setting to 1 specifies to return to start address in each loop. Setting to 0 specified to resume the address pattern from where the previous loop left off.
0x100	TG_SEQ_ADDR_INCR	6	Read/Write	Address Pattern	Array of increments for fields 0-5 to be used for each address field when address is to be incremented sequentially. This value should be at least the value of TG_BURST_LENGTH for field 0.
0x140	TG_SEQ_START_ADDR_RD	6	Read/Write	Address Pattern	Start address for reads; used as seed address in Random Mode.
0x180	TG_ADDR_MODE_RD	6	Read/Write	Address Pattern	Array that selects address generator mode for each address field for reads TG_ADDR_MODE==0: Fixed TG_ADDR_MODE==1: Random TG_ADDR_MODE==2: Sequential TG_ADDR_MODE==3 Field Unused.
0x1C0	TG_PASS	1	Read Only	Status	Reading a 1 indicates that the traffic generator passed at the end of all the test stages.
0x1C4	TG_FAIL	1	Read Only	Status	Reading a 1 indicates that the traffic generator failed at the end of all the test stages.

*continued...*

<b>Symbol Address</b>	<b>Register Name</b>	<b>Number of Registers</b>	<b>Access Type</b>	<b>Register Section</b>	<b>Register Description</b>
0x1C8	TG_FAIL_COUNT_L	1	Read Only	Status	Number of failed reads (lower 32 bits).
0x1CC	TG_FAIL_COUNT_H	1	Read Only	Status	Number of failed reads (upper 32 bits).
0x1D0	TG_FIRST_FAIL_ADDR_L	1	Read Only	Status	Address of the first failed read (lower 32 bits).
0x1D4	TG_FIRST_FAIL_ADDR_H	1	Read Only	Status	Address of the first failed read (upper 32 bits).
0x1D8	TG_TOTAL_READ_COUNT_L	1	Read Only	Status	Number of read operations executed - sent and received (lower 32 bits).
0x1DC	TG_TOTAL_READ_COUNT_H	1	Read Only	Status	Number of read operations executed - sent and received (upper 32 bits).
0x1E0	TG_TEST_COMPLETE	1	Read Only	Status	Reading a 1 indicates that the traffic generator run is complete.
0x1E4	TG_INVERT_BYTEEN	1	Read/ Write	Data/ Byte- Enable Pattern	Setting to 1 specifies to invert byte-enable values and writedata.
0x1E8	TG_RESTART_DEFAULT_TRAFFIC	1	Read/ Write	Data/ Byte- Enable Pattern	Perform a write to this register to restart the default mode.
0x1EC	TG_USER_WORM_EN	1	Read/ Write	Data/ Byte- Enable Pattern	Setting to 1 enables WORM (Write Once Read Many) Mode. If a data mismatch is encountered in WORM mode TG2 stops at the first data mismatch and issues a second read to the same address.
0x1F0	TG_TEST_BYTEEN	1	Read/ Write	Data/ Byte- Enable Pattern	Perform a write to this register to indicate TG is in invert byte-enable testing mode.
0x1F4	TG_TIMEOUT	1	Read Only	Status	Reading a 1 indicates that the traffic generator timeout.
0x1F8	TG_NUM_DATA_GEN	1	Read Only	Data/ Byte- Enable Pattern	Number of data generators.
0x1FC	TG_NUM_BYTEEN_GEN	1	Read Only	Data/ Byte- Enable Pattern	Number of byte enable generators.
0x200	TG_RDATA_WIDTH	1	Read Only	Data/ Byte- Enable Pattern	Width of read data and PNF signals.

*continued...*

Symbol Address	Register Name	Number of Registers	Access Type	Register Section	Register Description
0x204	TG_ERROR_REPORT	1	Read Only	Data/Byte-Enable Pattern	Reports illegal configurations of TG2.
0x208	TG_DATA_RATE_WIDTH_RATIO	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	18	Read Only	Status	Persistent PNF (Pass Not Fail) signal. Bus Width = TG_RDATA_WIDTH.
0x340	TG_FAIL_EXPECTED_DATA	1	Read Only	Status	Expected data on the first failure. Bus Width = TG_RDATA_WIDTH.
0x440	TG_FAIL_READ_DATA	1	Read Only	Status	Received data on the first failure. Bus Width = TG_RDATA_WIDTH.
0x540	TG_DATA_SEED	TG_NUM_DATA_GEN	Read/Write	Data Generator	Seed or starting value for each data generator (DG). This consists of TG_NUM_DATA_GEN entries.
0x580	TG_BYTEEN_SEED	TG_NUM_BYTEEN_GEN	Read/Write	Data/Byte-Enable Pattern	Seed or starting value for each byte-enable generator (BEG). This consists of TG_NUM_BYTEEN_GEN entries.
0x5C0	TG_PPPG_SEL	TG_NUM_DATA_GEN	Read/Write	Data Generator	Select from available pattern modes 0: Fixed; 1: PRBS7; 2: PRBS15; 3: PRBS31; 4: Rotating.
0x600	TG_BYTEEN_SEL	TG_NUM_BYTEEN_GEN	Read/Write	Data/Byte-Enable Pattern	Select from available pattern modes 0: Fixed; 1: PRBS7; 2: PRBS15; 3: PRBS31; 4: Rotating.
0x640	TG_ADDR_FIELD_RELATIVE_FREQ	6	Read/Write	Address Pattern	Array to set the frequency of each field. The value of the relative frequency is an integer that specifies after how many unique address commands the field updates.
0x680	TG_ADDR_FIELD_MSB_INDEX	5	Read/Write	Address Pattern	Array to set the MSB position of fields 0-4. Field 5 is redundant and does not need to be specified. MSB position implies the field width.
0x6C0	TG_BURSTLENGTH_OVERFLOW_OCCURRED	1	Read Only	Status	Reading a 1 indicates that an attempt was made to write outside of the address space. This occurs when the current address plus the

*continued...*

<b>Symbol Address</b>	<b>Register Name</b>	<b>Number of Registers</b>	<b>Access Type</b>	<b>Register Section</b>	<b>Register Description</b>
					burstlength is greater than the total address space.
0x700	TG_BURSTLENGTH_FAIL_ADDR_L	1	Read Only	Status	Indicates the address where an address overflow occurred due to burstlength being greater than the difference between the write/read address and the last memory address (lower 32 bits).
0x704	TG_BURSTLENGTH_FAIL_ADDR_H	1	Read Only	Status	Indicates the address where an address overflow occurred due to burstlength being greater than the difference between the write/read address and the last memory address (lower 32 bits).
0x740	TG_WORM_MODE_TARGETTED_D ATA	1	Read Only	Status	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.

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.

#### Related Information

[Register Map IP-XACT Support for Intel Agilex 7 F-Series and I-Series EMIF DDR4 IP on page 148](#)

### 11.9.5. User Pattern

The following topics describe available traffic patterns.

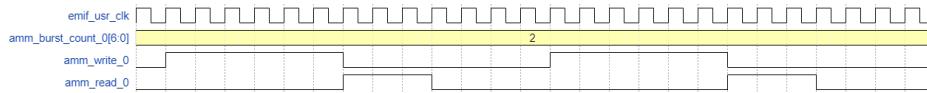
#### 11.9.5.1. Test Duration / Instruction pattern

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 211. Timing Diagram for Example Test Pattern



### 11.9.5.2. 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 six address generators available. The address pattern is configured by specifying modes, positions, and relative frequencies for each of the six address generators.

#### 11.9.5.2.1. 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.

#### 11.9.5.2.2. 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 spans bits 0-5 (inclusive) and field 1 spans bits 6-9 (inclusive), giving field 0 a width of 6 and field 1 a width of 4.

#### 11.9.5.2.3. 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 the [Error Codes](#) table for information on error codes).

#### 11.9.5.2.4. 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 generate a new address every 8th read or write operation.

#### 11.9.5.2.5. 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:

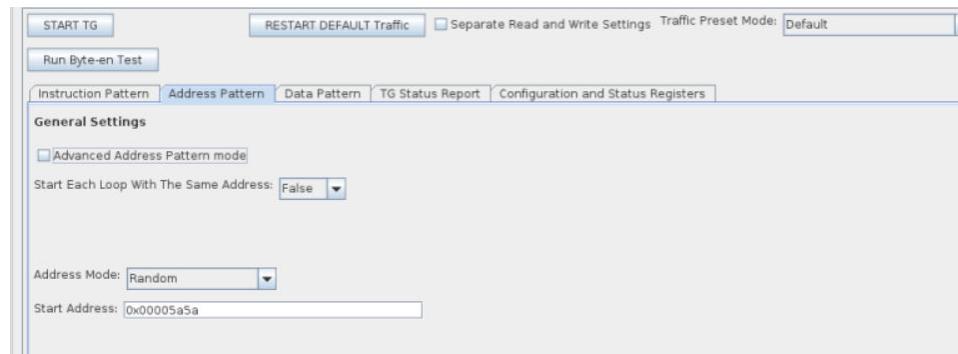
- The data width on the memory side.
- Whether a configured EMIF IP is quarter rate, half rate, or full rate.
- Whether the memory interface is double data rate or quarter data rate.

### Example 1: Random Address Mode

Consider the following instruction pattern:

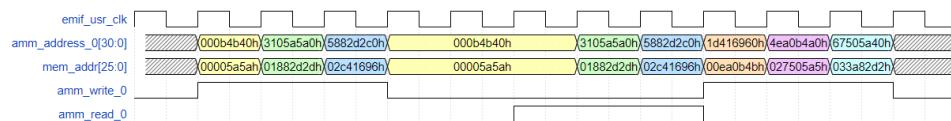
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=2
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=0
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

**Figure 212. 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 213. Random Address Mode**

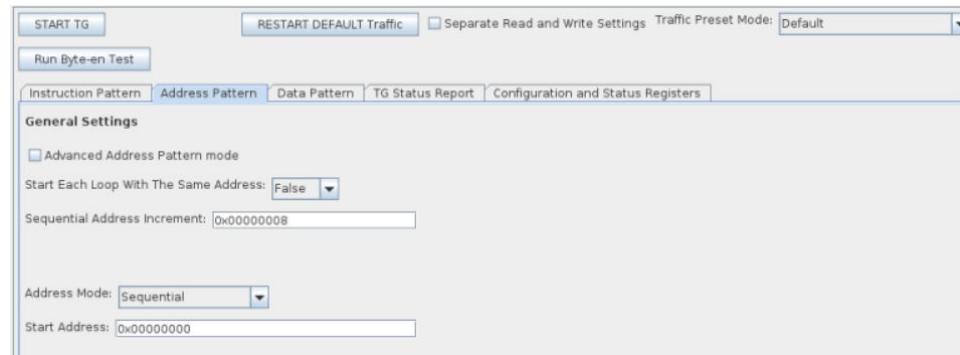


### Example 2: Sequential Address Mode

Consider the following instruction pattern:

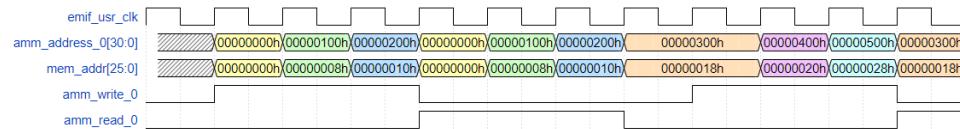
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

**Figure 214.** 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 215. Sequential Address Mode**

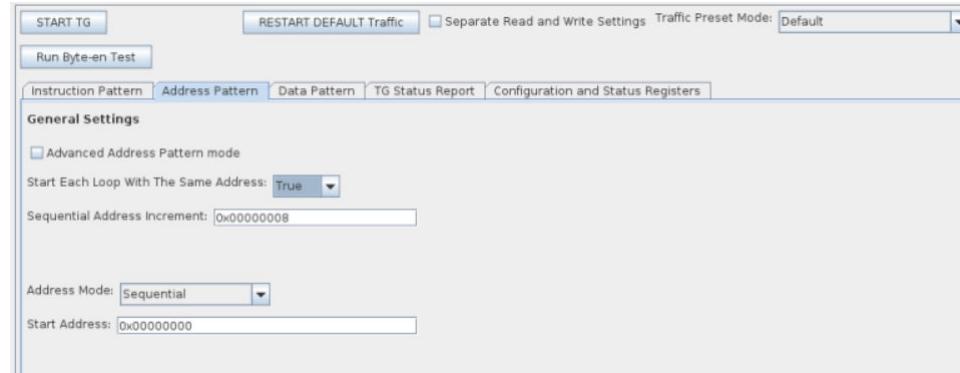


### Example 3: Sequential Address Mode with TG\_RETURN\_TO\_START\_ADDR=1

Consider the following instruction pattern:

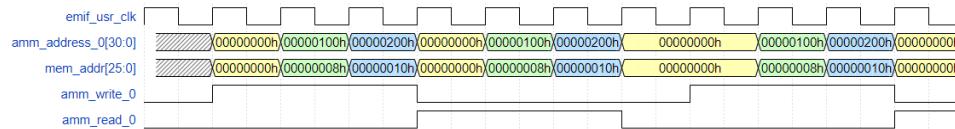
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

**Figure 216. 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 217. Sequential Address Mode with TG\_RETURN\_TO\_START\_ADDR=1**

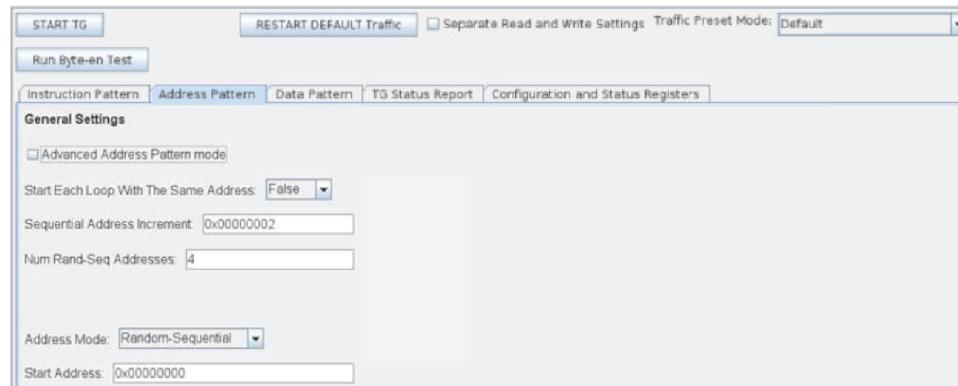


#### Example 4: Random Sequential Address Mode

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 218. Setting the Address Pattern in the Traffic Generator Configuration Interface**

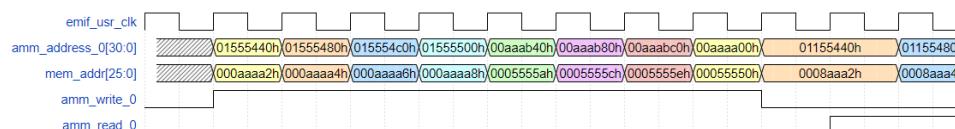


The goal of this address pattern is to create a random-sequential pattern, where:

- The bottom 4 bits of the address increment by 2 for each new address.
- The remaining upper bits are generated randomly for every fourth new address.

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 219. Random-Sequential Address Mode**



### 11.9.5.2.6. 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_ammm` 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

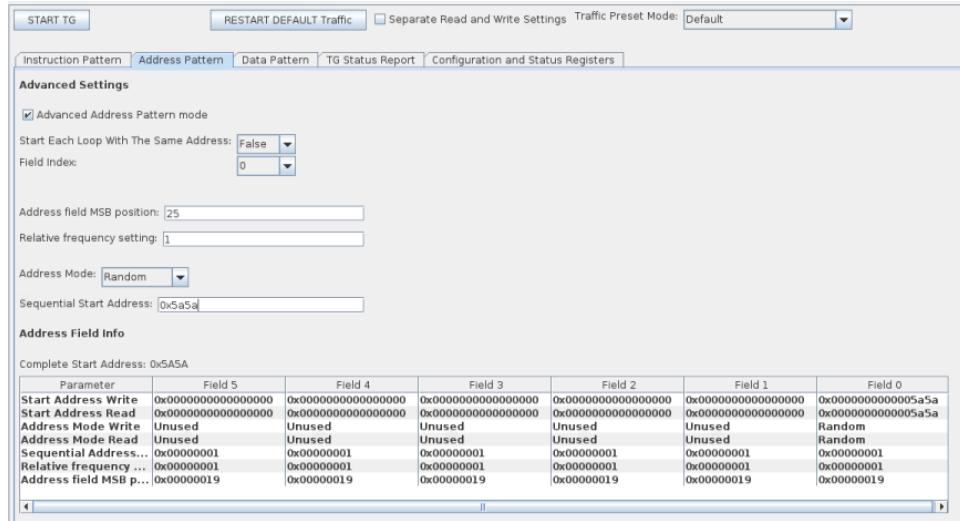
Consider the following instruction pattern:

```
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
```

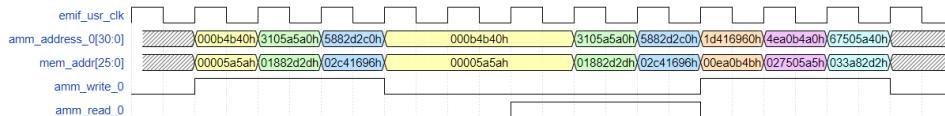
**Table 160. Address Pattern**

Write Start Addresses: <code>TG_SEQ_START_ADDR_WR</code> =0x5a5a <code>TG_SEQ_START_ADDR_WR+1</code> =0x0000 <code>TG_SEQ_START_ADDR_WR+2</code> =X ... <code>TG_SEQ_START_ADDR_WR+11</code> =X	Read Start Addresses: <code>TG_SEQ_START_ADDR_RD</code> =0x5a5a <code>TG_SEQ_START_ADDR_RD+1</code> =0x0000 <code>TG_SEQ_START_ADDR_RD+2</code> =X ... <code>TG_SEQ_START_ADDR_RD+11</code> =X
Write Address Modes: <code>TG_ADDR_MODE_WR</code> =1 <code>TG_ADDR_MODE_WR+1</code> =3 ... <code>TG_ADDR_MODE_WR+5</code> =3	Read Address Modes: <code>TG_ADDR_MODE_RD</code> =1 <code>TG_ADDR_MODE_RD+1</code> =3 ... <code>TG_ADDR_MODE_RD+5</code> =3
Sequential Address Increments: <code>TG_SEQ_ADDR_INCR</code> =X <code>TG_SEQ_ADDR_INCR+1</code> =X ... <code>TG_SEQ_ADDR_INCR+5</code> =X	Return to Start Address: <code>TG_RETURN_TO_START_ADDR</code> =0
Relative Frequencies: <code>TG_ADDR_FIELD_RELATIVE_FREQ</code> =1 <code>TG_ADDR_FIELD_RELATIVE_FREQ+1</code> =X ... <code>TG_ADDR_FIELD_RELATIVE_FREQ+5</code> =X	MSB Indices: <code>TG_ADDR_FIELD_MSB_INDEX</code> = <code>AMM_WORD_ADDRESS_WIDTH</code> -1 <code>TG_ADDR_FIELD_MSB_INDEX+1</code> =X ... <code>TG_ADDR_FIELD_MSB_INDEX+4</code> =X

**Figure 220. Setting the Address Pattern in the Traffic Generator Configuration Interface**



**Figure 221. 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
```

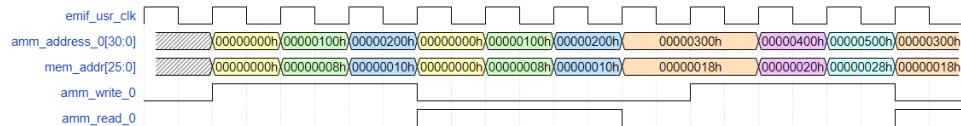
**Table 161. Address Pattern**

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 222. Sequential Address Mode**



### Example 3: Sequential Address Mode with TG\_RETURN\_TO\_START\_ADDR\_1

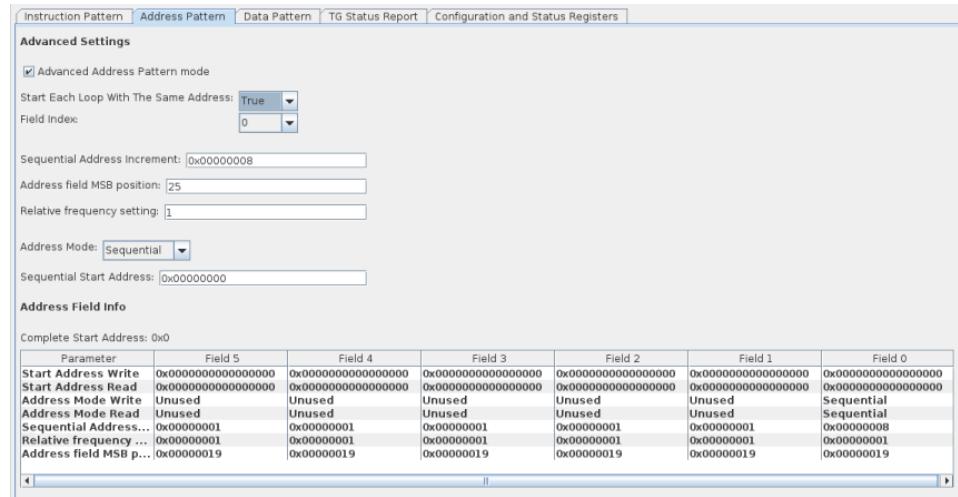
Consider the following instruction pattern:

```
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
```

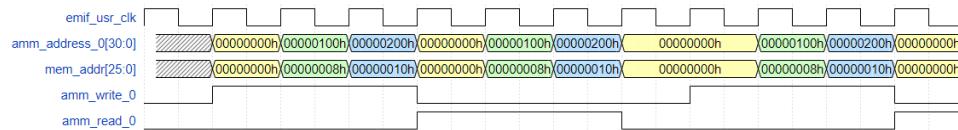
**Table 162. Address Pattern**

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 223. Setting the Address Pattern in the Traffic Generator Configuration Interface**



**Figure 224. 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
```

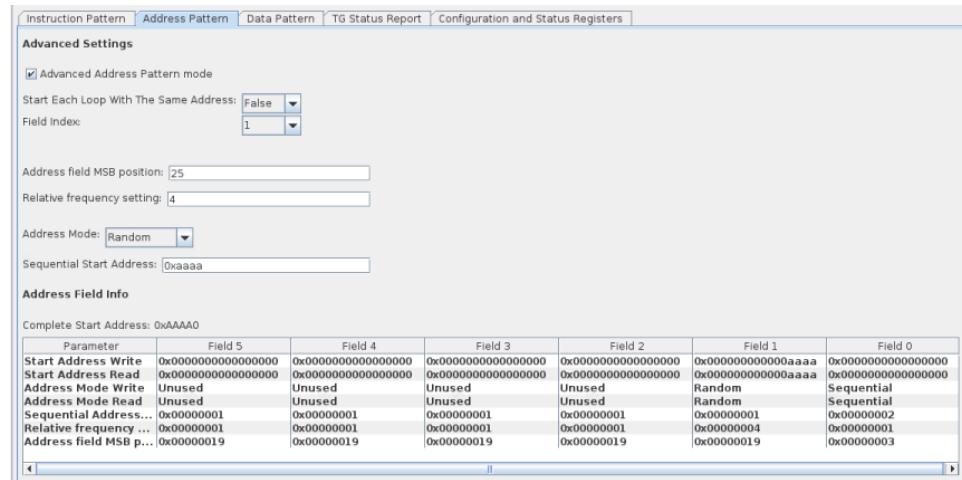
**Table 163. Address Pattern**

Write Start Addresses: TG_SEQ_START_ADDR_WR =0x0000 TG_SEQ_START_ADDR_WR+1=0x0000 TG_SEQ_START_ADDR_WR+2=0aaaa 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=0aaaa 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
Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=4 TG_ADDR_FIELD_RELATIVE_FREQ+2=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=3 TG_ADDR_FIELD_MSB_INDEX+1=AMM_WORD_ADDRESS_WIDTH-1

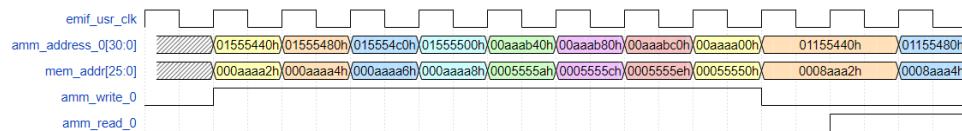
*continued...*

TG_ADDR_FIELD_RELATIVE_FREQ+3=X TG_ADDR_FIELD_RELATIVE_FREQ+4=X TG_ADDR_FIELD_RELATIVE_FREQ+5=X	TG_ADDR_FIELD_MSB_INDEX+4=X TG_ADDR_FIELD_MSB_INDEX+4=X TG_ADDR_FIELD_MSB_INDEX+4=X
---	---

**Figure 225.** Setting the Address Pattern in the Traffic Generator Configuration Interface



**Figure 226.** Random-Sequential Address Mode



#### Example 5: Using Multiple Address Fields for Traversing Memory Hierarchy

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 164.**

Hierarchy Level	Parameter	Value of Parameter
Rank	MEM_DDR4_DISCRETE_CS_WIDTH (cs)	1
Bank Group	MEM_DDR4_BANK_GROUP_WIDTH (BG)	2

*continued...*

Hierarchy Level	Parameter	Value of Parameter
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:

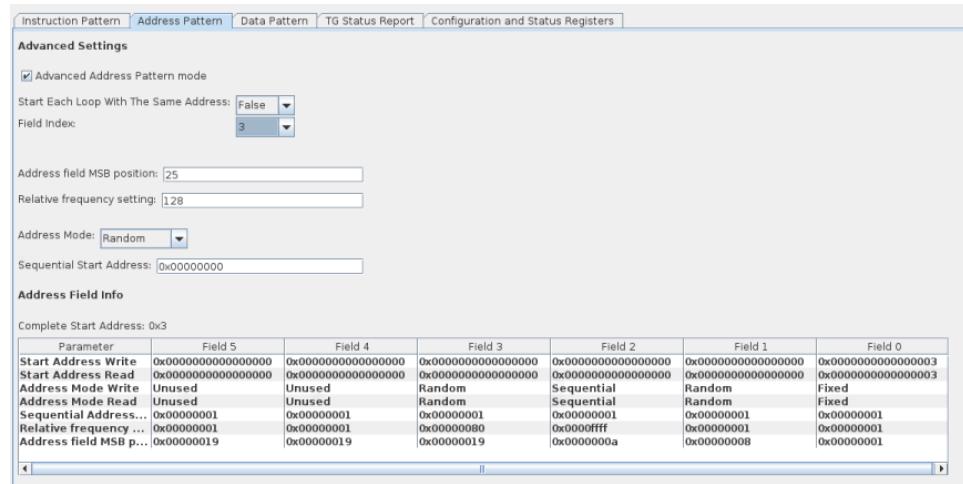
2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	B A	B A	C	C	C	C	C	C	B G	B G	

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  $2^{16} \times 2^7 = 2^{23}$  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.

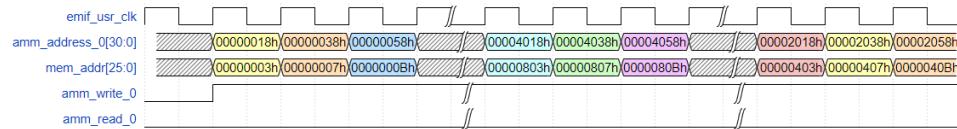
Write Start Addresses: TG_SEQ_START_ADDR_WR =0x0003 TG_SEQ_START_ADDR_WR+1=0x0000 TG_SEQ_START_ADDR_WR+2=0x0000 TG_SEQ_START_ADDR_WR+3=0x0000 TG_SEQ_START_ADDR_WR+4=0x0000 TG_SEQ_START_ADDR_WR+5=0x0000 TG_SEQ_START_ADDR_WR+6=0x0000 TG_SEQ_START_ADDR_WR+7=0x0000 TG_SEQ_START_ADDR_WR+8=X ... TG_SEQ_START_ADDR_WR+11=X	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
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=2 $^{16}-1$ TG_ADDR_FIELD_RELATIVE_FREQ+3=2 $^7$ 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 227. Setting this Address Pattern Configuration in the Traffic Generator Configuration Interface**



This address pattern can be performed in advanced mode only.

**Figure 228. 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:

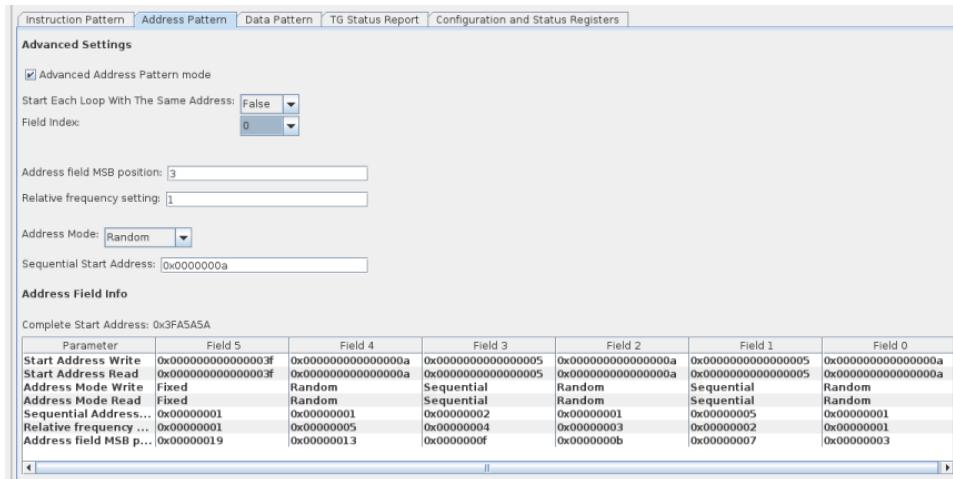
This address pattern illustrates the abilities of the advanced mode, by tying different address bits to a variety of different address generators, each with a different relative frequency.

Write Start Addresses:	Read Start Addresses:
TG_SEQ_START_ADDR_WR = 0x000a	TG_SEQ_START_ADDR_RD = 0x000a
TG_SEQ_START_ADDR_WR+1=0x0000	TG_SEQ_START_ADDR_RD+1=0x0000
TG_SEQ_START_ADDR_WR+2=0x0005	TG_SEQ_START_ADDR_RD+2=0x0005
TG_SEQ_START_ADDR_WR+3=0x0000	TG_SEQ_START_ADDR_RD+3=0x0000
TG_SEQ_START_ADDR_WR+4=0x000a	TG_SEQ_START_ADDR_RD+4=0x000a
TG_SEQ_START_ADDR_WR+5=0x0000	TG_SEQ_START_ADDR_RD+5=0x0000
TG_SEQ_START_ADDR_WR+6=0x0005	TG_SEQ_START_ADDR_RD+6=0x0005
TG_SEQ_START_ADDR_WR+7=0x0000	TG_SEQ_START_ADDR_RD+7=0x0000
TG_SEQ_START_ADDR_WR+8=0x000a	TG_SEQ_START_ADDR_RD+8=0x000a

*continued...*

TG_SEQ_START_ADDR_WR+9=0x0000 TG_SEQ_START_ADDR_WR+10=0x007f TG_SEQ_START_ADDR_WR+11=0x0000	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=X TG_SEQ_ADDR_INCR+2=X TG_SEQ_ADDR_INCR+3=X 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 229. How to set this address pattern configuration in the Traffic Generator Configuration Interface**



This address pattern can be performed in advanced mode only.

**Figure 230. Multiple Address Fields**



### 11.9.5.3. 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:

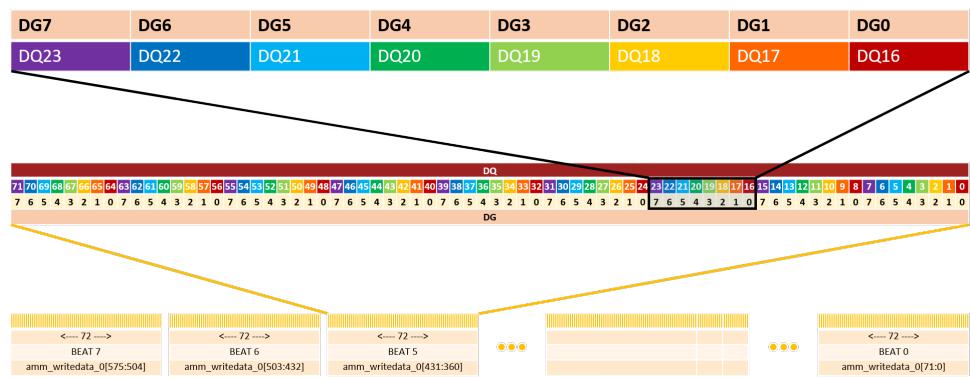
- DQ Width: x72 (without ECC enabled)
- Memory Protocol: DDR4
- DQ/DQS: 8
- Rate: Quarter-rate

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, we 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 231. Data Pattern Example**



For each data generator, you can create a unique data pattern by configuring the start seed (32 bits wide) and pattern mode by writing to the TG\_DATA\_SEED register and the TG\_PPPG\_SEL register, respectively. Refer to [Configuration and Status Registers](#) for the supported options.

The following table illustrates examples of data patterns that can be generated on each data generator by configuring the TG\_DATA\_SEED and TG\_PPPG\_SEL registers.

**Table 165. 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.

*continued...*

Mode	Description
	<p>Example: A seed of 0x76543210 for DG0, where TG_DATA_RATE_WIDTH_RATIO=8 results in the following pattern seen on DQ0, sequentially:</p>
PRBS7	<p>Pseudorandom Binary Sequence. Uses the 8 least-significant bits (LSB) of the input seed, and the monic polynomial: <math>x^7 + x^6 + 1</math> A seed of '0' produces no unique patterns.</p>
PRBS15	<p>Pseudorandom Binary Sequence. Uses the 16 least-significant bits of the input seed, and the monic polynomial: <math>x^{15} + x^{14} + 1</math> A seed of '0' produces no unique patterns.</p>
PRBS31	<p>Pseudorandom Binary Sequence. Uses all 32 bits of the input seed, and the monic polynomial: <math>x^{31} + x^{28} + 1</math> A seed of '0' produces no unique patterns.</p>
Rotating (custom)	<p>The data pattern on the pin is 32 bits long, as specified by the user. The pattern appears on the pin least-significant bit to most-significant bit.</p> <p>Example: A seed of 0x76543210 for DG0 results in the following pattern seen on DQ0, sequentially:</p>

There is one byte-enable generator for each byte in the interface. The byte-enable generator options are identical to the data generator options.

## 11.9.6. 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:

$$\text{TG\_PNF}[\text{N}_{\text{PNF\_reg}}] = (\text{Symbol Address of TG\_PNF}) + 4 * \text{N}_{\text{PNF\_reg}}$$

The maximum PNF width is 511 bits, so the bus width is split up across  $\text{N}_{\text{PNF\_ISSP}}$  ISSPs:

$$\text{N}_{\text{PNF\_ISSP}} = \text{ceil}(\text{TG\_RDATA\_WIDTH} / 511)$$

PNF ISSPs have the index in their name, starting with PNFO.

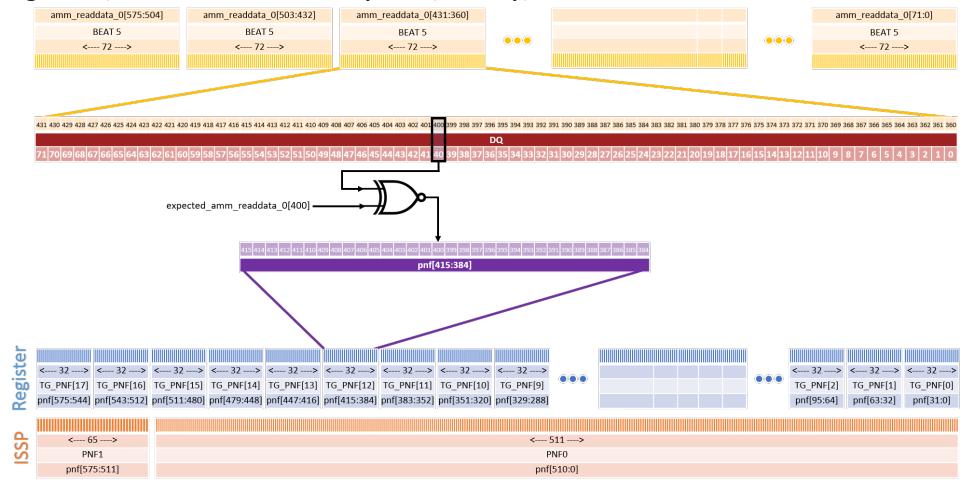
Example:

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,  $\text{TG\_DATA\_RATE\_WIDTH\_RATIO} = 8$ .

Therefore  $\text{TG\_RDATA\_WIDTH} = 72 * 8 = 576$ . This means that there are 18 TG\_PNF registers, and two PNF ISSPs (PNFO, PNFI), 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 166. 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.

## 11.9.7. 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 write to `TG_START` (any value) is issued at the end of the pattern, to trigger starting the 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.

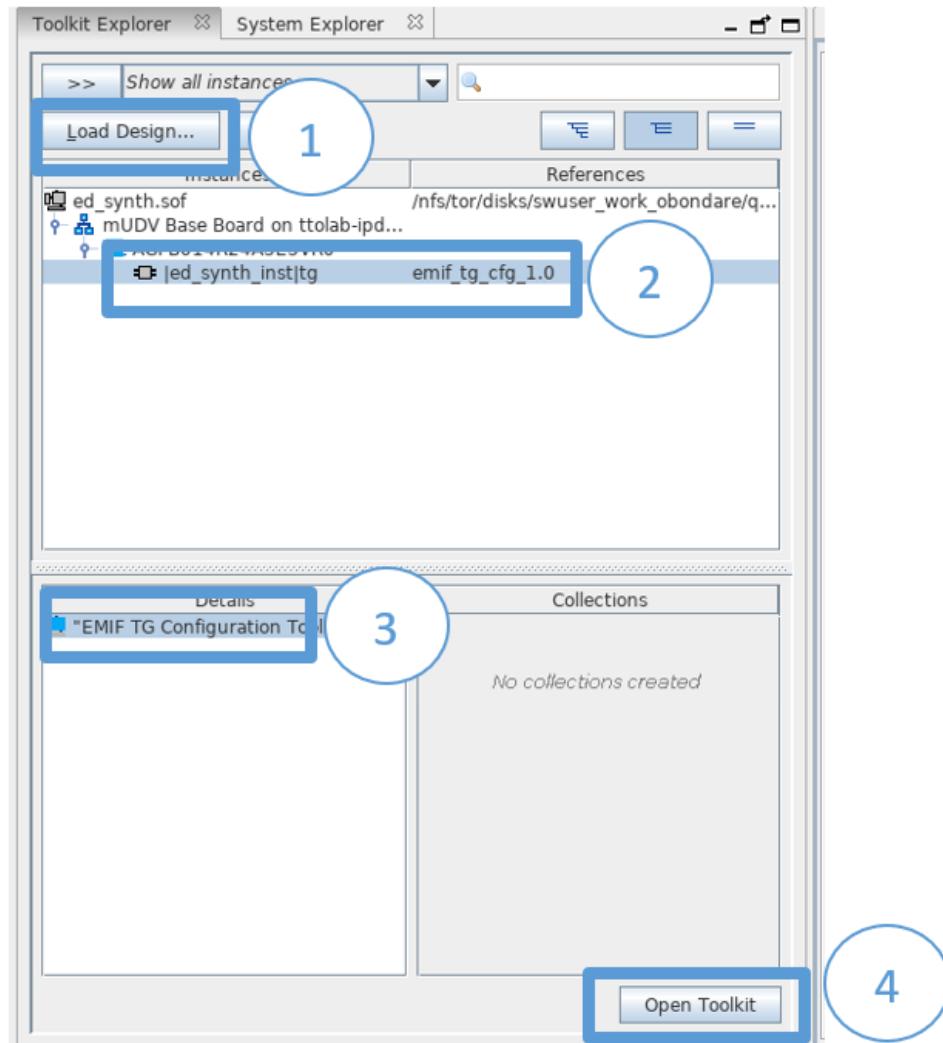
## 11.9.8. Traffic Generator Configuration User Interface

The following topics describe the traffic generator user interface.

### 11.9.8.1. 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 232. Connecting the Traffic Generator



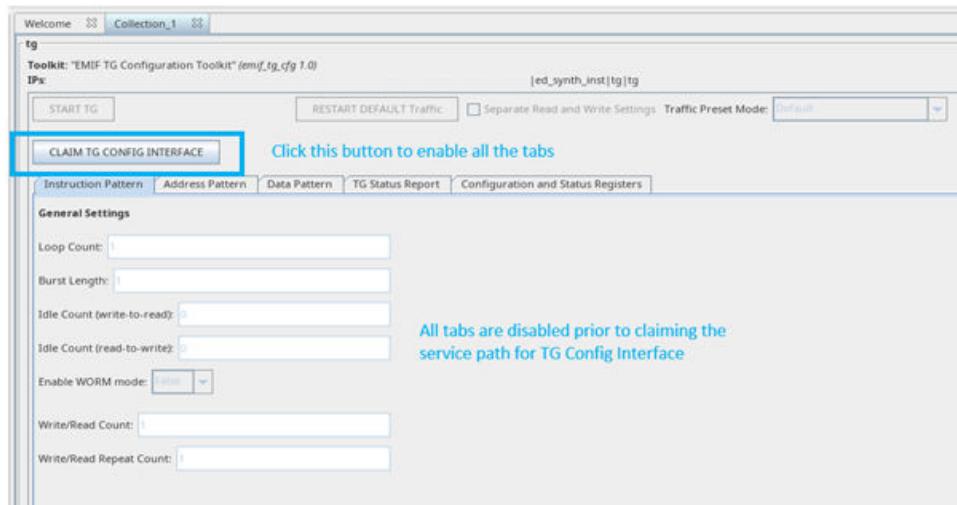
### 11.9.8.2. Claiming/Releasing the TG Config Interface

Before using the Traffic Generator (TG2) interface, you must claim the service to the TG Config interface. All the tabs in the traffic generator interface are disabled until you claim the service.

#### Graphical User Interface (GUI) Method

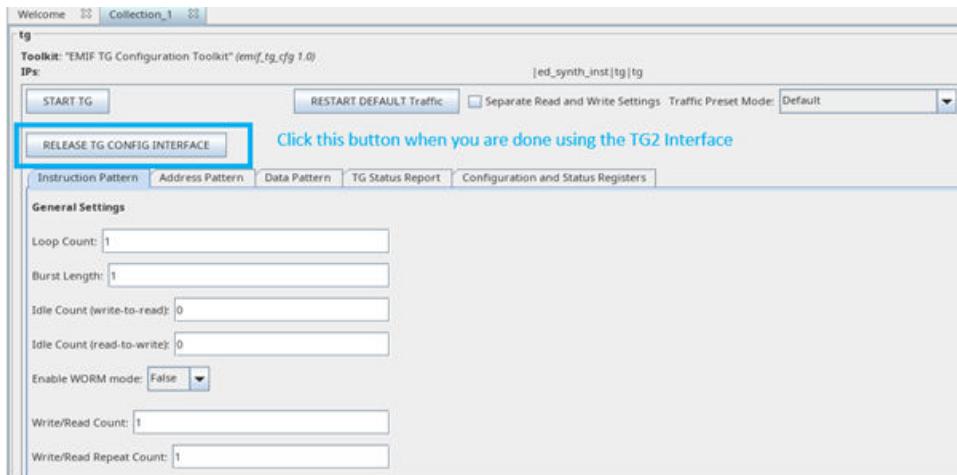
To claim the TG Config service, click **CLAIM TG CONFIG INTERFACE**. The tabs are enabled after you claim the service successfully.

**Figure 233. Click CLAIM TG CONFIG INTERFACE Before Using the TG2 Interface**



After you are finished using the TG2 interface, click **RELEASE TG CONFIG INTERFACE** to release the service instance and direct the System Console to make the resource available to other users.

**Figure 234. Click RELEASE TG CONFIG INTERFACE to Release the Service for Other Users**



### Scripting Method

If you are writing your own script to access the traffic generator, you must claim the TG Config service. Refer to the example below for claiming and releasing the service.

The following is the service path for the TG2 traffic generator:

```
*/alt_sld_fab_0_alt_sld_fab_0_host_link_jtag.h2t/
alt_sld_fab_0_alt_sld_fab_0_stfabric.h2t_0/
alt_sld_fab_0_alt_sld_fab_0_memfabric_transacto.avalon_master/
master_1_0.slave
```

Example code to claim and release the TG2 service:

```
#Example Code to claim/release the TG2 service
#Check for available service paths.
set service_paths [get_service_paths slave]

#Set the TG2 service path. Assume index 2 is the TG2 service path in this example
#User needs to check the service_paths list to identify the index for the TG2 in
your system
set tg_service_path [lindex $service_paths 2]

#Open the slave service.
set claimed_path [claim_service slave $tg_service_path mylib]

#User code to write/read to TG2
    #Read the TG Version
    puts "TG Version : [master_read_32 $claimed_path 0x0 1]"

    #Clear the TG_STATUS before Retest with Default Traffic
    puts "\nClear TG_STATUS"
    master_write_32 $claimed_path 0x20 0xf

    #Read the Read_Count-Low
    puts "Read Count - Low : [master_read_32 $claimed_path 0x1d8 1]"
    #Read the Read_Count-High
    puts "Read Count - High : [master_read_32 $claimed_path 0x1dc 1]"

    #Restart Default Traffic
    puts "\nWrite to TG_RESTART_DEFAULT_TRAFFIC"
    master_write_32 $claimed_path 0x1e8 1

    set TG_TEST_COMPLETE 0

    while { $TG_TEST_COMPLETE < 1 } {
        set TG_TEST_COMPLETE [expr [ master_read_32 $claimed_path 0x1e0 1] & 0x1 ]
        puts "\nTG_TEST_COMPLETE : $TG_TEST_COMPLETE"
    }

    #Read the Read_Count-Low
    puts "Read Count - Low : [master_read_32 $claimed_path 0x1d8 1]"
    #Read the Read_Count-High
    puts "Read Count - High : [master_read_32 $claimed_path 0x1dc 1]"
    #Read the TG_PASS
    puts "Pass : [expr [master_read_32 $claimed_path 0x1c0 1] & 0x1 ]"
    #Read the TG_FAIL
    puts "Fail : [expr [master_read_32 $claimed_path 0x1c4 1] & 0x1 ]"
#End of write/read to TG2

#Close the slave service.
close_service slave $claimed_path
```

### 11.9.8.3. 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 235. Instruction Pattern Tab**

The screenshot shows the 'Instruction Pattern' tab selected in a navigation bar. Below it is a section titled 'General Settings' containing the following fields:

- Loop Count:
- Burst Length:
- Idle Count (write-to-read):
- Idle Count (read-to-write):
- Enable WORM Mode:
- Write/Read Count:
- Write/Read Repeat Count:

**Figure 236. Instruction Pattern tab: Separate Read and Write Settings**

The screenshot shows the 'Instruction Pattern' tab selected in a navigation bar. Below it is a section titled 'General Settings' containing the following fields:

- Loop Count:
- Burst Length:
- Idle Count (write-to-read):
- Idle Count (read-to-write):
- Enable WORM Mode:

Below the general settings, there are two sections: 'Write Settings' and 'Read Settings'.

Write/Read Count: <input type="text" value="0"/> Write/Read Repeat Count: <input type="text" value="1"/>	Write/Read Count: <input type="text" value="0"/> Write/Read Repeat Count: <input type="text" value="1"/>
---	---

**Figure 238. Address Pattern Tab – Basic Mode**

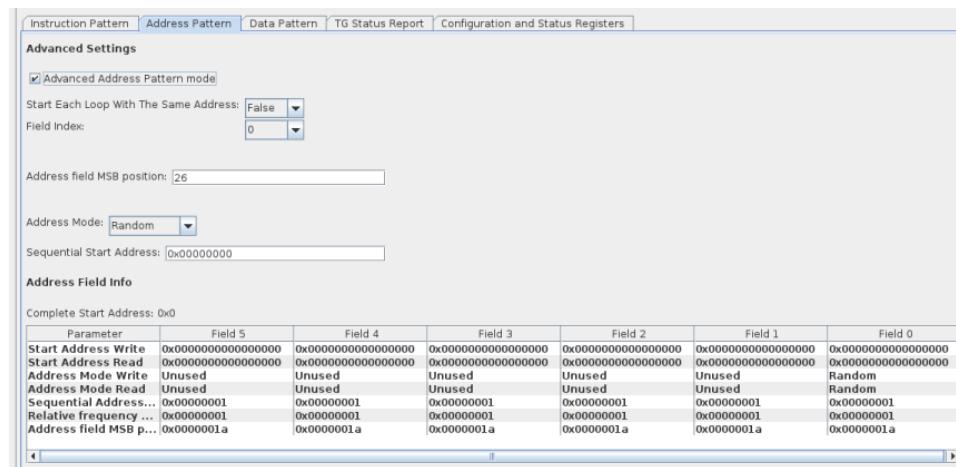
The screenshot shows the 'Address Pattern' tab selected in a navigation bar. Below it is a section titled 'General Settings' containing the following fields:

- Advanced Address Pattern mode
- Start Each Loop With The Same Address:

Below these, there are two more fields:

- Address Mode:
- Start Address:

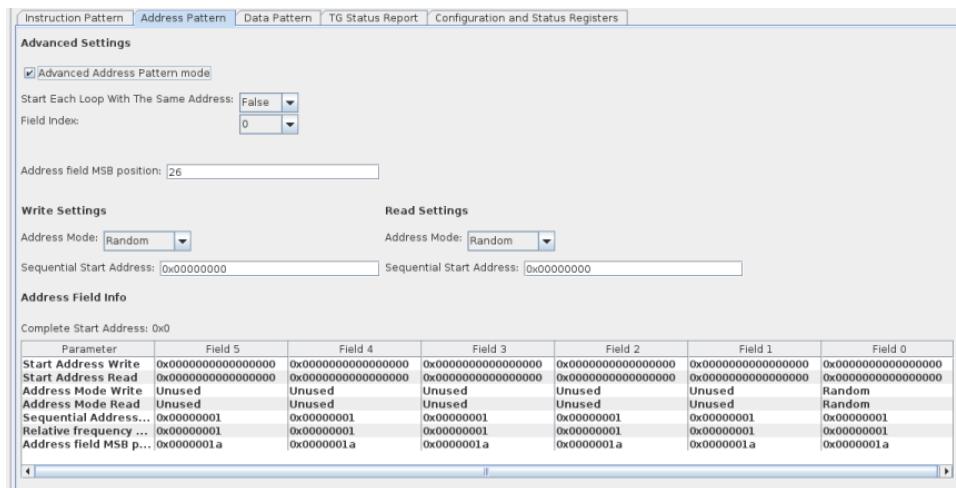
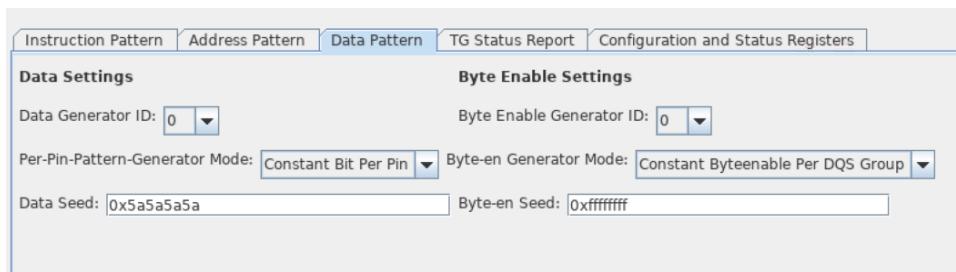
**Figure 239. Address Pattern Tab – Advanced Mode**



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 240. Address Pattern tab: Separate Read and Write Settings – Basic Mode**



**Figure 241. Address Pattern tab: Separate Read and Write Settings – Advanced Mode**

**Figure 242. 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 244. Configurations Tab**

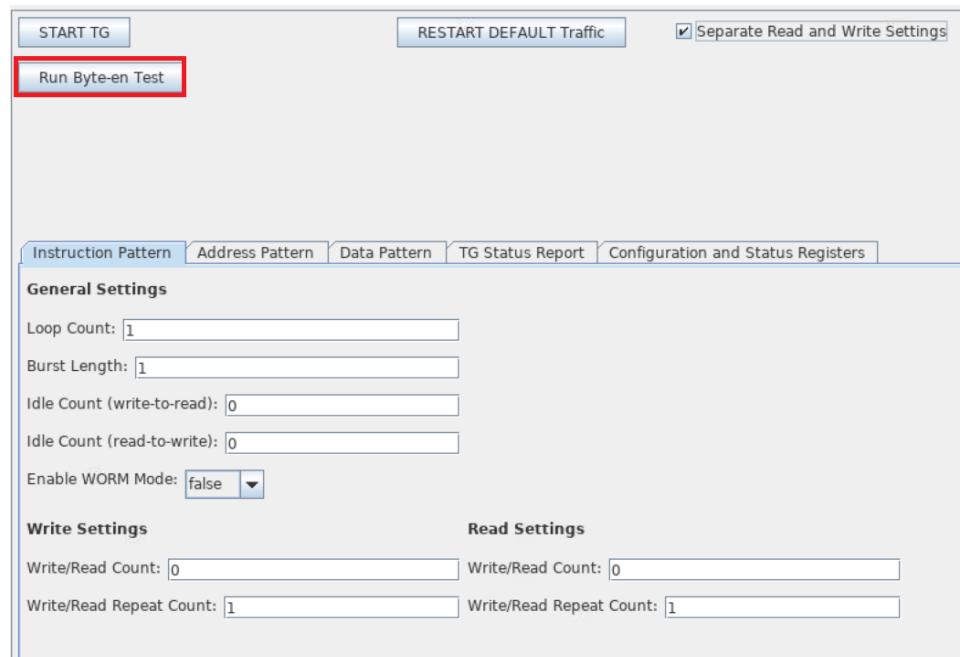
Parameter	Register Name	Address	Value
TG Version	TG VERSION	0x0	0x000000ab
Start User Traffic	TG START	0x1	0x00000000
Loop Count	TG LOOP_COUNT	0x2	0x00000001
Write Count	TG_WRITE_COUNT	0x3	0x00000001
Read Count	TG_READ_COUNT	0x4	0x00000001
Write Repeat Count	TG_WRITE_REPEAT_COUNT	0x5	0x00000001
Read Repeat Count	TG_READ_REPEAT_COUNT	0x6	0x00000001
Burst Length	TG_BURST_LENGTH	0x7	0x00000001
Clear Status	TG_CLEAR_STATUS	0x8	0x00000000
Idle Count (write-to-write)	TG_RW_GEN_IDLE_COUNT	0x9	0x00000000
Idle Count (read-to-write)	TG_RW_GEN_LOOP_IDLE_COUNT	0xf	0x00000000
Start Address Write - 0 - Low	TG_SEQ_START_ADDR_WR	0x10	0x00000000
Start Address Write - 0 - High	TG_SEQ_START_ADDR_WR	0x11	0x00000000
Start Address Write - 1 - Low	TG_SEQ_START_ADDR_WR	0x12	0x00000000
Start Address Write - 1 - High	TG_SEQ_START_ADDR_WR	0x13	0x00000000
Start Address Write - 2 - Low	TG_SEQ_START_ADDR_WR	0x14	0x00000000
Start Address Write - 2 - High	TG_SEQ_START_ADDR_WR	0x15	0x00000000
Start Address Write - 3 - Low	TG_SEQ_START_ADDR_WR	0x16	0x00000000
Start Address Write - 3 - High	TG_SEQ_START_ADDR_WR	0x17	0x00000000
Start Address Write - 4 - Low	TG_SEQ_START_ADDR_WR	0x18	0x00000000
Start Address Write - 4 - High	TG_SEQ_START_ADDR_WR	0x19	0x00000000
Start Address Write - 5 - Low	TG_SEQ_START_ADDR_WR	0x1a	0x00000000
Start Address Write - 5 - High	TG_SEQ_START_ADDR_WR	0x1b	0x00000000

### 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\_BYTEN (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\_BYTEN (symbol address 0xB8).

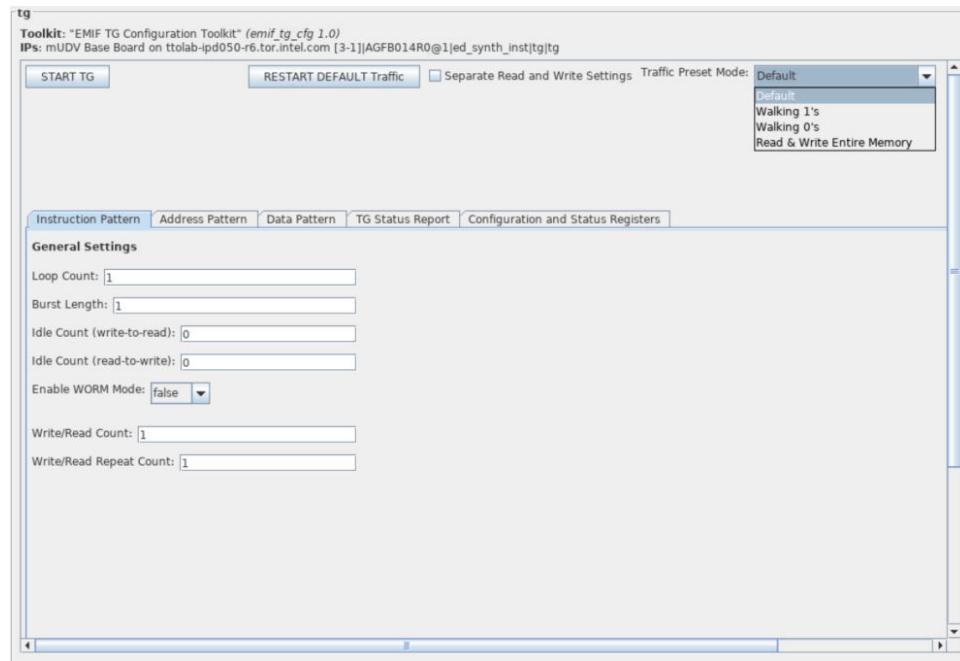
**Figure 245. Byte-enable Test**



#### 11.9.8.4. 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 246.** Traffic Generator Presets



### Default Preset

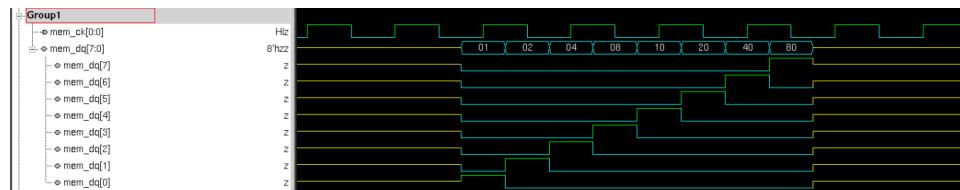
The **Default** preset loads the default values to the configuration registers. This sets the traffic generator to perform one read and one write at address 0 using the data seed *0x5a5a5a5a*.

### 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 247.** 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 248. 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.

#### **11.9.8.5. 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 249. TG Status Report (Passing Traffic Pattern)**

TG Status Report																																																																																																																																																																																																																																																																													
Configuration and Status Registers																																																																																																																																																																																																																																																																													
<b>Export TG Status Report</b>																																																																																																																																																																																																																																																																													
TG Status: <span style="color: green;">●</span>																																																																																																																																																																																																																																																																													
Write Overflow Status: <span style="color: red;">●</span>																																																																																																																																																																																																																																																																													
Number of Avalon Read Commands Issued: 260								<b>Clear</b>																																																																																																																																																																																																																																																																					
<b>Pass Not Fail Signal</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">DQ PIN</th> <th style="text-align: center; padding: 2px;">beat0</th> <th style="text-align: center; padding: 2px;">beat1</th> <th style="text-align: center; padding: 2px;">beat2</th> <th style="text-align: center; padding: 2px;">beat3</th> <th style="text-align: center; padding: 2px;">beat4</th> <th style="text-align: center; padding: 2px;">beat5</th> <th style="text-align: center; padding: 2px;">beat6</th> <th style="text-align: center; padding: 2px;">beat7</th> </tr> </thead> <tbody> <tr><td style="text-align: left; padding: 2px;">0</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">1</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">2</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">3</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">4</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">5</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">6</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">7</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">8</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">9</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">10</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">11</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">12</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">13</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">14</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">15</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">16</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">17</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">18</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">19</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">20</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">21</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">22</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">23</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">24</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">25</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">26</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> <tr><td style="text-align: left; padding: 2px;">27</td><td style="text-align: center; padding: 2px;">pass</td><td style="text-align: center; padding: 2px;">pass</td></tr> </tbody> </table>	DQ PIN	beat0	beat1	beat2	beat3	beat4	beat5	beat6	beat7	0	pass	1	pass	2	pass	3	pass	4	pass	5	pass	6	pass	7	pass	8	pass	9	pass	10	pass	11	pass	12	pass	13	pass	14	pass	15	pass	16	pass	17	pass	18	pass	19	pass	20	pass	21	pass	22	pass	23	pass	24	pass	25	pass	26	pass	27	pass	<b>Clear Failed Bits</b>																																																																																																																																																																																																											
DQ PIN	beat0	beat1	beat2	beat3	beat4	beat5	beat6	beat7																																																																																																																																																																																																																																																																					
0	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
1	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
2	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
3	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
4	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
5	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
6	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
7	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
8	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
9	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
10	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
11	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
12	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
13	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
14	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
15	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
16	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
17	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
18	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
19	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
20	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
21	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
22	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
23	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
24	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
25	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
26	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					
27	pass	pass	pass	pass	pass	pass	pass	pass																																																																																																																																																																																																																																																																					

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 250. TG Status Report (Failing Traffic Pattern)**

The screenshot shows the 'TG Status Report' tab of a system console. At the top, there are several status indicators and clear buttons:

- TG Status: **●**
- Write Overflow Status: **●**
- Burstlength overflow address: 0x0000000007ffff0
- Number of Avalon Read Commands issued: 3995
- Fail Count: 2000
- First Fail Addr: 0x00000000007ff885
- First Failure - Read Data: 0x0000000045f08600000000005b22a200155c008800000000842a028600000000
- First Failure - Expected Data: 0x0000000fefefe000000000fefefe00fffffff00000000fffffff00000000

Below these is a table titled 'Pass Not Fail Signal' with columns for DQ PIN and beat0 through beat7. The table shows a mix of 'pass' and 'fail' values across the 28 rows.

DQ PIN	beat0	beat1	beat2	beat3	beat4	beat5	beat6	beat7
0	fail							
1	fail							
2	fail							
3	fail							
4	fail							
5	fail							
6	fail							
7	fail							
8	pass	fail	pass	fail	fail	pass	fail	pass
9	pass	fail	pass	fail	fail	pass	fail	pass
10	pass	fail	pass	fail	pass	fail	pass	pass
11	pass	fail	pass	fail	fail	pass	fail	pass
12	pass	fail	pass	fail	fail	pass	fail	pass
13	pass	fail	pass	fail	fail	pass	fail	pass
14	pass	fail	pass	fail	fail	pass	fail	pass
15	pass	fail	pass	fail	fail	pass	fail	pass
16	pass	fail	pass	fail	fail	pass	fail	pass
17	pass	fail	pass	fail	fail	pass	fail	pass
18	pass	fail	pass	fail	fail	pass	fail	pass
19	pass	fail	pass	fail	fail	pass	fail	pass
20	pass	fail	pass	fail	fail	pass	fail	pass
21	pass	fail	pass	fail	fail	pass	fail	pass
22	pass	fail	pass	fail	fail	pass	fail	pass
23	pass	fail	pass	fail	fail	pass	fail	pass
24	pass	fail	pass	fail	fail	pass	fail	pass
25	pass	fail	pass	fail	fail	pass	fail	pass
26	pass	fail	pass	fail	fail	pass	fail	pass
27	pass	fail	pass	fail	fail	pass	fail	pass

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.

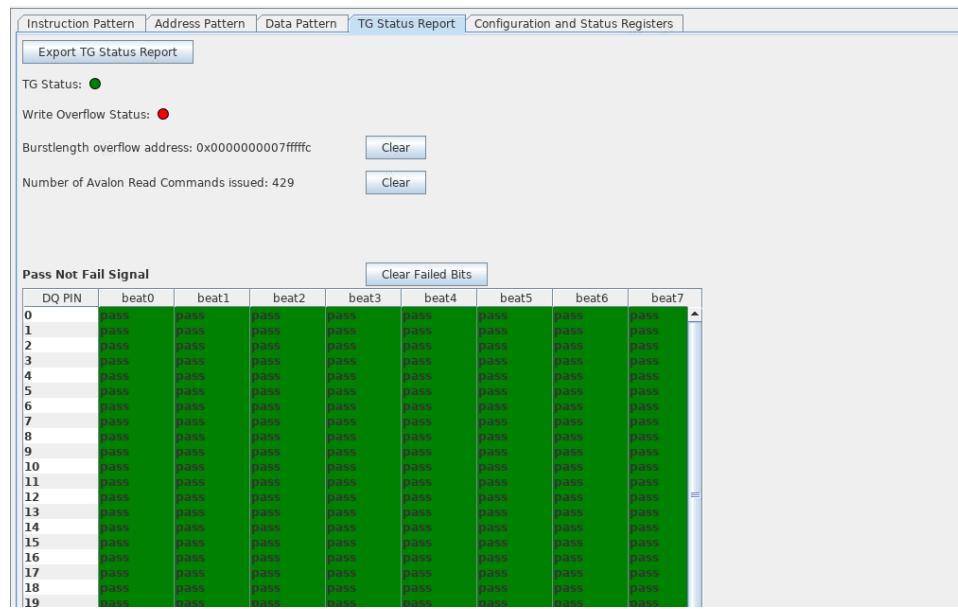
The PNF signal is logged in raw hex format. For information on reading and interpreting this format, refer to *Reading PNF Registers or ISSPs* in the [Traffic Generator Status](#) topic.

When the traffic generator is running in infinite user mode, you can update the status report only by clicking **Generate TG Status Report**.

**Figure 251. TG Status Report (While Running Infinite Traffic)**

TG Status Report												
Configuration and Status Registers												
<b>Export TG Status Report</b>					<b>Generate TG Status Report</b>							
TG Status:												
Write Overflow Status:												
Number of Avalon Read Commands issued: 118953859					<b>Clear</b>							
<b>Pass Not Fail Signal</b>												
<b>Clear Failed Bits</b>												
DQ PIN	beat0	beat1	beat2	beat3	beat4	beat5	beat6	beat7				
0	pass	pass	pass	pass	pass	pass	pass	pass				
1	pass	pass	pass	pass	pass	pass	pass	pass				
2	pass	pass	pass	pass	pass	pass	pass	pass				
3	pass	pass	pass	pass	pass	pass	pass	pass				
4	pass	pass	pass	pass	pass	pass	pass	pass				
5	pass	pass	pass	pass	pass	pass	pass	pass				
6	pass	pass	pass	pass	pass	pass	pass	pass				
7	pass	pass	pass	pass	pass	pass	pass	pass				
8	pass	pass	pass	pass	pass	pass	pass	pass				
9	pass	pass	pass	pass	pass	pass	pass	pass				
10	pass	pass	pass	pass	pass	pass	pass	pass				
11	pass	pass	pass	pass	pass	pass	pass	pass				
12	pass	pass	pass	pass	pass	pass	pass	pass				
13	pass	pass	pass	pass	pass	pass	pass	pass				
14	pass	pass	pass	pass	pass	pass	pass	pass				
15	pass	pass	pass	pass	pass	pass	pass	pass				
16	pass	pass	pass	pass	pass	pass	pass	pass				
17	pass	pass	pass	pass	pass	pass	pass	pass				
18	pass	pass	pass	pass	pass	pass	pass	pass				
19	pass	pass	pass	pass	pass	pass	pass	pass				
20	pass	pass	pass	pass	pass	pass	pass	pass				
21	pass	pass	pass	pass	pass	pass	pass	pass				
22	pass	pass	pass	pass	pass	pass	pass	pass				
23	pass	pass	pass	pass	pass	pass	pass	pass				
24	pass	pass	pass	pass	pass	pass	pass	pass				
25	pass	pass	pass	pass	pass	pass	pass	pass				
26	pass	pass	pass	pass	pass	pass	pass	pass				
27	pass	pass	pass	pass	pass	pass	pass	pass				

Figure 252. TG Status Report (Attempt to Overflow Address Space)



If you configure the traffic generator such that a generated address plus the value of the burst length is greater than the address space, a burst length overflow occurs. The traffic generator prevents this invalid command from being issued by setting the burstlength to the highest valid value for the one command where it would overflow. The toolkit displays the address at which the burst length would have overflowed the address space. You can use the clear button to clear the write overflow status and burst length overflow address between successive runs.

#### 11.9.8.6. 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 253. Address Width for Memory IP**

```

Instantiation Template <@pg-iccf0298>
You can copy the example HDL below to declare an instance of afs.
HDL Language: Verilog ▾

Example HDL


```

.mem_alert_n    (_connected_to_mem_alert_n_),           // input, width = 1, .mem_a
.mem_dqs        (_connected_to_mem_dqs_),             // inout, width = 9, .mem_dq
.mem_dqs_n      (_connected_to_mem_dqs_n_),           // inout, width = 9, .mem_dq
.mem_dq         (_connected_to_mem_dq_),              // inout, width = 72, .mem_dq
.mem_dbi_n      (_connected_to_mem_dbi_n_),           // inout, width = 9, .mem_dbi
.local_cal_success (_connected_to_local_cal_success_), // output, width = 1, status.local
.local_cal_fail  (_connected_to_local_cal_fail_),       // output, width = 1, .local
.emif_usr_reset_n (_connected_to_emif_usr_reset_n_),   // output, width = 1, emif_usr_reset_n.reset_
.emif_usr_clk     (_connected_to_emif_usr_clk_),        // output, width = 1, emif_usr_clk.clk
.amm_ready_0     (_connected_to_amm_ready_0_),          // output, width = 1, ctrl_amm_0.waitre
.amm_read_0      (_connected_to_amm_read_0_),           // input, width = 1, .read
.amm_write_0     (_connected_to_amm_write_0_),          // input, width = 1, .write
.amm_address_0   (_connected_to_amm_address_0_),         // input, width = 27, .address
.amm_readdata_0  (_connected_to_amm_readdata_0_),       // output, width = 576, .readdata
.amm_writedata_0 (_connected_to_amm_writedata_0_),       // input, width = 576, .writedat
.amm_burstcount_0 (_connected_to_amm_burstcount_0_),     // input, width = 7, .burstc
.amm_readdatavalid_0 (_connected_to_amm_readdatavalid_0_), // output, width = 1, .readdatavalid
.calbus_read     (_connected_to_calbus_read_),           // input, width = 1, emif_calbus.calbus_
.calbus_write    (_connected_to_calbus_write_),           // input, width = 1, .calbus
.calbus_address  (_connected_to_calbus_address_),          // input, width = 20, .calbus
.calbus_wdata    (_connected_to_calbus_wdata_),           // input, width = 32, .calbus
.calbus_rdata    (_connected_to_calbus_rdata_),           // output, width = 32, .calbus
.calbus_seq_param_tbl (_connected_to_calbus_seq_param_tbl_), // output, width = 4096, .calbus
.calbus_clk      (_connected_to_calbus_clk_)             // input, width = 1, emif_calbus_clk.clk
);

```


```

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)
= 227/64
= 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 167. 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_C OUNT	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_AD DR	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 DQ0 (DQ0/8/16/24/32/40/48/56/64).

**continued...**

<b>Address</b>	<b>Register Name</b>	<b>Value</b>	<b>Remarks</b>
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.

#### **11.10. EMIF On-Chip Debug Port**

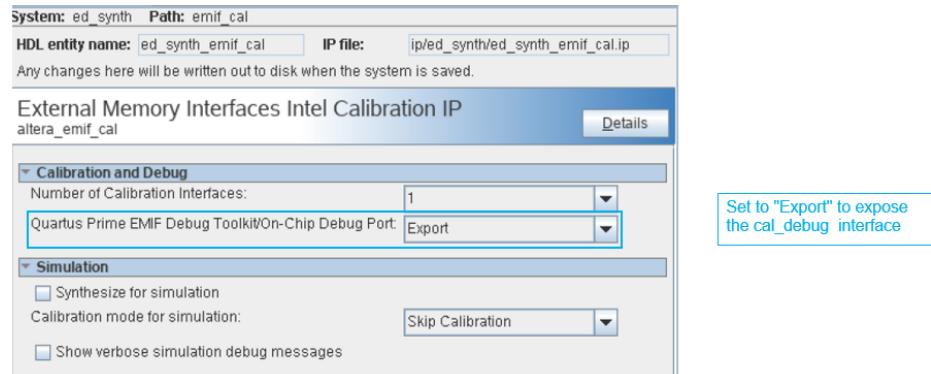
The On-Chip Debug Port (cal\_debug) provides access to the I/O SSM user-ram and calbus bridge, which contain debug information collected during EMIF calibration, and tools that can help analyze or improve interface stability.

The cal\_debug is an Avalon memory-mapped interface. This port does not provide access outside of the address ranges of the user-ram and calbus bridge.

### 11.10.1. Enabling the On-Chip Debug Port

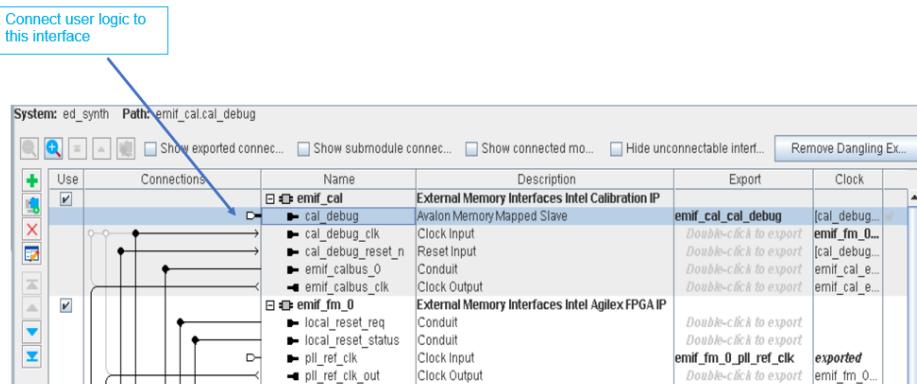
To export the cal\_debug port, set the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter to **Export**, when parameterizing the emif\_cal IP.

**Figure 254. Enabling the On-Chip Debug Port**



You may then create your own logic to perform the desired read/write commands on the cal\_debug Avalon memory-mapped interface.

**Figure 255. Connecting Your Own Logic**



### 11.10.2. I/O SSM calbus Bridge Data Structures and Usage

The I/O SSM calbus bridge spans addresses in the range: 0x0300\_0000 - 0x033f\_ffff, where each address represents a 32-bit word. The calbus bridge is composed of several data structures, which are outlined in the tables below.

At the base of the calbus bridge address range is an array of sub-bank structures, which mimic the physical sub-bank structure (described in the *I/O Bank* topic in the *Architecture* chapter of this user guide). The size of the sub-bank structure is 0x0001\_0000 bytes, such that sub-bank 0 appears at the base of the calbus bridge, sub-bank 1 is at 0x0301\_0000, sub-bank 2 is at 0x0302\_0000, and so on.

**Note:** Be aware that Intel does not recommend changing the calibrated delay setting value via the calbus bridge, as doing so may cause corruption of the PHY.

Each sub-bank structure instantiates four lane structures, a tile center structure, and an hmc structure.

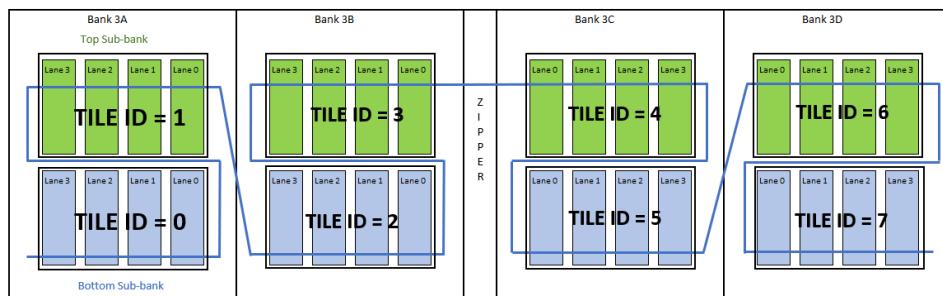
**Table 168. Sub-bank Structure**

Address	Structure Type	Structure Instance Name	Description
tile_addr_base_address = 0x0300_0000 + 0x0001_0000*tile_id	lane_struct	lane_0	Lane data structure, which mimics the physical structure of the lanes within a sub-bank.
tile_addr_base_address + 0x2000	lane_struct	lane_1	Lane data structure, which mimics the physical structure of the lanes within a sub-bank.
tile_addr_base_address + 0x4000	lane_struct	lane_2	Lane data structure, which mimics the physical structure of the lanes within a sub-bank.
tile_addr_base_address + 0x6000	lane_struct	lane_3	Lane data structure, which mimics the physical structure of the lanes within a sub-bank.
tile_addr_base_address + 0x8000	sub_bank_centre_struct	center	Data structure which mimics the physical structure of a tile-center (described in the <i>I/O Bank</i> topic in the <i>Architecture</i> chapter)
tile_addr_base_address + 0xA000	hmc_struct	hmc	Data structure which mimics the physical structure of the MMR space (described in the <i>MMR Tables</i> topic in the <i>End-User Signals</i> chapter).

*tile\_id* is the numerical position of a sub-bank when the sub-banks are chained in an I/O row. A *tile id* value of 0 denotes the bottom sub-bank within the left-most I/O bank.

The following figure depicts the tile IDs for the top I/O row in Intel Agilex 7 AGF012 and AGF014 devices, package R24A.

**Figure 256. Tile ID for Top I/O Row in Intel Agilex 7 AGF012 and AGF014, package R24A**



The lane structure consists of settings for the pins within the lane, as well as settings for the overall lane (such as DQS tree settings).

**Table 169. Lane Structure**

Address	Structure Type	Structure Instance Name
io_lane_addr (base address of lane_0, lane_1, lane_2, or lane_3)	pin_configuration_struct	pin_0
io_lane_addr + 0x0100	pin_configuration_struct	pin_1
io_lane_addr + 0x0200	pin_configuration_struct	pin_2
<i>continued...</i>		

Address	Structure Type	Structure Instance Name
io_lane_addr + 0x0300	pin_configuration_struct	pin_3
io_lane_addr + 0x0400	pin_configuration_struct	pin_4
io_lane_addr + 0x0500	pin_configuration_struct	pin_5
io_lane_addr + 0x0600	pin_configuration_struct	pin_6
io_lane_addr + 0x0700	pin_configuration_struct	pin_7
io_lane_addr + 0x0800	pin_configuration_struct	pin_8
io_lane_addr + 0x0900	pin_configuration_struct	pin_9
io_lane_addr + 0xa00	pin_configuration_struct	pin_10
io_lane_addr + 0xb00	pin_configuration_struct	pin_11
io_lane_addr + 0x1800	dqs_tree_struct	dqs

**Table 170. Pin Configuration Structure**

Size (bytes)	Register Name	Description	Offset from Structure Base Address
4	reg_counter_b_out	Data Output Delay.	0xC0
4	reg_dqs_toggle_count_clr	DQS Toggle Counter Clear.	0xCC
4	reg_out_phase_rank[4]	Data Output Delay Per Rank; each array element corresponds to a memory rank (if applicable).	0xD0
4	reg_io_pin_value	Direct read of pin value.	0xE4

**Table 171. DQS Tree Structure**

Size (bytes)	Register Name	Description	Offset from Structure Base Address
4	reg_sel_vref	Internal VREF Control (Vref-in).	0x14
4	reg_dq_in_delay_pin0[4]	DQ Input delay of Pin 0; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x80
4	reg_dq_in_delay_pin1[4]	DQ Input delay of Pin 1; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x90
4	reg_dq_in_delay_pin2[4]	DQ Input delay of Pin 2; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xA0
4	reg_dq_in_delay_pin3[4]	DQ Input delay of Pin 3; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xB0
4	reg_dq_in_delay_pin4[4]	DQ Input delay of Pin 4; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xC0
4	reg_dq_in_delay_pin5[4]	DQ Input delay of Pin 5; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xD0

*continued...*

<b>Size (bytes)</b>	<b>Register Name</b>	<b>Description</b>	<b>Offset from Structure Base Address</b>
4	reg_dqs_in_delay_a[4]	DQS input delay A; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xE0
4	reg_dqs_preamble_delay_a[4]	DQS preamble input delay A; each array element corresponds to a memory rank (if applicable). Bit[15] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0xF0
4	reg_dq_in_delay_pin6[4]	DQ Input delay of Pin 6; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x100
4	reg_dq_in_delay_pin7[4]	DQ Input delay of Pin 7; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x110
4	reg_dq_in_delay_pin8[4]	DQ Input delay of Pin 8; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x120
4	reg_dq_in_delay_pin9[4]	DQ Input delay of Pin 9; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x130
4	reg_dq_in_delay_pin10[4]	DQ Input delay of Pin 10; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x140
4	reg_dq_in_delay_pin11[4]	DQ Input delay of Pin 11; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x150
4	reg_dqs_in_delay_b[4]	DQS input delay B; each array element corresponds to a memory rank (if applicable). Bit[12] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x160
4	reg_dqs_preamble_delay_b[4]	DQS preamble input delay B; each array element corresponds to a memory rank (if applicable). Bit[15] is the enable bit (enable=1, disable=0). Bit[11:0] is the delay tap.	0x170

### 11.10.3. I/O SSM User-RAM Data Structures and Usage

The I/O SSM User-Ram spans addresses in the range: 0x0500\_0000 – 0x0500\_0fff, where each address represents a 32-bit word.

At the base address of the user-ram, is the Global Parameter Table , which contains pointers to the per-interface parameter table. The per-interface parameter table contains a pointer to the Debug Data Structure. There is one debug data structure per emif\_cal IP (that is, one per I/O row).

#### 11.10.3.1. Parameter Tables

##### 11.10.3.1.1. Global Parameter Table

The Global Parameter Table's base address is 0x0500\_0000 (which is the base address of the user RAM).

**Table 172. Global Parameters**

Offset from GPT Base	Field Name	Field Width (in bytes)	Description
0x0	gpt_GLOBAL_PAR_VER	4	Global Parameter Table version number.
0x4	gpt_NIOS_C_VER	4	Firmware version.
0x8	gpt_COLUMN_ID	4	ID of I/O row.
0xC	gpt_NUM_IOPACKS	4	Maximum number of I/O tiles in the row that can be used as memory interface.
0x10	gpt_NIOS_CLK_FREQ_KHZ	4	Nios clock frequency (KHz).
0x14	gpt_PARAM_TABLE_SIZE	4	Size (in bytes) of the global and mem interface parameter tables combined.
0x1C	gpt_GLOBAL_CAL_CONFIG	4	Asserts whether the debug toolkit is enabled (enabled if bits 0 and 2 are asserted).
0x20	gpt_SLAVE_CLK_DIVIDER	4	Divider for the calbus clock. Value 0 may be observed, indicating that firmware set an optimal value, assuming there is no other master accessing the calbus at the same time.
0x24	gpt_INTERFACE_PAR_PTRS	4	Pointers to memory interface parameter tables. The number of array elements is equal to gpt_NUM_IOPACKS. offset 0x24 contains the pointer to the first interface (interface ID=0). offset 0x28 contains the pointer to the second interface (interface ID=1). offset 0x32 contains the pointer to the third interface, and so forth. The index of the cal_bus to which an interface is connected determines its interface ID. An interface connected to cal_bus_0 is interface 0, while an interface connected to cal_bus_1 is interface 1, and so forth. bit[31:16]: Reserved. bit [15:0]: The pointer to memory interface parameter table, coded as the offset from the start of the user RAM. When a pointer has value 0, it means the memory interface is not used and its parameter table does not exist.

#### 11.10.3.1.2. Per-interface Parameter Table Structure

The Per-Interface parameter table's base address is equal to: *(user-ram base address) + (interface parameter table offset, read from gpt\_INTERFACE\_PAR\_PTRS array)*.

**Table 173. Per-interface Parameters**

<b>Address</b>	<b>Data Type</b>	<b>Bit Position</b>	<b>Field Name</b>	<b>Description</b>
base address for per-interface parameter table (per_interface_base_address) = 0x0500_0000 + per-interface offset	alt_u16	Bit[15:0]	pt_IP_VER	IP ACDS version encoded as a 16-bit value, where: <ul style="list-style-type: none"><li>• Bit 15 is reserved.</li><li>• Bits 14-10 encode the major release number.</li><li>• Bits 9-6 encode the minor release number.</li><li>• Bits 5-3 encode the service pack number.</li><li>• Bits 2-0 encode whether this is a special variant.</li></ul>
	alt_u16	Bit[31:16]	pt_INTERFACE_PAR_VER	Version of this parameter table (Nios checks compatibility).
per_interface_base_address + 0x04	alt_u16	Bit[15:0]	pt_DEBUG_DATA_PTR	Debug_data_struct offset from the starting address of the user-ram.
	alt_u16	Bit[31:16]	pt_UNUSED	Unused.
per_interface_base_address + 0x08	alt_u8	Bit[7:0]	pt_MEMORY_TYPE	Memory type, defined by ENUM_MEM_TYPE.
	alt_u8	Bit[15:8]	pt_DIMM_TYPE	DIMM type, defined by ENUM_DIMM_TYPE.
	alt_u8	Bit[31:16]	pt_RESERVED	Reserved entry.
per_interface_base_address + 0x0C	alt_u32	Bit[31:0]	pt_AFI_CLK_FREQ_KHZ	AFI clock frequency in KHz.
per_interface_base_address + 0x10	alt_u8	Bit[7:0]	pt_BURST_LEN	Burst length.
	alt_u8	Bit[15:8]	pt_READ_LATENCY	Effective Read Latency. The value is a fixed-point number in 7.1 format: bit [6:0] is the integer part and bit 7 is the fractional part. For example: <ul style="list-style-type: none"><li>• value 0x2 means RL=2</li><li>• value 0x82 means RL=2.5 (useful for QDR memory)</li></ul> The value should be equal to $(CL + AL + PL)$ , where: $CL$ is CAS Latency, $AL$ is Additive Latency (for the memories supporting it; 0 otherwise), and $PL$ is Parity Latency (for memories supporting it if parity feature is enabled; 0 otherwise).
	alt_u8	Bit[23:16]	pt_WRITE_LATENCY	Effective Write Latency. Equals $(WL + AL + PL)$ , where: $WL$ is Write Latency, and $AL$ and $PL$ are the same as those used in READ_LATENCY above.

*continued...*

Address	Data Type	Bit Position	Field Name	Description
	alt_u8	Bit[31:24]	pt_NUM_RANKS	Number of electrical loads of DQ/DQS.
per_interface_base_address + 0x14	alt_u8	Bit[7:0]	pt_NUM_DIMMS	Number of DIMM slots.
	alt_u8	Bit[15:8]	pt_NUM_DQS_WR	Number of write DQS pins.
	alt_u8	Bit[23:16]	pt_NUM_DQS_RD	Number of read DQS pins.
	alt_u8	Bit[31:24]	pt_NUM_DQ	Number of DQ pins.
per_interface_base_address + 0x18	alt_u8	Bit[7:0]	pt_NUM_DM	Number of DM pins (or DM/DBI pins for DDR4).
	alt_u8	Bit[15:8]	pt_ADDR_WIDTH	Number of address pins.
	alt_u8	Bit[23:16]	pt_BANK_WIDTH	Bank Address width; equals log2(number-of-banks).
	alt_u8	Bit[31:24]	pt_CS_WIDTH	CS width; in most cases equals NUM_RANKS.
per_interface_base_address + 0x1C	alt_u8	Bit[7:0]	pt_CKE_WIDTH	CKE width; in most cases equals CS_WIDTH.
	alt_u8	Bit[15:8]	pt_ODT_WIDTH	ODT width; in most cases equals CS_WIDTH.
	alt_u8	Bit[23:16]	pt_C_WIDTH	Chip ID width for DDR4.
	alt_u8	Bit[31:24]	pt_BANK_GROUP_WIDTH	Bank group width for DDR4.
per_interface_base_address + 0x20	alt_u8	Bit[7:0]	pt_ADDR_MIRROR	Address mirroring configuration; one bit enable per rank enables.
	alt_u8	Bit[15:8]	pt_CK_WIDTH	Number of pairs of CK/CK_N; in most cases equals to CS_WIDTH.
	alt_u8	Bit[23:16]	pt_CAL_DATA_SIZE	Size of the pt_CAL_DATA_PTR array (in bytes).
	alt_u8	Bit[31:24]	pt_NUM_LRDIMM_CFG	(LRDIMM only) Number of the triplets of code words for LRDIMM. On non-LRDIMM configuration this entry should be set to 0.
per_interface_base_address + 0x24	alt_u8	Bit[7:0]	pt_NUM_AC_ROM_ENUMS	Number of AC ROM enums for the current protocol.
	alt_u8	Bit[15:8]	pt_NUM_CENTERS	Number of tiles used by this interface.
	alt_u8	Bit[23:16]	pt_NUM_CA_LANES	Number of command/address lanes.
	alt_u8	Bit[31:24]	pt_NUM_DATA_LANES	Number of data lanes.

*continued...*

<b>Address</b>	<b>Data Type</b>	<b>Bit Position</b>	<b>Field Name</b>	<b>Description</b>
per_interface_base_address + 0x28	alt_u32	Bit[31:0]	pt_ODT_TABLE_LO	ODT table; 4 bits as in ENUM_ODT_TABLE order: [odt3_cs1, odt2_cs1, ..., odt0_cs0]
per_interface_base_address + 0x2C	alt_u32	Bit[31:0]	pt_ODT_TABLE_HI	ODT table; 4 bits as in ENUM_ODT_TABLE order: [odt3_cs3, odt2_cs3, ..., odt0_cs2]
per_interface_base_address + 0x34	alt_u16	Bit[15:0]	pt_RESERVED	Field not used currently.
	alt_u16	Bit[31:16]	pt_CAL_DATA_PTR	cal_data array offset from the starting address of the user-ram. This array's structure is described in the Parameter Table Arrays section.
per_interface_base_address + 0x38	alt_u32	Bit[31:0]	pt_DBG_SKIP_RANKS	Each set bit indicates that storing calibration report information for the corresponding rank should be skipped. For example, if bit[1]==1, then in a 2-rank design, the second rank's calibration debug information is not stored.
per_interface_base_address + 0x3C	alt_u32	Bit[31:0]	pt_DBG_SKIP_GROUPS	Each set bit indicates that storing calibration report information for the corresponding DQS group should be skipped.
per_interface_base_address + 0x40	alt_u32	Bit[31:0]	pt_DBG_SKIP_STEPS	Each set bit represents a calibration step which should be skipped, as defined in ENUM_DBG_CALIB_SKIP.
per_interface_base_address + 0x44	alt_u8	Bit[7:0]	pt_NUM_MR	Number of words that store values to be written to Mode Registers.
	alt_u8	Bit[15:8]	pt_NUM_DIMM_MR	Number of words that store control words for DIMM.
	alt_u16	Bit[31:16]	pt_TILE_ID_PTR	tile_id array pointer, as an offset from the starting address of the user-ram; This array maps IDs of lanes used by the memory interface to the tiles they are placed in.
per_interface_base_address + 0x48	alt_u16	Bit[15:0]	pt_PIN_ADDR_PTR	pin_addr array pointer, as an offset from the starting address of the user-ram; This array stores the pin locations (on the calbus) of command/address and data pins.
	alt_u16	Bit[31:16]	pt_MR_PTR	Mode Register and RDIMM/LRDIMM control word array pointer, as an offset from the starting address of the user RAM.

### 11.10.3.1.3. Parameter Table Arrays

This topic describes the structure of some arrays whose addresses are stored in the parameter table. These arrays are used during calibration and may be useful for debugging your design.

#### Cal Data Array (pt\_CAL\_DATA\_PTR)

The Cal Data Array's base address is equal to: (user\_ram base address) + (offset for cal\_data array, read from pt\_CAL\_DATA\_PTR)

**Table 174. cal\_data Array**

Offset from Pointer Address	Entry	Size (bytes)
0x0	STARTING_VREFIN	4
0x4	CAL_TREFI	4
0x8	CAL_TRFC	4
0xC	CAL_ADDR0	4
0x10	CAL_ADDR1	4

#### MR Array (pt\_MR\_PTR)

The MR Array's base address is equal to: (user\_ram base address) + (offset for MR array, read from pt\_MR\_PTR)

Each array element is 32 bits wide. There can be up to three types of information in this array, as follows:

- Values to be written into Mode Registers. For ordering refer to ENUM\_MR\_INDEX. The number of entries of this type is *pt\_NUM\_MR*, and should be 0 for non-DDR protocols. Each entry is coded as 32 bits: {19'b0, A[12:0]} (Bank ADDR not included).
- On RDIMM or LRDIMM, the data above is followed by 64 bit RDIMM configuration, comprising 16 control words of 4 bits each.
- On LRDIMM, the data above is followed by (*pt\_NUM\_LRDIMM\_CFG* \* 3) number of bytes for extra LRDIMM configuration, each entry being 3 bytes of data with no padding bytes between triplets. Each triplet is defined as {func\_sel, address, value}. For example, {3, 2, 6} means F3RC2=6 .

#### Pin Address Array (pt\_PIN\_ADDR\_PTR)

Each array element is 8 bits wide. The order of the pins is as follows:

- All the pins in the appropriate ENUM\_AC\_ROM\_\* (based on the protocol used by this design) for command/address pins.
- All data pins in the following order, where each item in the brackets below can be a single bit, multi-bit bus (from LSB to MSB), or zero entry (non-existing bus):
  - DDR4: [DQS\_T, DQS\_C, DM, DQ]
  - QDR-IV: [QKA, QKB, QKA\_N, QKB\_N, DKA, DKB, DKA\_N, DKB\_N, DIVA, DIVB, DQA, DQB]

For example:

The following table depicts the pin address array for a DDR4 interface of x16 data width, split into two DQS groups:

**Table 175. Example**

Section	Address	Element
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr = user_ram base address + pt_PIN_ADDR_PTR	DDR4_CKE_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1	DDR4_CKE_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2	DDR4_CKE_2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x3	DDR4_CKE_3
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x4	DDR4_ODT_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x5	DDR4_ODT_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x6	DDR4_ODT_2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x7	DDR4_ODT_3
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x8	DDR4_RESET
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x9	DDR4_ACT
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xA	DDR4_CS_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xB	DDR4_CS_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xC	DDR4_CS_2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xD	DDR4_CS_3
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xE	DDR4_C_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0xF	DDR4_C_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x10	DDR4_C_2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x11	DDR4_BA_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x12	DDR4_BA_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x13	DDR4_BG_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x14	DDR4_BG_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x15	DDR4_ADD_0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x16	DDR4_ADD_1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x17	DDR4_ADD_2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x18	DDR4_ADD_3
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x19	DDR4_ADD_4
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1A	DDR4_ADD_5
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1B	DDR4_ADD_6
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1C	DDR4_ADD_7
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1D	DDR4_ADD_8
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1E	DDR4_ADD_9

*continued...*

Section	Address	Element
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x1F	DDR4_ADD_10
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x20	DDR4_ADD_11
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x21	DDR4_ADD_12
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x22	DDR4_ADD_13
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x23	DDR4_ADD_14
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x24	DDR4_ADD_15
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x25	DDR4_ADD_16
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x26	DDR4_ADD_17
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x27	DDR4_ADD_18
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x28	DDR4_ADD_19
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x29	DDR4_PAR_IN
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2A	REAL_AC_PIN_DDR4_NUM
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2A	DDR4_RDATA_EN
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2B	DDR4_MRNK_RD
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2C	DDR4_WDATA_VALID
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2D	DDR4_MRNK_WRT
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2E	AC_DDR4_NUM
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2A	DDR4_ALERT0_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2B	DDR4_ALERT1_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2C	DDR4_CK0
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2D	DDR4_CK0_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2E	DDR4_CK1
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x2F	DDR4_CK1_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x30	DDR4_CK2
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x31	DDR4_CK2_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x32	DDR4_CK3
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x33	DDR4_CK3_N
1 (ENUM_AC_ROM_DDR4)	Pin Address Array Base Addr + 0x34	AC_PIN_DDR4_NUM
2 (DQS_T)	Pin Address Array Base Addr + 0x35	DQS_0
2 (DQS_T)	Pin Address Array Base Addr + 0x36	DQS_1
2 (DQS_C)	Pin Address Array Base Addr + 0x37	DQS_N_0
2 (DQS_C)	Pin Address Array Base Addr + 0x38	DQS_N_1
2 (DM)	Pin Address Array Base Addr + 0x39	DM_0
2 (DM)	Pin Address Array Base Addr + 0x40	DM_1
2 (DQ)	Pin Address Array Base Addr + 0x41	DQ_0

*continued...*

<b>Section</b>	<b>Address</b>	<b>Element</b>
2 (DQ)	Pin Address Array Base Addr + 0x42	DQ_1
2 (DQ)	Pin Address Array Base Addr + 0x43	DQ_2
2 (DQ)	Pin Address Array Base Addr + 0x44	DQ_3
2 (DQ)	Pin Address Array Base Addr + 0x45	DQ_4
2 (DQ)	Pin Address Array Base Addr + 0x46	DQ_5
2 (DQ)	Pin Address Array Base Addr + 0x47	DQ_6
2 (DQ)	Pin Address Array Base Addr + 0x48	DQ_7
2 (DQ)	Pin Address Array Base Addr + 0x49	DQ_8
2 (DQ)	Pin Address Array Base Addr + 0x50	DQ_9
2 (DQ)	Pin Address Array Base Addr + 0x51	DQ_10
2 (DQ)	Pin Address Array Base Addr + 0x52	DQ_11
2 (DQ)	Pin Address Array Base Addr + 0x53	DQ_12
2 (DQ)	Pin Address Array Base Addr + 0x54	DQ_13
2 (DQ)	Pin Address Array Base Addr + 0x55	DQ_14
2 (DQ)	Pin Address Array Base Addr + 0x56	DQ_15

In most cases, each entry is encoded as 8 bits: {lane-index[3:0], pin-index[3:0]}, where lane-index is the index to the lane addresses in *pt\_TILE\_ID\_PTR* and pin-index is the physical pin index in that lane.

16-bit encoding is used when the number of data groups exceeds 16, so 4 bits are insufficient. 16-bit encoding is also used for CA pins such as alert\_n, which are not necessarily physically located in a CA lane. If an alert\_n pin is placed in a data lane, the absolute tile-id value is used: {1'b0, data-lane-index[6:0], pin-index[3:0], 4'b1111}, where data-lane-index is similar to lane-index in 8-bit encoding, except that it is always based off data lanes, regardless of pin types. For 16-bit code, the LSB byte appears first, followed by the MSB byte.

The lane-index is based off of the corresponding group, specifically either the CA lanes or the data lanes. (For an explanation of CA lanes and data lanes, refer to the comments of *pt\_TILE\_ID\_PTR*, above).

For example:

In a DDR4 interface, the array element for DQ\_0 = 8'b0001\_0010

lane\_index = 4'b0001

pin\_index = 4'b0010

Because this is a data pin, you should refer to index 1 in DATA\_LANES (that is, "DATA\_LANES: TILE ID 1" in Tile ID Array ) to get the actual tile and lane location for DQ0 data pin for this interface.

The unused slots (such as CKE\_3 when there are only 2 CKE pins in the design) must be filled with 8'b0.

### Tile ID Array (pt\_TILE\_ID\_PTR)

Each array element is an 8-bit-wide tile-id: {tile-pos[4:0], lane-id[2:0]}. This array contains tile IDs of three groups, in the following order:

1. Centers.
2. CA lanes.
3. Data lanes.

pt\_NUM\_CENTERS, pt\_NUM\_CA\_LANES, and pt\_NUM\_DATA\_LANES indicate how many tile IDs exist in Centers, CA lanes, and Data lanes respectively.

The table below depicts the Tile ID Array for an example with the following characteristics:

- pt\_NUM\_CENTERS = 0x03
- pt\_NUM\_CA\_LANES = 0x03
- pt\_NUM\_DATA\_LANES = 0x09

**Table 176. Tile ID Array Example**

Address	Group	Element
Base Address for Tile ID Array = user-ram base address + pt_TILE_ID_PTR offset	CENTERS	CENTERS: Tile ID 0
Base Address for Tile ID Array + 0x1	CENTERS	CENTERS: Tile ID 1
Base Address for Tile ID Array + 0x2	CENTERS	CENTERS: Tile ID 2
Base Address for Tile ID Array + 0x3	CA_LANES	CA_LANES: Tile ID 0
Base Address for Tile ID Array + 0x4	CA_LANES	CA_LANES: Tile ID 1
Base Address for Tile ID Array + 0x5	CA_LANES	CA_LANES: Tile ID 2
Base Address for Tile ID Array + 0x6	DATA_LANES	DATA_LANES: TILE ID 0
Base Address for Tile ID Array + 0x7	DATA_LANES	DATA_LANES: TILE ID 1
Base Address for Tile ID Array + 0x8	DATA_LANES	DATA_LANES: TILE ID 2
Base Address for Tile ID Array + 0x9	DATA_LANES	DATA_LANES: TILE ID 3
Base Address for Tile ID Array + 0xA	DATA_LANES	DATA_LANES: TILE ID 4
Base Address for Tile ID Array + 0xB	DATA_LANES	DATA_LANES: TILE ID 5
Base Address for Tile ID Array + 0xC	DATA_LANES	DATA_LANES: TILE ID 6
Base Address for Tile ID Array + 0xD	DATA_LANES	DATA_LANES: TILE ID 7
Base Address for Tile ID Array + 0xE	DATA_LANES	DATA_LANES: TILE ID 8

The order of lanes inside each group is arbitrary, except that the first address must be the center address with an active sequencer.

#### 11.10.3.2. Parameter Table Enums

This topic describes enums that are used during calibration and can be useful when debugging your design.

**ENUM\_AC\_ROM\_DDR4**

<b>Entry</b>	<b>Value</b>
AC_ROM_DDR4_CKE_0	0
AC_ROM_DDR4_CKE_1	1
AC_ROM_DDR4_CKE_2	2
AC_ROM_DDR4_CKE_3	3
AC_ROM_DDR4_ODT_0	4
AC_ROM_DDR4_ODT_1	5
AC_ROM_DDR4_ODT_2	6
AC_ROM_DDR4_ODT_3	7
AC_ROM_DDR4_RESET	8
AC_ROM_DDR4_ACT	9
AC_ROM_DDR4_CS_0	10
AC_ROM_DDR4_CS_1	11
AC_ROM_DDR4_CS_2	12
AC_ROM_DDR4_CS_3	13
AC_ROM_DDR4_C_0	14
AC_ROM_DDR4_C_1	15
AC_ROM_DDR4_C_2	16
AC_ROM_DDR4_BA_0	17
AC_ROM_DDR4_BA_1	18
AC_ROM_DDR4_BG_0	19
AC_ROM_DDR4_BG_1	20
AC_ROM_DDR4_ADD_0	21
AC_ROM_DDR4_ADD_1	22
AC_ROM_DDR4_ADD_2	23
AC_ROM_DDR4_ADD_3	24
AC_ROM_DDR4_ADD_4	25
AC_ROM_DDR4_ADD_5	26
AC_ROM_DDR4_ADD_6	27
AC_ROM_DDR4_ADD_7	28
AC_ROM_DDR4_ADD_8	29
AC_ROM_DDR4_ADD_9	30
AC_ROM_DDR4_ADD_10	31
AC_ROM_DDR4_ADD_11	32
AC_ROM_DDR4_ADD_12	33
AC_ROM_DDR4_ADD_13	34

*continued...*

Entry	Value
AC_ROM_DDR4_ADD_14	35
AC_ROM_DDR4_ADD_15	36
AC_ROM_DDR4_ADD_16	37
AC_ROM_DDR4_ADD_17	38
AC_ROM_DDR4_ADD_18	39
AC_ROM_DDR4_ADD_19	40
AC_ROM_DDR4_PAR_IN	41
REAL_AC_PIN_DDR4_NUM	42
AC_ROM_DDR4_RDATA_EN	REAL_AC_PIN_DDR4_NUM
AC_ROM_DDR4_MRNK_RD	43
AC_ROM_DDR4_WDATA_VALID	44
AC_ROM_DDR4_MRNK_WRT	45
AC_DDR4_NUM	46
AC_ROM_DDR4_ALERT0_N	REAL_AC_PIN_DDR4_NUM + 0
AC_ROM_DDR4_ALERT1_N	REAL_AC_PIN_DDR4_NUM + 1
AC_ROM_DDR4_CK0	REAL_AC_PIN_DDR4_NUM + 2
AC_ROM_DDR4_CK0_N	REAL_AC_PIN_DDR4_NUM + 3
AC_ROM_DDR4_CK1	REAL_AC_PIN_DDR4_NUM + 4
AC_ROM_DDR4_CK1_N	REAL_AC_PIN_DDR4_NUM + 5
AC_ROM_DDR4_CK2	REAL_AC_PIN_DDR4_NUM + 6
AC_ROM_DDR4_CK2_N	REAL_AC_PIN_DDR4_NUM + 7
AC_ROM_DDR4_CK3	REAL_AC_PIN_DDR4_NUM + 8
AC_ROM_DDR4_CK3_N	REAL_AC_PIN_DDR4_NUM + 9
AC_PIN_DDR4_NUM	52

#### ENUM\_AC\_ROM\_QDR4

Entry	Value
AC_ROM_QDRIV_RESET	0
AC_ROM_QDRIV_CFG	1
AC_ROM_QDRIV_LDA	2
AC_ROM_QDRIV_LDB	3
AC_ROM_QDRIV_RWA	4
AC_ROM_QDRIV_RWB	5
AC_ROM_QDRIV_AINV	6
AC_ROM_QDRIV_AP	7
AC_ROM_QDRIV_ADD_0	8

*continued...*

<b>Entry</b>	<b>Value</b>
AC_ROM_QDRIV_ADD_1	9
AC_ROM_QDRIV_ADD_2	10
AC_ROM_QDRIV_ADD_3	11
AC_ROM_QDRIV_ADD_4	12
AC_ROM_QDRIV_ADD_5	13
AC_ROM_QDRIV_ADD_6	14
AC_ROM_QDRIV_ADD_7	15
AC_ROM_QDRIV_ADD_8	16
AC_ROM_QDRIV_ADD_9	17
AC_ROM_QDRIV_ADD_10	18
AC_ROM_QDRIV_ADD_11	19
AC_ROM_QDRIV_ADD_12	20
AC_ROM_QDRIV_ADD_13	21
AC_ROM_QDRIV_ADD_14	22
AC_ROM_QDRIV_ADD_15	23
AC_ROM_QDRIV_ADD_16	24
AC_ROM_QDRIV_ADD_17	25
AC_ROM_QDRIV_ADD_18	26
AC_ROM_QDRIV_ADD_19	27
AC_ROM_QDRIV_ADD_20	28
AC_ROM_QDRIV_ADD_21	29
AC_ROM_QDRIV_ADD_22	30
AC_ROM_QDRIV_ADD_23	31
AC_ROM_QDRIV_ADD_24	32
AC_ROM_QDRIV_LBK_0	33
AC_ROM_QDRIV_LBK_1	34
REAL_AC_PIN_QDRIV_NUM	35
AC_ROM_QDRIV_RDATA_EN	REAL_AC_PIN_QDRIV_NUM
AC_ROM_QDRIV_MRNK_RD	36
AC_ROM_QDRIV_WDATA_VALID	37
AC_ROM_QDRIV_MRNK_WRT	38
AC_QDRIV_NUM	39
AC_ROM_QDRIV_PE	REAL_AC_PIN_QDRIV_NUM + 0
AC_ROM_QDRIV_CK	REAL_AC_PIN_QDRIV_NUM + 1
AC_ROM_QDRIV_CK_N	REAL_AC_PIN_QDRIV_NUM + 2
AC_PIN_QDRIV_NUM	38

### ENUM\_DBG\_CALIB\_SKIP

**Table 177.**

Entry	Value
CALIB_SKIP_VREFIN_CAL	(1 << 14)
CALIB_SKIP_VREFOUT_CAL	(1 << 15)

### ENUM\_DIMM\_TYPE

Entry	Value
DIMM_COMPONENT	0
DIMM_UDIMM	1
DIMM_RDIMM	2
DIMM_SODIMM	3
DIMM_LRDIMM	4
DIMM_CLAMSHELL	16

### ENUM\_MEM\_TYPE

Entry	Value
MEM_DDR4	1
MEM_QDRIV	5
MEM_NUMM	15

### ENUM\_MR\_INDEX

Entry	Value
DDR4_AC_ROM_MR0	0
DDR4_AC_ROM_MR1	1
DDR4_AC_ROM_MR2	2
DDR4_AC_ROM_MR3	3
DDR4_AC_ROM_MR4	4
DDR4_AC_ROM_MR5	5
DDR4_AC_ROM_MR6	6
QDRIV_AC_ROM_MR0	0
QDRIV_AC_ROM_MR1	1
QDRIV_AC_ROM_MR2	2

### ENUM\_ODT\_TABLE

Entry	Value
ODT_HIGH_ON_IDLE	(1 << 0)
ODT_HIGH_ON_READ	(1 << 1)
ODT_HIGH_ON_WRITE	(1 << 2)

**Table 178.**

ODT_TABLE_VALID_USE			
IDLE	READ	WRITE	Behavior
1	1	1	Always 1.
0	1	1	1 on both read and write.
0	0	1	1 on write only
0	1	0	1 on read only.
0	0	0	Always 0.

## 11.10.4. Debug Interface Structure

The topics in this section outline the structure of the debug interface.

These structures are populated during calibration and can be accessed afterwards to see the results of the calibration. If you change values in these structures the values in hardware do not change; to change values in hardware, you must change this information in the calibration bus instead. These structures are populated only if you have enabled debugging in this row, in the emif\_cal IP.

### 11.10.4.1. Debug Data Structures

This topic describes how to navigate the debug structures in the user-RAM.

#### **debug\_data\_struct (pt\_DEBUG\_DATA\_PTR)**

The debug\_data\_struct's base address is equal to: (user\_ram base address) + (offset for debug\_data\_struct, read from pt\_DEBUG\_DATA\_PTR)

If you have enabled debug in this row, this structure is populated and the pointer (pt\_DEBUG\_DATA\_PTR) is stored in the parameter table after calibration. This structure provides the following:

1. An interface by which to view the state of calibration, by checking the status bit in the debug\_data\_struct:
  - Bit 1 is set if calibration has started.
  - Bit 2 is set if calibration has finished.
2. An interface by which to send commands to the firmware (available commands are described by ENUM\_INTERFACE\_COMMANDS) and receive response codes about the command status (recognized status codes are described by ENUM\_DEBUG\_INTERFACE\_COMMAND\_STATUS\_CODES). Sending a command should follow this procedure:
  - a. Wait for command\_status to be TX\_STATUS\_CMD\_READY.
  - b. Set any relevant command\_parameters for the desired command (described in ENUM\_INTERFACE\_COMMANDS).
  - c. Set the requested\_command code (ENUM\_INTERFACE\_COMMANDS).

- d. Wait for command\_status to be TX\_STATUS\_RESPONSE\_READY.
- e. Set requested\_command code to CMD\_RESPONSE\_ACK to acknowledge the response was received.
- f. Any following command can be send once command\_status returns to the value TX\_STATUS\_CMD\_READY.
3. A pointer to mem\_summary\_report, which provides a more detailed status of calibration.
4. A pointer to mem\_cal\_report, which provides details of settings that were decided during calibration (such as delay settings, margins, vref settings, and so forth).

#### **Example 1: Steps to Request Full EMIF Recalibration**

1. Read from command\_status. Wait until the readdata=32'h0000\_0000 (TX\_STATUS\_CMD\_READY)
2. Write to command\_parameters[0] with wridata=Interface\_ID that you want to calibrate. (If you have only one interface, set the wridata=32'h0000\_0000.)
3. Write to command\_parameters[1] with wridata=32'h0000\_0003(DYNAMIC\_FULL\_RECAL). (Refer to *ENUM\_INIT\_MODE* table for other supported calibration options.)
4. Write to requested\_command with wridata=32'h0000\_0005(RUN\_MEM\_CALIBRATE).
5. Read from command\_status. Wait until the readdata=32'h0000\_0003 (TX\_STATUS\_RESPONSE\_READY).
6. Write to requested\_command with wridata=32'h0000\_0001(CMD\_RESPONSE\_ACK).

#### **Example 2: Set the Starting value of VREF\_OUT and Recalibrate EMIF using New Starting Value**

1. Read from command\_status. Wait till the readdata=32'h0000\_0000 (TX\_STATUS\_CMD\_READY).
2. Set the new VREF\_OUT. Refer to *debug\_cal\_data\_struct* for information on how to determine the Vref\_setting and Vref\_range.
  - a. Write to command\_parameters[0] with wridata=Vref\_setting.
  - b. Write to command\_parameters[1] with wridata=Vref\_range.
  - c. Write to requested\_command with wridata=32'h0000\_001B(SET\_VREF\_OUT).
  - d. Read from command\_status. Wait till the readdata=32'h0000\_0003 (TX\_STATUS\_RESPONSE\_READY).
  - e. Write to requested\_command with wridata=32'h0000\_0001(CMD\_RESPONSE\_ACK).
3. Request Recalibration. Follow steps in Example 1 **but set init\_mode to SKIP\_INIT\_VREF**.

#### **Example 3: Hardcode the VREF\_IN and VREF\_OUT and bypass the VREFCAL in next EMIF Recalibration**

1. Set the new VREF\_OUT. Refer to *debug\_cal\_data\_struct* for information on how to determine the Vref\_setting and Vref\_range.

- a. Read from command\_status. Wait till the readdata=32'h0000\_0000 (TX\_STATUS\_CMD\_READY).
  - b. Write to command\_parameters[0] with writedata=Vref\_setting.
  - c. Write to command\_parameters[1] with writedata=Vref\_range.
  - d. Write to requested\_command with writedata=32'h0000\_001B(SET\_VREF\_OUT).
  - e. Read from command\_status. Wait till the readdata=32'h0000\_0003 (TX\_STATUS\_RESPONSE\_READY).
  - f. Write to requested\_command with writedata=32'h0000\_0001(CMD\_RESPONSE\_ACK).
2. Set the new VREF\_IN. Refer to *debug\_cal\_data\_struct* for information on how to determine the Vref\_setting and Vref\_range.
    - a. Read from command\_status. Wait till the readdata=32'h0000\_0000 (TX\_STATUS\_CMD\_READY).
    - b. Write to command\_parameters[0] with writedata=Vref\_setting.
    - c. Write to requested\_command with writedata=32'h0000\_001A(SET\_VREF\_IN).
    - d. Read from command\_status. Wait till the readdata=32'h0000\_0003 (TX\_STATUS\_RESPONSE\_READY).
    - e. Write to requested\_command with writedata=32'h0000\_0001(CMD\_RESPONSE\_ACK).
  3. Set the calibration to skip VREF\_IN can VREF\_OUT.
    - a. Read from command\_status. Wait till the readdata=32'h0000\_0000 (TX\_STATUS\_CMD\_READY).
    - b. Write to command\_parameters[0] with writedata=32'h 0000 C000 (CALIB\_SKIP\_VREFIN\_CAL | CALIB\_SKIP\_VREFOUT\_CAL).
    - c. Write to requested\_command with writedata= 32'h 0000 001E (SET\_SKIP\_STEPS).
    - d. Read from command\_status. Wait till the readdata=32'h0000\_0003 (TX\_STATUS\_RESPONSE\_READY).
  4. Start EMIF calibration. Follow the same steps as in example 1, but set init\_mode to SKIP\_INIT\_VREF.

Parameter	Offset Within Structure	Size	Purpose
data_size	0	32	size of this structure (in bytes).
status	4	32	output: calibration status (ENUM_CAL_ERROR).
requested_command	8	32	input: a command that the user wishes for firmware to perform (ENUM_INTERFACE_COMMANDS).
command_status	12	32	output: status of the command that a user requested (ENUM_DEBUG_INTERFACE_COMMAND_STATUS_CODES).
command_parameters[0]	16	32	input: a parameter for a requested_command. The relevant parameters for each command are described in ENUM_DEBUG_INTERFACE_COMMANDS.

*continued...*

Parameter	Offset Within Structure	Size	Purpose
command_parameters[1]	20	32	input: a parameter for a requested_command. The relevant parameters for each command are described in ENUM_DEBUG_INTERFACE_COMMANDS.
command_parameters[2]	24	32	input: a parameter for a requested_command. The relevant parameters for each command are described in ENUM_DEBUG_INTERFACE_COMMANDS.
command_parameters[3]	28	32	input: a parameter for a requested_command. The relevant parameters for each command are described in ENUM_DEBUG_INTERFACE_COMMANDS.
mem_summary_report_pointer	32	32	output: pointer to the mem_summary_report structure.
mem_cal_report_pointer	36	32	output: pointer to the mem_cal_report structure.

### mem\_summary\_report (mem\_summary\_report\_pointer)

This structure provides details about the status of calibration.

Parameter	Offset Within Structure	Size	Comments
data_size	0	32	The size of this structure (in bytes).
report_flags	4	32	bit[0]: is 1 if report is ready, and all the registers below are valid. bits[23:1]: reserved for future use. bits[31:24]: version number of this report.
error_stage	12	32	The first stage at which calibration has failed (ENUM_CAL_STAGE).
error_group	16	32	Each bit corresponds to a dqs group. If a bit is set 1, then the corresponding group failed in the stage indicated by error_stage.
error_code	20	32	Detailed calibration status (ENUM_CAL_ERROR).
in_out_rate	72	8	bits [7:4] = out_rate = vco : mem_clk. bits [3:0] = in_rate = mem_clk : phy_clk ratio.
cur_interface_idx	32	32	ID of the interface to which the stored debug information applies — that is, the latest interface to have been calibrated.

### mem\_cal\_report (mem\_cal\_report\_pointer)

This structure provides details of settings observed during calibration (such as delay settings, margins, vref settings, etc.).

Parameter	Offset Within Structure	Number of Elements in Array (1 if not an array)	Size of Each Element in Array
data_size	0	1	32
debug_cal_data_struct_pointer__cal_data_dq_in	4	num_dq	32
debug_cal_data_struct_pointer__cal_data_dq_out	8	num_dq	32
debug_cal_data_struct_pointer__cal_data_dm_dbi_in	12	num_dm	32
<i>continued...</i>			

Parameter	Offset Within Structure	Number of Elements in Array (1 if not an array)	Size of Each Element in Array
debug_cal_data_struct_pointer__cal_data_dm_dbio_ut	16	num_dm	32
debug_cal_data_struct_pointer__cal_data_dqs_in	20	num_dqs_rd	32
debug_cal_data_struct_pointer__cal_data_dqs_en	24	num_dqs_rd	32
debug_cal_data_struct_pointer__cal_data_dqs_en_b	28	num_dqs_rd	32
debug_cal_data_struct_pointer__cal_data_dqs_out	32	num_dqs_wr	32
debug_cal_data_struct_pointer__vrefin	36	num_dqs_rd	32
debug_cal_data_struct_pointer__vrefout	40	num_dqs_wr	32
debug_cal_data_struct_pointer__cal_data_ca	44	num_ac_romEnums	32
debug_cal_data_struct_pointer__vfifo	52	num_dqs_rd	8
debug_cal_data_struct_pointer__lfifo	56	num_dqs_rd	8
debug_cal_data_struct_pointer__dcc_dq_in	60	num_dq	8
debug_cal_data_struct_pointer__dcc_dq_out	64	num_dq	8
debug_cal_data_struct_pointer__dcc_dm_dbio_in	68	num_dm	8
debug_cal_data_struct_pointer__dcc_dm_dbio_out	72	num_dm	8
debug_cal_data_struct_pointer__dcc_dqs_in	76	num_dqs_rd	8
debug_cal_data_struct_pointer__dcc_dqs_out	80	num_dqs_wr	8
debug_cal_data_struct_pointer__dcc_ca	84	num_ac_romEnums	8
debug_cal_data_struct_pointer__vrefout_all_ranks	88	num_dqs_wr	8
debug_cal_data_struct_pointer__ctle_out	92	num_dqs_wr	8
debug_cal_data_struct_pointer__ctle_in_dq	96	num_dq	8
debug_cal_data_struct_pointer__ctle_in_dqs	100	num_dqs_rd	8
write_lat	108	1	32
read_lat	112	1	32
rank_skew_data_out	116	1	32
rank_skew_dqsen	120	1	32
extra_rank_delay_any_to_read	124	1	32
extra_rank_delay_any_to_write	128	1	32

### debug\_cal\_data\_struct

This data structure is instantiated many times and is pointed to by each `debug_cal_data_struct_pointer_*` in the `mem_cal_report`. The `setting` field stores the chosen setting for the given parameter, while the `left_edge` and `right_edge` fields store the offset from the setting, for which transactions were found to still pass. To interpret the values stored, observe the following:

- For a timing parameter, all three fields are stored in taps.
- For a voltage parameter, the `setting` field is as follows:
  - Bits[15:8] = `vref_range`. This is set during IP parameterization.
    - Value 0 has a range of 60-92.5% of the VCCIO/VREFDQ.
    - Value 1 has range of 45-77.5% of the VCCIO/VREFDQ.
  - Bits[7:0] = `vref_setting` decided during calibration, as the number of incremental steps within the range (where each step is 0.65%).

For example, in DDR4, the  $V_{CCIO}$  voltage is 1.2V. So, given the setting = 0x0122:

`Vref_range=1, vref_setting=34`

Therefore, to calculate the  $V_{ref}$  value in volts:  $((34 \times 0.0065) + 0.45) \times 1.2V = 0.805V$

Parameter	Offset Within Structure	Size
setting	0	16
left_edge	2	8
right_edge	3	8

### 11.10.4.2. Debug Enums

This topic describes enums used in the debug data structures.

#### ENUM\_CAL\_ERROR

Entry	Value
SUCCESS	0
CALIBRATION_FAILED	1
MALLOC_ERROR	2
RIGHT_EDGE_NOT_FOUND	11
HARDWARE_TIMEOUT	12
REFRESH_MISSED	13
BAD_LOCK_SPEED	14
RANK_SKEW_TOO_LARGE	15

### **ENUM\_CAL\_INIT\_MODE**

<b>Entry</b>	<b>Value</b>	<b>Behavior if this is the Value of Field</b>
FULL_RECAL	3	Full re-calibration. Mostly same as power up calibration. Sequencer does not have to reset. Typical recalibration requests should use this option.
SKIP_INIT_VREF	6	No vref init.

### **ENUM\_CAL\_STAGE**

<b>Entry</b>	<b>Value</b>
NIL	0
CA_LEVEL	1
CA_DESKEW	2
DQS_EN	3
READ_DESKEW	4
WRITE_LEVEL	5
WRITE_DESKEW	6
LFIFO	7
CA_RANK_CENTER	8
VREF_IN	9
VREF_OUT	10
CTLE_IN	11
CTLE_OUT	12
CLEANUP	13

### **ENUM\_DEBUG\_INTERFACE\_COMMAND\_STATUS\_CODES**

<b>Entry</b>	<b>Value</b>	<b>Meaning</b>
TX_STATUS_CMD_READY	0	Interface ready to accept commands.
TX_STATUS_CMD_EXE	1	Response not ready as command is being executed.
TX_STATUS_ILLEGAL_CMD	2	Illegal command received.
TX_STATUS_RESPONSE_READY	3	Response ready.

### **ENUM\_DEBUG\_INTERFACE\_COMMANDS**

<b>Entry</b>	<b>Value</b>	<b>Command Description</b>
CMD_WAIT_CMD	1000	The wait command. When the debug data command is set to CMD_WAIT_CMD and the status is TX_STATUS_CMD_READY, the interface is ready to accept commands from user.
CMD_NOP	0	No operation command.
CMD_RESPONSE_ACK	1	Command response acknowledged.
RUN_MEM_CALIBRATE	5	Run memory calibration: command_parameters[0]: Interface ID; command_parameters[1]: Initialization mode.

*continued...*

Entry	Value	Command Description
MARK_ALL_RANKS_AS_VALID	17	Mark all ranks as being valid for calibration.
MARK_RANK_AS_SKIP	18	Mark a specific rank to be skipped for calibration: command_parameters[0]: Rank to skip.
SET_VREF_IN	26	Set the starting value of VREF IN for the next recalibration: command_parameters[0]: VREF setting.
SET_VREF_OUT	27	Set the starting value of VREF OUT for the next recalibration: command_parameters[0]: VREF setting. command_parameters[1]: VREF range.
SET_SKIP_STEPS	30	Set the calibration steps to skip: command_parameters[0]: flags indicating steps to skip (ENUM_DBG_CALIB_SKIP).

### 11.10.5. Example: Reading Calibration Results and Margins with the On-Chip Debug Port

This example provides instructions for reading calibration results and margins using the on-chip debug port.

The values in the example below are for illustrative purposes and are obtained from an EMIF example design using DDR4 RDIMM x72, implemented on the Intel Agilex 7 F-Series FPGA Development Kit.

1. Assume that you obtain the following data from the Global Parameter Table (GPT):

Address	Field Name	Value
0x500_0000	gpt_GLOBAL_PAR_VER	0x00000002
0x500_0004	gpt_NIOS_C_VER	0x00000001
0x500_0008	gpt_COLUMN_ID	0x00000001
0x500_000c	gpt_NUM_IOPACKS	0x00000010
0x500_0010	gpt_NIOS_CLK_FREQ_KHZ	0x0003D090
0x500_0014	gpt_PARAM_TABLE_SIZE	0x000001A0
0x500_0018	gpt_RESERVED	0x00000008
0x500_001c	gpt_GLOBAL_CAL_CONFIG	0x00000505
0x500_0020	gpt_SLAVE_CLK_DIVIDER	0x0000001C
0x500_0024	gpt_INTERFACE_PAR_PTRS_0	0x00000064
0x500_0028	gpt_INTERFACE_PAR_PTRS_1	0x00000000
0x500_002c	gpt_INTERFACE_PAR_PTRS_2	0x00000000
0x500_0030	gpt_INTERFACE_PAR_PTRS_3	0x00000000
0x500_0034	gpt_INTERFACE_PAR_PTRS_4	0x00000000
0x500_0038	gpt_INTERFACE_PAR_PTRS_5	0x00000000
0x500_003c	gpt_INTERFACE_PAR_PTRS_6	0x00000000
0x500_0040	gpt_INTERFACE_PAR_PTRS_7	0x00000000
0x500_0044	gpt_INTERFACE_PAR_PTRS_8	0x00000000
0x500_0048	gpt_INTERFACE_PAR_PTRS_9	0x00000000

*continued...*

Address	Field Name	Value
0x500_004c	gpt_INTERFACE_PAR_PTRS_10	0x00000000
0x500_0050	gpt_INTERFACE_PAR_PTRS_11	0x00000000
0x500_0054	gpt_INTERFACE_PAR_PTRS_12	0x00000000
0x500_0058	gpt_INTERFACE_PAR_PTRS_13	0x00000000
0x500_005c	gpt_INTERFACE_PAR_PTRS_14	0x00000000
0x500_0060	gpt_INTERFACE_PAR_PTRS_15	0x00000000

2. Determine the base address for the per-interface parameter table. There is only one EMIF interface in the I/O row, because only one gpt\_INTERFACE\_PAR\_PTRS has a non-zero value.

The base address for the per-interface parameter table = 0x500\_0000 + 0x64  
= 0x500\_0064.

3. Read the per-interface parameter table starting from its base address of 0x500\_0064. The content of the per-interface parameter table is shown below.

Address	Field Name				Value		
	Bit [31:24]	Bit [23:16]	Bit [15:8]	Bit [7:0]			
0x500_0064	pt_INTERFACE_PAR_VER		pt_IP_VER		0x00024C40		
0x500_0068	pt_UNUSED		pt_DEBUG_DATA_PTR		0x000001A0		
0x500_006C	pt_RESERVED	pt_CONTROLLER_TYPE	pt_DIMM_TYPE	pt_MEMORY_TYPE	0x00000201		
0x500_0070	pt_AFI_CLK_FREQ_KHZ				0x000411AC		
0x500_0074	pt_NUM_RANKS	pt_WRITE_LATENCY	pt_READ_LATENCY	pt_BURST_LEN	0x01121708		
0x500_0078	pt_NUM_DQ	pt_NUM_DQS_RD	pt_NUM_DQS_WR	pt_NUM_DIMMS	0x48090901		
0x500_007C	pt_CS_WIDTH	pt_BANK_WIDTH	pt_ADDR_WIDTH	pt_NUM_DM	0x01021109		
0x500_0080	pt_BANK_GROUP_WIDTH	pt_C_WIDTH	pt_ODT_WIDTH	pt_CKE_WIDTH	0x02000101		
0x500_0084	pt_NUM_LRDIMM_CFG	pt_CAL_DATA_SIZE	pt_CK_WIDTH	pt_ADDR_MIRROR	0x05140100		
0x500_0088	pt_NUM_DATA_LANES	pt_NUM_CA_LANES	pt_NUM_CENTERS	pt_NUM_AC_ROM_ENUMS	0x09040434		
0x500_008C	pt_ODT_TABLE_LO				0x00000004		
0x500_0090	pt_ODT_TABLE_HI				0x00000000		
0x500_0094	pt_RESERVED				0x0D00C081		
0x500_0098	pt_CAL_DATA_PTR		pt_RESERVED		0x00B00000		
0x500_009C	pt_DBG_SKIP_RANKS				0x00000000		
0x500_00A0	pt_DBG_SKIP_GROUPS				0x00000000		

*continued...*

Address	Field Name				Value
	Bit [31:24]	Bit [23:16]	Bit [15:8]	Bit [7:0]	
0x500_00A4	pt_DBG_SKIP_STEPS				0x00670000
0x500_00A8	pt_TILE_ID_PTR		pt_NUM_DIMM	pt_NUM_MR	0x00C4070C
0x500_00AC	pt_MR_PTR		pt_PIN_ADDR_PTR		0x017000D8

4. Determine the base address for debug\_data\_struct.

```
Base address for debug_data_struct = 0x500_0000 + pt_DEBUG_DATA_PTR
= 0x500_0000 + 0x1A0
= 0x500_01A0
```

5. Read the debug\_data\_struct starting from its base address of 0x500\_01A0. Below is the content for the debug\_data\_struct:

Address	Parameter	Value
0x500_01A0	data_size	0x00000000
0x500_01A4	status	0x00000006
0x500_01A8	requested_command	0x000003E8
0x500_01AC	command_status	0x00000000
0x500_01B0	command_parameters[0]	0x00000000
0x500_01B4	command_parameters[1]	0x00000000
0x500_01B8	command_parameters[2]	0x00000000
0x500_01BC	command_parameters[3]	0x00000000
0x500_01C0	mem_summary_report_pointer	0x05000354
0x500_01C4	mem_cal_report_pointer	0x050003A0

6. Read the memory\_summary\_report from the address starting at 0x500\_0354.

- Read from report\_flags (address = 0x500\_0358). Ensure the LSB of the readdata is 1'b1 (indicating that the calibration report is ready), before reading the other members in the *memory\_summary\_report* structure or the *mem\_cal\_report* structure.
- In this example, *error\_stage*, *error\_group* and *error\_code* are all 0, because the calibration is successful.

Address	Parameter	Value
0x5000354	data_size	0x00000013
0x5000358	report_flags	0x01000001
0x5000360	error_stage	0x00000000
0x5000364	error_group	0x00000000
0x5000368	error_code	0x00000000
0x500039C	in_out_rate	0x00000014

7. Read the mem\_cal\_report from the address beginning at 0x500\_03A0.

Address	Parameter	Value
0x500_03A0	data_size	0x00000021
0x500_03A4	debug_cal_data_struct_pointer__cal_data_dq_in	0x05000424
0x500_03A8	debug_cal_data_struct_pointer__cal_data_dq_out	0x05000544
0x500_03AC	debug_cal_data_struct_pointer__cal_data_dm_dbi_in	0x05000664
0x500_03B0	debug_cal_data_struct_pointer__cal_data_dm_dbi_out	0x05000688
0x500_03B4	debug_cal_data_struct_pointer__cal_data_dqs_in	0x050006AC
0x500_03B8	debug_cal_data_struct_pointer__cal_data_dqs_en	0x050006D0
0x500_03BC	debug_cal_data_struct_pointer__cal_data_dqs_en_b	0x050006F4
0x500_03C0	debug_cal_data_struct_pointer__cal_data_dqs_out	0x05000718
0x500_03C4	debug_cal_data_struct_pointer__vrefin	0x0500073C
0x500_03C8	debug_cal_data_struct_pointer__vrefout	0x05000760
0x500_03CC	debug_cal_data_struct_pointer__cal_data_ca	0x05000784
0x500_03D4	debug_cal_data_struct_pointer__vfifo*	0x05000878
0x500_03D8	debug_cal_data_struct_pointer__lfifo*	0x05000884
0x500_03DC	debug_cal_data_struct_pointer__dcc_dq_in*	0x05000890
0x500_03E0	debug_cal_data_struct_pointer__dcc_dq_out*	0x050008D8
0x500_03E4	debug_cal_data_struct_pointer__dcc_dm_dbi_in*	0x05000920
0x500_03E8	debug_cal_data_struct_pointer__dcc_dm_dbi_out*	0x0500092C
0x500_03EC	debug_cal_data_struct_pointer__dcc_dqs_in*	0x05000938
0x500_03F0	debug_cal_data_struct_pointer__dcc_dqs_out*	0x05000944
0x500_03F4	debug_cal_data_struct_pointer__dcc_ca*	0x05000950
0x500_03F8	debug_cal_data_struct_pointer__vrefout_all_ranks*	0x05000984
0x500_03FC	debug_cal_data_struct_pointer__ctle_out*	0x05000990
0x500_0400	debug_cal_data_struct_pointer__ctle_in_dq*	0x0500099C
0x500_0404	debug_cal_data_struct_pointer__ctle_in_dqs*	0x050009E4
0x500_040C	write_lat	0x00000005
0x500_0410	read_lat	0x0000000D
0x500_0414	rank_skew_data_out	0x00000000
0x500_0418	rank_skew_dqsen	0x00000000
0x500_041C	extra_rank_delay_any_to_read	0x00000000
0x500_0420	extra_rank_delay_any_to_write	0x00000000

Note: \* Each address stores the setting for 4 pins or 4 groups.

8. Read the calibration result for DQ input setting (read path) starting from address 0x0500\_0424.

- The data at address 0x0500\_0424 has the calibration result for DQ[0].
- The data at address 0x0500\_0428 has the calibration result for DQ[1].
- The data at address 0x0500\_042C has the calibration result for DQ[2], and so forth.

For data at each address:

- Bit[15:0] corresponds to the calibrated setting.
- Bit[23:16] corresponds to the left edge margin.
- Bit[31:24] corresponds to the right edge margin.

9. Repeat step 8 for other pins.
10. Read the Vrefin setting starting from address 0x0500\_073C.
  - The data at address 0x0500\_073C has the calibration result for group 0.
  - The data at address 0x0500\_0740 has the calibration result for group 1, and so forth.

For Vrefin/Vrefout calibration setting:

- Bit[15:8] corresponds to Vref\_range:
  - Vref\_range = 0 means range of 60-92.5% of VCCIO/VREFDQ.
  - Vref\_range = 1 means range of 45-77.5% of VCCIO/VREFDQ.
- Bit[7:0] corresponds to Vref\_setting where each step is 0.65%.

For example, in DDR4, VCCIO/VREFDQ is 1.2V. Assuming you obtain 0x122:

- Vref\_range = 0x1
- Vref\_setting = 0x22 = 34 (decimal)

The Vref value in volts = ((34 \* 0.0065) + 0.45) x 1.2V = 0.805V.

11. Read the mem\_summary\_report starting from address 0x05000354.
  - The data at 0x5000360 indicates the stage at which calibration first failed. (ENUM\_CAL\_STAGE).
  - The data at 0x5000364 contains per-group status. Each bit corresponds to a dqs group. If a bit is set 1, then the corresponding group failed in the stage indicated by error\_stage.
  - The data at 0x5000368 contains detailed calibration status (ENUM\_CAL\_ERROR).
12. For a calibration setting that is stored as 8-bit data (for example, vfifo, lfifo, dcc\*, ctle\*), each address stores the setting for 4 pins or 4 groups. If you want to read the vfifo setting for this interface implemented in 9 DQS groups, reading the data from address 0x500\_0878, 0x500\_087C and 0x500\_0880 is adequate.
  - Bit [ 7:0 ] @ address 0x500\_0878 = vfifo setting for group 0
  - Bit [15:8] @ address 0x500\_0878 = vfifo setting for group 1
  - Bit [23:16] @ address 0x500\_0878 = vfifo setting for group 2
  - Bit [31:24] @ address 0x500\_0878 = vfifo setting for group 3
  - Bit [ 7:0 ] @ address 0x500\_087C = vfifo setting for group 4
  - Bit [15:8] @ address 0x500\_087C = vfifo setting for group 5

- Bit [23:16] @ address 0x500\_087C = vfifo setting for group 6
  - Bit [31:24] @ address 0x500\_087C = vfifo setting for group 7
  - Bit [ 7:0] @ address 0x500\_0880 = vfifo setting for group 9
13. All the debug\_cal\_data is initialized with an all-1 value. Getting an all-1 value for given calibration data (such as dcc-related calibration) means that the value is not overwritten after calibration and you can ignore it.

## 11.11. Efficiency Monitor

You can instantiate an 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.

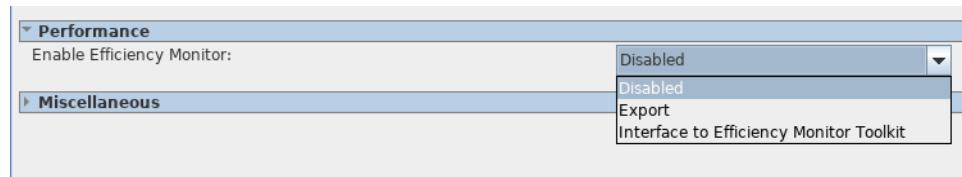
### 11.11.1. Enabling the Efficiency Monitor in a Design Example

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 the Unified Toolkit graphical user interface (in the System Console). Refer to *Opening the Efficiency Monitor Toolkit* for more information.

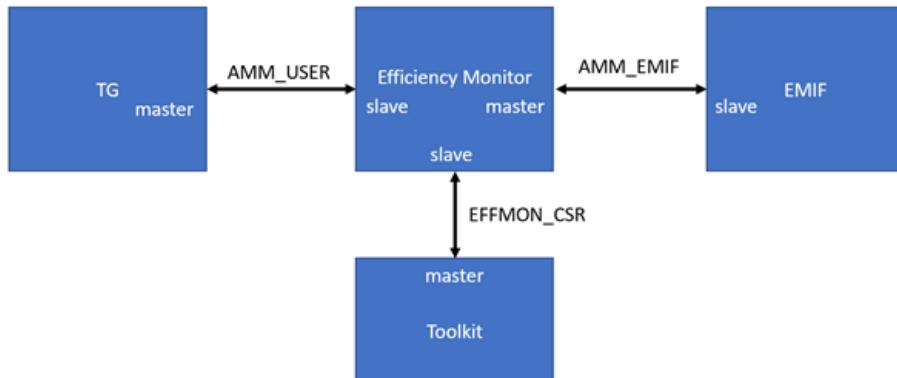
**Figure 257. Efficiency Monitor Mode Setting**



### 11.11.2. 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 258. 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.

### 11.11.3. 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 179. Control and Status Registers

Symbol Address	Register Name	Readable or Writeable	Register Description
0x0	EFFMON_START	Readable and Writeable	<ul style="list-style-type: none"><li>Write a value of 1 to enable the Efficiency Monitor.</li><li>Write a value of 0 to disable the Efficiency Monitor.</li></ul>
0x4	EFFMON_READ_COUNTER	Readable	Number of read commands issued.
0x8	EFFMON_WRITE_COUNTER	Readable	Number of write commands issued.

*continued...*

<b>Symbol Address</b>	<b>Register Name</b>	<b>Readable or Writeable</b>	<b>Register Description</b>
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	<ul style="list-style-type: none"> <li>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.</li> <li>A value of 0 indicates the counter has not saturated.</li> </ul>
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_WRIDDLE_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.

### Related Information

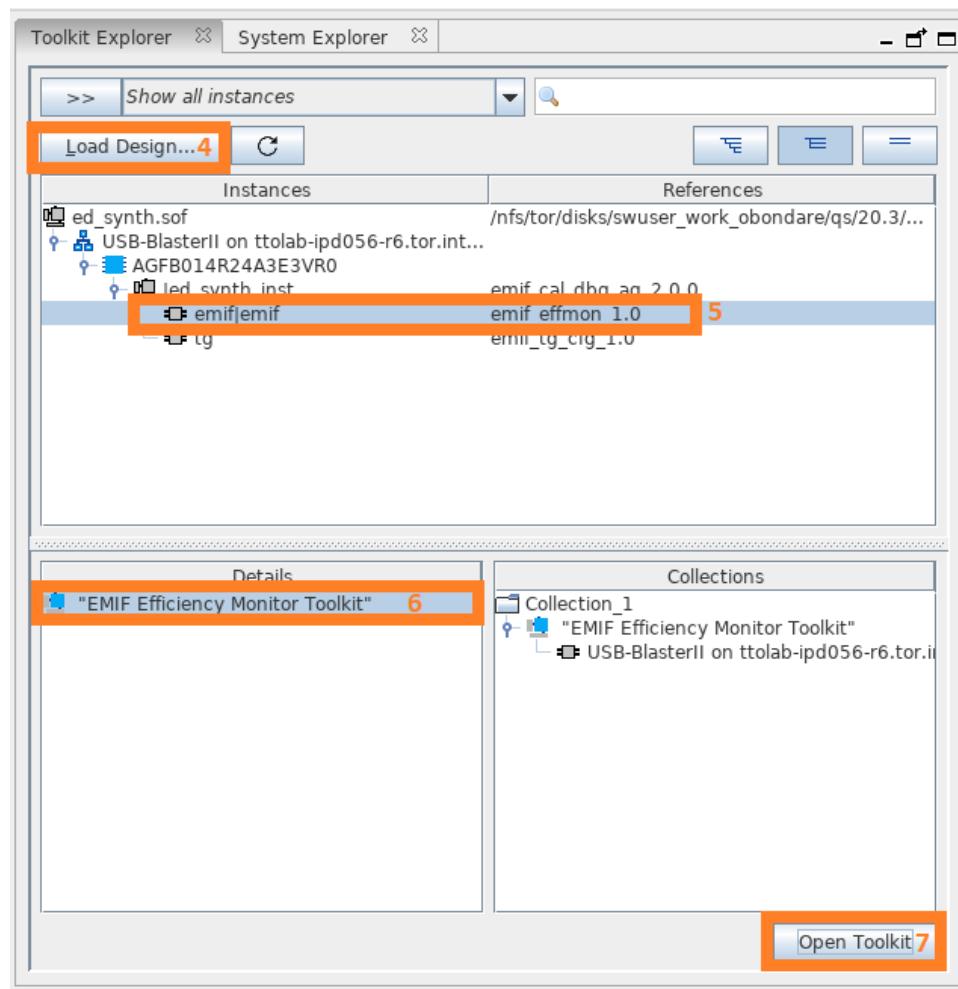
Register Map IP-XACT Support for Intel Agilex 7 F-Series and I-Series EMIF DDR4 IP on page 148

#### 11.11.4. Opening the Efficiency Monitor Toolkit

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.

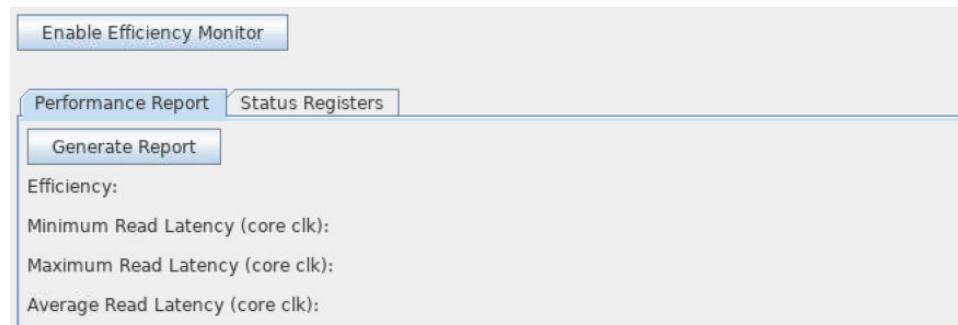
##### Connecting the Efficiency Monitor

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 259. Connecting the Efficiency Monitor**


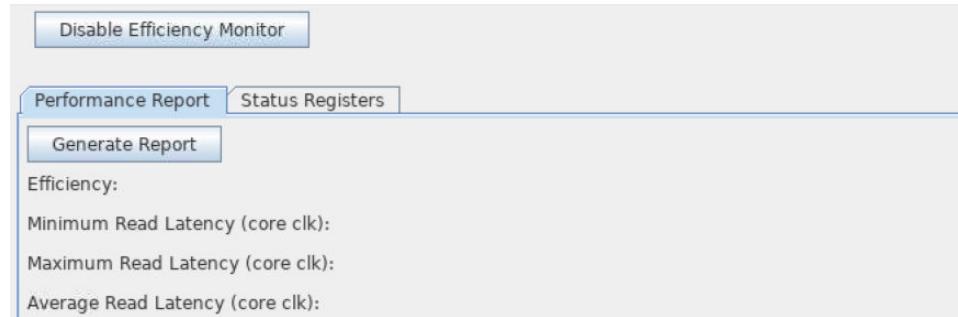
### Starting and Stopping the Efficiency Monitor

To start collecting information on the Avalon interface, click **Enable Efficiency Monitor**.

**Figure 260. Enable Efficiency Monitor**


To stop the Efficiency Monitor, click **Disable Efficiency Monitor**.

**Figure 261. Disable Efficiency Monitor**

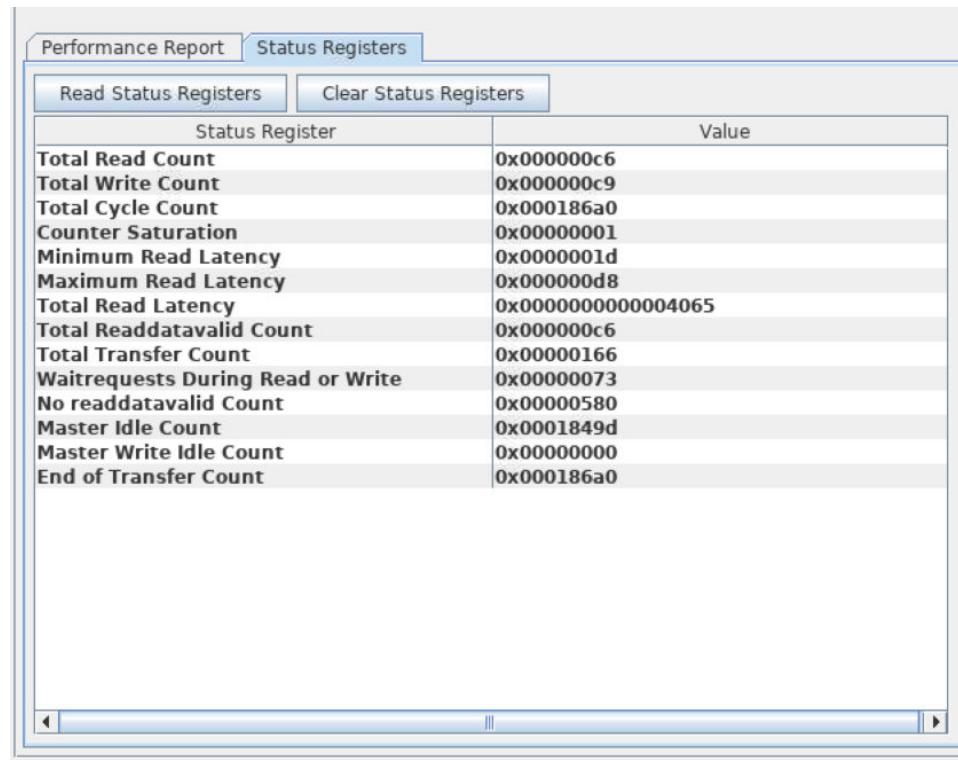


### Status Registers

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 262. Status Registers**

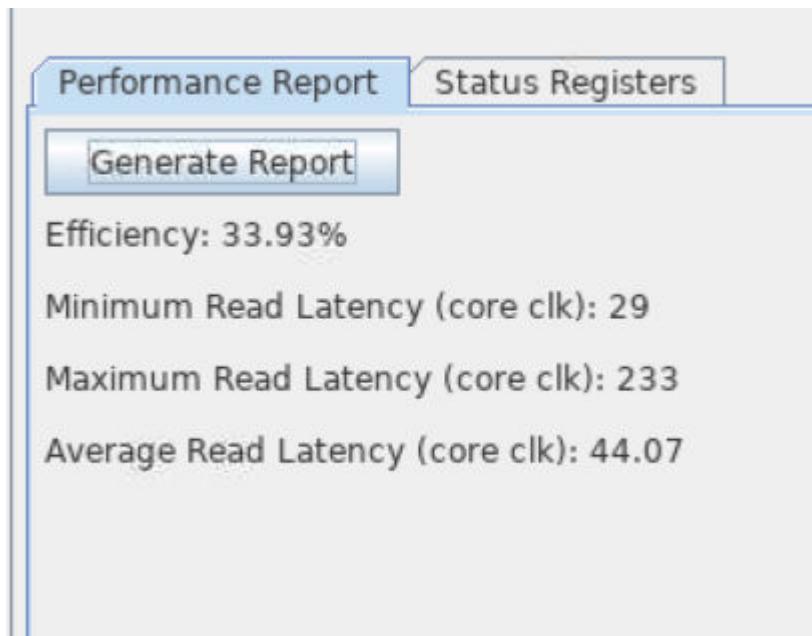


The screenshot shows a software window titled 'Status Registers'. At the top left is a small button labeled 'Performance Report' (disabled). Below it is a horizontal tab bar with two tabs: 'Performance Report' (disabled) and 'Status Registers' (selected). Underneath the tabs is a small button labeled 'Clear Status Registers'. The main area is a table with two columns: 'Status Register' and 'Value'. The table lists various status registers and their corresponding hex values:

Status Register	Value
Total Read Count	0x000000c6
Total Write Count	0x000000c9
Total Cycle Count	0x000186a0
Counter Saturation	0x00000001
Minimum Read Latency	0x0000001d
Maximum Read Latency	0x000000d8
Total Read Latency	0x0000000000004065
Total Readdatavalid Count	0x000000c6
Total Transfer Count	0x00000166
Waitrequests During Read or Write	0x00000073
No readdatavalid Count	0x00000580
Master Idle Count	0x0001849d
Master Write Idle Count	0x00000000
End of Transfer Count	0x000186a0

### Performance Report

The **Performance Report** tab displays efficiency as a percentage, and the read latency report.

**Figure 263. Performance Report Tab**

The reported values are calculated using values in the status registers, as follows:

- Efficiency =  $(\text{EFFMON\_TRANSFER\_COUNTER} \div \text{EFFMON\_END\_OF\_TRANS\_COUNTER}) \times 100\%$
- Minimum Read Latency = `EFFMON_RDLAT_MIN`
- Maximum Read Latency = `EFFMON_RDLAT_MAX`
- Average Read Latency =  $\text{EFFMON\_RDLAT\_TOTAL} \div \text{EFFMON\_READDATAVALID\_COUNTER}$

## 12. External Memory Interfaces Intel Agilex 7 F-Series and I-Series 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/683216/>. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

## 13. Document Revision History for External Memory Interfaces Intel Agilex 7 F-Series and I-Series FPGA IP User Guide

Document version	Intel Quartus Prime Version	IP Version	Changes
2023.06.26	23.2	2.7.1	<ul style="list-style-type: none"> <li>• In the <i>Architecture</i> chapter: <ul style="list-style-type: none"> <li>— Added new figures 44 and 45 to the <i>I/O Banks</i> topic.</li> <li>— In the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic, added one new row to both the <i>DDR x72 EMIF With AVST and Address/Command Scheme with 4 I/O Lanes</i> and <i>DDR x72 EMIF With AVST and Address/Command Scheme with 3 I/O Lanes</i> tables.</li> </ul> </li> <li>• In the <i>Debugging</i> chapter: <ul style="list-style-type: none"> <li>— In the <i>Rerunning the Traffic Generator</i> topic, implemented a minor terminology change in the note.</li> <li>— In the <i>Guidelines for Debugging Calibration Issues</i> topic, inserted a new guideline 5.</li> <li>— In the <i>Using the Default Traffic Generator</i> topic, added an additional figure and additional text.</li> <li>— In the <i>Data Pattern and Byte Enable</i> topic, expanded the text introducing the table.</li> </ul> </li> </ul>
2023.04.03	23.1	2.7.0	<ul style="list-style-type: none"> <li>• Updated product family name to <i>Intel Agilex 7</i>.</li> <li>• In the <i>Architecture</i> chapter: <ul style="list-style-type: none"> <li>— Added new figures 42 and 43 to the <i>I/O Bank</i> topic.</li> <li>— In the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic, added two rows to the <i>DDR x72 EMIF With AVST and Address/Command Scheme with 4 I/O Lanes</i> and <i>DDR x72 EMIF With AVST and Address/Command Scheme with 3 I/O Lanes</i> tables.</li> <li>— In the <i>QDR-IV Read Calibration</i> topic, corrected an error in the <i>Vref-in Calibration</i> description.</li> </ul> </li> </ul>

**continued...**



Document version	Intel Quartus Prime Version	IP Version	Changes
2022.12.19	22.4	2.6.1	<ul style="list-style-type: none"> <li>In the <i>Architecture</i> chapter:           <ul style="list-style-type: none"> <li>Added sub-bank ordering figures in the <i>I/O Bank</i> topic.</li> <li>Added rows to the tables in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic. Also added <i>Restriction for using JB26, JH26, JP26 and JL27 on AGI041, Package R29D</i> subsection.</li> </ul> </li> <li>In the <i>DDR4</i> chapter, modified the <i>PLL reference clock jitter</i> parameter description in the <i>DDR4 Parameters: General</i> topic.</li> <li>In the <i>QDR-IV</i> chapter, modified the <i>PLL reference clock jitter</i> parameter description in the <i>QDR-IV Parameters: General</i> topic.</li> </ul>
2022.11.03	22.3	2.6.1	<ul style="list-style-type: none"> <li>In the <i>Debugging</i> chapter, modified the introduction to the <i>Prerequisites for Using the EMIF Debug Toolkit</i> topic.</li> </ul>
2022.09.26	22.3	2.6.1	<ul style="list-style-type: none"> <li>In the <i>Architecture</i> chapter:           <ul style="list-style-type: none"> <li>Added sub-bank ordering figures in the <i>I/O Bank</i> topic.</li> <li>Added a note about interface widths for DDR4 hard PHY-only configurations, to the <i>I/O Sub-Bank Usage</i> section of the <i>I/O Bank</i> topic.</li> <li>Added rows to the tables in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> <li>Changed two instances of <i>bank</i> to <i>sub-bank</i> in the <i>PLL Reference Clock Networks</i> topic.</li> </ul> </li> <li>In the <i>Debugging</i> chapter, added the <i>Claiming/Releasing the TG Config Interface</i> topic.</li> </ul>
2022.06.20	22.2	2.6.1	<ul style="list-style-type: none"> <li>In the <i>Architecture</i> chapter:           <ul style="list-style-type: none"> <li>Added sub-bank ordering figures in the <i>I/O Bank</i> topic.</li> <li>Added rows to the tables in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> </ul> </li> <li>In the <i>Simulation</i> chapter, added the <i>Simulating the Design Example with Mentor Graphics® AXI4 Master BFM (Intel FPGA Edition)</i> section.</li> <li>In the <i>DDR4</i> chapter, modified the tables in the <i>Skew Matching Guidelines for DDR4 DIMM Configurations</i> and <i>Skew Matching Guidelines for DDR4 Discrete Configurations</i> topics.</li> <li>In the <i>QDR-IV</i> chapter, modified the table in the <i>Skew Matching Guidelines for QDR-IV Configurations</i> topic.</li> </ul>

**continued...**

Document version	Intel Quartus Prime Version	IP Version	Changes
2022.03.28	22.1	2.6.1	<ul style="list-style-type: none"><li>• In the <i>Architecture</i> chapter:<ul style="list-style-type: none"><li>— Inserted a new step 1 in the <i>Guidelines for Debugging Calibration Issues</i> topic.</li><li>— Added <i>Using a Custom Controller with the Hard PHY</i> topic.</li></ul></li><li>• In the <i>Simulating Memory IP</i> chapter, added an <i>Abstract I/O SSM</i> section to the <i>Simulation Options</i> topic.</li><li>• In the <i>Board Design Guidelines</i> section of the <i>DDR4</i> chapter, modified the figure and the second bullet point in the <i>Intel Agilex 7 EMIF -Specific Routing Guidelines for Various DDR4 Topologies</i> topic.</li></ul>

*continued...*

Document version	Intel Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> <li>• In the <i>Board Design Guidelines</i> section of the <i>QDR-IV</i> chapter:           <ul style="list-style-type: none"> <li>— Added an additional figure and associated text to the <i>Reference Stackup</i> topic.</li> <li>— Added a reference to related information following the table in the <i>QDR-IV Single Device Memory Topology</i> topic.</li> <li>— Modified one row and added one row to the table in the <i>Skew Matching Guidelines for QDR-IV Configurations</i> topic.</li> </ul> </li> <li>• In the <i>Intel Agilex 7 EMIF IP - I/O Timing Closure</i> chapter, added the <i>Understanding the *_ip_parameters.dat File and Making a Mask Polygon</i> topic.</li> <li>• In the <i>Debugging</i> chapter:           <ul style="list-style-type: none"> <li>— Added the <i>Memory Timing Parameter Evaluation and Verify that the Board Has the Correct Memory Component or DIMM Installed</i> topics to the <i>Categorizing Hardware Issues</i> section.</li> <li>— Inserted a new step 1 in the <i>Guidelines for Debugging Calibration Issues</i> topic.</li> </ul> </li> </ul>
2022.01.31	21.4	2.6.0	<ul style="list-style-type: none"> <li>• In the <i>Using the Configurable Traffic Generator (TG2)</i> section of the <i>Debugging</i> chapter:           <ul style="list-style-type: none"> <li>— Modified the table in the <i>Configuration and Status Registers</i> topic.</li> <li>— Modified the last dialog box image and description in the <i>Address Pattern Examples - Basic Mode</i> topic.</li> </ul> </li> </ul>
2021.12.13	21.4	2.6.0	<ul style="list-style-type: none"> <li>• In the <i>Product Architecture</i> chapter:           <ul style="list-style-type: none"> <li>— Added <i>PHY-Only Mode</i> to the <i>Introduction</i> topic.</li> <li>— In the <i>I/O Banks</i> topic, modified the first sentence of the short paragraph preceding the <i>Zipper Block</i> section, and the last sentence of the first paragraph in the <i>I/O Sub-Bank Usage</i> section.</li> <li>— Removed a note from the <i>User-requested Reset in Intel Agilex 7 EMIF IP</i> topic.</li> </ul> </li> <li>• In the <i>End-User Signals</i> chapter, added the <i>AFI Signals</i> and <i>AFI 4.0 Timing Diagrams</i> sections.</li> <li>• In the <i>DDR4</i> chapter:           <ul style="list-style-type: none"> <li>— In the <i>Intel Agilex EMIF IP DDR4 Parameters: Memory</i> topic, added notes about custom DIMM support to the <i>Memory format</i> parameter description in the <i>Group: Memory / Topology</i> table, and added description of the <i>Fine granularity refresh</i> parameter to the <i>Mode Register Settings</i> table.</li> <li>— In the <i>Single Rank x8 Discrete (Component) Topology</i> and <i>Single Rank x16 Discrete (Component) Topology</i> topics in the <i>Board Design Guidelines</i> section, added mention of the <i>alert_n</i> pin requiring an external pull-up resistor of approximately 10KΩ.</li> <li>— In the <i>Intel Agilex 7 EMIF Pin Swapping Guidelines</i> topic, expanded the note discussing the pin swapping rules enforced by the Intel Quartus Prime software.</li> </ul> </li> <li>• In the <i>QDR-IV</i> chapter, removed the <i>Core Clock Network</i> section from the <i>Resource Sharing Guidelines (Multiple Interfaces)</i> topic.</li> </ul>

Document Version	Intel Quartus Prime Version	IP Version	Changes
2021.10.04	21.3	2.5.0	<ul style="list-style-type: none"> <li>• In the <i>Architecture</i> chapter:           <ul style="list-style-type: none"> <li>— Modified captions on existing figures in the <i>I/O Bank</i> topic, and added four additional figures.</li> <li>— Modified the two tables in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> </ul> </li> <li>• In the <i>Simulating</i> chapter, changed <i>Mentor Graphics</i> to <i>Siemens EDA</i>, and <i>ModelSim - Intel FPGA Edition</i> to <i>Questa - Intel FPGA Edition</i>.</li> <li>• In the <i>DDR4</i> chapter:           <ul style="list-style-type: none"> <li>— Modified the Example Designs with Multi-IPs table in the <i>Intel Agilex EMIF IP DDR4 Parameters: Example Designs</i> topic.</li> <li>— Added <i>Register Map IP-XACT Support for Intel Agilex EMIF DDR4 IP</i> topic.</li> <li>— Added a note to step 9 in the <i>General Guidelines</i> topic in the <i>Pin Guidelines</i> section.</li> <li>— In the <i>Skew Matching Guidelines for DIMM Configurations</i> topic, added two rows to the table.</li> <li>— In the <i>Skew Matching Guidelines for DDR4 Discrete Configurations</i> topic, added two rows to the table.</li> </ul> </li> <li>• In the <i>Controller Optimization</i> chapter, added the <i>Controller Pre-pay and Post-pay Refresh (DDR4 Only)</i> topic.</li> <li>• In the <i>Debugging</i> chapter:           <ul style="list-style-type: none"> <li>— Updated the table in the <i>Configuration and Status Registers</i> topic.</li> <li>— Updated the <i>Address Pattern</i> topic.</li> <li>— Updated the <i>Address Pattern</i> topic and added several new topics:               <ul style="list-style-type: none"> <li>• <i>Address Generator Modes</i></li> <li>• <i>Address Generator MSB Indices</i></li> <li>• <i>Address Generator Effective Width</i></li> <li>• <i>Address Generator Relative Frequencies</i></li> <li>• <i>Address Pattern Examples - Basic Mode</i></li> <li>• <i>Address Pattern Examples - Advanced Mode</i></li> </ul> </li> <li>— Updated the table in the <i>Traffic Generator Status</i> topic.</li> <li>— In the <i>Configuring the Traffic Generator</i> topic, updated the <i>Configurations Tab</i> figure, and added several new figures.</li> </ul> </li> </ul>
2021.07.09	21.2	2.4.2	In the <i>Debugging</i> chapter, modified the <i>Code Value</i> column in the <i>Error Codes</i> table in the <i>Traffic Generator Status</i> topic.

**continued...**



Document Version	Intel Quartus Prime Version	IP Version	Changes
2021.06.21	21.2	2.4.2	<ul style="list-style-type: none"> <li>• In the <i>Architecture</i> chapter, <ul style="list-style-type: none"> <li>— Added the <i>Calibration Algorithms</i> section.</li> <li>— Added information to the <i>ECC Support</i> description in the table in the <i>Hard Memory Controller Features</i> topic.</li> <li>— Added information to the <i>ECC Controller</i> description in the table in the <i>Hard Memory Controller Main Control Path</i> topic.</li> </ul> </li> <li>• In the <i>Simulation</i> chapter, added information to the <i>Skip Calibration Mode</i> description in the <i>Calibration Modes</i> topic.</li> <li>• In the <i>Debugging</i> chapter: <ul style="list-style-type: none"> <li>— Added <i>Guidelines for Debugging Calibration Issues</i> to the <i>Using the EMIF Debug Toolkit</i> section.</li> <li>— Added descriptions of the <code>TG_VERSION</code> and <code>TG_START</code> registers to the table in the <i>Configuration and Status Registers</i> topic.</li> <li>— Modified a sentence in the <i>User Traffic</i> section of the <i>Starting Traffic with the Traffic Generator</i> topic.</li> <li>— Added the <i>Examples of Configuring the TG2 Traffic Generator</i> topic.</li> </ul> </li> </ul>
2021.03.29	21.1	2.4.0	<ul style="list-style-type: none"> <li>• In the <i>End-User Signals</i> chapter, removed references to <i>Ping-Pong PHY</i> from the port descriptions in the <i>Interface: ctrl_ecc_status</i> table.</li> <li>• In the <i>Product Architecture</i> chapter: <ul style="list-style-type: none"> <li>— In the <i>I/O Bank</i> topic, added I/O chaining diagrams for R29A, R31B, and R31C package devices.</li> <li>— In the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic, added three rows to the <i>DDR4x72 EMIF With AVST and Address/Command Scheme with 4 I/O Lanes</i> table.</li> </ul> </li> <li>• In the <i>DDR4</i> chapter, removed a redundant paragraph from the <i>Data, Data Strobes, DM/DBI, and Optional ECC Signals</i> topic.</li> <li>• In the <i>Debugging</i> chapter: <ul style="list-style-type: none"> <li>— In <i>Table 154. Configuration and Status Registers</i>, modified the <code>TG_CLEAR</code> register description, and added three additional rows at the bottom of the table.</li> <li>— In <i>Table 156. Error Codes</i>, added an additional row to the bottom of the table, for the <code>ERR_BURSTLENGTH_OVERFLOW_ON_FIRST_WRITE</code> code name.</li> <li>— In the <i>Traffic Generator Status Report</i> topic: <ul style="list-style-type: none"> <li>• Replaced the figure <i>TG Status Report (Passing Traffic Pattern)</i>.</li> <li>• Replaced the figure <i>TG Status Report (Failing Traffic Pattern)</i>.</li> <li>• Replaced the figure <i>TG Status Report (While Running Infinite Traffic)</i>.</li> <li>• Added the figure <i>TG Status Report (Attempt to Overflow Address Space)</i>.</li> </ul> </li> <li>— In the <i>Control and Status Registers</i> table, changed the register name for the <code>0x40</code> entry.</li> </ul> </li> </ul>

**continued...**

Document Version	Intel Quartus Prime Version	IP Version	Changes
2021.01.20	20.4	2.3.0	<ul style="list-style-type: none"> <li>In the <i>DDR4</i> chapter, added the <i>Group: Example Designs / Example Design with Multi-IPs</i> table to the <i>Intel Agilex EMIF IP DDR4 Parameters: Example Designs</i> topic.</li> </ul>
2020.12.14	20.4	2.3.0	<ul style="list-style-type: none"> <li>In the <i>Product Architecture</i> chapter, recast the text around the table in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> <li>In the <i>End-User Signals</i> chapter, removed the <i>AFI Signals</i> and <i>AFI 4.0 Timing Diagrams</i> sections.</li> <li>In the <i>DDR4</i> chapter: <ul style="list-style-type: none"> <li>Modified the <i>x4 DIMM Implementation</i> topic.</li> <li>Modified the <i>ADDR/CMD Reference Voltage/RESET Signal Routing Guidelines for Single Rank x 8 and R Rank x 16 Discrete (Component) Topologies</i> topic.</li> </ul> </li> <li>In the <i>Debugging</i> chapter: <ul style="list-style-type: none"> <li>Replaced the <i>Adding Interfaces to a Design Example</i> topic with <i>Creating a Design Example with Multiple EMIF Interfaces</i>.</li> <li>Replaced the <i>Using the Traffic Generator with the Generated Design Example</i> topic with the <i>Using the Default Traffic Generator</i> section.</li> <li>Retitled the <i>Configurable Traffic Generator (TG2)</i> section to <i>Using the Configurable Traffic Generator (TG2)</i>.</li> </ul> </li> </ul>
2020.10.05	20.3	2.3.0	<ul style="list-style-type: none"> <li>In the <i>About</i> chapter, updated the <i>Release Information</i> topic.</li> <li>In the <i>Product Architecture</i> chapter, added device packages to the <i>I/O Bank</i> topic, and expanded the table in the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> <li>In the <i>MMR Tables</i> section of the <i>End-User Signals</i> chapter, added ECC error information to the <i>ecc6: Address of Most Recent Correction Command Dropped</i> topic.</li> </ul>

**continued...**

Document Version	Intel Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> <li>• In the <i>Timing Closure</i> chapter, made a minor addition to the first sentence.</li> <li>• Added the <i>Intel Agilex 7 FPGA EMIF IP – Timing Closure</i> chapter.</li> <li>• In the <i>Debugging</i> chapter, made changes to the following: <ul style="list-style-type: none"> <li>– In the <i>Debugging with the External Memory Interface Debug Toolkit</i> section, made changes to the following topics: <ul style="list-style-type: none"> <li>• <i>Rerunning the Traffic Generator</i> (added information on changing address ordering)</li> <li>• <i>Calibration Report Tab</i></li> <li>• <i>Calibrate Termination Tab</i> (recast text and added image)</li> <li>• <i>JSSPs Tab</i> (added row to bottom of table)</li> <li>• <i>Viewing Reports Graphically in the Eye Viewer</i> (added an additional eye diagram)</li> </ul> </li> <li>– In the <i>Configurable Traffic Generator (TG2) Description</i> section, made changes to the following topics: <ul style="list-style-type: none"> <li>• <i>Enabling the Traffic Generator in a Design Example</i></li> <li>• <i>Configuration and Status Registers</i> (changed description of TG_TEST_BYTEEN register)</li> <li>• <i>Configuring the Traffic Generator</i> (updated images)</li> <li>• <i>Traffic Generator Preset Selection</i> (new topic)</li> <li>• <i>Traffic Generator Status Report</i> (updated images)</li> </ul> </li> <li>– In the <i>On-Chip Debug Port</i> section, made changes to the following topics: <ul style="list-style-type: none"> <li>• <i>I/O SSM calbus Bridge Data Structures and Usage</i> (added note and figure, modified DQS Tree Structure table)</li> <li>• <i>Parameter Table Arrays</i> (modified 2 (DQS_C) entry in Example table)</li> <li>• <i>Debug Data Structures</i> (modified cur_interface_idx entry in the mem_summary_report table, and corrected text in debug_cal_data_struct section)</li> <li>• <i>Example: Reading Calibration Results and Margins with the On-Chip Debug Port</i></li> </ul> </li> <li>– In the <i>Efficiency Monitor</i> section, made changes to the following topics: <ul style="list-style-type: none"> <li>• <i>Enabling the Efficiency Monitor in a Design Example</i> (updated second bullet and image)</li> <li>• <i>Control and Status Registers</i> (updated last row in table)</li> <li>• <i>Opening the Efficiency Monitor Toolkit</i> (updated images)</li> </ul> </li> </ul> </li> </ul>

*continued...*

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.06.22	20.2	2.2.0	<ul style="list-style-type: none"> <li>• In the <i>Product Architecture</i> chapter, made minor changes to the <i>Intel Agilex 7 Calibration Stages</i> and <i>Intel Agilex 7 Calibration Algorithms</i> topics.</li> <li>• In the <i>Functional Simulation</i> chapter, added text and a figure to the <i>Simulation Walkthrough</i> topic.</li> <li>• In the <i>Debugging</i> chapter, updated images throughout the <i>Debugging with the External Memory Interface Debug Toolkit</i> section.</li> <li>• In the <i>Debugging</i> chapter, made several changes to the <i>Configurable Traffic Generator (TG2)</i> section: <ul style="list-style-type: none"> <li>— Minor change to terminology in the <i>Default Traffic Pattern, Configuration and Status Registers, Test Duration / Instruction Pattern, and Address Pattern</i> topics.</li> <li>— Expanded content in the <i>Starting Traffic with the Traffic Generator</i> topic.</li> <li>— Updated images in the <i>Configuring the Traffic Generator</i> and <i>Traffic Generator Status Report</i> topics.</li> </ul> </li> <li>• In the <i>Debugging</i> chapter, added the <i>EMIF On-chip Debug Port</i> section.</li> </ul>
2020.04.27	20.1	2.1.0	<ul style="list-style-type: none"> <li>• In the <i>DDR4</i> chapter, modified guidelines 2 and 3, and added guideline 11, in the <i>General Guidelines</i> topic.</li> <li>• In the <i>QDR-IV</i> chapter: <ul style="list-style-type: none"> <li>— Modified guidelines 2 and 3, and added guideline 11, in the <i>General Guidelines</i> topic.</li> <li>— Added guideline 4 to the <i>I/O Bank</i> section of the <i>Resource Sharing Guidelines (Multiple Interfaces)</i> topic.</li> </ul> </li> </ul>
2020.04.13	20.1	2.1.0	<ul style="list-style-type: none"> <li>• In the <i>Introduction</i> chapter: <ul style="list-style-type: none"> <li>— Added QDR-IV support to the <i>Intel Agilex 7 EMIF IP Protocol and Feature Support</i> topic.</li> <li>— Added links to <i>QDR-IV Parameter Descriptions</i> to the <i>Intel Agilex 7 EMIF IP Design Checklist</i> topic.</li> </ul> </li> <li>• In the <i>Architecture</i> chapter: <ul style="list-style-type: none"> <li>— Added QDR-IV support to the <i>Introduction</i> topic.</li> </ul> </li> <li>• In the <i>End-User Signals</i> chapter: <ul style="list-style-type: none"> <li>— Added QDR-IV interfaces and signals to the <i>Interface and Signal Descriptions</i> section.</li> <li>— In the <i>AFI 4.0 Timing Diagrams</i> section, removed <i>Write data sequence with CRC</i>.</li> <li>— Implemented changes to the <i>ctrlcfg1</i>, <i>sbcfg1</i>, and <i>caltiming4</i> tables in the <i>Memory Mapped Register (MMR) Tables</i> section.</li> </ul> </li> <li>• Added the <i>Intel Agilex 7 FPGA EMIF IP – QDR-IV Support</i> chapter.</li> <li>• In the <i>Debugging</i> chapter: <ul style="list-style-type: none"> <li>— Implemented numerous changes to the <i>Debugging with the External Memory Interface Debug Toolkit</i> section.</li> <li>— Added the <i>Configurable Traffic Generator (TG2)</i> <i>Description</i> section.</li> </ul> </li> </ul>

**continued...**



Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.02.10	19.4	2.0.0	<ul style="list-style-type: none"> <li>• In the <i>DDR4</i> chapter:           <ul style="list-style-type: none"> <li>— Added the <i>x4 DIMM Implementation</i> topic to the <i>Pin and Resource Planning</i> section.</li> <li>— Added the <i>Clamshell Topology</i> topic to the <i>DDR4 Board Design Guidelines</i> section.</li> </ul> </li> <li>• In the <i>Debugging</i> chapter, modified the last sentence of the <i>Intermittent Issue Evaluation</i> topic.</li> </ul>
2019.12.16	19.4	2.0.0	<ul style="list-style-type: none"> <li>• In the <i>Architecture</i> chapter:           <ul style="list-style-type: none"> <li>— Modified the first sentence following Figure 7 in the <i>I/O Bank</i> topic.</li> <li>— Added the <i>Considerations for Designing DDR4 x72 Interface Together with an AVST x8/x16/x32 Configuration Scheme</i> topic.</li> </ul> </li> <li>• In the <i>DDR4</i> chapter:           <ul style="list-style-type: none"> <li>— Expanded the <i>DDR4 Board Design Guidelines</i> section.</li> </ul> </li> <li>• In the <i>Debugging</i> chapter:           <ul style="list-style-type: none"> <li>— Added the <i>Debugging With the External Memory Interface Debug Toolkit</i> section.</li> </ul> </li> <li>• Added <i>External Memory Interfaces Intel Agilex 7 FPGA IP User Guide Archives</i> topic.</li> </ul>
2019.10.18	19.3		<ul style="list-style-type: none"> <li>• In the <i>Introduction</i> chapter, revised the <i>EMIF IP Design Flow</i> flowchart.</li> <li>• In the <i>Product Architecture</i> chapter:           <ul style="list-style-type: none"> <li>— Updated the <i>Intel Agilex I/O Subsystem</i> figure in the <i>EMIF Architecture: I/O Subsystem</i> topic.</li> <li>— Modified the first paragraph, and updated the connectivity diagrams in the <i>EMIF Architecture: I/O SSD</i> topic.</li> <li>— Modified the <i>Pin Index Mapping</i> table in the <i>EMIF Architecture: I/O Lane</i> topic.</li> <li>— Added an explanation of Quasi-1T to the description of the Arbiter component in the <i>Main Control Path Components</i> table in the <i>Hard Memory Controller Main Control Path</i> topic.</li> <li>— Removed the note from the beginning of the <i>Intel Agilex 7 EMIF for Hard Processor Subsystem</i> topic.</li> </ul> </li> <li>• In the <i>DDR4 Support</i> chapter:           <ul style="list-style-type: none"> <li>— Removed the <i>Board Skew Equations</i> section.</li> <li>— Modified entries for the Clock pin in the <i>UDIMM, RDIMM, and LRDIMM Pin Options for DDR4</i> table.</li> <li>— Removed the <i>Channel Signal Integrity Measurement and Package Deskew</i> sections.</li> </ul> </li> </ul>

*continued...*

Document Version	Intel Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> <li>• In the <i>Timing Closure</i> chapter:           <ul style="list-style-type: none"> <li>— Modified the <i>Core to periphery (C2P)</i> and <i>Periphery to core (P2C)</i> descriptions in the <i>Timing Closure</i> topic.</li> <li>— Removed references to <i>Early I/O Timing Estimation</i>.</li> </ul> </li> <li>• In the <i>Controller Optimization</i> chapter:           <ul style="list-style-type: none"> <li>— In the <i>Bank Interleaving</i> topic, modified the names of the three supported interleaving options.</li> <li>— Added content to the paragraph introducing the examples, in the <i>Using Auto-precharge to Achieve Highest Memory Bandwidth for DDR4 Interfaces</i> topic.</li> </ul> </li> <li>• In the <i>Debugging</i> chapter:           <ul style="list-style-type: none"> <li>— Added the <i>Debugging with the External Memory Interface Debug Toolkit</i> section.</li> <li>— Added the <i>Using the Traffic Generator with the Generated Design Example</i> topic.</li> </ul> </li> </ul>
2019.07.31	19.2	1.2.0	<ul style="list-style-type: none"> <li>• Added the <i>About the External Memory Interfaces Intel Agilex 7 FPGA IP</i> chapter.</li> <li>• In the topic <i>Intel Agilex 7 EMIF for Hard Processor Subsystem</i>, in the <i>Product Architecture</i> chapter, changed the description of <i>Memory format</i> in the table from 16GB support to 32GB support.</li> <li>• In the <i>Intel Agilex 7 FPGA EMIF IP — End-User Signals</i> chapter:           <ul style="list-style-type: none"> <li>— Removed <i>emif_usr_reset_n_sec</i> and <i>emif_usr_clk_sec</i> from Table 10, <i>Interfaces for DDR4</i>.</li> <li>— In the topic <i>3.1.1.10, mem for DDR4</i>, modified the description of <i>mem_a</i> in the table.</li> <li>— Removed the <i>3.1.1.18 emif_usr_reset_n_sec for DDR4</i> and <i>3.1.1.19 emif_usr_clk_sec for DDR4</i> topics.</li> <li>— Removed <i>sbcfg1</i>, <i>sideband2</i>, <i>sideband3</i>, <i>sideband5</i>, <i>sideband8</i>, <i>sideband10</i>, and <i>sideband15</i> from the <i>Intel Agilex 7 EMIF IP Memory Mapped Register (MMR) Tables</i>.</li> <li>— Changed the <i>Bit High</i> value for the second row in the <i>dramtiming0</i> MMR table.</li> <li>— Changed the <i>Field</i> name and <i>Description</i> text for the fourth row in the <i>caltiming4</i> MMR table.</li> <li>— Made several changes in the <i>Description</i> column of the <i>sideband13</i> MMR table.</li> <li>— Changed the <i>Field</i>, <i>Bit High</i>, <i>Bit Low</i>, and <i>Description</i> values in the <i>sideband14</i> MMR table.</li> </ul> </li> <li>• In the <i>Intel Agilex 7 EMIF IP DDR4 Parameters: Memory</i> topic of the DDR4 chapter:           <ul style="list-style-type: none"> <li>— Removed the <i>Enable ALERT#/PAR pins</i> parameter and recast the description of the <i>ALERT# pin placement</i> parameter in the <i>Group: Memory / Topology</i> table.</li> </ul> </li> <li>• In the <i>Intel Agilex 7 EMIF IP DDR4 Parameters: Mem I/O</i> topic, revised the description for the <i>SPD Byte 145-147 - DB MDQ Drive Strength and RTT</i> parameter.</li> </ul>

*continued...*



Document Version	Intel Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"><li>• In the <i>Intel Agilex 7 EMIF IP DDR4 Parameters: Diagnostics</i> topic:<ul style="list-style-type: none"><li>— Added the <i>Group: Diagnostics / Example Design</i> table.</li><li>— Added the <i>Group: Diagnostics / Traffic Generator</i> table (marked as future support).</li><li>— Added the <i>Group: Diagnostics / Performance</i> and <i>Group: Diagnostics / Miscellaneous</i> tables.</li></ul></li><li>• Added the <i>Intel Agilex 7 EMIF IP DDR4 Parameters: Example Designs</i> topic.</li><li>• In the <i>Intel Agilex 7 FPGA EMIF IP — Timing Closure</i> chapter, revised the <i>Optimizing Timing</i> topic.</li><li>• Removed occurrences of <i>Ping Pong PHY</i> throughout.</li></ul>
2019.04.02	19.1		<ul style="list-style-type: none"><li>• Initial release.</li></ul>