External Memory Interfaces Intel® Agilex™ FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.3**

Subscribe

Send Feedback

UG-20218 | 2019.10.18

Latest document on the web: PDF | HTML

Contents

1. About the External Memory Interfaces Intel® Agilex™ 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 versioning scheme (X.Y.Z) number changes from one 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	1.2.0
Intel Quartus Prime	19.3
Release Date	2019.09.30

2. Intel Agilex™ 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™ FPGA IP* (referred to hereafter as the *Intel Agilex 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* https://www.intel.com/content/www/us/en/programmable/support/support-resources/support-centers/external-memory-interfaces-support/emif.html.

2.1. Intel Agilex EMIF IP Protocol and Feature Support

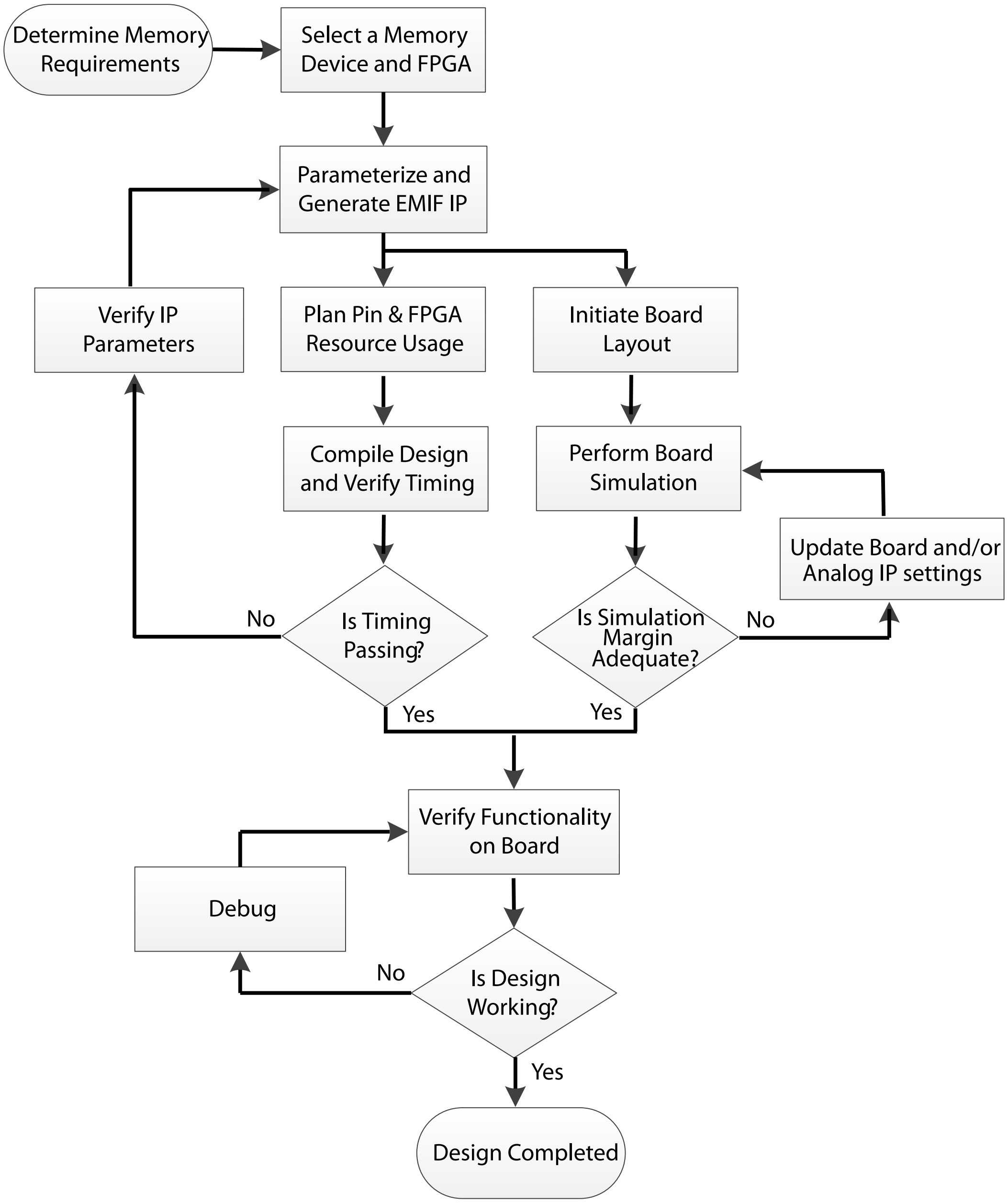
- Supports DDR4 with hard memory controller and hard PHY.

2.2. Intel Agilex 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


Determine Memory Requirements
Select a Memory Device and FPGA
Parameterize and Generate EMIF IP
Verify IP Parameters
Plan Pin & FPGA Resource Usage
Initiate Board Layout
Compile Design and Verify Timing
Perform Board Simulation
Update Board and/or Analog IP settings
No
Is Timing Passing?
Is Simulation Margin Adequate?
No
Yes
Yes
Verify Functionality on Board
Debug
No
Is Design Working?
Yes
Design Completed

2.3. Intel Agilex 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.	• Intel FPGA Product Selector • External Memory Interface Device Selector • External Memory Interface Spec Estimator
Parameterize the IP	Correct IP parameterization is important for good EMIF IP operation. The resources listed in the right column define the memory parameters during IP generation.	• DDR4 Parameter Descriptions
Generate initial IP and example design	After you have parameterized the EMIF IP, you can generate the IP, along with an optional example design. Refer to the Quick-Start Guide for a walkthrough of this process.	• Design Example Quick Start Guide
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.	• Design Example Quick Start Guide • Simulating Memory IP
Make pin assignments	For guidance on pin placement, refer to the resources listed in the right column.	• DDR4 Parameter Descriptions • Device Pin Tables
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.	• Board Design Guidelines • Board Skew Parameter Tool
Verify timing closure	For information regarding compilation, system-level timing closure and timing reports refer to the Timing Closure section of this User Guide.	• Timing Closure
Run the design on hardware	For instructions on how to program a FPGA refer to the Quick-Start section of the Design Example User Guide.	• Design Example Quick Start Guide
Debug issues with preceding steps	Operational problems can generally be attributed to one of the following: interface configuration, pin/resource planning, signal integrity, or timing. The resources listed in the right column contain information on typical debug procedures and available tools to help diagnose hardware issues.	• Debugging • External Memory Interfaces Support Center

3. Intel Agilex FPGA EMIF IP – Product Architecture

This chapter describes the Intel Agilex FPGA EMIF IP product architecture.

3.1. Intel Agilex EMIF Architecture: Introduction

The Intel Agilex 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 FPGA IP supports the DDR4 memory protocol. Future versions will include support for QDR-IV and RLDRAM 3 protocols.

The following are key hardware features of the Intel Agilex 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 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.

High-Speed PHY Clock Tree

Dedicated high speed PHY clock networks clock the I/O buffers in Intel Agilex 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.

3.1.1. Intel Agilex EMIF Architecture: I/O Subsystem

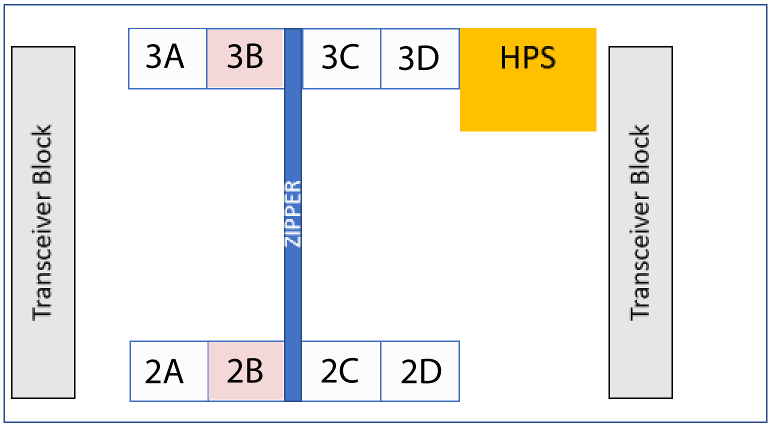
In Intel Agilex devices, the I/O subsystem consists of two rows at the edge of the core.

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 I/O Subsystem

The following figure depicts the I/O subsystem structure for 014 series devices. I/O banks 2B and 3B are the locations of calibration I/O SSMs in these devices.



3.1.2. Intel Agilex 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. It's location is fixed in one of the I/O banks along each edge and depends on the base die.

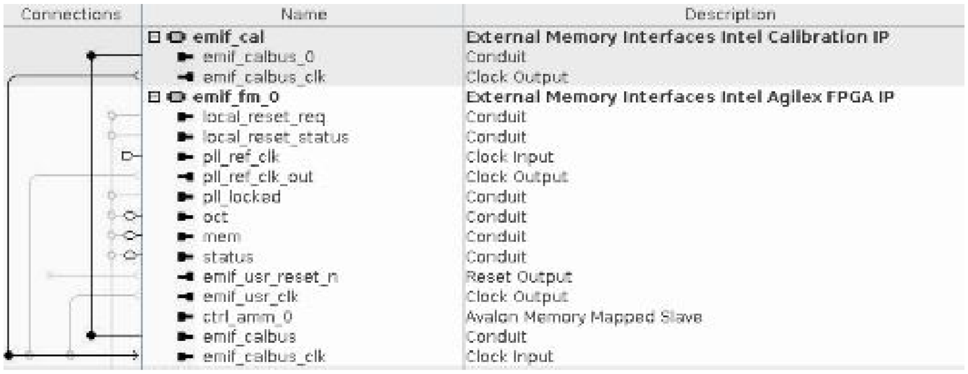
The I/O SSM includes dedicated memory which stores both the calibration algorithm and calibration run-time data. The hardened Nios II processor and the dedicated memory can be used only by an external memory interface, and cannot be employed for any other use. The I/O SSM can interface with soft logic,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

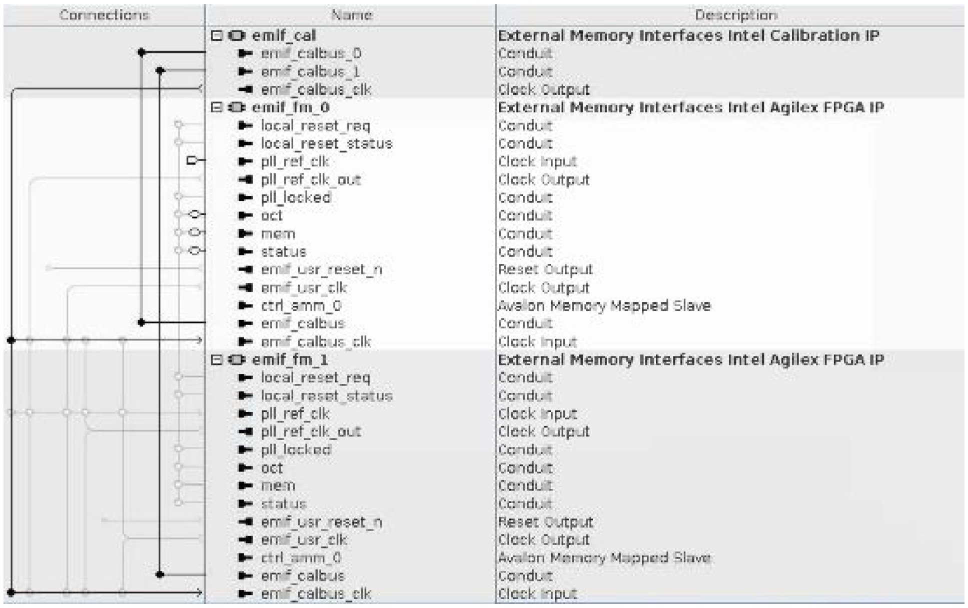
Only one calibration IP is allowed for each I/O row. All the EMIFs in the same I/O row must be connected to the same calibration I/P. You can specify the number of EMIF interfaces to be connected to the calibration IP when parameterizing the IP. Connect the `emif_calbus` and `emif_calbus_clk` on the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



Connections	Name	Description
	⊟ **emif_cal**	**External Memory Interfaces Intel Calibration IP**
	emif_calbus_0	Conduit
	emif_calbus_clk	Clock Output
	⊟ **emif_fm_0**	**External Memory Interfaces Intel Agilex FPGA IP**
	local_reset_req	Conduit
	local_reset_status	Conduit
	pll_ref_clk	Clock Input
	pll_ref_clk_out	Clock Output
	pll_locked	Conduit
	oct	Conduit
	mem	Conduit
	status	Conduit
	emif_usr_reset_n	Reset Output
	emif_usr_clk	Clock Output
	ctrl_amm_0	Avalon Memory Mapped Slave
	emif_calbus	Conduit
	emif_calbus_clk	Clock Input

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



Connections	Name	Description
	emif_cal	**External Memory Interfaces Intel Calibration IP**
	emif_calbus_0	Conduit
	emif_calbus_1	Conduit
	emif_calbus_clk	Clock Output
	emif_fm_0	**External Memory Interfaces Intel Agilex FPGA IP**
	local_reset_req	Conduit
	local_reset_status	Conduit
	pll_ref_clk	Clock Input
	pll_ref_clk_out	Clock Output
	pll_locked	Conduit
	oct	Conduit
	mem	Conduit
	status	Conduit
	emif_usr_reset_n	Reset Output
	emif_usr_clk	Clock Output
	ctrl_amm_0	Avalon Memory Mapped Slave
	emif_calbus	Conduit
	emif_calbus_clk	Clock Input
	emif_fm_1	**External Memory Interfaces Intel Agilex FPGA IP**
	local_reset_req	Conduit
	local_reset_status	Conduit
	pll_ref_clk	Clock Input
	pll_ref_clk_out	Clock Output
	pll_locked	Conduit
	oct	Conduit
	mem	Conduit
	status	Conduit
	emif_usr_reset_n	Reset Output
	emif_usr_clk	Clock Output
	ctrl_amm_0	Avalon Memory Mapped Slave
	emif_calbus	Conduit
	emif_calbus_clk	Clock Input

3.1.3. Intel Agilex EMIF Architecture: I/O Bank

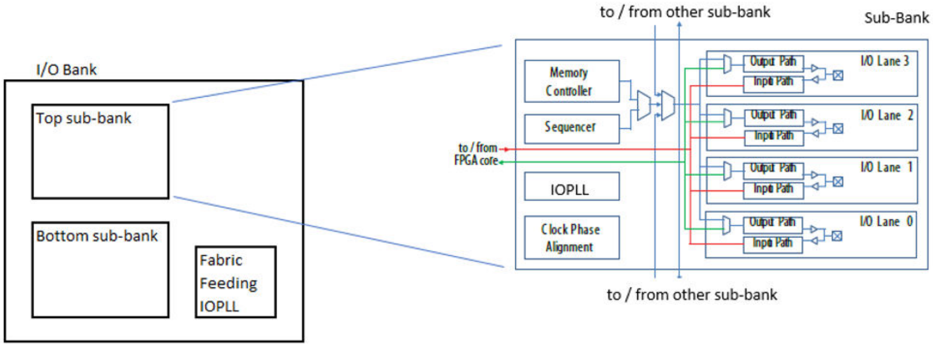
Each I/O row contains up to four I/O banks; the exact number of banks depends on device size and pin package.

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 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 Intel Agilex AGF012 and AGF014 device variants, respectively. 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 AGF012 and AGF014 devices

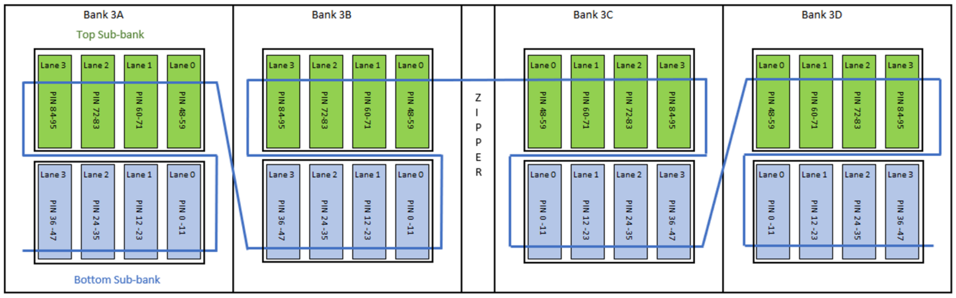
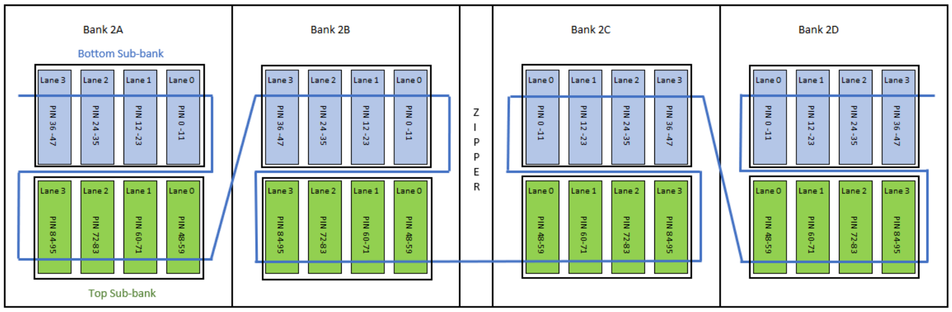


Figure 7. Sub-Bank Ordering in Bottom I/O Row in Intel Agilex AGF012 and AGF014 devices



The two sub-banks within an I/O bank are adjacent to each other, unless any of the sub-banks are not bonded out or partially bonded out. The blue line in the above figures shows the connectivity between the sub-banks.

For example, in the top row in Intel Agilex 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.

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 banks in a multi-bank interface.

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 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 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` in the device pinout.

Table 3. Pin Index Mapping

Pin Index	Lane	Sub-bank Location
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.

Table 4. Lanes Used Per Group

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

Figure 8. x4 Group

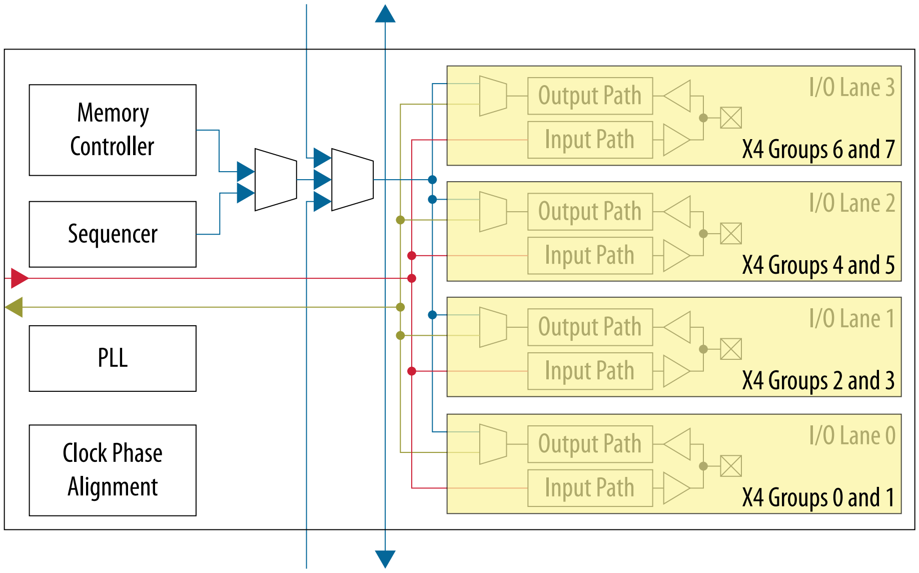


Figure 9. x8 Group

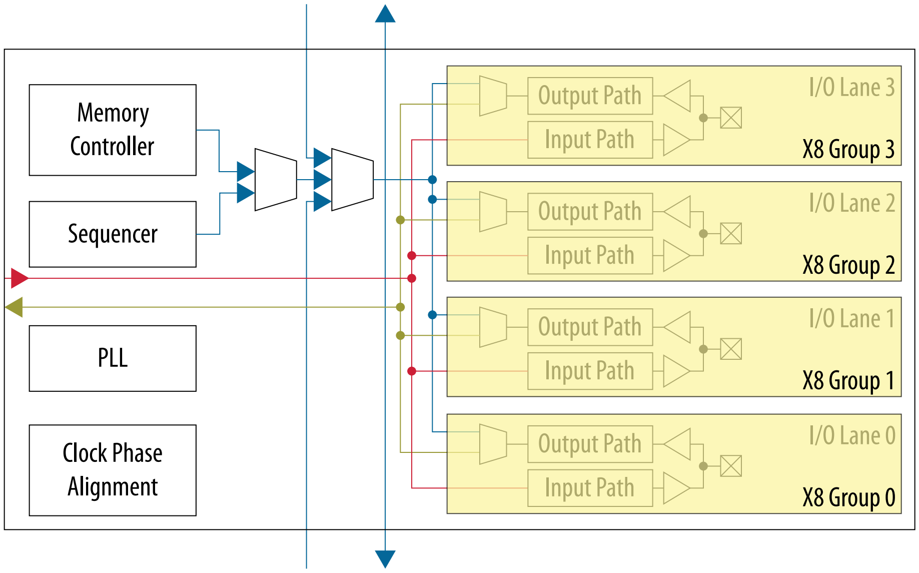
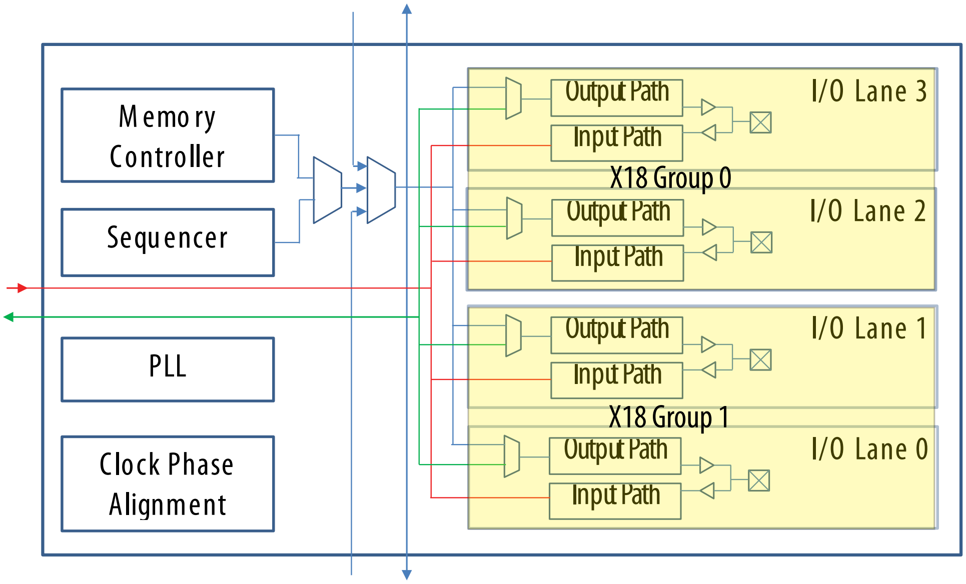


Figure 10. x18 Group



3.1.5. Intel Agilex 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 5. Pins Usable as Read Capture Clock / Strobe Pair

Group Size	Index of Lanes Spanned by Clock Tree [1]	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
x8 / x9	1		16	17

continued...

Group Size	Index of Lanes Spanned by Clock Tree [1]	Sub-Bank	Index of Pins Usable as Read Capture Clock / Strobe Pair	
			DQS p	DQS n
x8 / x9	2		28	29
x8 / x9	3		40	41
x18	0, 1		4	5
x18	2, 3		28	29
x4	0A	Top	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: [1] *A* and *B* refer to the two nibbles within the lane.

3.1.6. Intel Agilex EMIF Architecture: PHY Clock Tree

Dedicated high-speed clock networks drive I/Os in Intel Agilex 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 devices can run as fast as 1.6 GHz. All Intel Agilex external memory interfaces use the PHY clock trees.

3.1.7. Intel Agilex 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.

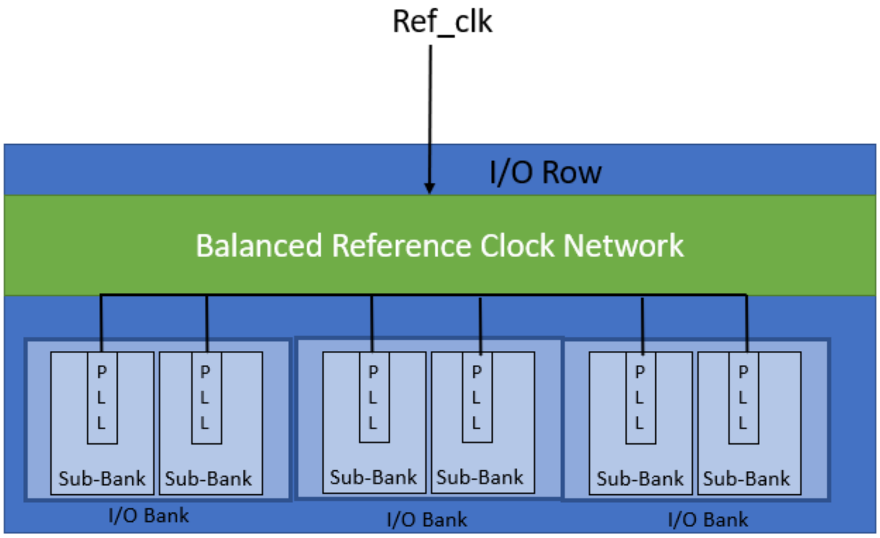
- A single-ended reference clock must be constrained to pin index 0 in lane 2. You cannot use pin index 1 in lane 2 as a general purpose I/O pin.
- Differential reference clocks must be constrained to pin indices 0 and 1 in lane 2.

Intel Agilex external memory interfaces that span multiple banks use the PLL in each bank. The Intel Agilex 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 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 11. PLL Balanced Reference Clock Tree



3.1.8. Intel Agilex EMIF Architecture: Clock Phase Alignment

In Intel Agilex 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 external memory interfaces employ clock phase alignment circuitry.

Figure 12. Clock Phase Alignment Illustration

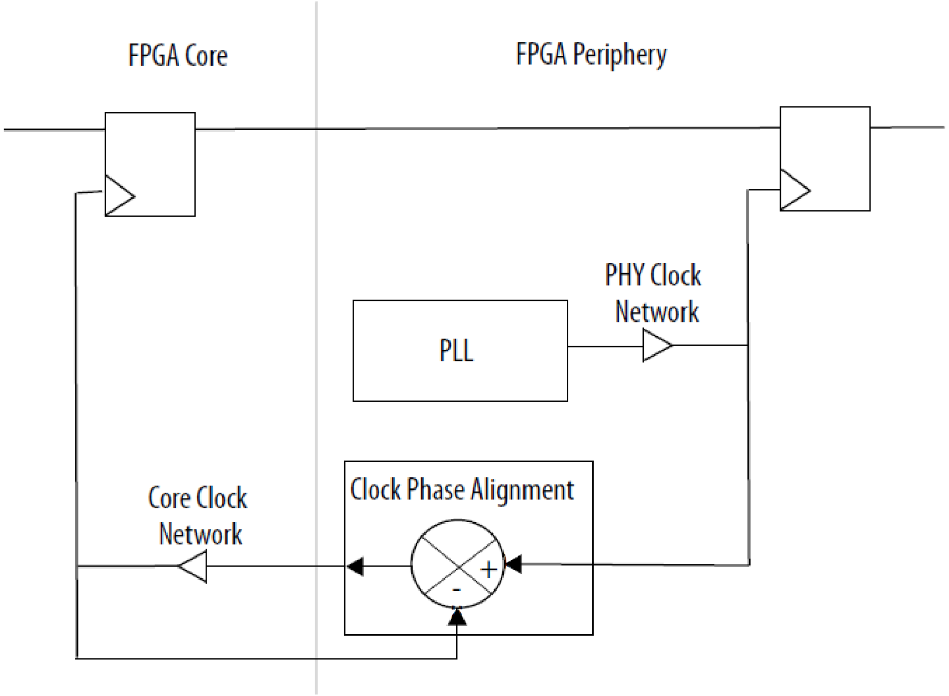
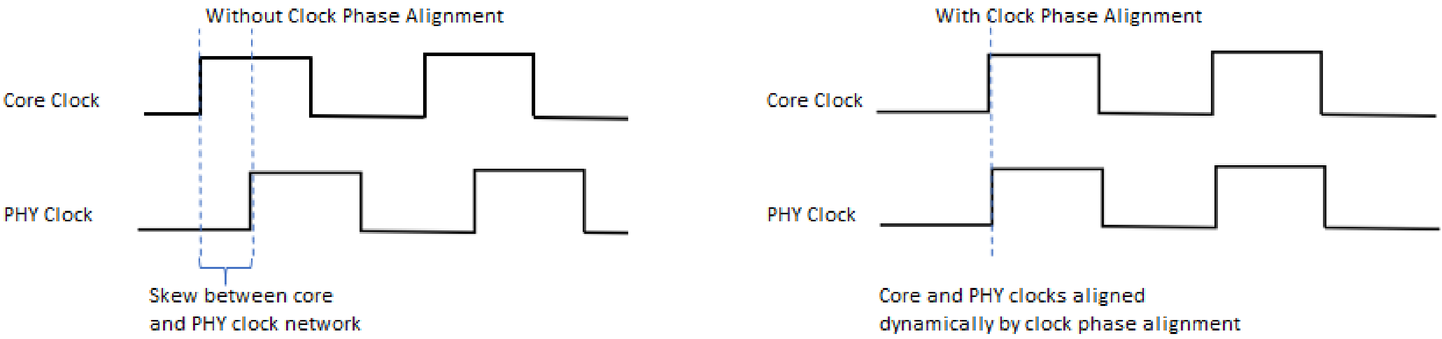


Figure 13. Effect of Clock Phase Alignment


Without Clock Phase Alignment
Core Clock
PHY Clock
Skew between core
and PHY clock network
With Clock Phase Alignment
Core Clock
PHY Clock
Core and PHY clocks aligned
dynamically by clock phase alignment

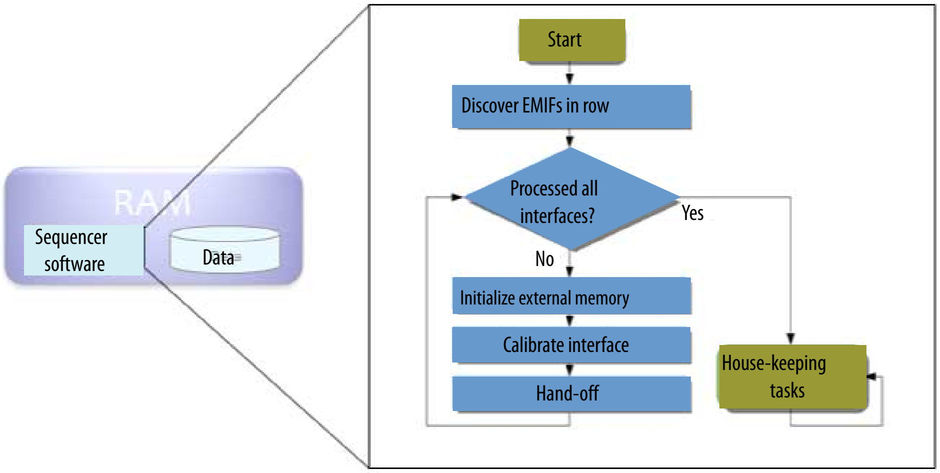
3.2. Intel Agilex EMIF Sequencer

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

The Intel Agilex 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 14. Intel Agilex EMIF Sequencer Operation



3.3. Intel Agilex 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 Calibration Stages

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

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

Table 6. Calibration Stages by Protocol

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

3.3.2. Intel Agilex 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

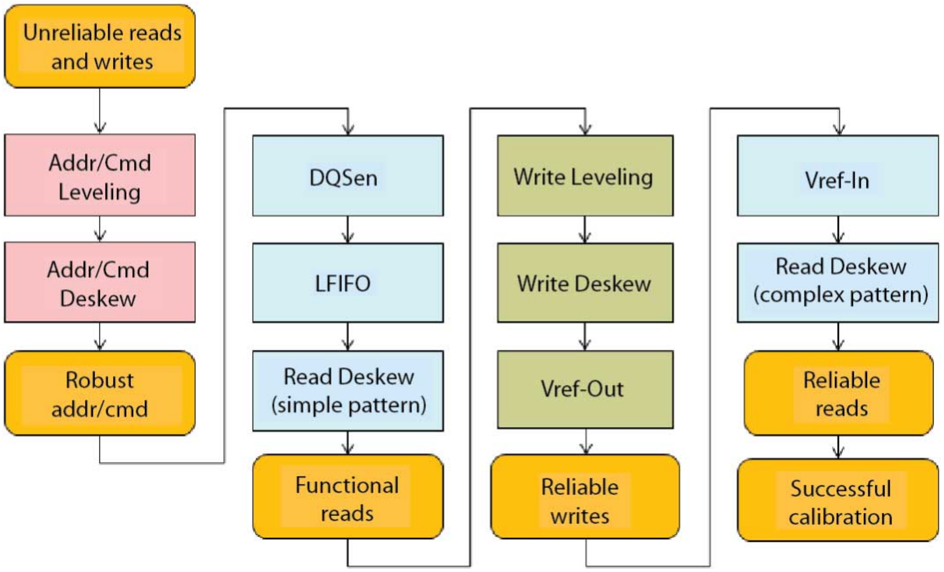
Write calibration consists of the following parts:

- 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 Calibration Flowchart

The following flowchart illustrates the Intel Agilex calibration flow.

Figure 15. Calibration Flowchart



3.3.4. Intel Agilex Calibration Algorithms

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

Address and Command Calibration

Address and command calibration consists of the following parts:

- Leveling calibration— (DDR4 only) Toggles the CS# and CAS# signals to send read commands while keeping other address and command signals constant. The algorithm monitors for incoming DQS signals, and if the DQS signal toggles, it indicates that the read commands have been accepted. The algorithm then repeats using different delay values, to find the optimal window.

- Deskew calibration— (DDR4 and QDR-IV only)
  - (DDR4) Uses the DDR4 address and command parity feature. The FPGA sends the address and command parity bit, and the DDR4 memory device responds with an alert signal if the parity bit is detected. The alert signal from the memory device tells the FPGA that the parity bit was received.

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

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

Read Calibration

- DQSen calibration— (DDR4, RLDRAM 3, and QDR-IV) DQSen calibration occurs before Read deskew, therefore only a single DQ bit is required to pass in order to achieve a successful read pass.
  - (DDR4) The DQSen calibration algorithm searches the DQS preamble using a hardware state machine. The algorithm sends many back-to-back reads with a one clock cycle gap between. The hardware state machine searches for the DQS gap while sweeping DQSen delay values. The algorithm then increments the VFIFO value, and repeats the process until a pattern is found. The process then repeats for all other read DQS groups.
  - (RLDRAM 3 and QDR-IV) The DQSen calibration algorithm does not use a hardware state machine; rather, it calibrates cycle-level delays using software and subcycle delays using DQS tracking hardware. The algorithm requires good data in memory, and therefore relies on guaranteed writes. (Writing a burst of 0s to one location, and a burst of 1s to another; back-to-back reads from these two locations are used for read calibration.)

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

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

Write Calibration

- Leveling calibration— Write leveling calibration aligns the write strobe and clock to the memory clock, to compensate for skews. In general, leveling calibration tries a variety of delay values to determine the edges of the write window, and then selects an appropriate value to center the window. The details of the algorithm vary, depending on the memory protocol.
    - (DDR4) Write leveling occurs before write deskew, therefore only one successful DQ bit is required to register a pass. Write leveling staggers the DQ bus to ensure that at least one DQ bit falls within the valid write window.
    - (RLDRAM 3) Optimizes for the CK versus DK relationship.
    - (QDR-IV) Optimizes for the CK versus DK relationship. It is covered by address and command deskew using the loopback mode.
- Deskew calibration— Performs per-bit deskew of write data relative to the write strobe and clock. Write deskew calibration does not change `dqs_out` delays; the write clock is aligned to the CK clock during write leveling.
- VREF-Out calibration— (DDR4) Calibrates the VREF level at the memory device. The VREF-Out calibration algorithm is similar to the VREF-In calibration algorithm.

3.4. Intel Agilex EMIF Controller

3.4.1. Hard Memory Controller

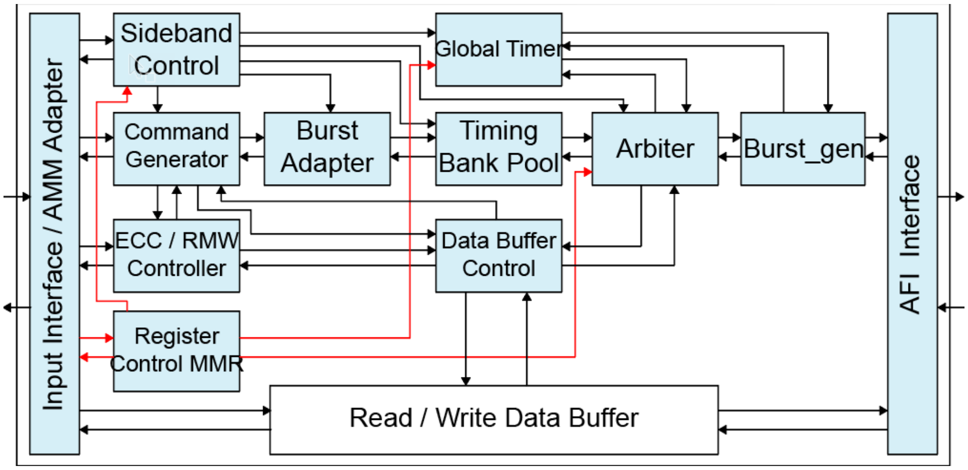
The Intel Agilex hard memory controller is designed for high speed, high performance, high flexibility, and area efficiency. The Intel Agilex hard memory controller supports the DDR4 memory standard.

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 16. Hard Memory Controller Architecture



The hard memory controller consists of the following logic blocks:

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

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

3.4.1.1. Hard Memory Controller Features

Table 7. Features of the Intel Agilex Hard Memory Controller

Feature	Description
Memory standards support	Supports DDR4 SDRAM.
Memory devices support	Supports the following memory devices: • Discrete • UDIMM • RDIMM • LRDIMM • SODIMM
3D Stacked Die support	Supports 2 and 4 height of 3D stacked die for DDR4 to increase memory capacity.
Memory controller bypass mode (Future support.)	You can use this configurable mode to bypass the hard memory controller and use your own customized controller.
Interface protocols support	• Supports Avalon-MM interface. • The PHY interface adheres to the AFI protocol.
Rate support	The legal options are:

continued...

Feature	Description
	• HMC half-rate, user logic half-rate (extremely slow interfaces only) • HMC half-rate, user-logic quarter-rate • HMC quarter-rate, user-logic quarter-rate (extremely high-speed interfaces only)
Configurable memory interface width	Supports data widths from 8 to 72 bits, in 8 bit increments
Multiple ranks support	Supports: • 4 ranks with single slot • 2 ranks with dual slots
Burst adapter	Able to accept burst lengths of 1–127 on the local interface of the controller and map the bursts to efficient memory commands. For applications that must strictly adhere to the -MM specification, the maximum burst length is 64. No burst chop support for DDR4.
Efficiency optimization features	• Open-page policy—by default, opens page on every access. However, the controller intelligently closes a row based on incoming traffic, which improves the efficiency of the controller especially for random traffic. • Pre-emptive bank management—the controller issues bank management commands early, which ensures that the required row is open when the read or write occurs. • Data reordering—the controller reorders read/write commands. • Additive latency—the controller can issue a `READ/WRITE` command after the `ACTIVATE` command to the memory bank prior to t_{RCD}, which increases the command efficiency.
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	• User-controlled refresh timing—optionally, you can control when refreshes occur and this allows you to prevent important read or write operations from clashing with the refresh lock-out time. • Per-rank refresh—allows refresh for each individual rank. • Controller-controlled refresh.
ECC support	• 8 bit ECC code; single error correction, double error detection (SECDED). • User ECC supporting pass through user ECC bits as part of data bits.

continued...

Feature	Description
Power saving features	• Low power modes (power down and self-refresh)—optionally, you can request the controller to put the memory into one of the two low power states. • Automatic power down—puts the memory device in power down mode when the controller is idle. You can configure the idle waiting time. • Memory clock gating.
DDR4 features	• Bank group support—supports different timing parameters for between bank groups. • Command/Address parity—command and address bus parity check. • Support Direct Dual CS Mode and Direct QuadCS Mode for DDR4 LRDIMM devices. • Support Encoded Quad CSMode for single CS assertion memory mapping for DDR4 LRDIMM devices.
User ZQ calibration	Long or short ZQ calibration request for DDR4.

3.4.1.2. Hard Memory Controller Main Control Path

The main control path performs the following functions:

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

Table 8. Main Control Path Components

Component	Description
Input interface	• Accepts memory access commands from the core logic at half or quarter rate. • Uses the Avalon-MM protocol. • You can connect the Avalon-MM interface to an AXI bus master in Platform Designer. To connect the Avalon-MM interface, implement the AXI bus master as a Platform Designer component and connect the AXI bus master to the Avalon-MM slave. The Platform Designer interconnect performs the bus translation between the AXI and Avalon-MM bus interfaces.
Command generator and burst adapter	• Drains your commands from the input interface and feeds them to the timing bank pool. • If read-modify-write is required, inserts the necessary read-modify-write read and write commands into the stream. • The burst adapter chops your arbitrary burst length to the number specified by the memory types.
Timing Bank Pool	• Key component in the memory controller. • Sets parallel queues to track command dependencies. • Signals the ready status of each command being tracked to the arbiter for the final dispatch. • Big scoreboard structure. The number of entries is currently sized to 16 where it monitors up to 16 commands at the same time. • Handles the memory access hazards such as Read After Write (RAW), Write After Read (WAR), and Write After Write (WAW), while part of the timing constraints are being tracked. • Assist the arbiter in reordering row commands and column commands. • When the pool is full, a flow control signal is sent back upstream to stall the traffic.

continued...

Component	Description
Arbiter	• Enforces the arbitration rules. • Performs the final arbitration to select a command from all ready commands, and issues the selected command to the memory. • Supports Quasi-1T mode for half rate mode. • For the quasi modes, a row command must be paired with a column command. *Note:* For quasi modes, a row command must be paired with a column command. Quasi-1T mode indicates only 1 command will be executed within 1 cycle; the command can be a row command or a column column. Quasi-2T allows 2 commands to be executed within 1 cycle and the 2 commands must be a row command paired with a column command.
Global Timer	Tracks the global timing constraints including: • t_{FAW}—the Four Activates Window parameter that specifies the time period in which only four activate commands are allowed. • t_{RRD}—the delay between back-to-back activate commands to different banks. • Some of the bus turnaround time parameters.
MMR/IOCSR	• The host of all the configuration registers. • Uses Avalon-MM bus to talk to the core. • Core logic can read and write all the configuration bits.
Sideband	Executes the refresh and power down features.
ECC controller	Although ECC encoding and decoding is performed in soft logic[1], the ECC controller maintains the read-modify-write state machine in the hard solution.
AFI interface	The memory controller communicates with the PHY using this interface.

3.4.1.3. Data Buffer Controller

The data buffer controller performs the following operations:

- 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 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.

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

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 EMIF IP

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

Table 9.

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

continued...

	Description
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.

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

Step 2 - Reset Request

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

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

Optional - Detecting local_reset_done deassertion and assertion

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

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

3.6. Intel Agilex EMIF for Hard Processor Subsystem

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

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

Supported Modes

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

Table 10. Intel Agilex 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. • Discrete components with up to 2 chip selects * • Non-3DS UDIMM or RDIMM with up to 2 chip selects * • SODIMM with up to 2 ranks *

* 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.

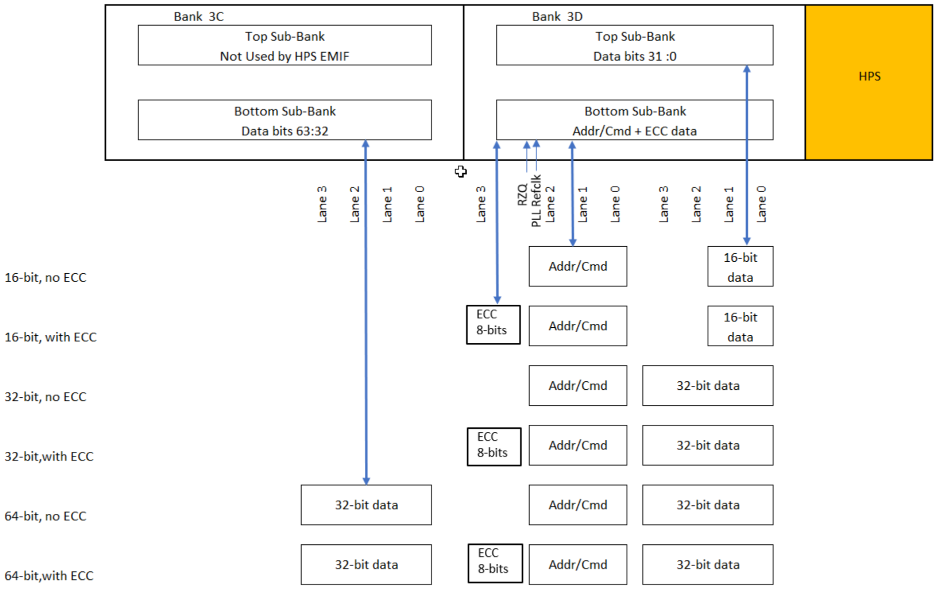
3.6.1. Restrictions on I/O Bank Usage for Intel Agilex EMIF IP with HPS

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

The restrictions on I/O bank usage result from the Intel Agilex 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:

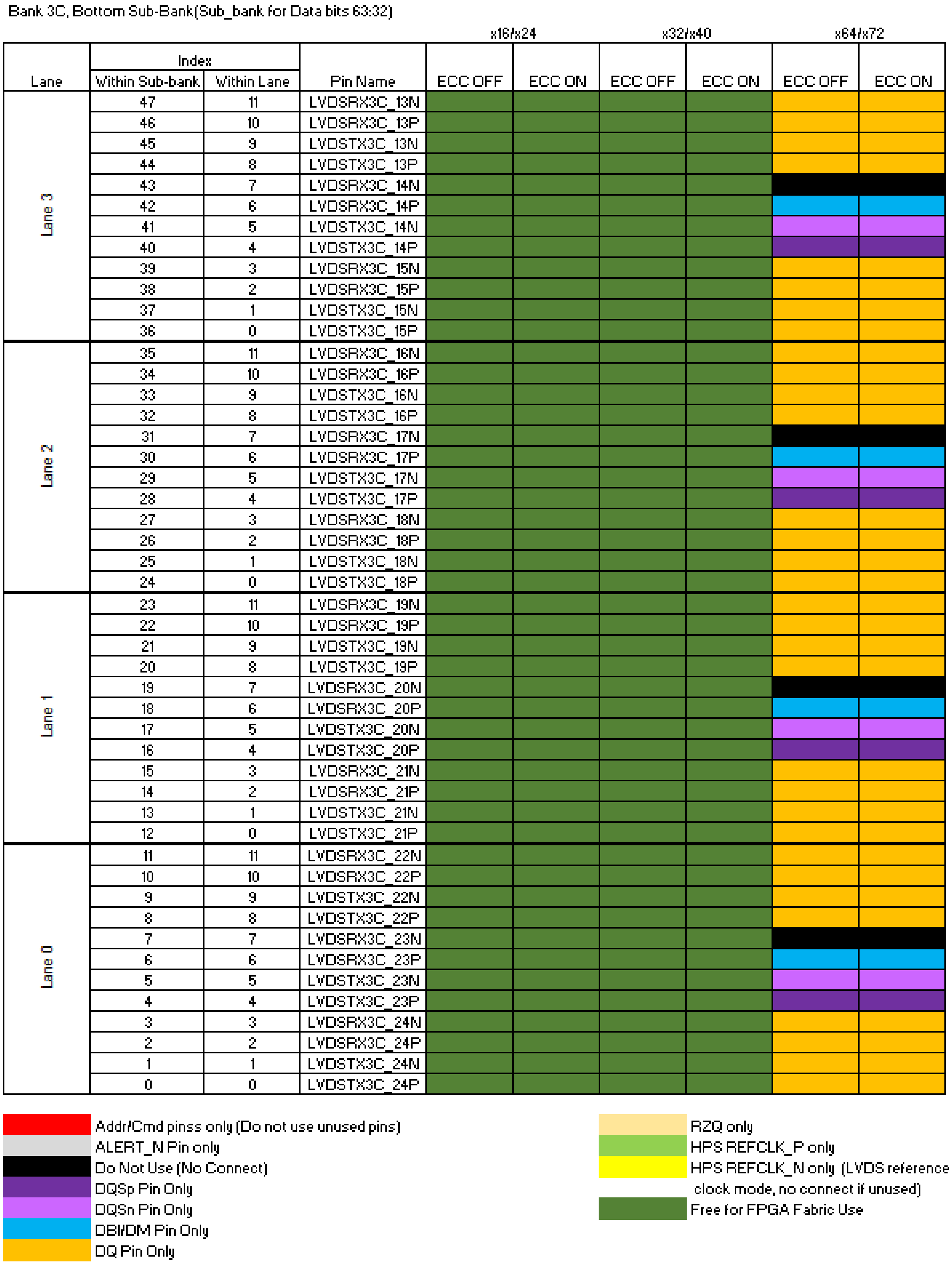
Figure 17. Intel Agilex HPS - EMIF I/O Bank and Lanes Usage



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 18. I/O Pin Usage Restrictions for Intel Agilex 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	🟩	🟩	🟩	🟩	🟧	🟧
Lane 2	35	11	LVDSRX3C_16N	🟩	🟩	🟩	🟩	🟧	🟧
	34	10	LVDSRX3C_16P	🟩	🟩	🟩	🟩	🟧	🟧
	33	9	LVDSTX3C_16N	🟩	🟩	🟩	🟩	🟧	🟧
	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	🟩	🟩	🟩	🟩	🟧	🟧
Lane 1	23	11	LVDSRX3C_19N	🟩	🟩	🟩	🟩	🟧	🟧
	22	10	LVDSRX3C_19P	🟩	🟩	🟩	🟩	🟧	🟧
	21	9	LVDSTX3C_19N	🟩	🟩	🟩	🟩	🟧	🟧
	20	8	LVDSTX3C_19P	🟩	🟩	🟩	🟩	🟧	🟧
	19	7	LVDSRX3C_20N	🟩	🟩	🟩	🟩	⬛	⬛
	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	🟩	🟩	🟩	🟩	🟧	🟧
Lane 0	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 pinss 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		

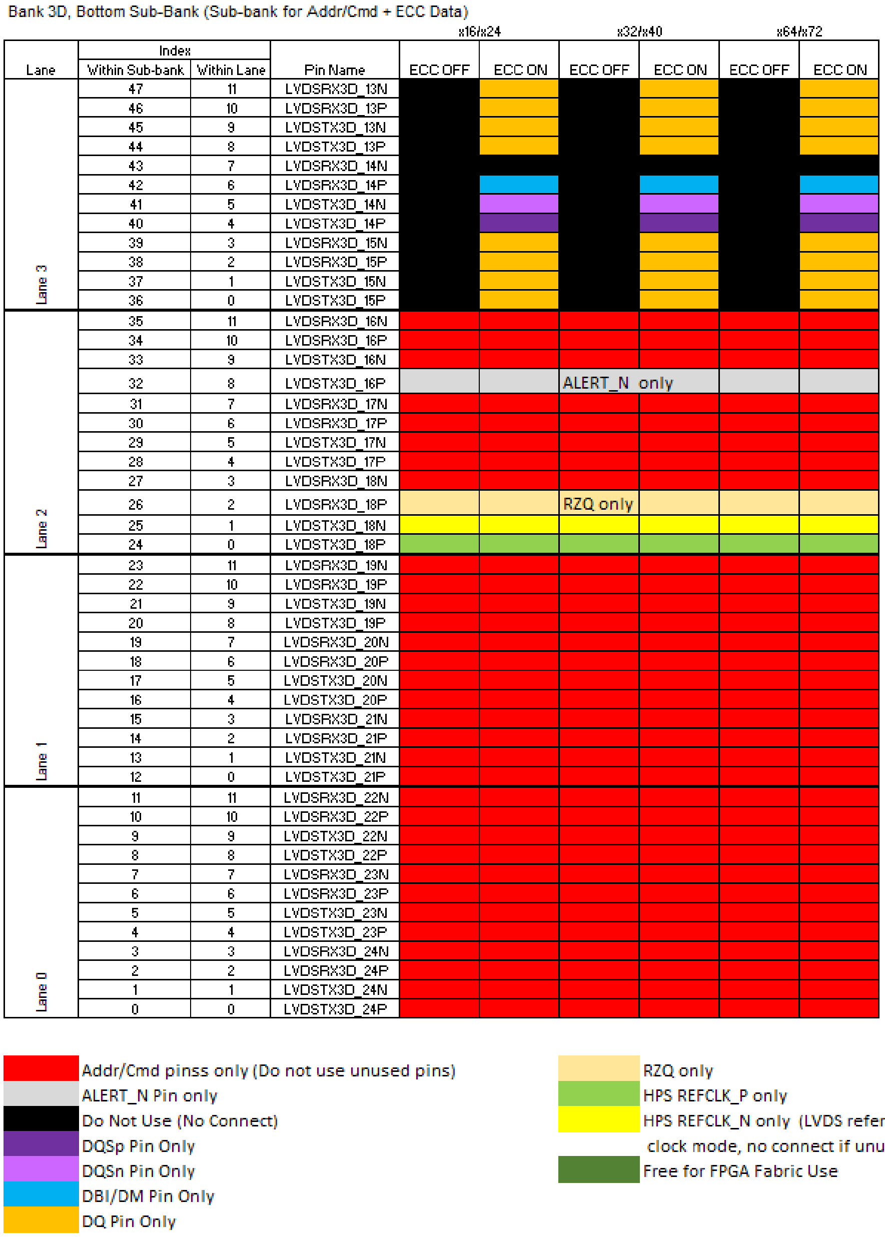
Figure 19. I/O Pin Usage Restrictions for Intel Agilex External Memory Interface with HPS (2 of 3)

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

	Index			x16/x24		x32/x40		x64/x72	
Lane	Within Sub-bank	Within Lane	Pin Name	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	LVDSRX3D_2N	⬛	⬛	⬛	⬛	⬛	⬛
	90	6	LVDSRX3D_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	LVDSRX3D_4N	⬛	⬛	🟧	🟧	🟧	🟧
	82	10	LVDSRX3D_4P	⬛	⬛	🟧	🟧	🟧	🟧
	81	9	LVDSTX3D_4N	⬛	⬛	🟧	🟧	🟧	🟧
	80	8	LVDSTX3D_4P	⬛	⬛	🟧	🟧	🟧	🟧
	79	7	LVDSRX3D_5N	⬛	⬛	⬛	⬛	⬛	⬛
	78	6	LVDSRX3D_5P	⬛	⬛	🟦	🟦	🟦	🟦
	77	5	LVDSTX3D_5N	⬛	⬛	🟣	🟣	🟣	🟣
	76	4	LVDSTX3D_5P	⬛	⬛	🟪	🟪	🟪	🟪
	75	3	LVDSRX3D_6N	⬛	⬛	🟧	🟧	🟧	🟧
	74	2	LVDSRX3D_6P	⬛	⬛	🟧	🟧	🟧	🟧
	73	1	LVDSTX3D_6N	⬛	⬛	🟧	🟧	🟧	🟧
	72	0	LVDSTX3D_6P	⬛	⬛	🟧	🟧	🟧	🟧
Lane 1	71	11	LVDSRX3D_7N	🟧	🟧	🟧	🟧	🟧	🟧
	70	10	LVDSRX3D_7P	🟧	🟧	🟧	🟧	🟧	🟧
	69	9	LVDSTX3D_7N	🟧	🟧	🟧	🟧	🟧	🟧
	68	8	LVDSTX3D_7P	🟧	🟧	🟧	🟧	🟧	🟧
	67	7	LVDSRX3D_8N	⬛	⬛	⬛	⬛	⬛	⬛
	66	6	LVDSRX3D_8P	🟦	🟦	🟦	🟦	🟦	🟦
	65	5	LVDSTX3D_8N	🟣	🟣	🟣	🟣	🟣	🟣
	64	4	LVDSTX3D_8P	🟪	🟪	🟪	🟪	🟪	🟪
	63	3	LVDSRX3D_9N	🟧	🟧	🟧	🟧	🟧	🟧
	62	2	LVDSRX3D_9P	🟧	🟧	🟧	🟧	🟧	🟧
	61	1	LVDSTX3D_9N	🟧	🟧	🟧	🟧	🟧	🟧
	60	0	LVDSTX3D_9P	🟧	🟧	🟧	🟧	🟧	🟧
Lane 0	59	11	LVDSRX3D_10N	🟧	🟧	🟧	🟧	🟧	🟧
	58	10	LVDSRX3D_10P	🟧	🟧	🟧	🟧	🟧	🟧
	57	9	LVDSTX3D_10N	🟧	🟧	🟧	🟧	🟧	🟧
	56	8	LVDSTX3D_10P	🟧	🟧	🟧	🟧	🟧	🟧
	55	7	LVDSRX3D_11N	⬛	⬛	⬛	⬛	⬛	⬛
	54	6	LVDSRX3D_11P	🟦	🟦	🟦	🟦	🟦	🟦
	53	5	LVDSTX3D_11N	🟣	🟣	🟣	🟣	🟣	🟣
	52	4	LVDSTX3D_11P	🟪	🟪	🟪	🟪	🟪	🟪
	51	3	LVDSRX3D_12N	🟧	🟧	🟧	🟧	🟧	🟧
	50	2	LVDSRX3D_12P	🟧	🟧	🟧	🟧	🟧	🟧
	49	1	LVDSTX3D_12N	🟧	🟧	🟧	🟧	🟧	🟧
	48	0	LVDSTX3D_12P	🟧	🟧	🟧	🟧	🟧	🟧

🟥	Addr/Cmd pinss 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		

Figure 20. I/O Pin Usage Restrictions for Intel Agilex External Memory Interface with HPS (3 of 3)



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

	Index			x16/x24		x32/x40		x64/x72	
Lane	Within Sub-bank	Within Lane	Pin Name	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_14N	⬛	🟣	⬛	🟣	⬛	🟣
	40	4	LVDSTX3D_14P	⬛	🟪	⬛	🟪	⬛	🟪
	39	3	LVDSRX3D_15N	⬛	🟧	⬛	🟧	⬛	🟧
	38	2	LVDSRX3D_15P	⬛	🟧	⬛	🟧	⬛	🟧
	37	1	LVDSTX3D_15N	⬛	🟧	⬛	🟧	⬛	🟧
	36	0	LVDSTX3D_15P	⬛	🟧	⬛	🟧	⬛	🟧
Lane 2	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 1	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 0	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	🟥	🟥	🟥	🟥	🟥	🟥

🟥	Addr/Cmd pinss 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 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 will cause 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 will cause 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.

4. Intel Agilex FPGA EMIF IP – End-User Signals

4.1. Intel Agilex EMIF IP Interface and Signal Descriptions

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

4.1.1. Intel Agilex EMIF IP Interfaces for DDR4

The interfaces in the Intel Agilex 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 11. Interfaces for DDR4

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

continued...

Interface Name	Interface Type	Description
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
ctrl_auto_precharge	Conduit	Controller auto-precharge interface
ctrl_user_priority	Conduit	Controller user-requested priority interface
ctrl_ecc_user_interrupt	Conduit	Controller ECC user interrupt interface
ctrl_ecc_readdataerror	Conduit	Controller ECC read data error indication interface
ctrl_mmr_slave	Avalon Memory-Mapped Slave	Controller MMR slave interface
hps_emif	Conduit	Conduit between Hard Processor Subsystem and memory interface

4.1.1.1. local_reset_req for DDR4

Local reset request. Output signal from local_reset_combiner

Table 12. Interface: local_reset_req

Interface type: Conduit

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

4.1.1.2. local_reset_done for DDR4

Local reset status. Input signal to the local_reset_combiner

Table 13. Interface: local_reset_done

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 14. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.1.4. pll_locked for DDR4

PLL locked signal

Table 15. Interface: pll_locked

Interface type: Conduit

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

4.1.1.5. pll_extra_clk_0 for DDR4

Additional core clock 0

Table 16. Interface: pll_extra_clk_0

Interface type: Clock Output

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

4.1.1.6. pll_extra_clk_1 for DDR4

Additional core clock 1

Table 17. Interface: pll_extra_clk_1

Interface type: Clock Output

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

4.1.1.7. pll_extra_clk_2 for DDR4

Additional core clock 2

Table 18. Interface: pll_extra_clk_2

Interface type: Clock Output

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

4.1.1.8. pll_extra_clk_3 for DDR4

Additional core clock 3

Table 19. Interface: pll_extra_clk_3

Interface type: Clock Output

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

4.1.1.9. oct for DDR4

On-Chip Termination (OCT) interface

Table 20. Interface: oct

Interface type: Conduit

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

4.1.1.10. mem for DDR4

Interface between FPGA and external memory

Table 21. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_ck	Output	CK clock
mem_ck_n	Output	CK clock (negative leg)
mem_a	Output	Address. Address bit A17 is defined only for the x4 configuration of 16 Gb SDRAM.
mem_ba	Output	Bank address
mem_bg	Output	Bank group

continued...

Port Name	Direction	Description
mem_cke	Output	Clock enable
mem_cs_n	Output	Chip select
mem_odt	Output	On-die termination
mem_reset_n	Output	Asynchronous reset
mem_act_n	Output	Activation command
mem_par	Output	Command and address parity
mem_dq	Bidirectional	Read/write data
mem_dbi_n	Bidirectional	Acts as either the data bus inversion pin, or the data mask pin, depending on configuration.
mem_dqs	Bidirectional	Data strobe
mem_dqs_n	Bidirectional	Data strobe (negative leg)
mem_alert_n	Input	Alert flag

4.1.1.11. status for DDR4

PHY calibration status interface

Table 22. Interface: status

Interface type: Conduit

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

4.1.1.12. afi_reset_n for DDR4

AFI reset interface

Table 23. Interface: afi_reset_n

Interface type: Reset Output

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

4.1.1.13. afi_clk for DDR4

AFI clock interface

Table 24. Interface: afi_clk

Interface type: Clock Output

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

4.1.1.14. afi_half_clk for DDR4

AFI half-rate clock interface

Table 25. Interface: afi_half_clk

Interface type: Clock Output

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

4.1.1.15. afi for DDR4

Altera PHY Interface (AFI)

Table 26. Interface: afi

Interface type: Conduit

Port Name	Direction	Description
afi_cal_success	Output	Signals calibration successful completion
afi_cal_fail	Output	Signals calibration failure
afi_cal_req	Input	When asserted, the interface is recalibrated
afi_rlat	Output	Latency in afi_clk cycles between read command and read data valid
afi_wlat	Output	Latency in afi_clk cycles between write command and write data valid
afi_addr	Input	Address
afi_ba	Input	Bank address
afi_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

continued...

Port Name	Direction	Description
afi_rdata_valid	Output	Asserted by the PHY to indicate that afi_rdata contains valid read data
afi_rrank	Input	Asserted by the controller to indicate which rank is being read from, to control shadow register switching
afi_wrank	Input	Asserted by the controller to indicate which rank is being written to, to control shadow register switching

4.1.1.16. emif_usr_reset_n for DDR4

User clock domain reset interface

Table 27. Interface: emif_usr_reset_n

Interface type: Reset Output

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

4.1.1.17. emif_usr_clk for DDR4

User clock interface

Table 28. Interface: emif_usr_clk

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

4.1.1.18. clks_sharing_master_out for DDR4

Core clocks sharing master interface

Table 29. Interface: clks_sharing_master_out

Interface type: Conduit

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

4.1.1.19. clks_sharing_slave_in for DDR4

Core clocks sharing slave input interface

Table 30. Interface: clks_sharing_slave_in

Interface type: Conduit

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

4.1.1.20. clks_sharing_slave_out for DDR4

Core clocks sharing slave output interface

Table 31. Interface: clks_sharing_slave_out

Interface type: Conduit

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

4.1.1.21. ctrl_amm for DDR4

Controller Avalon Memory-Mapped interface

Table 32. Interface: ctrl_amm

Interface type: Avalon Memory-Mapped Slave

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

4.1.1.22. ctrl_auto_precharge for DDR4

Controller auto-precharge interface

Table 33. Interface: ctrl_auto_precharge

Interface type: Conduit

Port Name	Direction	Description
ctrl_auto_precharge_req	Input	When asserted high along with a read or write request to the memory controller, indicates that the controller should close the currently opened page after the read or write burst. Ctrl_auto_precharge_req must remain asserted (along with the read or write request) until it is accepted (via the Avalon ready signal being asserted).

4.1.1.23. ctrl_user_priority for DDR4

Controller user-requested priority interface

Table 34. 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.

4.1.1.24. ctrl_ecc_user_interrupt for DDR4

Controller ECC user interrupt interface

Table 35. 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

4.1.1.25. ctrl_ecc_readdataerror for DDR4

Controller ECC read data error indication interface

Table 36. 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.

4.1.1.26. ctrl_mmr_slave for DDR4

Controller MMR slave interface

Table 37. 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

continued...

Port Name	Direction	Description
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

4.1.1.27. hps_emif for DDR4

Conduit between Hard Processor Subsystem and memory interface

Table 38. 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

4.2. Intel Agilex 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.

4.2.1. AFI Clock and Reset Signals

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

Table 39. 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 40. Address and Command Signals

Signal Name	Direction	Width	Description
afi_addr	Input	AFI_ADDR_WIDTH	Address.
afi_bg	Input	AFI_BANKGROUP_WIDTH	Bank group (DDR4 only).
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address.
afi_cke	Input	AFI_CLK_EN_WIDTH	Clock enable.
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs and LRDIMMs require a minimum of 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device data sheet for information about chip select signal width.)
afi_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 41. Write Data Signals

Signal Name	Direction	Width	Description
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. The mem_dm and mem_dbi 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 42. 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 43. Calibration Status Signals

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat *(1)*	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Note to Table:

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

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 44. 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 45. Shadow Register Management Signals

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

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

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

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

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

Additional Requirements for Shadow Register Support

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

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

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

4.3. Intel Agilex 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, half and quarter rate respectively.

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

Figure 21. AFI Address and Command Full-Rate

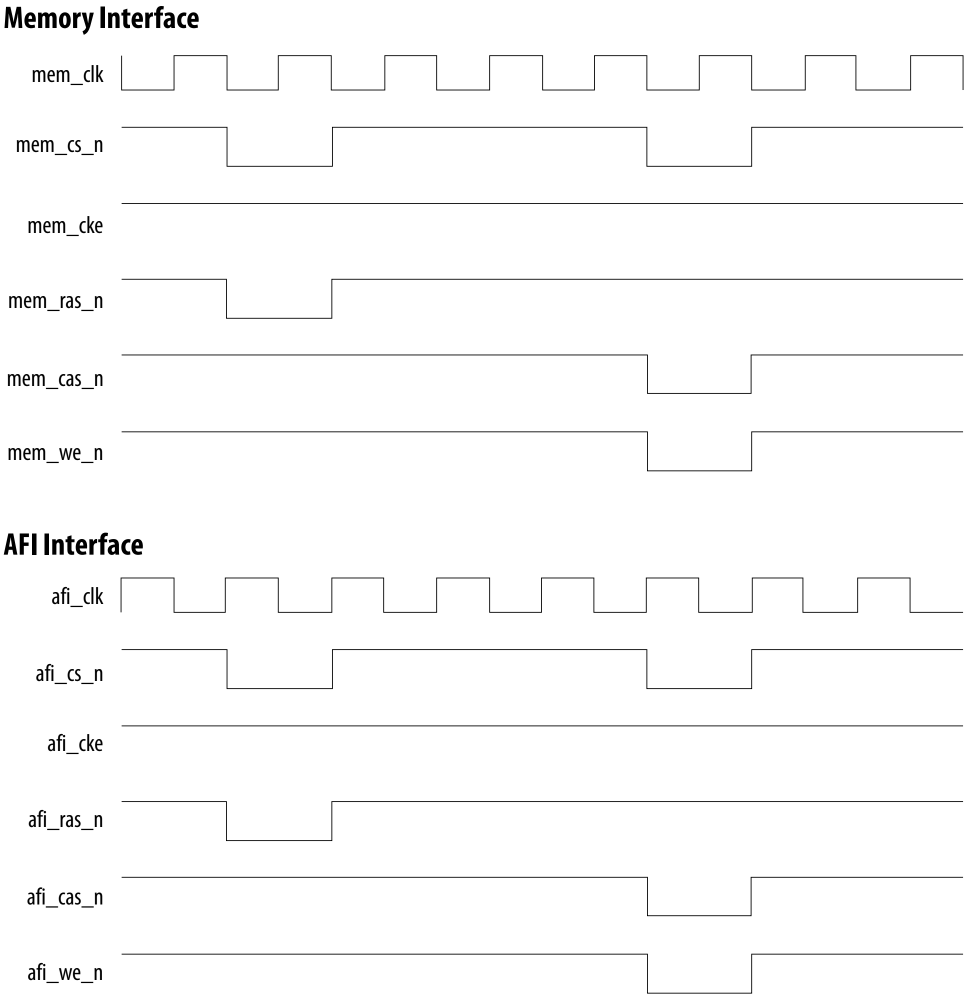


Figure 22. AFI Address and Command Half-Rate

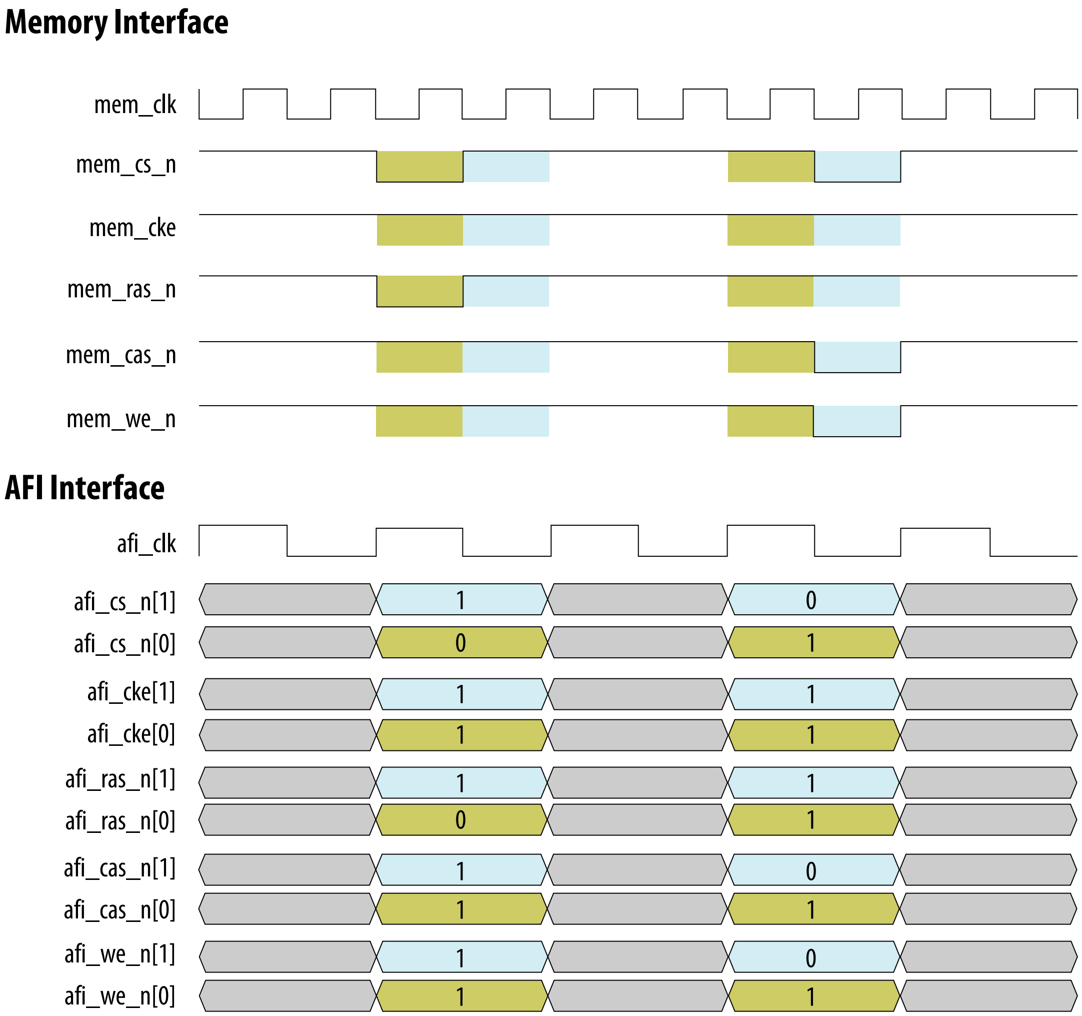
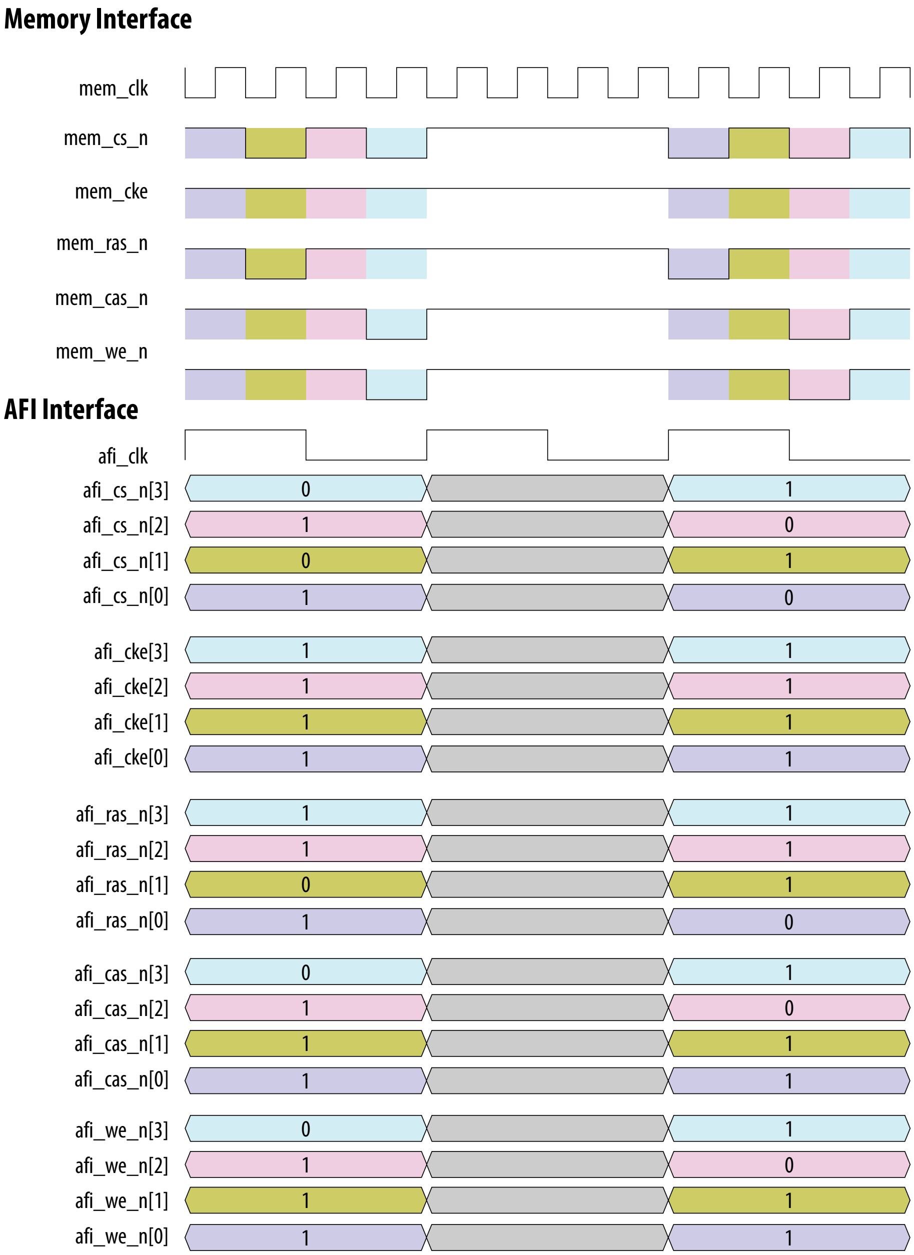


Figure 23. AFI Address and Command Quarter-Rate



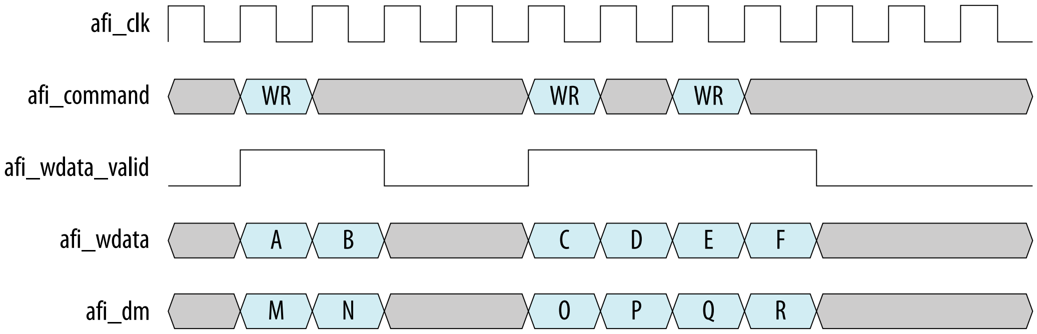
4.3.2. AFI Write Sequence Timing Diagrams

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

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

Write sequences with wlat=0

Figure 24. AFI Write Data Full-Rate, wlat=0



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

Figure 25. AFI Write Data Half-Rate, wlat=0

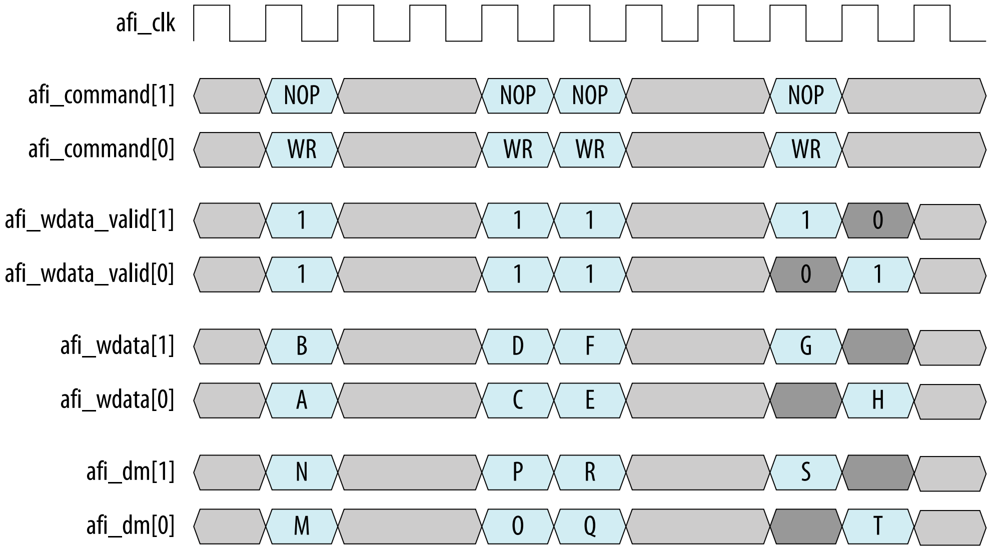
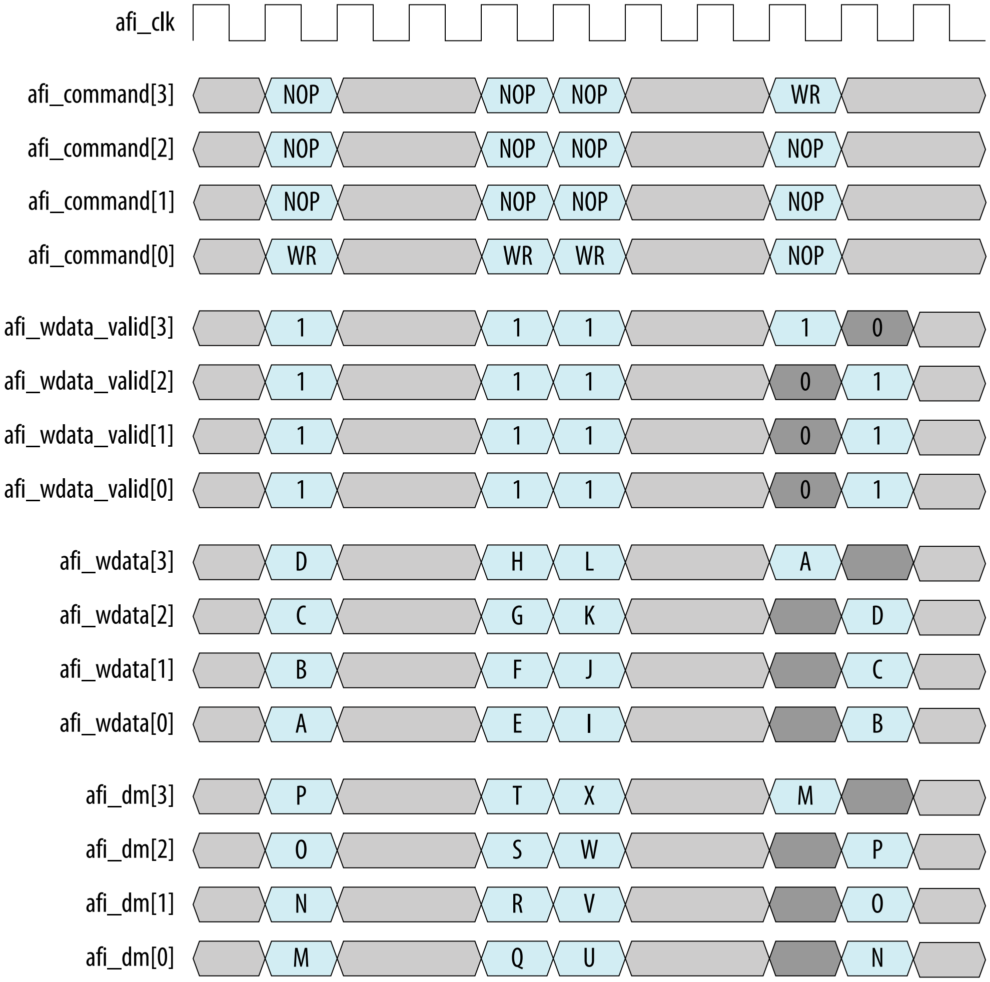


Figure 26. AFI Write Data Quarter-Rate, wlat=0



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 27. AFI Write Data Full-Rate, wlat=1

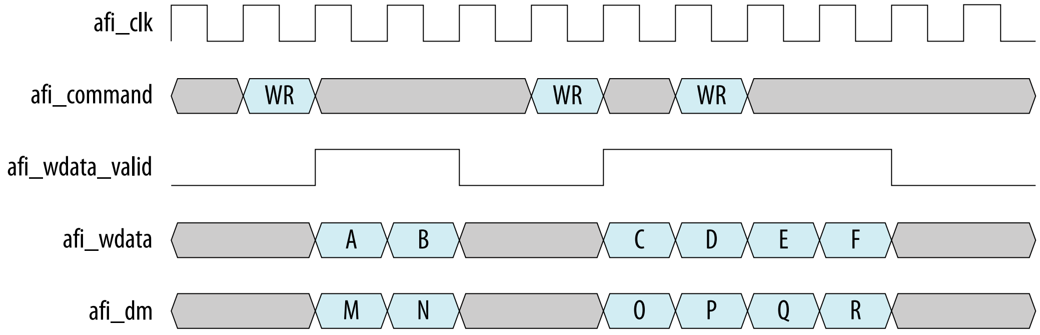


Figure 28. AFI Write Data Half-Rate, wlat=1

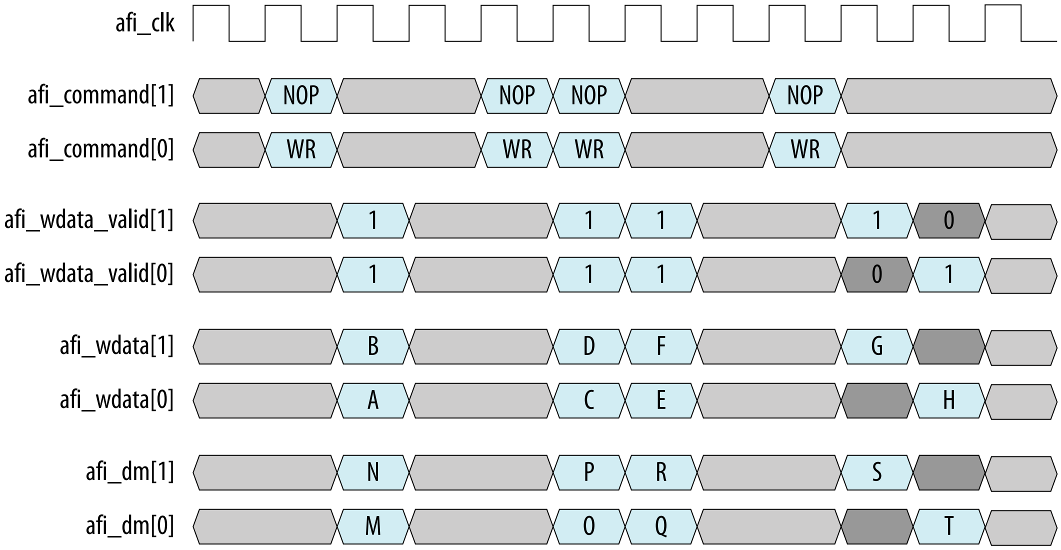
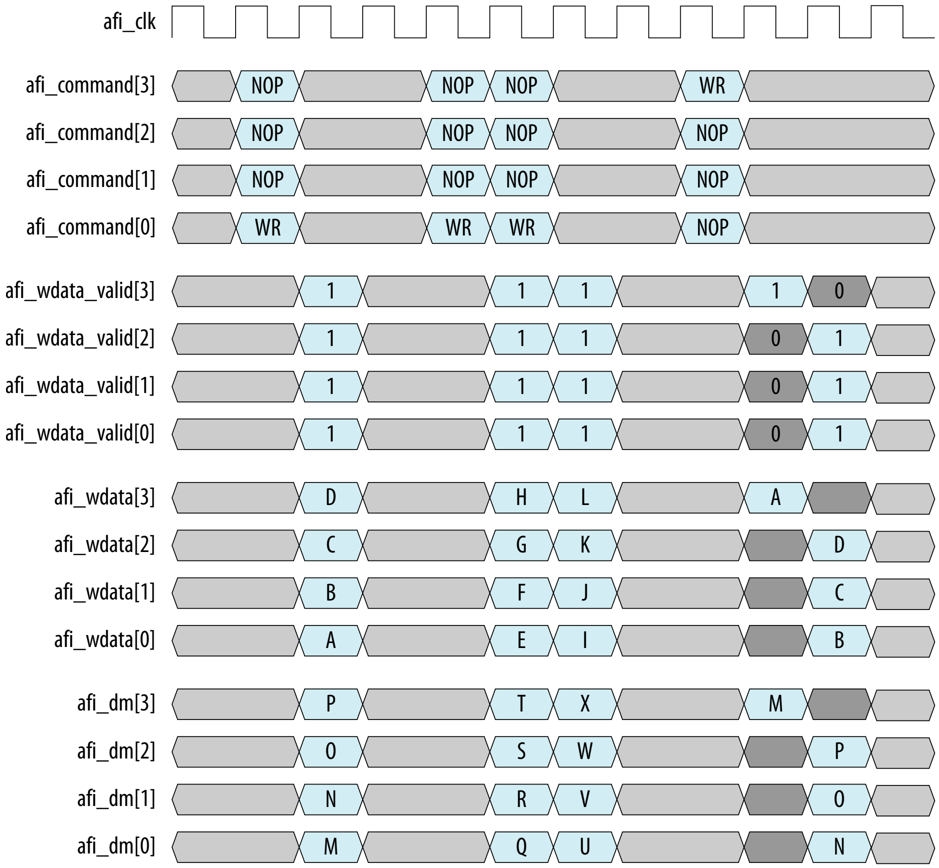


Figure 29. AFI Write Data Quarter-Rate, wlat=1



DQS burst

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

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

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

Figure 30. AFI DQS Burst Full-Rate, wlat=1

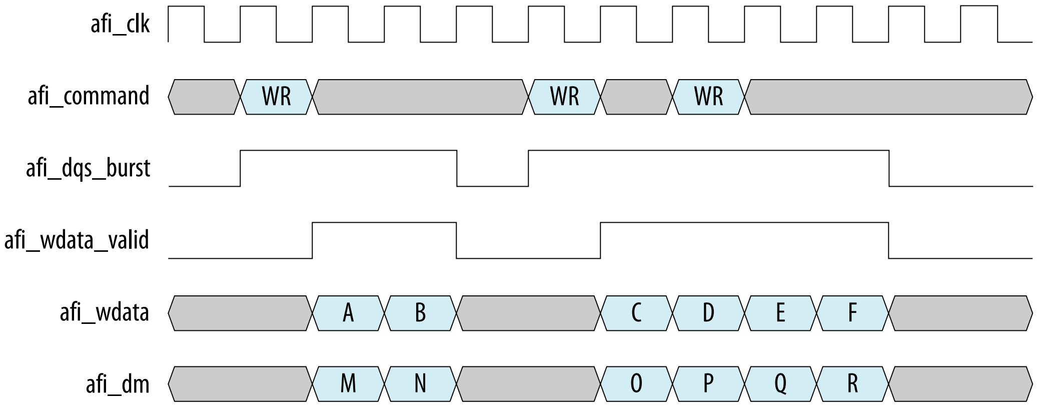


Figure 31. AFI DQS Burst Half-Rate, wlat=1

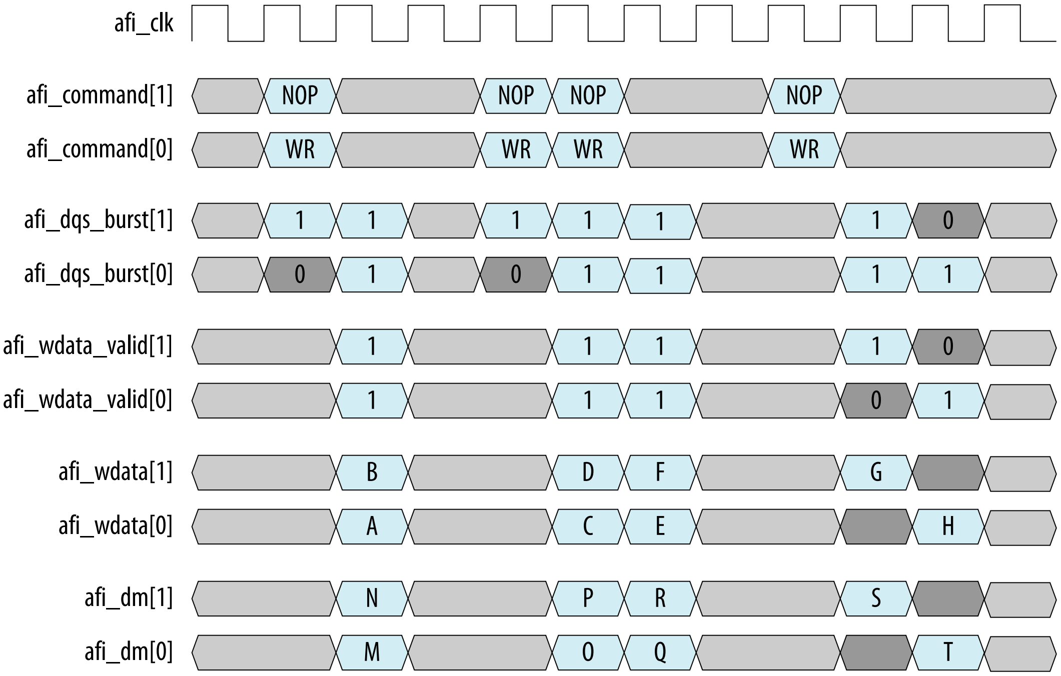
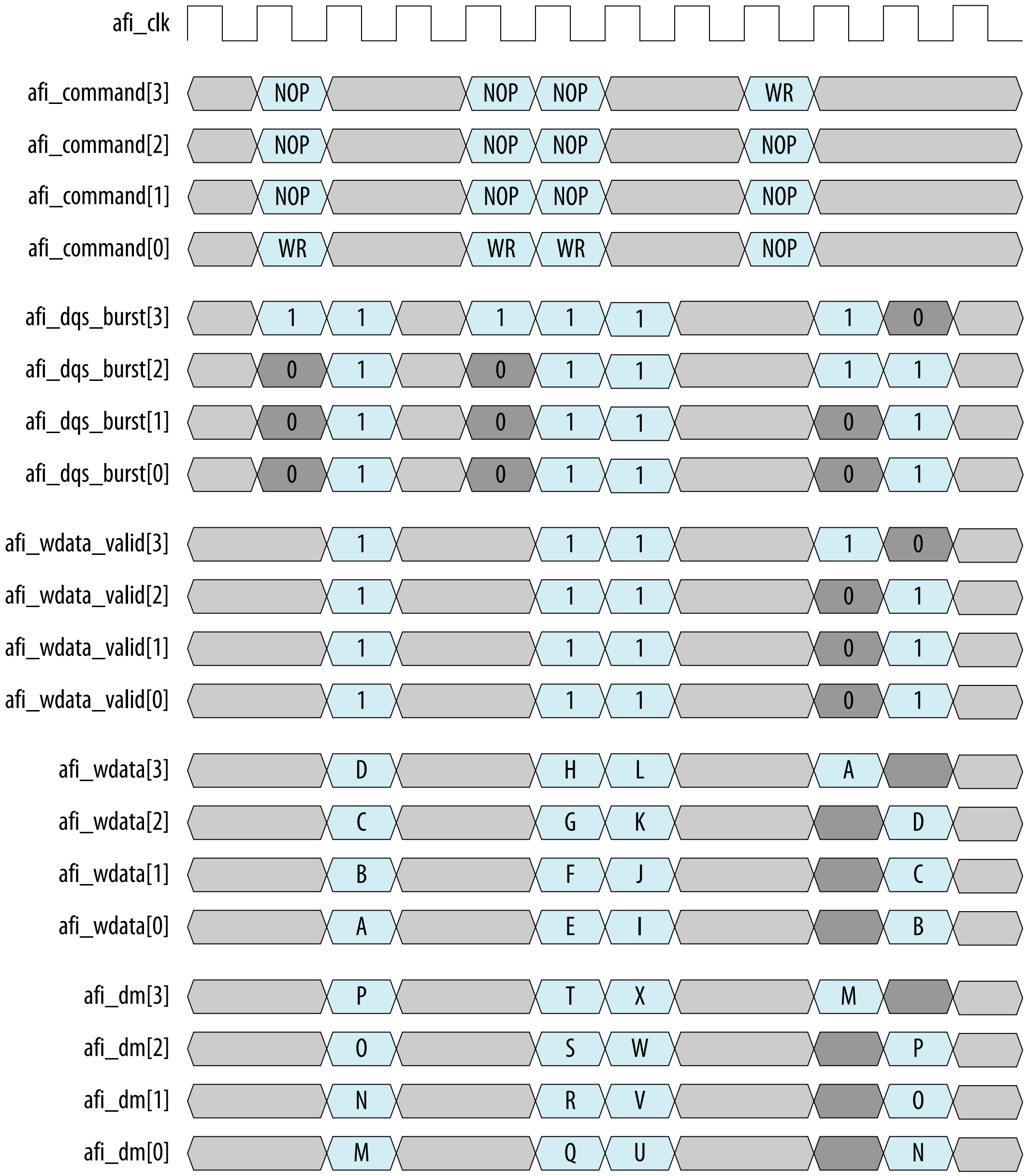


Figure 32. AFI DQS Burst Quarter-Rate, wlat=1


afi_clk
afi_command[3] NOP NOP NOP WR
afi_command[2] NOP NOP NOP NOP
afi_command[1] NOP NOP NOP NOP
afi_command[0] WR WR WR NOP
afi_dqs_burst[3] 1 1 1 1 1 1 0
afi_dqs_burst[2] 0 1 0 1 1 1 1
afi_dqs_burst[1] 0 1 0 1 1 0 1
afi_dqs_burst[0] 0 1 0 1 1 0 1
afi_wdata_valid[3] 1 1 1 1 0
afi_wdata_valid[2] 1 1 1 0 1
afi_wdata_valid[1] 1 1 1 0 1
afi_wdata_valid[0] 1 1 1 0 1
afi_wdata[3] D H L A
afi_wdata[2] C G K D
afi_wdata[1] B F J C
afi_wdata[0] A E I B
afi_dm[3] P T X M
afi_dm[2] O S W P
afi_dm[1] N R V O
afi_dm[0] M Q U N

Write data sequence with DBI (DDR4 and QDRIV only)

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

Write data sequence with CRC (DDR4 only)

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

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

Figure 33. AFI Write Data with CRC Half-Rate, wlat=2

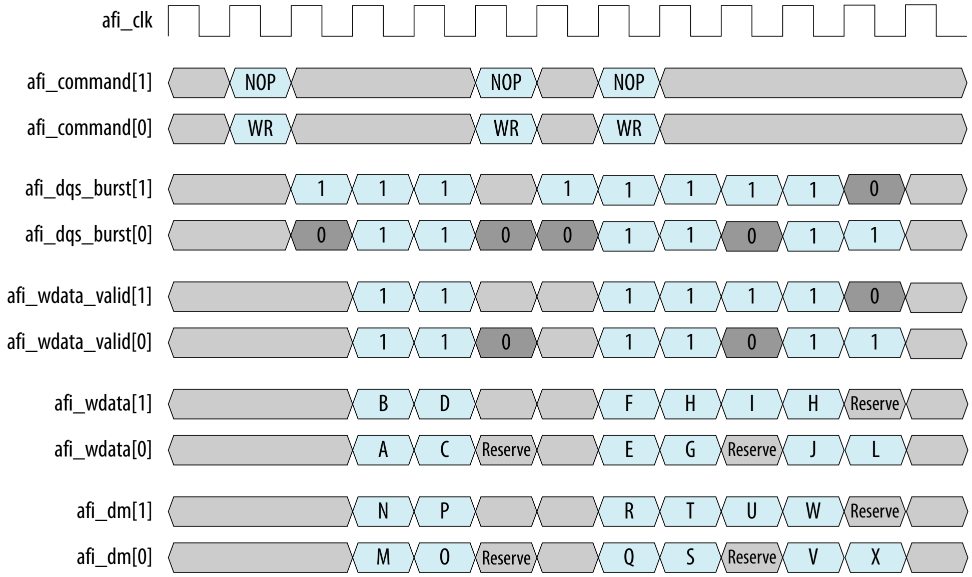
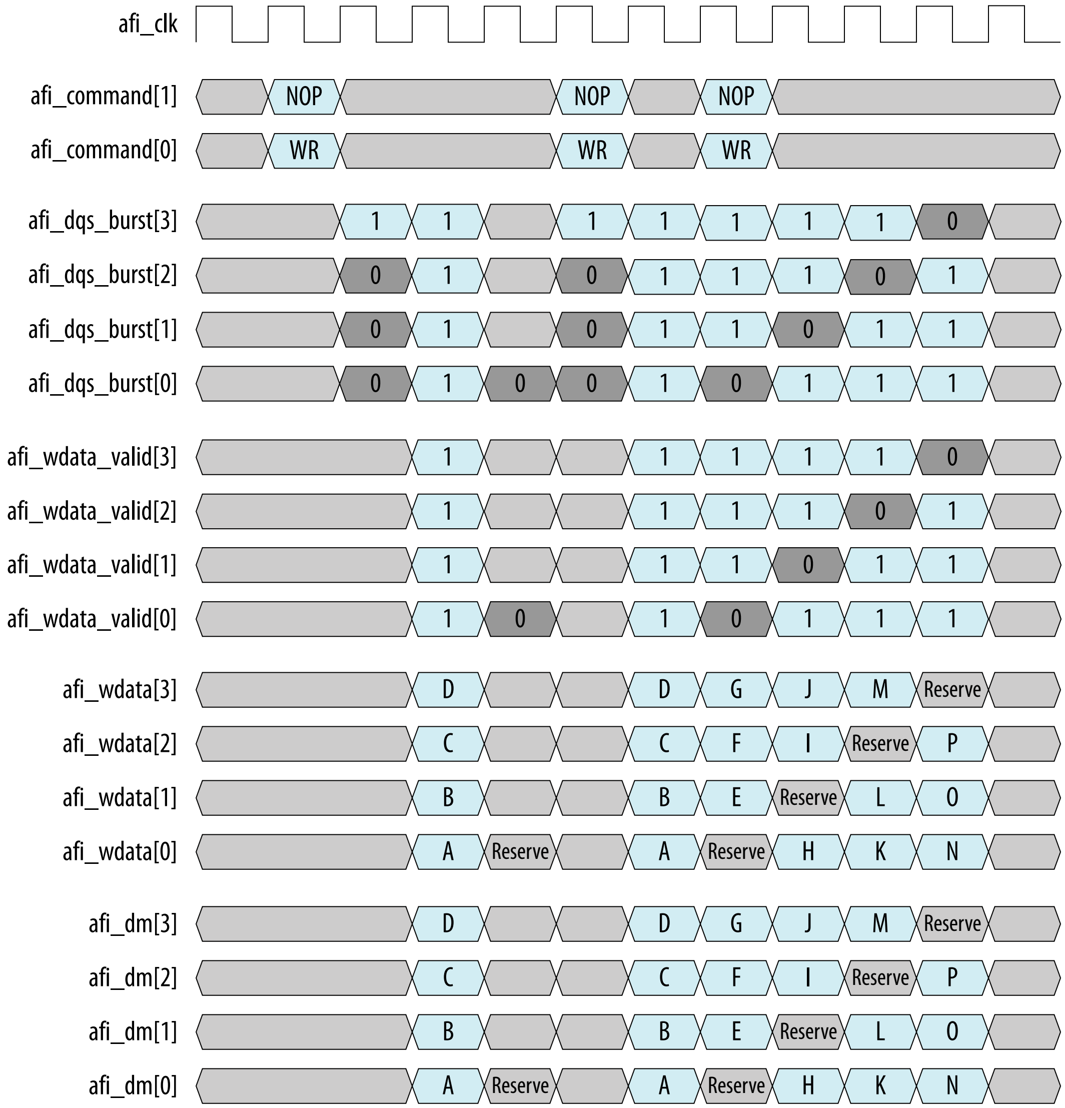


Figure 34. AFI Write Data with CRC Quarter-Rate, wlat=2


afi_clk
afi_command[1] NOP NOP NOP
afi_command[0] WR WR WR
afi_dqs_burst[3] 1 1 1 1 1 1 1 0
afi_dqs_burst[2] 0 1 0 1 1 1 0 1
afi_dqs_burst[1] 0 1 0 1 1 0 1 1
afi_dqs_burst[0] 0 1 0 0 1 0 1 1 1
afi_wdata_valid[3] 1 1 1 1 1 0
afi_wdata_valid[2] 1 1 1 1 0 1
afi_wdata_valid[1] 1 1 1 0 1 1
afi_wdata_valid[0] 1 0 1 0 1 1 1
afi_wdata[3] D D G J M Reserve
afi_wdata[2] C C F I Reserve P
afi_wdata[1] B B E Reserve L O
afi_wdata[0] A Reserve A Reserve H K N
afi_dm[3] D D G J M Reserve
afi_dm[2] C C F I Reserve P
afi_dm[1] B B E Reserve L O
afi_dm[0] A Reserve A Reserve H K N

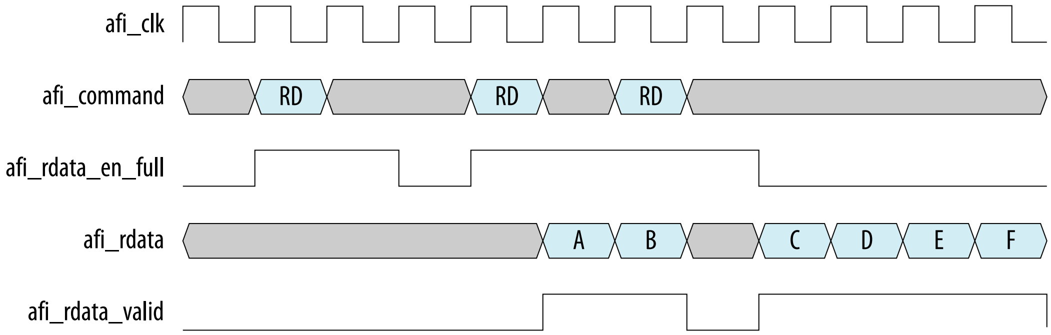
4.3.3. AFI Read Sequence Timing Diagrams

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

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

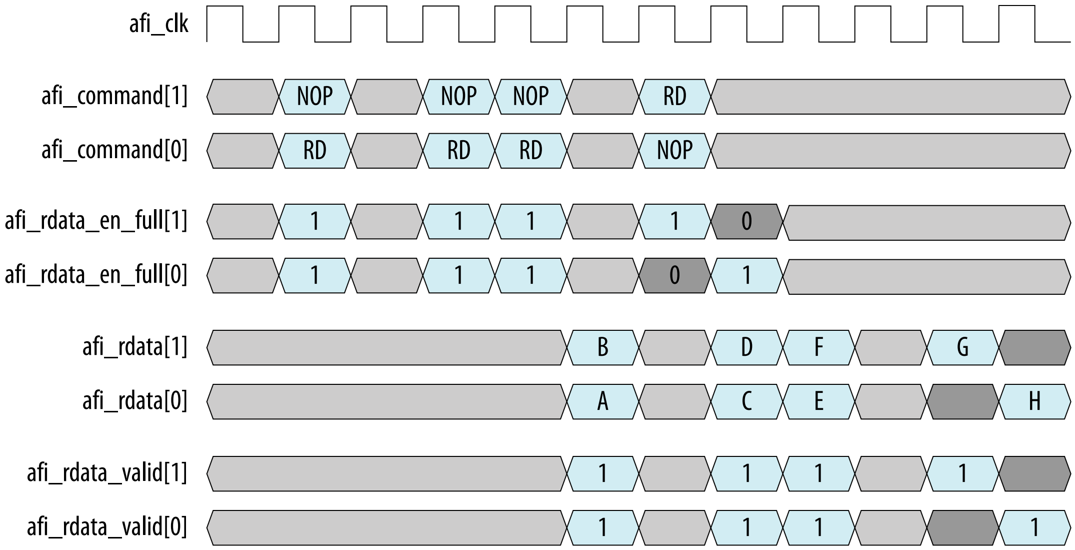
Aligned and unaligned access for read commands is similar to write commands; however, the `afi_rdata_en_full` signal must be sent on the same memory clock in a PHY clock as the read command. That is, if a read command is sent on the second memory clock in a PHY clock, `afi_rdata_en_full` must also be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

Figure 35. AFI Read Data Full-Rate



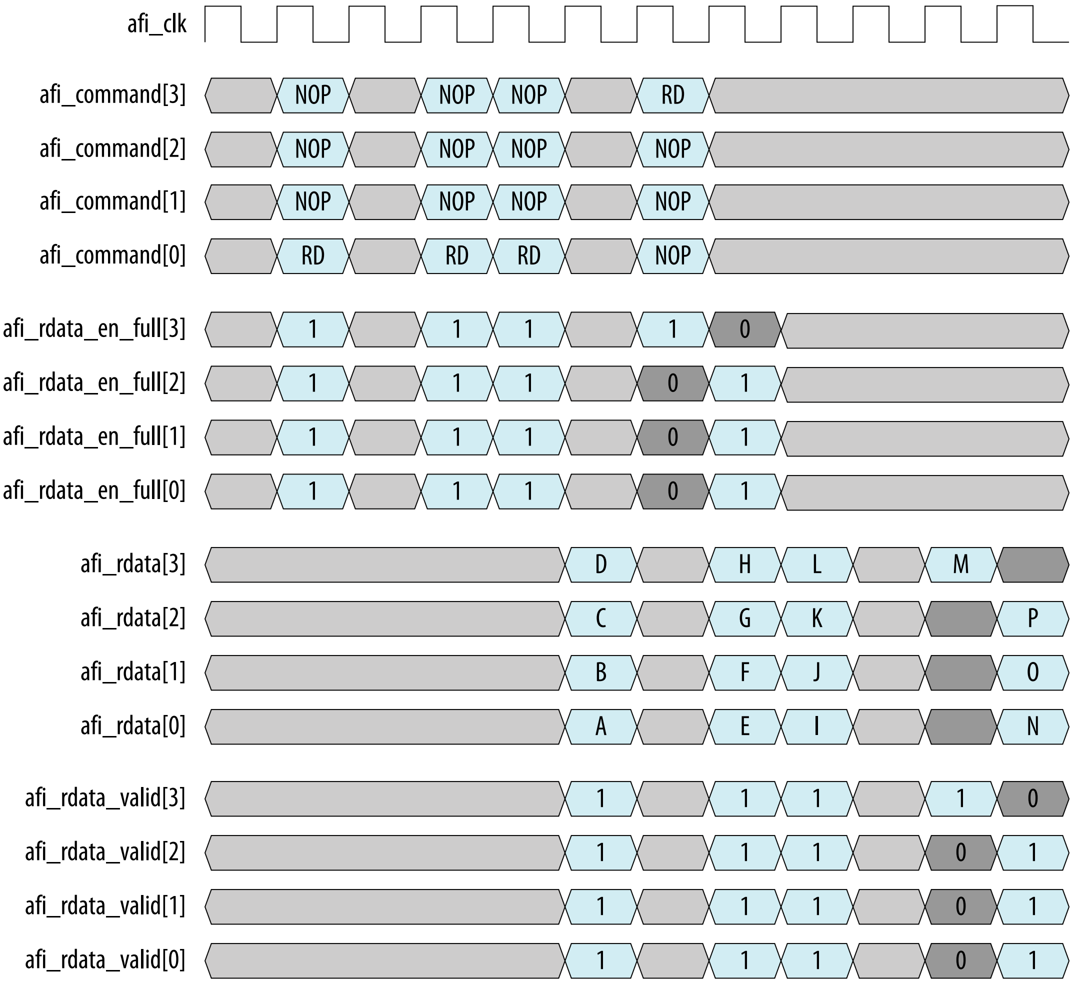
The following figure illustrates that the second and third reads require only the first and second half of data, respectively. The first three `read` commands are aligned accesses where they are issued on the LSB of `afi_command`. The fourth `read` command is unaligned access, where it is issued on a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

Figure 36. AFI Read Data Half-Rate



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

Figure 37. AFI Read Data Quarter-Rate



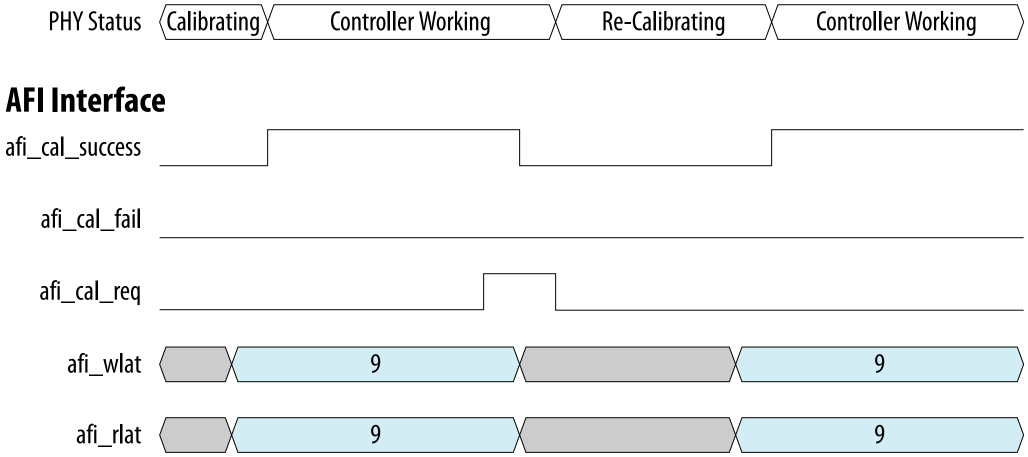
4.3.4. AFI Calibration Status Timing Diagram

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 38. Calibration



4.4. Intel Agilex 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.

Register Summary

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

continued...

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

Note: Addresses are in decimal format.

4.4.1. ctrlcfg0

address=10(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_mem_type	3	0	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	Read

continued...

Field	Bit High	Bit Low	Description	Access
			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.	
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	25	Reserved.	Read
cfg_ctrl_enable_dm	26	26	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc0_enable_dm	27	27	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc1_enable_dm	28	28	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc2_enable_dm	29	29	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc3_enable_dm	30	30	Set to 1 to enable DRAM operation if DM pins are connected.	Read

4.4.3. dramtiming0

address=20(32 bit)

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

4.4.4. caltiming0

address=31(32 bit)

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

4.4.5. caltiming1

address=32(32 bit)

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

4.4.6. caltiming2

address=33(32 bit)

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

4.4.7. caltiming3

address=34(32 bit)

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

4.4.8. caltiming4

address=35(32 bit)

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

4.4.9. caltiming9

address=40(32 bit)

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

4.4.10. dramaddrw

address=42(32 bit)

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

4.4.11. sideband0

address=43(32 bit)

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

4.4.12. sideband1

address=44(32 bit)

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

4.4.13. sideband4

address=47(32 bit)

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

4.4.14. sideband6

address=49(32 bit)

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

4.4.15. sideband7

address=50(32 bit)

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

4.4.16. sideband9

address=52(32 bit)

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

4.4.17. sideband11

address=54(32 bit)

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

4.4.18. sideband12

address=55(32 bit)

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

4.4.19. sideband13

address=56(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_opcode	31	0	Register Command Opcode. Information used for register command.	Read/Write
			DDR4	
			[26:24] C2:C0	
			[23] ACT	
			[22:21] BG1:BG0	
			[20] Reserved	
			[19:18] BA1:BA0	
			[17] A17	

continued...

Field	Bit High	Bit Low	Description	Access
			[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]	

4.4.20. sideband14

address=57(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_cid	3	1	DDR4 3DS Chip ID Refresh. When asserted, indicates logical rank chip ID for 3DS Refresh.	Read

4.4.21. dramsts

address=59(32 bit)

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

4.4.22. niosreserve0

address=68(32 bit)

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

4.4.23. niosreserve1

address=69(32 bit)

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

4.4.24. sideband16

address=79(32 bit)

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

4.4.25. ecc3: ECC Error and Interrupt Configuration

address=130(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_gen_sbe	0	0	A value of 1 enables the generate SBE feature. Generates a single bit error during the write process.	Read/Write
cfg_gen_dbe	1	1	A value of 1 enables the generate DBE feature. Generates a double bit error during the write process.	Read/Write
cfg_enable_intr	2	2	A value of 1 enables the interrupt feature. The interrupt signal notifies if an error condition occurs. The condition is configurable.	Read/Write
cfg_mask_sbe_intr	3	3	A value of 1 masks the interrupt signal when SBE occurs.	Read/Write

continued...

Field	Bit High	Bit Low	Description	Access
cfg_mask_dbe_intr	4	4	A value of 1 masks the interrupt signal when DBE occurs.	Read/Write
cfg_mask_corr_dropped_intr	5	5	A value of 1 masks the interrupt signal when the auto correction command can’t be scheduled, due to back-pressure (FIFO full).	Read/Write
cfg_mask_hmi_intr	6	6	A value of 1 masks the interrupt signal when the hard memory interface asserts an interrupt signal via the `hmi_interrupt` port.	Read/Write
cfg_clr_intr	7	7	Writing a vale of 1 to this self-clearing bit clears the interrupt signal, error status, and address.	Read/Write
Reserved	31	8	Reserved.	Read

4.4.26. ecc4: Status and Error Information

address=144(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_ecc_intr	0	0	Indicates the interrupt status; a value of 1 indicates an interrupt occurred.	Read
sts_sbe_error	1	1	Indicates the SBE status; a value of 1 indicates SBE occurred.	Read
sts_dbe_error	2	2	Indicates the DBE status; a value of 1 indicates DBE occurred.	Read
sts_corr_dropped	3	3	Indicates the status of correction command dropped; a value of 1 indicates correction command dropped.	Read
sts_sbe_count	7	4	Indicates the number of times SBE error has occurred. The counter will overflow.	Read
sts_dbe_count	11	8	Indicates the number of times DBE error has occurred. The counter will overflow.	Read
sts_corr_dropped_count	15	12	Indicates the number of times correction command has dropped. The counter will overflow.	Read
Reserved	31	16	Reserved.	Read

4.4.27. ecc5: Address of Most Recent SBE/DBE

address=145(32 bit)

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

4.4.28. ecc6: Address of Most Recent Correction Command Dropped

address=146(32 bit)

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

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

address=147(32 bit)

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

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

address=148(32 bit)

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

5. Intel Agilex 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:
  - Mentor Graphics* ModelSim
  - Mentor Graphics 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 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.

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

Table 46. Typical Simulation Times Using Intel Agilex EMIF IP

Calibration Mode/Run Time [1]	Estimated Simulation Time	
	Small Interface (×8 Single Rank)	Large Interface (×72 Quad Rank)
Skip • Skip calibration • Preloads calculated settings	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.

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 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 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.

To perform functional simulation for an Intel Agilex 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 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.

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 ModelSim* - Intel FPGA Edition simulator, which is supplied with the Intel Quartus Prime software. Submodules normally provided as cleartext SystemVerilog files are encrypted using IEEE Verilog HDL encryption for ModelSim - 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 Mentor Graphics.

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 Mentor Graphics 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 *ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim* 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 ModelSim - 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, Mentor Graphics, 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\vcsmx\vcsmx_setup.sh`
- `sim\ed_sim\synopsys\vcs\vcs_setup.sh`
- `sim\ed_sim\cadence\ncsim_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 *ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim* 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 Mentor Graphics 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 Mentor Graphics 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 Mentor Graphics simulator, refer to the *Simulating External Memory Interface IP With ModelSim* chapter in the *Intel Agilex 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.

6. Intel Agilex FPGA EMIF IP – DDR4 Support

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

6.1. Intel Agilex 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 EMIF IP DDR4 Parameters: General

Table 48. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_DDR4_CONFIG_ENUM)
Use clamshell layout	When clamshell layout is used, each rank requires two CS pins to configure the top and bottom memory chips separately. (Identifier: PHY_DDR4_USER_CLAMSHELL_EN)

Table 49. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_DDR4_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_DDR4_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_DDR4_USER_REF_CLK_FREQ_MHZ)

continued...

Display Name	Description
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_DDR4_REF_CLK_JITTER_PS)
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_DDR4_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_DDR4_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_DDR4_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 50. Group: General / Clocks / Additional Core Clocks

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

Table 51. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

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

Table 52. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

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

Table 53. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

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

Table 54. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

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

6.1.2. Intel Agilex 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 55. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_DDR4_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results.* (Identifier: PHY_DDR4_DEFAULT_IO)

Table 56. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR4_USER_AC_SLEW_RATE_ENUM)

Table 57. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR4_USER_CK_SLEW_RATE_ENUM)

Table 58. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_DATA_IN_MODE_ENUM)
Use recommended initial Vrefin	Specifies that the initial Vrefin setting is calculated automatically, to a reasonable value based on termination settings. (Identifier: PHY_DDR4_USER_AUTO_STARTING_VREFIN_EN)
Initial Vrefin	Specifies the **initial value for the reference voltage on the data pins(Vrefin)**. This value is entered as a percentage of the supply voltage level on the I/O pins. The specified value serves as a starting point and may be overridden by calibration to provide better timing margins. If you choose to **skip Vref calibration** (**Diagnostics** tab), this is the value that is used as the Vref for the interface. (Identifier: PHY_DDR4_USER_STARTING_VREFIN)

Table 59. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

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

6.1.3. Intel Agilex EMIF IP DDR4 Parameters: Memory

Table 60. Group: Memory / Topology

Display Name	Description
Memory format	Specifies the format of the external memory device. The following formats are supported: **Component** - a Discrete memory device; **UDIMM** - Unregistered/Unbuffered DIMM where address/control, clock, and data are unbuffered; **RDIMM** - Registered DIMM where address/control and clock are buffered; **LRDIMM** - Load Reduction DIMM where address/control, clock, and data are buffered. **LRDIMM** reduces the load to increase memory speed and supports higher densities than RDIMM; **SODIMM** - Small Outline DIMM is similar to UDIMM but smaller in size and is typically used for systems with limited space. Some memory protocols may not be available in all formats. (Identifier: MEM_DDR4_FORMAT_ENUM)
DQ width	Specifies the total number of data pins in the interface. (Identifier: MEM_DDR4_DQ_WIDTH)
DQ pins per DQS group	Specifies the total number of DQ pins per DQS group. (Identifier: MEM_DDR4_DQ_PER_DQS)
Number of clocks	Specifies the number of CK/CK# clock pairs exposed by the memory interface. Usually more than 1 pair is required for RDIMM/LRDIMM formats. The value of this parameter depends on the memory device selected; *refer to the data sheet for your memory device.* (Identifier: MEM_DDR4_CK_WIDTH)
Number of chip selects	Specifies the total number of chip selects in the interface, up to a maximum of 4. This parameter applies to **discrete components only**. (Identifier: MEM_DDR4_DISCRETE_CS_WIDTH)
Number of DIMMs	Total number of DIMMs. (Identifier: MEM_DDR4_NUM_OF_DIMMS)
Chip ID width	Specifies the number of chip ID pins. Only applicable to *registered and load-reduced DIMMs* that use 3DS/TSV memory devices. (Identifier: MEM_DDR4_CHIP_ID_WIDTH)
Number of physical ranks per DIMM	Number of ranks per DIMM. For LRDIMM, this represents the number of physical ranks on the DIMM behind the memory buffer (Identifier: MEM_DDR4_RANKS_PER_DIMM)
Row address width	Specifies the number of row address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available rows. (Identifier: MEM_DDR4_ROW_ADDR_WIDTH)
Column address width	Specifies the number of column address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available columns. (Identifier: MEM_DDR4_COL_ADDR_WIDTH)
Bank address width	Specifies the number of bank address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of bank address pins needed for access to all available banks. (Identifier: MEM_DDR4_BANK_ADDR_WIDTH)

continued...

Display Name	Description
Bank group width	Specifies the number of bank group pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of bank group pins needed for access to all available bank groups. (Identifier: MEM_DDR4_BANK_GROUP_WIDTH)
Data mask	Indicates whether the interface uses data mask (DM) pins. This feature allows specified portions of the data bus to be written to memory (not available in x4 mode). **One DM pin exists per DQS group.** (Identifier: MEM_DDR4_DM_EN)
Write DBI	Indicates whether the interface uses write data bus inversion (DBI). This feature provides **better signal integrity and write margin**. This feature is unavailable if Data Mask is enabled or in x4 mode. (Identifier: MEM_DDR4_WRITE_DBI)
Read DBI	Specifies whether the interface uses read data bus inversion (DBI). Enable this feature for **better signal integrity and read margin**. This feature is not available in x4 configurations. (Identifier: MEM_DDR4_READ_DBI)
Enable address mirroring for odd chip-selects	Enabling address mirroring for multi-CS discrete components. Typically used when components are arranged in a clamshell layout. (Identifier: MEM_DDR4_DISCRETE_MIRROR_ADDRESSING_EN)
Enable address mirroring for odd ranks	Enabling address mirroring for dual-rank or quad-rank DIMM. (Identifier: MEM_DDR4_MIRROR_ADDRESSING_EN)
ALERT# pin placement	Specifies placement for the `mem_alert_n` signal. If you select **"I/O Lane with Address/Command Pins"**, you can pick the I/O lane and pin index in the add/cmd bank with the subsequent drop down menus. If you select **"I/O Lane with DQS Group"**, you can specify the DQS group with which to place the `mem_alert_n` pin. If you select **"Automatically select a location"**, the IP automatically selects a pin for the `mem_alert_n` signal. If you select this option, no additional location constraints can be applied to the `mem_alert_n` pin, or a fitter error will result during compilation. For optimum signal integrity, you should choose **"I/O Lane with Address/Command Pins"**. For interfaces containing multiple memory devices, it is recommended to connect the ALERT# pins together to the ALERT# pin on the FPGA. (Identifier: MEM_DDR4_ALERT_N_PLACEMENT_ENUM)
DQS group of ALERT#	Select the DQS group with which the ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_DQS_GROUP)
Address/command I/O lane of ALERT#	Select the lane of the Address/Command I/O Tile where ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_AC_LANE)
Pin index of ALERT#	Select the pin of the Address/Command I/O Lane where ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_AC_PIN)

Table 61. Group: Memory / Latency and Burst

Display Name	Description
Memory CAS latency setting	Specifies the number of clock cycles between the read command and the availability of the first bit of output data at the memory device. Overall read latency equals the additive latency (AL) + the CAS latency (CL). Overall read latency depends on the memory device selected; *refer to the datasheet for your device.* (Identifier: MEM_DDR4_TCL)
Memory write CAS latency setting	Specifies the number of clock cycles from the release of internal write to the latching of the first data in at the memory device. This value depends on the memory device selected; *refer to the datasheet for your device.* (Identifier: MEM_DDR4_WTCL)
Memory additive CAS latency setting	Determines the posted CAS additive latency of the memory device. Enable this feature to **improve command and bus efficiency, and increase system bandwidth**. (Identifier: MEM_DDR4_ATCL_ENUM)

6.1.4. Intel Agilex EMIF IP DDR4 Parameters: Mem I/O

Table 62. Group: Mem I/O / Memory I/O Settings

Display Name	Description
Output drive strength setting	Specifies the output driver impedance setting at the memory device. To obtain optimum signal integrity performance, **select option based on board simulation results.** (Identifier: MEM_DDR4_DRV_STR_ENUM)
Dynamic ODT (Rtt_WR) value	Specifies the mode of the dynamic on-die termination (ODT) during writes to the memory device (used for **multi-rank configurations**). For optimum signal integrity performance, **select this option based on board simulation results**. (Identifier: MEM_DDR4_RTT_WR_ENUM)
ODT Rtt nominal value	Determines the nominal on-die termination value applied to the DRAM. The termination is applied any time that ODT is asserted. If you specify a different value for RTT_WR, that value takes precedence over the values mentioned here. For optimum signal integrity performance, **select your option based on board simulation results**. (Identifier: MEM_DDR4_RTT_NOM_ENUM)
RTT PARK	If set, the value is applied when the DRAM is not being written **AND** ODT is not asserted HIGH. (Identifier: MEM_DDR4_RTT_PARK)
RCD CA Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DA0..DA17, DBA0..DBA1, DBG0..DBG1, DACT_n, DC2, DPAR. This parameter determines the value of bits DA[1:0] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CA_IBT_ENUM)
RCD DCS[3:0]_n Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DCS[3:0]_n. This parameter determines the value of bits DA[3:2] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CS_IBT_ENUM)
RCD DCKE Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DCKE0, DCKE1. This parameter determines the value of bits DA[5:4] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CKE_IBT_ENUM)
RCD DODT Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DODT0, DODT1. This parameter determines the value of bits DA[7:6] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_ODT_IBT_ENUM)
DB Host Interface DQ RTT_NOM	Specifies the RTT_NOM setting for the host interface of the data buffer. Only "RTT_NOM disabled" is supported. This parameter determines the value of the control word BC00 of the data buffer. (Identifier: MEM_DDR4_DB_RTT_NOM_ENUM)
DB Host Interface DQ RTT_WR	Specifies the RTT_WR setting of the host interface of the data buffer. This parameter determines the value of the control word BC01 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_RTT_WR_ENUM)
DB Host Interface DQ RTT_PARK	Specifies the RTT_PARK setting for the host interface of the data buffer. This parameter determines the value of control word BC02 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_RTT_PARK_ENUM)

continued...

Display Name	Description
DB Host Interface DQ Driver	Specifies the driver impedance setting for the host interface of the data buffer. This parameter determines the value of the control word BC03 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_DQ_DRV_ENUM)
Use recommended initial VrefDQ value	Specifies to use the recommended initial VrefDQ value. This value is used as a starting point and may change after calibration. (Identifier: MEM_DDR4_DEFAULT_VREFOUT)
VrefDQ training value	VrefDQ training value. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_VALUE)
VrefDQ training range	VrefDQ training range. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_RANGE)

Table 63. Group: Mem I/O / RDIMM/LRDIMM Serial Presence Detect (SPD) Data

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

continued...

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

Table 64. Group: Mem I/O / ODT Activation

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

6.1.5. Intel Agilex 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 65. Group: Mem Timing / Parameters dependent on Speed Bin

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_DDR4_SPEEDBIN_ENUM)
tIS (base)	tIS (base) refers to the **setup time for the Address/Command/Control (A) bus** to the rising edge of CK. (Identifier: MEM_DDR4_TIS_PS)
tIS (base) AC level	tIS (base) AC level refers to the **voltage level which the address/ command signal must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_DDR4_TIS_AC_MV)
tIH (base)	tIH (base) refers to the **hold time for the Address/Command (A) bus** after the rising edge of CK. Depending on what AC level the user has chosen for a design, the hold margin can vary (this variance will be automatically determined when the user chooses the "**tIH (base) AC level**"). (Identifier: MEM_DDR4_TIH_PS)
tIH (base) DC level	tIH (base) DC level refers to the **voltage level which the address/ command signal must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_DDR4_TIH_DC_MV)

continued...

Display Name	Description
TdiVW_total	`TdiVW_total` describes the minimum horizontal width of the DQ eye opening required by the receiver (memory device/DIMM). It is measured in UI (1UI = half the memory clock period). (Identifier: MEM_DDR4_TDIVW_TOTAL_UI)
VdiVW_total	`VdiVW_total` describes the **Rx Mask voltage**, or the minimum vertical width of the DQ eye opening required by the receiver (memory device/DIMM). It is measured in mV. (Identifier: MEM_DDR4_VDIVW_TOTAL)
tDQSQ	tDQSQ describes the **latest valid transition of the associated DQ pins for a READ**. tDQSQ specifically refers to the DQS, DQS# to DQ skew. It is the length of time between the DQS, DQS# crossing to the last valid transition of the slowest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR4_TDQSQ_UI)
tQH	tQH specifies the **output hold time for the DQ in relation to DQS, DQS#**. It is the length of time between the DQS, DQS# crossing to the earliest invalid transition of the fastest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR4_TQH_UI)
tDVWp	Data valid window per device per pin (Identifier: MEM_DDR4_TDVWP_UI)
tDQSCK	tDQSCK describes the **skew between the memory clock (CK) and the input data strobes (DQS) used for reads**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSCK_PS)
tDQSS	tDQSS describes the **skew between the memory clock (CK) and the output data strobes used for writes**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSS_CYC)
tQSH	tQSH refers to the differential High Pulse Width, which is measured as a percentage of tCK. It is the **time during which the DQS is high for a read**. (Identifier: MEM_DDR4_TQSH_CYC)
tDSH	tDSH specifies the **write DQS hold time**. This is the time difference between the rising CK edge and the falling edge of DQS, measured as a percentage of tCK. (Identifier: MEM_DDR4_TDSH_CYC)
tDSS	tDSS describes the **time between the falling edge of DQS to the rising edge of the next CK transition**. (Identifier: MEM_DDR4_TDSS_CYC)
tWLS	tWLS describes the **write leveling setup time**. It is measured from the rising edge of CK to the rising edge of DQS. (Identifier: MEM_DDR4_TWLS_CYC)
tWLH	tWLH describes the **write leveling hold time**. It is measured from the rising edge of DQS to the rising edge of CK. (Identifier: MEM_DDR4_TWLH_CYC)
tINIT	tINIT describes the **time duration of the memory initialization after a device power-up**. After RESET_n is de-asserted, wait for another 500us until CKE becomes active. During this time, the DRAM will start internal initialization; this will be done independently of external clocks. (Identifier: MEM_DDR4_TINIT_US)
tMRD	The mode register set command cycle time, tMRD is the **minimum time period required between two MRS commands**. (Identifier: MEM_DDR4_TMRD_CK_CYC)
tRAS	tRAS describes the **activate to precharge duration**. A row cannot be deactivated until the tRAS time has been met. Therefore tRAS determines how long the memory has to wait after a activate command before a precharge command can be issued to close the row. (Identifier: MEM_DDR4_TRAS_NS)

continued...

Display Name	Description
tRCD	tRCD, **row command delay**, describes **the active to read/write time**. It is the amount of delay between the activation of a row through the RAS command and the access to the data through the CAS command. (Identifier: MEM_DDR4_TRCD_NS)
tRP	tRP refers to the **Precharge (PRE) command period**. It describes how long it takes for the memory to disable access to a row by precharging and before it is ready to activate a different row. (Identifier: MEM_DDR4_TRP_NS)
tWR	tWR refers to the **Write Recovery time**. It specifies the amount of clock cycles needed to complete a write before a precharge command can be issued. (Identifier: MEM_DDR4_TWR_NS)

Table 66. Group: Mem Timing / Parameters dependent on Speed Bin, Operating Frequency, and Page Size

Display Name	Description
tRRD_S	tRRD_S refers to the **Activate to Activate Command Period (short)**. It is the minimum time interval between two activate commands to the **different bank groups**. For 3DS devices, this parameter is the same as tRRD_S_slr (i.e. tRRD_S within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRRD_S_CYC)
tRRD_L	tRRD_L refers to the **Activate to Activate Command Period (long)**. It is the minimum time interval (measured in memory clock cycles) between two activate commands to the **same bank group**. For 3DS devices, this parameter is the same as tRRD_L_slr (i.e. tRRD_L within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRRD_L_CYC)
tRRD_dlr	tRRD_dlr refers to the **Activate to Activate Command Period to Different Logical Ranks**. It is the minimum time interval (measured in memory clock cycles) between two activate commands to different logical ranks within a 3DS DDR4 device. (Identifier: MEM_DDR4_TRRD_DLR_CYC)
tFAW	tFAW refers to the **four activate window time**. It describes the period of time during which only four banks can be active. For 3DS devices, this parameter is the same as tFAW_slr (i.e. tFAW within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TFAW_NS)
tFAW_dlr	tFAW_dlr refers to the **four activate window to different logical ranks**. It describes the period of time during which only four banks can be active across all logical ranks within a 3DS DDR4 device. (Identifier: MEM_DDR4_TFAW_DLR_CYC)
tCCD_S	tCCD_S refers to the **CAS_n-to-CAS_n delay (short)**. It is the minimum time interval between two read/write (CAS) commands to **different bank groups**. (Identifier: MEM_DDR4_TCCD_S_CYC)
tCCD_L	tCCD_L refers to the **CAS_n-to-CAS_n delay (long)**. It is the minimum time interval between two read/write (CAS) commands to the **same bank group**. (Identifier: MEM_DDR4_TCCD_L_CYC)
tWTR_S	tWTR_S or Write Timing Parameter refers to the **Write to Read period for different bank groups**. It describes the delay from start of internal write transaction to internal read command, for accesses to the different bank group. The delay is measured from the first rising memory clock edge after the last write data is received to the rising memory clock edge when a read command is received. (Identifier: MEM_DDR4_TWTR_S_CYC)
tWTR_L	tWTR_L or Write Timing Parameter refers to the **Write to Read period for the same bank group**. It describes the delay from start of internal write transaction to internal read command, for accesses to the same bank group. The delay is measured from the first rising memory clock edge after the last write data is received to the rising memory clock edge when a read command is received. (Identifier: MEM_DDR4_TWTR_L_CYC)

Table 67. Group: Mem Timing / Parameters dependent on Density and Temperature

Display Name	Description
tRFC	tRFC refers to the **Refresh Cycle Time**. It is the amount of delay after a refresh command before an activate command can be accepted by the memory. This parameter is dependent on the memory density and is necessary for proper hardware functionality. For 3DS devices, this parameter is the same as tRFC_slr (i.e. tRFC within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRFC_NS)
tRFC_dlr	tRFC_dlr refers to the **Refresh Cycle Time to different logical rank**. It is the amount of delay after a refresh command to one logical rank before an activate command can be accepted by another logical rank within a 3DS DDR4 device. This parameter is dependent on the memory density and is necessary for proper hardware functionality. (Identifier: MEM_DDR4_TRFC_DLR_NS)
tREFI	tREFI refers to the **average periodic refresh interval**. It is the maximum amount of time the memory can tolerate in between each refresh command (Identifier: MEM_DDR4_TREFI_US)

6.1.6. Intel Agilex EMIF IP DDR4 Parameters: Controller

Table 68. Group: Controller / Low Power Mode

Display Name	Description
Enable Auto Power-Down	Enable this parameter to have the controller automatically place the memory device into power-down mode after a specified number of idle controller clock cycles. The idle wait time is configurable. **All ranks must be idle to enter auto power-down.** (Identifier: CTRL_DDR4_AUTO_POWER_DOWN_EN)
Auto Power-Down Cycles	Specifies the number of idle controller cycles after which the memory device is placed into power-down mode. You can configure the idle waiting time. The supported range for number of cycles is from 1 to 65534. (Identifier: CTRL_DDR4_AUTO_POWER_DOWN_CYCS)

Table 69. Group: Controller / Efficiency

Display Name	Description
Enable User Refresh Control	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)
Enable Auto-Precharge Control	Select this parameter to enable the auto-precharge control on the controller top level. If you assert the auto-precharge control signal while requesting a read or write burst, you can specify whether the controller should close (auto-precharge) the currently open page at the end of the read or write burst, potentially making a future access to a different page of the same bank faster. (Identifier: CTRL_DDR4_AUTO_PRECHARGE_EN)
Address Ordering	Controls the mapping between Avalon addresses and memory device addresses. **By changing the value of this parameter, you can change the mappings between the Avalon-MM address and the DRAM address.** (*CS* = chip select, *CID* = chip ID in 3DS/TSV devices, *BG* = bank group address, *Bank* = bank address, *Row* = row address, *Col* = column address) (Identifier: CTRL_DDR4_ADDR_ORDER_ENUM)
Enable Reordering	Enable this parameter to allow the controller to perform command and data reordering. **Reordering can improve efficiency by reducing bus turnaround time and row/bank switching time.** Data reordering allows

continued...

Display Name	Description
	the single-port memory controller to change the order of read and write commands to achieve highest efficiency. Command reordering allows the controller to issue bank management commands early based on incoming patterns, so that the desired row in memory is already open when the command reaches the memory interface. *For more information, refer to the Data Reordering topic in the EMIF Handbook.* (Identifier: CTRL_DDR4_REORDER_EN)
Starvation limit for each command	Specifies the **number of commands that can be served before a waiting command is served**. The controller employs a counter to ensure that all requests are served after a pre-defined interval -- this ensures that low priority requests are not ignored, when doing data reordering for efficiency. The valid range for this parameter is from 1 to 63. *For more information, refer to the Starvation Control topic in the EMIF Handbook.* (Identifier: CTRL_DDR4_STARVE_LIMIT)
Enable Command Priority Control	Select this parameter to enable user-requested command priority control on the controller top level. This parameter instructs the controller to treat a read or write request as high-priority. The controller attempts to fill high-priority requests sooner, to reduce latency. **Connect this interface to the conduit of your logic block that determines when the external memory interface IP treats the read or write request as a high-priority command.** (Identifier: CTRL_DDR4_USER_PRIORITY_EN)

Table 70. Group: Controller / Configuration, Status and Error Handling

Display Name	Description
Enable Memory-Mapped Configuration and Status Register (MMR) Interface	Enable this parameter to change or read memory timing parameters, memory address size, mode register settings, controller status, and request sideband operations. (Identifier: CTRL_DDR4_MMR_EN)
Enable Error Detection and Correction Logic with ECC	Enables error-correction code (ECC) for **single-bit error correction and double-bit error detection**. Your memory interface must have a width of 16, 24, 40, or 72 bits to use ECC. *ECC is implemented as soft logic.* (Identifier: CTRL_DDR4_ECC_EN)
Enable Auto Error Correction to External Memory	Specifies that the controller automatically schedule and perform a write back to the external memory when a single-bit error is detected. Regardless of whether the option is enabled or disabled, the ECC feature always corrects single-bit errors before returning the read data to user logic. (Identifier: CTRL_DDR4_ECC_AUTO_CORRECTION_EN)
Enable ctrl_ecc_readdataerror signal to indicate uncorrectable data errors	Select this option to enable the `ctrl_ecc_readdataerror` signal on the controller top level. The signal has the same timing as the read data valid signal of the Controller Avalon Memory-Mapped interface, and is asserted high to indicate that the read data returned by the Controller in the same cycle contains errors uncorrectable by the ECC logic. (Identifier: CTRL_DDR4_ECC_READDATAERROR_EN)

Table 71. Group: Controller / Data Bus Turnaround Time

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

continued...

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

6.1.7. Intel Agilex EMIF IP DDR4 Parameters: Diagnostics

Table 72. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process. Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero. *If you enable this parameter, the interface still performs some memory initialization before starting normal operations.* Abstract PHY is supported with skip calibration. (Identifier: DIAG_DDR4_SIM_CAL_MODE_ENUM)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_DDR4_SIM_VERBOSE)

Group: Diagnostics / Calibration Debug Options

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

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic.

continued...

Display Name	Description
	If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only if the EMIF Debug Toolkit or On-Chip Debug Port is enabled.** *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_DDR4_INTERFACE_ID)
Skip address/command leveling calibration	Specifies to skip the address/command leveling stage during calibration. Address/command leveling attempts to center the memory clock edge against CS# by adjusting delay elements inside the PHY, and then applying the same delay offset to the rest of the address and command pins. (Identifier: DIAG_DDR4_SKIP_CA_LEVEL)
Skip address/command deskew calibration	Specifies to skip the address/command deskew calibration stage. Address/command deskew performs per-bit deskew for the address and command pins. (Identifier: DIAG_DDR4_SKIP_CA_DESKEW)
Skip VREF calibration	Specifies to skip the VREF stage of calibration. **Enable this parameter for debug purposes only**; generally, you should include the VREF calibration stage during normal operation. (Identifier: DIAG_DDR4_SKIP_VREF_CAL)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 73. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_DDR4_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_DDR4_EX_DESIGN_ISSP_EN)

Table 74. Group: Diagnostics / Traffic Generator

Display Name	Description
Use configurable Avalon traffic generator 2.0	This option allows users to add the new configurable Avalon traffic generator to the example design. (Identifier: DIAG_DDR4_USE_TG_AVL_2)
Bypass the default traffic pattern	Specifies that the controller/interface bypass the traffic generator 2.0 default pattern after reset. If you do not enable this parameter, the traffic generator does not assert a pass or fail status until the generator is configured and signaled to start by its Avalon configuration interface. (Identifier: DIAG_DDR4_BYPASS_DEFAULT_PATTERN)
Bypass the user-configured traffic stage	Specifies that the controller/interface bypass the user-configured traffic generator's pattern after reset. If you do not enable this parameter, the traffic generator does not assert a pass or fail status until the generator is configured and signaled to start by its Avalon configuration interface. Configuration can be done by connecting to the traffic generator via the EMIF Debug Toolkit, or by using custom logic connected to the Avalon-MM configuration slave port on the traffic generator. Configuration can also be simulated using the example testbench provided in the `altera_emif_avl_tg_2_tb.sv` file. (Identifier: DIAG_DDR4_BYPASS_USER_STAGE)
Run diagnostic on infinite test duration	Specifies that the traffic generator run indefinitely until the first error is detected. (Identifier: DIAG_DDR4_INFI_TG2_ERR_TEST)
TG2 Configuration Interface Mode	Specifies the connectivity of an Avalon slave interface for use by the TG Configuration Toolkit or user core logic. If you set this parameter to "**Export**", an Avalon slave interface named "`tg_cfg`" is exported from the IP. If you select "**JTAG**", a JTAG Avalon Master Endpoint is connected to the configuration interface, allowing the core to be accessed by the TG Configuration Toolkit. (Identifier: DIAG_DDR4_EXPORT_TG_CFG_AVALON_SLAVE)

Table 75. Group: Diagnostics / Performance

Display Name	Description
Enable Efficiency Monitor	Adds an Efficiency Monitor component to the Avalon-MM interface of the memory controller, allowing you to view efficiency statistics of the interface. You can access the efficiency statistics using the EMIF Debug Toolkit. (Identifier: DIAG_DDR4_EFFICIENCY_MONITOR)

Table 76. Group: Diagnostics / Miscellaneous

Display Name	Description
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

6.1.8. Intel Agilex EMIF IP DDR4 Parameters: Example Designs

Table 77. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_DDR4_SEL_DESIGN)

Table 78. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_DDR4_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a `make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_DDR4_GEN_SYNTH)

Table 79. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_DDR4_HDL_FORMAT)

6.2. Intel Agilex 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.

6.2.1. Intel Agilex FPGA EMIF IP Interface Pins

Any I/O banks that do not support transceiver operations in Intel Agilex 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 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.

6.2.1.1. 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, 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.

6.2.1.2. 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 80. UDIMM, RDIMM, and LRDIMM Pin Options for DDR4

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Data	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}
Data Mask	DM#/ DBI#[8:0] (1)	DM#/ DBI#[8:0](1)	DM#/ DBI#[8:0](1)	DM#/ DBI#[8:0](1)	—	—
Data Strobe	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]
Address	BA[1:0], BG[1:0], A[16:0] - 4GB: A[14:0] 8GB: A[15:0] 16GB: A[16:0] (2)	BA[1:0], BG[1:0], A[16:0] - 8GB: A[14:0] 16GB: A[15:0] 32GB: A[16:0] (2)	BA[1:0], BG[1:0], x8: A[16:0] - 4GB: A[14:0] 8GB: A[15:0] 16GB: A[16:0] (2) 32GB: A[17:0] (3)	BA[1:0], BG[1:0],x8: A[16:0] x4: A[17:0] - 8GB: A[14:0] 16GB: A[15:0] 32GB: A[16:0] (2) 64GB: A[17:0] (3)	BA[1:0], BG[1:0], A[17:0] - 16GB: A[15:0] 32GB: A[16:0] (2) 64GB: A[17:0] (3)	BA[1:0], BG[1:0], A[17:0] - 32GB: A[15:0] 64GB: A[16:0] (2) 128GB: A[17:0] (3)
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:

1. DM/DBI pins are available only for DIMMs constructed using x8 or greater components.
2. This density requires 4Gb x4 or 2Gb x8 DRAM components.
3. This density requires 8Gb x4 DRAM components.
4. This table assumes a single slot configuration. The Intel Agilex 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 cse, 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.

6.2.1.3. 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 devices, consult the EMIF Device Selector on 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*.

6.2.2. Intel Agilex FPGA EMIF IP Resources

The Intel Agilex FPGA memory interface IP uses several FPGA resources to implement the memory interface.

6.2.2.1. OCT

You require an OCT calibration block if you are using an Intel Agilex 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*.

6.2.2.2. 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*.

6.2.3. Pin Guidelines for Intel Agilex FPGA EMIF IP

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

Intel Agilex 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 Figure 6 on page 14, Figure 7 on page 14, and Figure 39 on page 107, and to the I/O Pin Count Tables located in the Intel Agilex General Purpose I/O and LVDS SERDES User Guide.

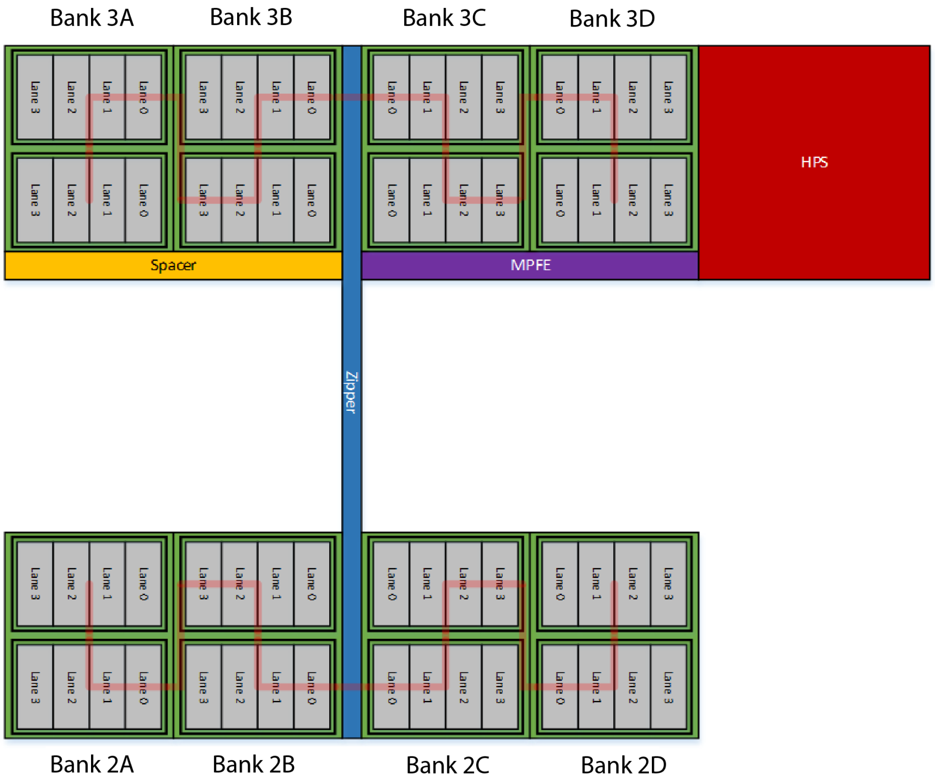
  In general, you can assume that I/O banks are adjacent within an I/O edge, unless the I/O bank is not bonded out on the package (indicated by the presence of the " - " symbol in the I/O table), or if the I/O bank does not contain 96 pins, indicating that it is only partially bonded out. If an I/O bank 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).

6.2.3.1. Intel Agilex FPGA EMIF IP Banks

Before you select pins for your Intel Agilex 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 FPGA with all banks bonded out to pins.

Figure 39.



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 FPGA EMIF IP Product Architecture chapter.

6.2.3.2. General Guidelines

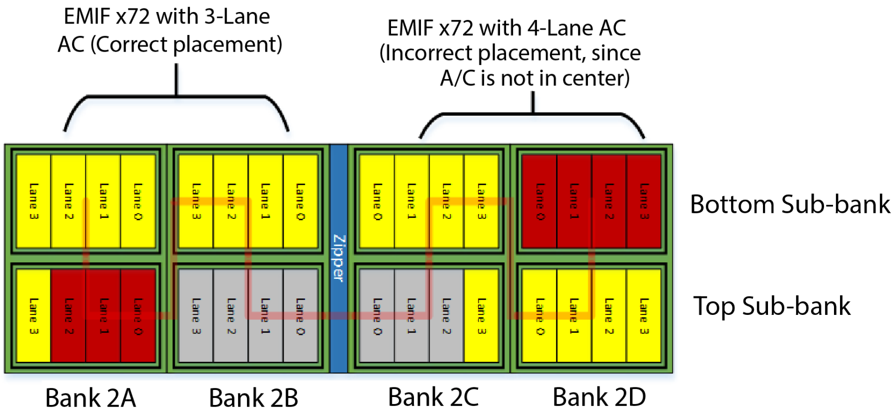
You should follow the recommended guidelines when performing pin placement for all external memory interface pins targeting Intel Agilex devices, whether you are using the hard memory controller or your own solution.

Note:

- EMIF IP pin-out requirements for the Intel Agilex 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, RLDRAM*x*, and QDR*x* are not supported with HPS.

Observe the following general guidelines when placing pins for your Intel Agilex 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.
   - If two interfaces share the same I/O 48 sub-block within the I/O 96 block, then both interfaces must use the same PLL frequency and the same reference clock.
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.
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:

   
   EMIF x72 with 3-Lane AC (Correct placement)
EMIF x72 with 4-Lane AC (Incorrect placement, since A/C is not in center)
Lane 3
Lane 2
Lane 1
Lane 0
Lane 0
Lane 1
Lane 2
Lane 3
Zipper
Bottom Sub-bank
Top Sub-bank
Bank 2A
Bank 2B
Bank 2C
Bank 2D

   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 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.

6.2.3.3. 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.

6.2.3.4. 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.

6.2.3.5. 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 ± 25% (± 90°) of the positive edge of the SDRAM clock input. Therefore, you should generate the CK and CK# signals using the DDR registers in the IOE to match with the DQS signal and reduce any variations across process, voltage, and temperature. The positive edge of the SDRAM clock, CK, is aligned with the DQS write to satisfy t_{DQSS}.

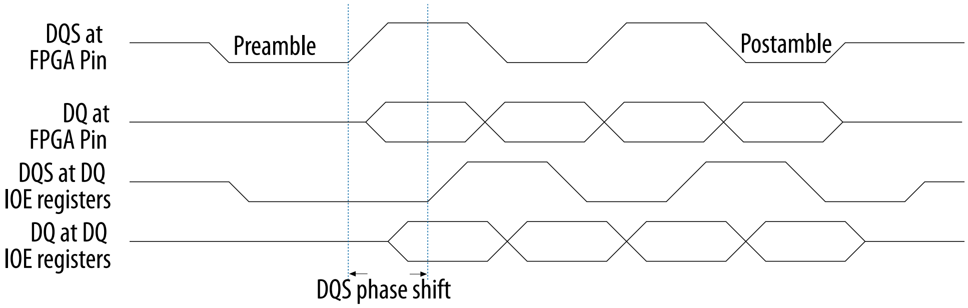
6.2.3.6. Data, Data Strobes, DM/DBI, and Optional ECC Signals

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

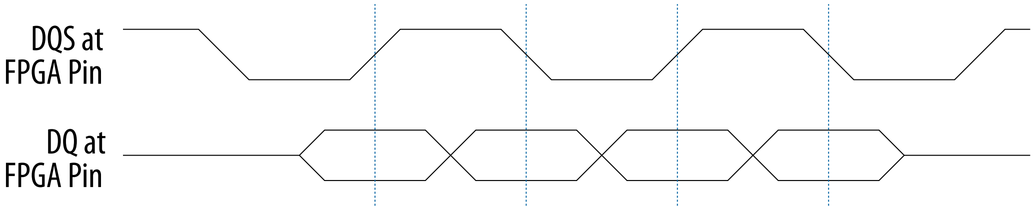
The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by –90 degrees during a write operation to center align the DQ and DQS signals. The PHY IP delays the 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 40. 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 41. 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 interfaces, the DM/DBI pins do not need to be paired with a DQ pin.

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 interfaces, DM/DBI (for DDR4) do not need to be paired with a DQ pin.

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

6.3. 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

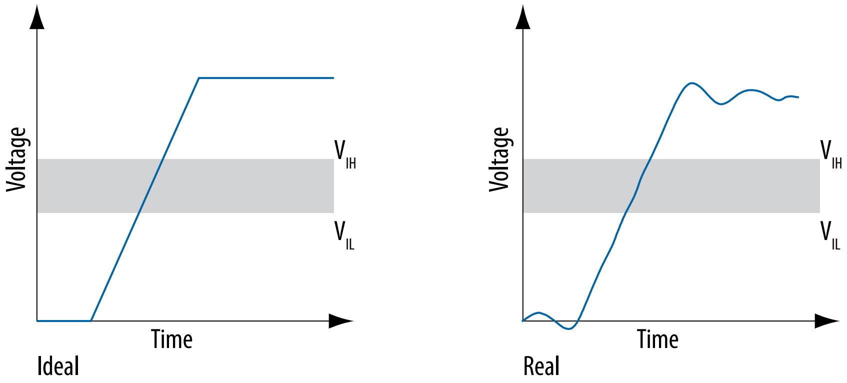
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

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 42. Ideal and Real Signal at the Receiver



6.3.1. Terminations for DDR4 with Intel Agilex Devices

The following topics describe considerations specific to DDR4 external memory interface protocols on Intel Agilex devices.

6.3.1.1. Dynamic On-Chip Termination (OCT)

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

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

6.3.1.2. 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.

6.3.1.3. Choosing Terminations on Intel Agilex 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.

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.

6.3.1.4. On-Chip Termination Recommendations for Intel Agilex 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.

6.3.2. Design Layout Guidelines

Contact Intel FPGA customer support for detailed information on external memory board design layout guidelines.

7. Intel Agilex FPGA EMIF IP – Timing Closure

This chapter describes timing analysis and optimization techniques that you can use to achieve timing closure.

7.1. 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.

7.1.1. Timing Analysis

Timing analysis of Intel Agilex EMIF IP is somewhat simpler than that of earlier device families, because Intel Agilex devices have more hardened blocks and fewer soft logic registers to be analyzed, because most are user logic registers.

Your Intel Agilex 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.

7.1.1.1. 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`.

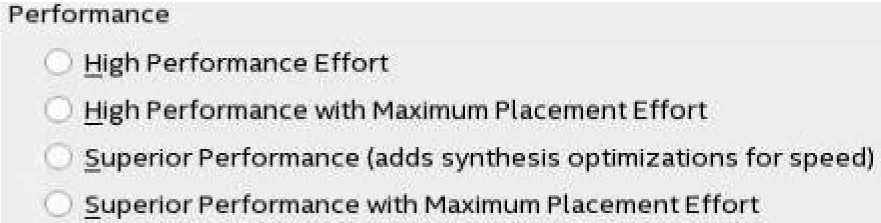
7.1.1.2. I/O Timing

I/O timing information for Intel Agilex EMIF IP will be available in a future release.

7.2. Optimizing Timing

The Intel Quartus Prime software offers several advanced features that you can use to assist in meeting core timing requirements.

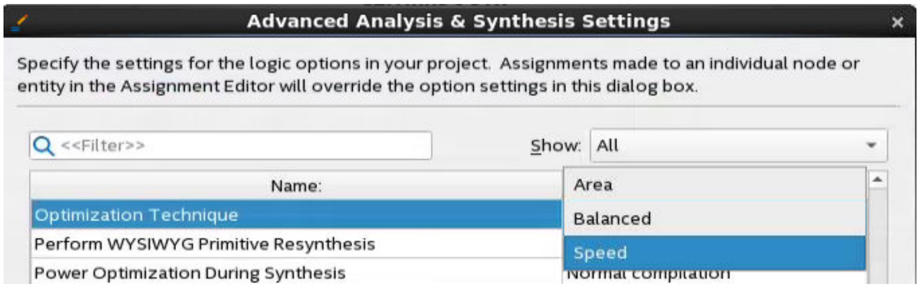
1. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings**. Under **Optimization mode**, select one of the **Performance** options.



Performance

○ High Performance Effort
○ High Performance with Maximum Placement Effort
○ Superior Performance (adds synthesis optimizations for speed)
○ Superior Performance with Maximum Placement Effort

2. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings ➤ Advanced Settings (Synthesis).** For **Optimization Technique**, select **Speed**.



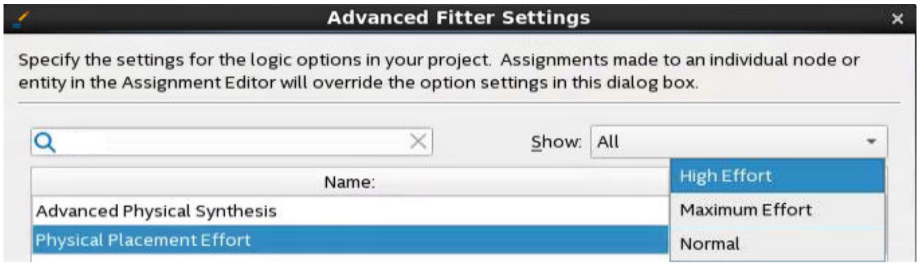
Advanced Analysis & Synthesis Settings

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

<<Filter>> Show: All

Name:	
Optimization Technique	Speed
Perform WYSIWYG Primitive Resynthesis	
Power Optimization During Synthesis	Normal compilation

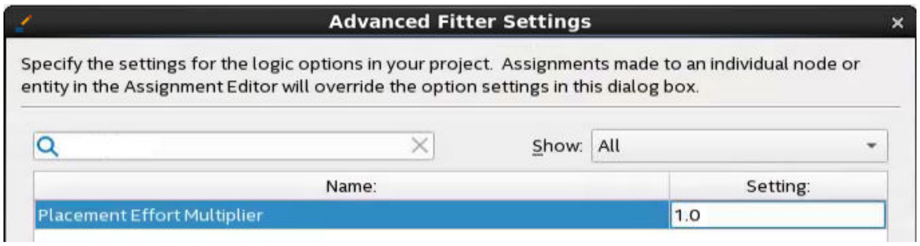
3. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings ➤ Advanced Settings (Fitter).** For **Physical Placement Effort**, select **High Effort** or **Maximum Effort**. The High and Maximum effort settings take additional compilation time to further optimize placement.



Advanced Fitter Settings

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

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.



Advanced Fitter Settings

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Name:	Setting:
Placement Effort Multiplier	1.0

8. Intel Agilex 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.

8.1. 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.

8.2. 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.

8.3. 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.

8.4. 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
- Starvation Control

- Command Reordering
- Bandwidth
- Enable Command Priority Control

The following sections discuss these methods in detail.

8.4.1. Auto-Precharge Commands

The auto-precharge read and write commands allow you to indicate to the memory device that a given read or write command is the last access to the currently opened row.

The memory device automatically closes or auto-precharges the page that is currently being accessed, so that the next access to the same bank is faster. The Auto-Precharge command is useful when you want to perform fast random memory accesses.

The Timer Bank Pool (TBP) block supports the dynamic page policy, where, depending on user input, 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.

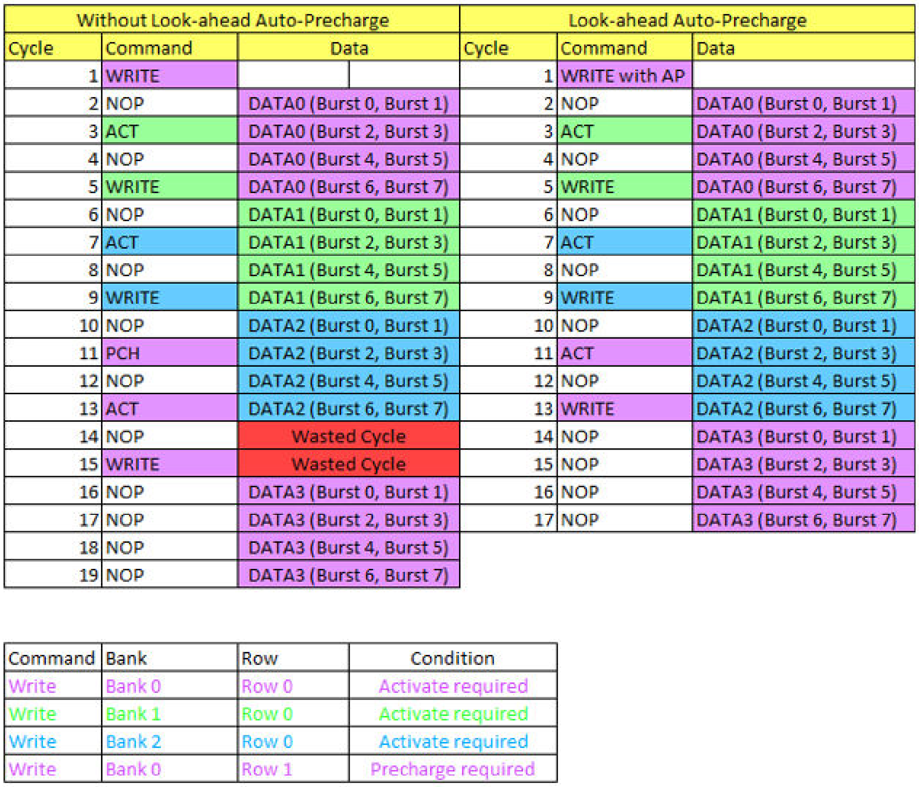
In an open-page policy, the page remains open after it is accessed for incoming commands. When the data pattern consists of repeated reads or writes to sequential addresses within the same page, the optimal system can achieve 100% efficiency for page-open transactions (ignoring the effects of periodic refreshes, which typically consume around 2-3% of total efficiency), with minimum latency for highest priority single transactions.

If you turn on **Enable Auto-Precharge Control**, you can instruct the controller to issue an auto-precharge read or write command. The next time you access that bank, the access will be faster because the controller does not have to precharge the bank before activating the row that you want to access.

The controller-derived 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 46. 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 will require only an activate command.

As shown in the preceding figure, the controller performs an auto-precharge for the write command to bank 0 at cycle 1. The controller detects that the next write at cycle 13 is to a different row in bank 0, and hence saves 2 data cycles.

The following efficiency results apply to the above figure:

Table 81. 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.

8.4.1.1. 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-MM accesses to the controller, the auto-precharge takes effect only on the last beat of the Avalon-MM 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.

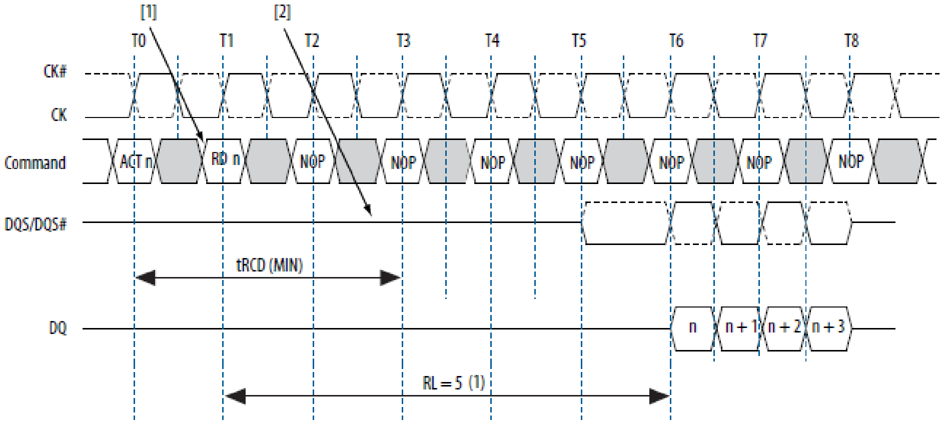
8.4.2. Additive Latency

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 48. 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}

8.4.3. 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.

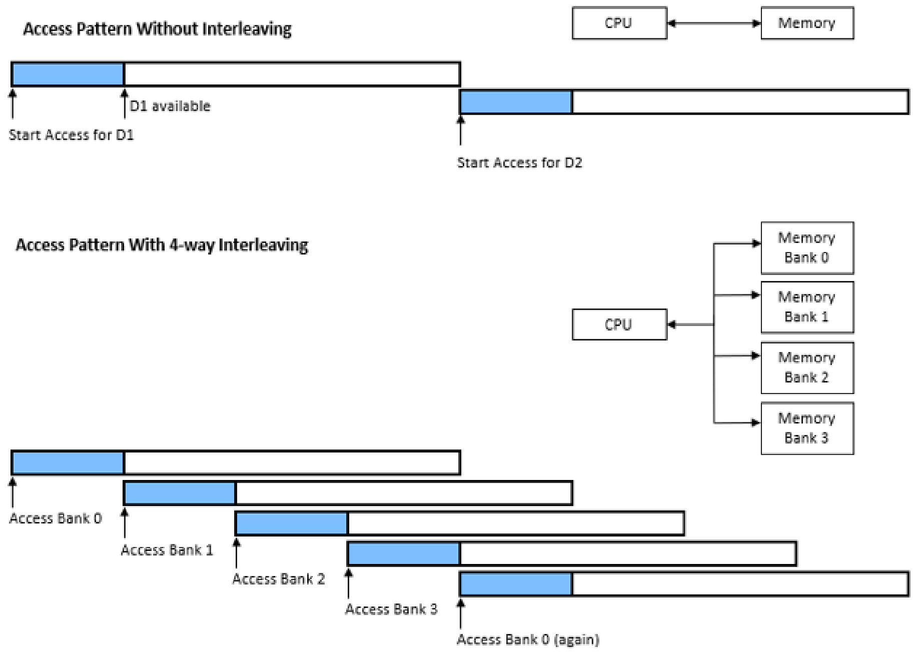
Interleaving allows banks of the SDRAM device to alternate their background operations and access cycles. One bank undergoes its precharge/activate cycle while another is being accessed. By alternating banks, the controller improves its performance by masking the precharge/activate time of each bank. 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 49. 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%.

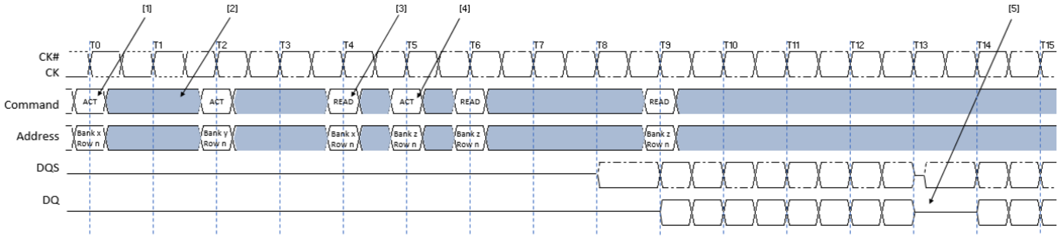
Note: Additive latency improves the efficiency of back-to-back interleaved reads or writes, but not individual random reads or writes.

8.4.4. 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 50. Bank Interleaving—Without Additive Latency



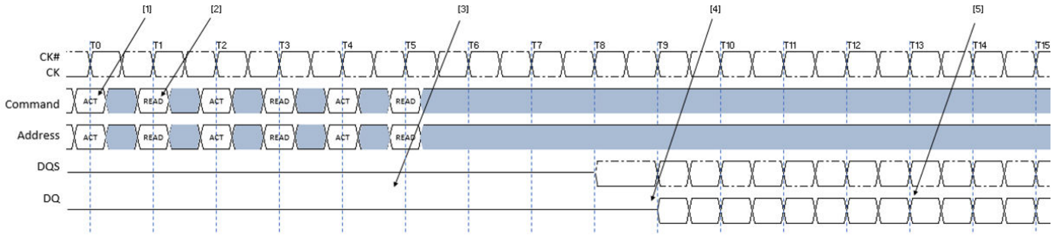
The following sequence of events describes the above figure:

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 51. 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.

8.4.5. 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.

8.4.6. 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.

8.4.7. 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.

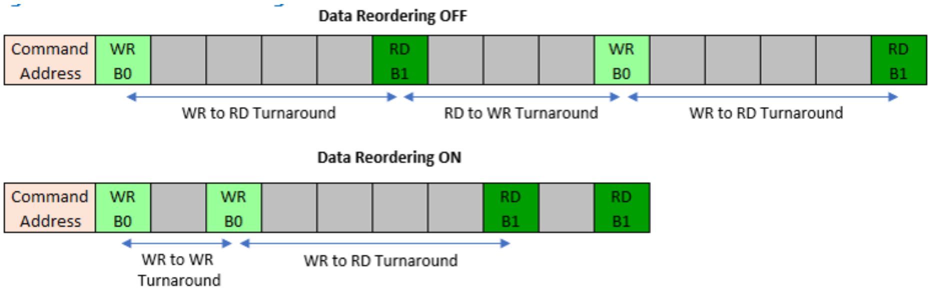
8.4.8. Data Reordering

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.

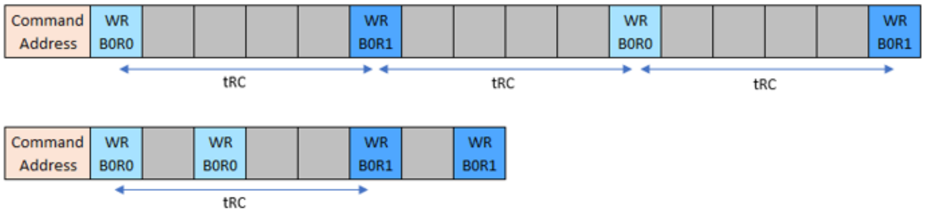
In the soft memory controller, inter-bank data reordering serves to minimize bus turnaround time by optimizing the ordering of read and write commands going to different banks; commands going to the same bank address are not reordered.

Figure 52. 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 53. Data Reordering for Minimum t_{RC}



8.4.9. 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 will be treated as high priority and served immediately, after ten other commands are served before it.

8.4.10. Command Reordering

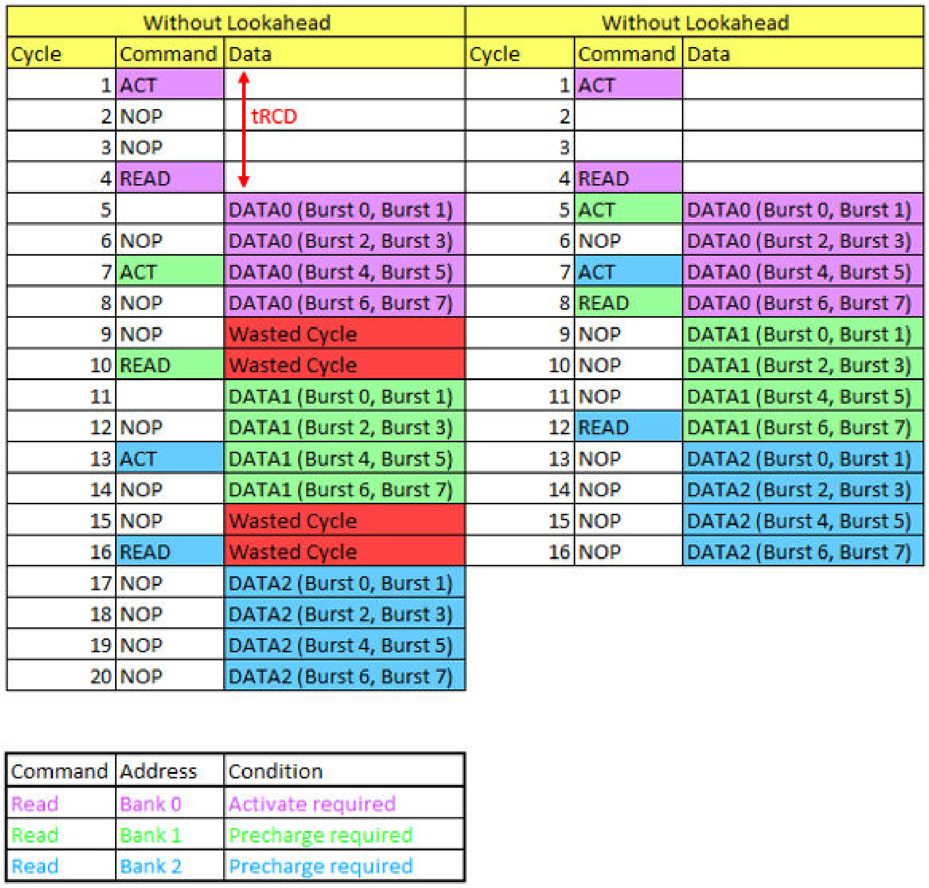
Data reordering and command reordering can both contribute towards achieving controller efficiency. You can enable command reordering by turning on **Enable Reordering** on the **Controller Settings** tab of the parameter editor.

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 54. Comparison With and Without Command Reordering Feature



Without Lookahead			Without Lookahead		
Cycle	Command	Data	Cycle	Command	Data
1	ACT		1	ACT	
2	NOP	tRCD	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 82. 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.

8.4.11. 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. For example, in a QDR-IV and RLDRAM 3 SRAM interface with separate write and read ports, the efficiency is 100% when there is an equal number of read and write operations on these memory interfaces.

8.4.12. 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.

9. Intel Agilex 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.

9.1. 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)

9.1.1. Interface Configuration Bottleneck and Efficiency Issues

Depending on the transaction types, efficiency issues can exist where the achieved data rate is lower than expected. Ideally, these issues should be assessed and resolved during the simulation stage because they are sometimes impossible to solve later without rearchitecting the product.

Any interface has a maximum theoretical data rate derived from the clock frequency, however, in practice this theoretical data rate can never be achieved continuously due to protocol overhead and bus turnaround times.

Simulate your desired configuration to ensure that you have specified a suitable external memory family and that your chosen controller configuration can achieve your required bandwidth.

Efficiency can be assessed in several different ways, and the primary requirement is an achievable continuous data rate. The local interface signals combined with the memory interface signals and a command decode trace should provide adequate visibility of the operation of the IP to understand whether your required data rate is sufficient and the cause of the efficiency issue.

To show if under ideal conditions the required data rate is possible in the chosen technology, follow these steps:

1. Use the memory 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.

9.2. 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.

9.2.1. 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.

9.2.2. 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.

9.2.3. 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.

9.2.4. 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.

9.3. 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.

9.3.1. 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 LogicLock regions or design partitions to better define the placement of your memory controllers. When you have your interface standalone placement, repeat for additional interfaces, combine, and finally add the rest of your design.

Additionally, use fitter seeds and increase the placement and router effort multiplier.

9.3.2. 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.

9.4. 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**

9.4.1. 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

9.5. Hardware Debugging Guidelines

Before debugging your design, confirm that it follows the recommended design flow. Refer to the *Intel Agilex 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.

9.5.1. 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.

9.5.2. 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.

9.5.3. 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.

9.5.4. 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.

9.5.5. 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

9.5.6. 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.

9.5.7. 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.

9.5.8. 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.

9.5.9. 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.

9.5.10. Debugging Checklist

The following checklist is a good starting point when debugging an external memory interface.

Table 83. Checklist

Check	Item
□	Try a different fit.
□	Check IP parameters at the operating frequency (t_{MRD}, t_{WTR} for example).
□	Ensure you have constrained your design with proper timing deration and have closed timing.
□	Simulate the design. If it fails in simulation, it will fail in hardware.
□	Analyze timing.
□	Place and assign R_{ZQ} (OCT).
□	Measure the power distribution network (PDN).
□	Measure signal integrity.
□	Measure setup and hold timing.
□	Measure FPGA voltages.
□	Vary voltages.
□	Heat and cool the PCB.
□	Operate at a lower or higher frequency.
□	Check board timing and trace Information.
□	Check LVDS and clock sources, I/O voltages and termination.
□	Check PLL clock source, specification, and jitter.
□	Retarget to a smaller interface width or a single bank.

9.6. 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

9.6.1. 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.

9.6.1.1. 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.

9.6.1.2. 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.

9.6.1.2.1. 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.

9.6.1.2.2. 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.

9.6.1.2.3. 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.

9.6.1.2.4. 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.

9.6.1.2.5. Read Data Valid Window and Eye Diagram

The memory generates the read signals. Take measurements at the FPGA end of the line.

To ease read diagram capture, modify the example driver to mask writes or modify the PHY to include a signal that you can trigger on when performing reads.

9.6.1.2.6. Write Data Valid Window and Eye Diagram

The FPGA generates the write signals. Take measurements at the memory device end of the line.

To ease write diagram capture, modify the example driver to mask reads or modify the PHY export a signal that is asserted when performing writes.

9.6.1.2.7. OCT and ODT Usage

Modern external memory interface designs typically use OCT for the FPGA end of the line, and ODT for the memory component end of the line. If either the OCT or ODT are incorrectly configured or enabled, signal integrity problems occur.

For the FPGA, ensure that you perform the following:

- Connect the R_{ZQ} 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_{ZQ} pins within a bank of the device that is operating at the same VCCIO voltage as the interface that is terminated.
- Review the Fitter Pin-Out file for R_{ZQ} pins to ensure that they are on the correct pins, and that only the correct number of calibration blocks exists in your design.
- Check in the fitter report that the input, output, and bidirectional signals with calibrated OCT all have the termination control block applicable to the associated R_{ZQ} pins.

For the memory components, ensure that you perform the following:

- Connect the required resistor to the correct pin on each and every component, and ensure that it is pulled to the correct voltage.
- Place the required resistor close to the memory component.
- Correctly configure the IP to enable the desired termination at initialization time.
- Check that the speed grade of memory component supports the selected ODT setting.
- Check that the second source part that may have been fitted to the PCB, supports the same ODT settings as the original.

9.6.2. 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.

9.6.2.1. 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.

9.6.2.2. 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.
- Capture the calibration error stage or error code, and use this information with whatever specifically occurs at that stage of calibration to assist with your debugging of the issue.

9.7. Debugging with the External Memory Interface Debug Toolkit

The External Memory Interface debug Toolkit for Intel Agilex FPGAs provides access to data collected by the Nios II sequencer during memory calibration, and allows you to perform various problem-resolution tasks.

The available data includes:

- General interface information, such as external memory protocol and interface width.
- Calibration results per group, including pass/fail status, failure stage, and delay settings.

The tasks that you may perform include:

- Requesting recalibration of the memory interface.
- Reading the probe data or writing the source data to the In-System Sources and Probes (ISSP) 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.

9.7.1. Prerequisites for Using the EMIF Debug Toolkit

To use the External Memory Interface Debug Toolkit, you must first do the following:

1. Add an EMIF debug interface to your design.
2. Compile your design.
3. Program the target device with the resulting SRAM Object File (`.sof`).

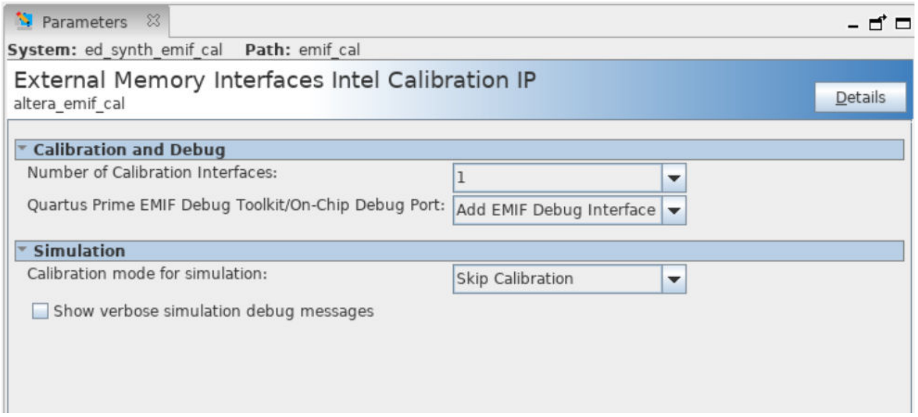
After you have completed the above steps, you are ready to run the EMIF Debug Toolkit.

9.7.2. Configuring the EMIF IP and Calibration IP for Use With the Debug Toolkit

All the external memory interfaces in a given I/O row must connect to the same calibration IP.

1. To enable the Debug Toolkit on the EMIF IP connected to the calibration IP, open the External Memory Interfaces Intel Calibration IP and select **Add EMIF Debug Interface** for the **Intel Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**.

Figure 55. Enabling the EMIF Debug Toolkit in the External Memory Interfaces Calibration IP



System: ed_synth_emif_cal **Path:** emif_cal

External Memory Interfaces Intel Calibration IP
altera_emif_cal

Calibration and Debug

Number of Calibration Interfaces: 1

Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port: Add EMIF Debug Interface

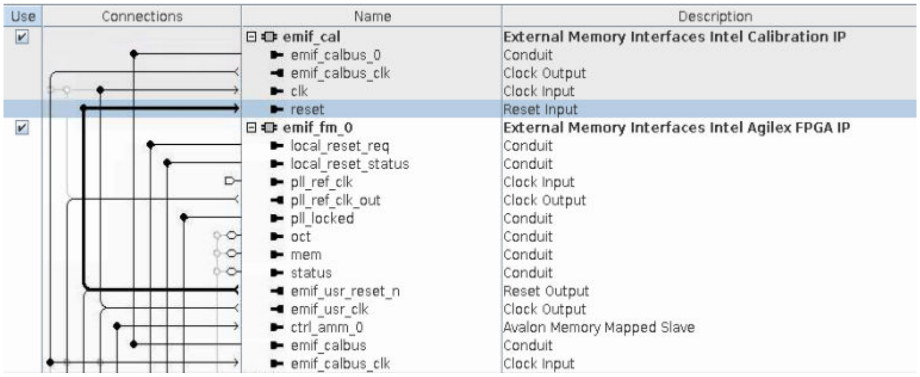
Simulation

Calibration mode for simulation: Skip Calibration

□ Show verbose simulation debug messages

2. Connect the `clk` and `reset` signals on the calibration IP to the `emif_usr_clk` and `emif_usr_reset_n`, respectively, on the EMIF IP.

Figure 56. Connectivity Between the Calibration IP and the EMIF Debug Toolkit



Use	Connections	Name	Description
☑		⊟ **emif_cal**	**External Memory Interfaces Intel Calibration IP**
		emif_calbus_0	Conduit
		emif_calbus_clk	Clock Output
		clk	Clock Input
		reset	Reset Input
☑		⊟ **emif_fm_0**	**External Memory Interfaces Intel Agilex FPGA IP**
		local_reset_req	Conduit
		local_reset_status	Conduit
		pll_ref_clk	Clock Input
		pll_ref_clk_out	Clock Output
		pll_locked	Conduit
		oct	Conduit
		mem	Conduit
		status	Conduit
		emif_usr_reset_n	Reset Output
		emif_usr_clk	Clock Output
		ctrl_amm_0	Avalon Memory Mapped Slave
		emif_calbus	Conduit
		emif_calbus_clk	Clock Input

Note: In the Intel Quartus Prime software version 19.3, the EMIF Debug Toolkit supports only a single EMIF interface. To use the toolkit, create your design with only one EMIF interface connected to the calibration IP.

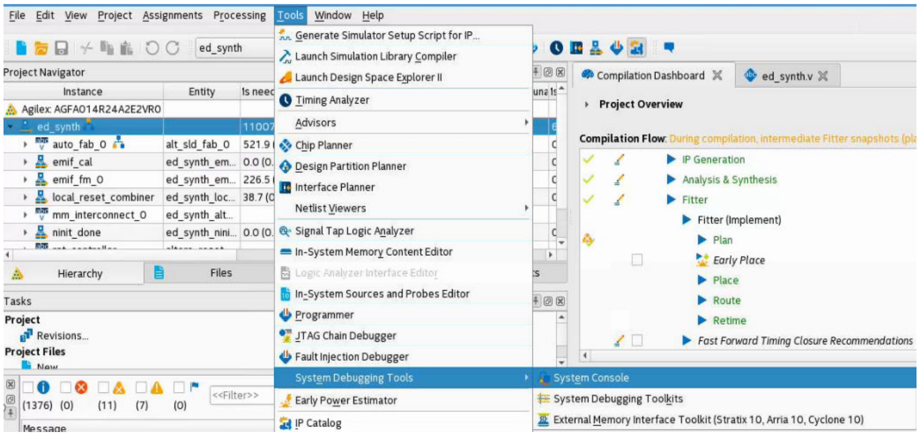
9.7.3. Launching the EMIF Debug Toolkit

Before launching the EMIF Debug Toolkit, ensure that you have configured your device with a programming file that has the EMIF Debug Toolkit enabled, as described in *Configuring the EMIF IP and Calibration IP for Use With the Debug Toolkit*.

To launch the EMIF Debug Toolkit, follow these steps:

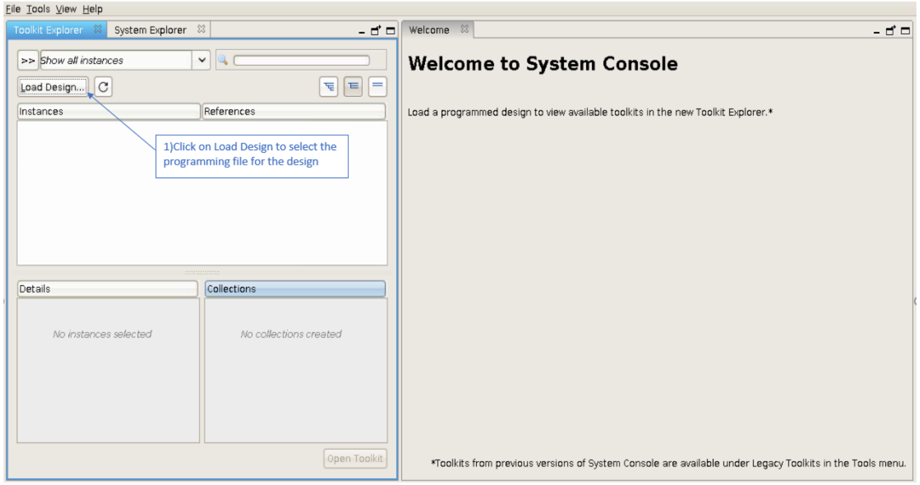
1. In the Intel Quartus Prime software, open the System Console, by clicking **Tools ➤ System Debugging Tools ➤ System Console**.

Figure 57. Opening the System Console in the Intel Quartus Prime Software



2. In the System Console, load the `.sof` file that you created in *Configuring Your EMIF and Calibration IPs for Use With the Debug Toolkit*.

Figure 58. Loading the Design in the System Console



1)Click on Load Design to select the programming file for the design

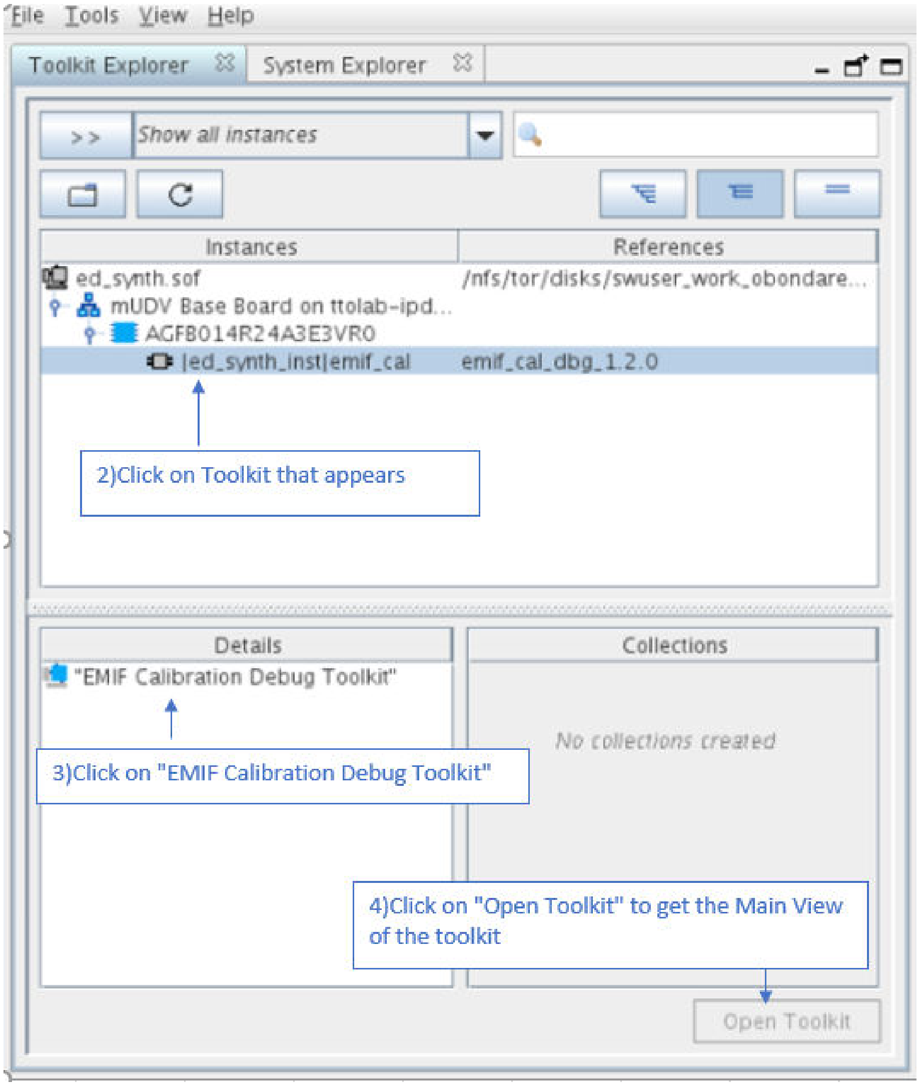
Welcome to System Console

Load a programmed design to view available toolkits in the new Toolkit Explorer.*

*Toolkits from previous versions of System Console are available under Legacy Toolkits in the Tools menu.

3. Click on the toolkit that appears in the upper portion of the dialog box, then select **EMIF Calibration Debug Toolkit** in the *Details* pane, and click the **Open Toolkit** button.

Figure 59. Launching the EMIF Debug Toolkit



2)Click on Toolkit that appears

3)Click on "EMIF Calibration Debug Toolkit"

4)Click on "Open Toolkit" to get the Main View of the toolkit

The Main View of the toolkit appears.

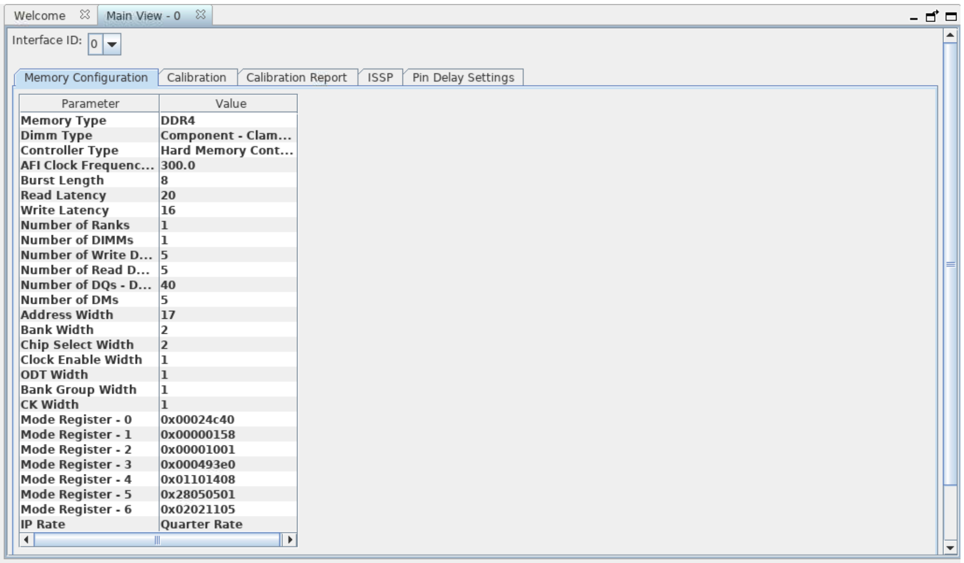
9.7.4. Using the EMIF Debug Toolkit

The Main View of the EMIF Debug Toolkit contains several tabs: **Memory Configuration**, **Calibration**, **Calibration Report**, **ISSP**, and **Pin Delay Settings**.

9.7.4.1. Memory Configuration Tab

The Memory Configuration tab shows the IP settings, which you defined when you parameterized the EMIF IP.

Figure 60. Memory Configuration Tab



Interface ID: 0

Parameter	Value
Memory Type	DDR4
Dimm Type	Component - Clam...
Controller Type	Hard Memory Cont...
AFI Clock Frequenc...	300.0
Burst Length	8
Read Latency	20
Write Latency	16
Number of Ranks	1
Number of DIMMs	1
Number of Write D...	5
Number of Read D...	5
Number of DQs - D...	40
Number of DMs	5
Address Width	17
Bank Width	2
Chip Select Width	2
Clock Enable Width	1
ODT Width	1
Bank Group Width	1
CK Width	1
Mode Register - 0	0x00024c40
Mode Register - 1	0x00000158
Mode Register - 2	0x00001001
Mode Register - 3	0x000493e0
Mode Register - 4	0x01101408
Mode Register - 5	0x28050501
Mode Register - 6	0x02021105
IP Rate	Quarter Rate

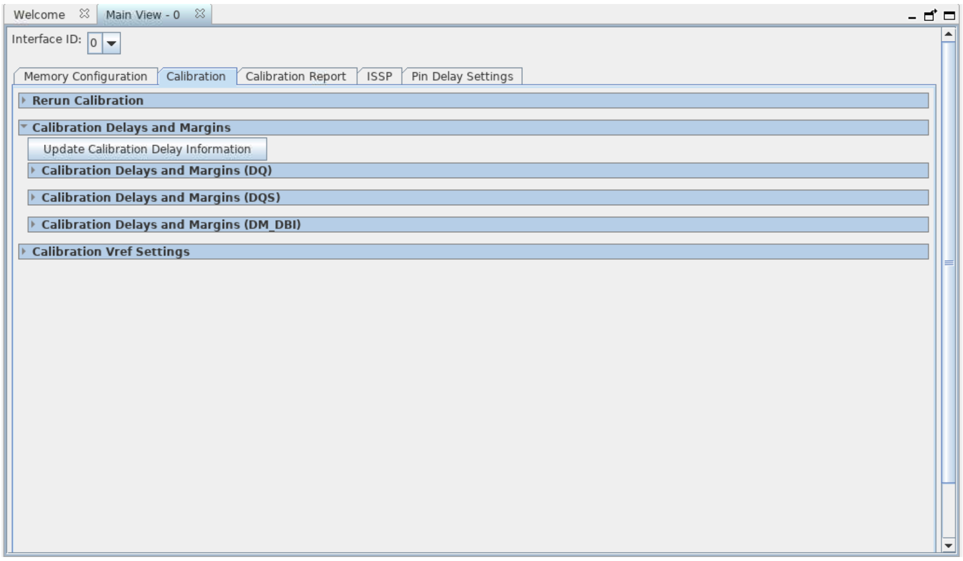
9.7.4.2. Calibration Tab

The Calibration tab allows you to rerun calibration, and reports the timing delays and margin observed on each pin (DQ, DQS, DM/DBI) during calibration.

The Calibration tab has several sections:

- Rerun Calibration
- Calibration Delays and Margins
  - Calibration Delays and Margins (DQ)
  - Calibration Delays and Margins (DQS)
  - Calibration Delays and Margins (DM/DBI)
- Calibration V_{REF} settings

Figure 61. Calibration Tab



Interface ID: 0

Welcome
Main View - 0
Memory Configuration
Calibration
Calibration Report
ISSP
Pin Delay Settings
Rerun Calibration
Calibration Delays and Margins
Update Calibration Delay Information
Calibration Delays and Margins (DQ)
Calibration Delays and Margins (DQS)
Calibration Delays and Margins (DM_DBI)
Calibration Vref Settings

9.7.4.2.1. Rerunning Calibration From the Calibration Tab of the EMIF Debug Toolkit

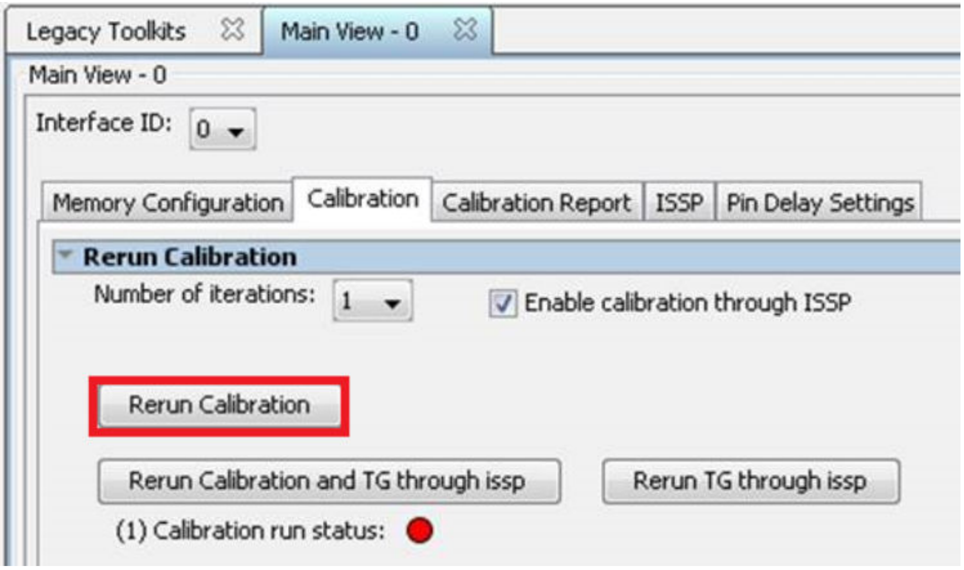
In the **Rerun Calibration** section of the **Calibration** tab, you can increase the number of iterations from the **Number of iterations** pull-down menu.

There are two methods that you can use to rerun calibration.

Method One: Using the Debug Interface

You can send a recalibration command for an interface—specified by the interface ID—through the debug interface by clicking the **Rerun Calibration** button.

Figure 62. Rerunning Calibration Method One



Interface ID: 0

Number of iterations: 1

☑ Enable calibration through ISSP

(1) Calibration run status: ●

The *Calibration run status* indicates the status of each iteration of calibration: a green icon indicates a pass, and a red icon indicates a failure.

Method Two: Using In-System Sources and Probes (ISSP)

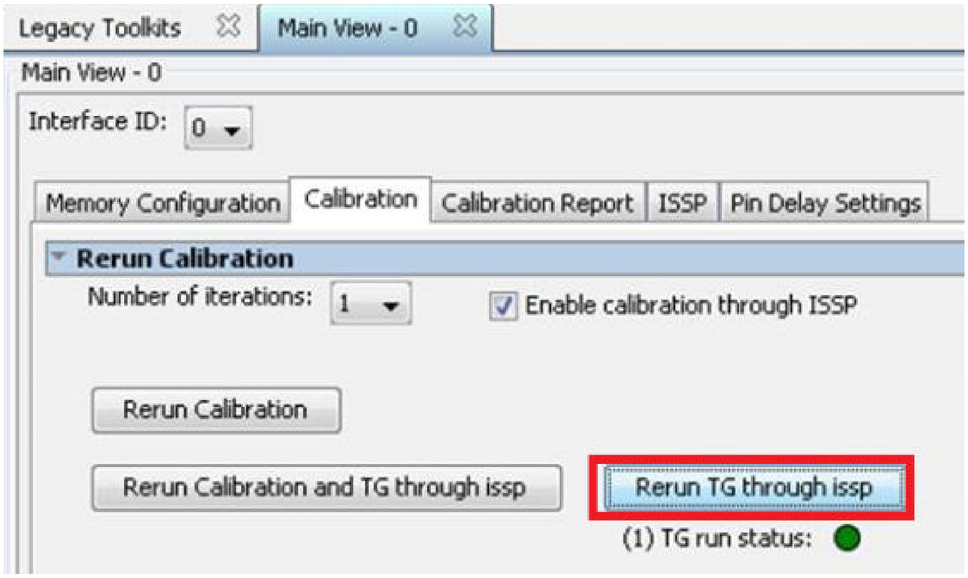
The In-System Sources and Probes method of recalibration works only on designs that have the required ISSPs enabled. If your design has a single external memory interface, you can enable the required ISSPs by adding the following line to your `.qsf` file:

```
set_global_assignment -name VERILOG_MACRO “ALTERA_EMIF_ENABLE_ISSP=1”
```

You can use the ISSPs in the design to restart the interface, which will rerun calibration and the traffic generator. To rerun calibration and the traffic generator, click the **Rerun Calibration and TG through ISSP** button.

Alternatively, you may choose to rerun the traffic generator without recalibration, by clicking the **Rerun TG through ISSP** button.

Figure 63. Rerunning Calibration Method Two



Interface ID: 0

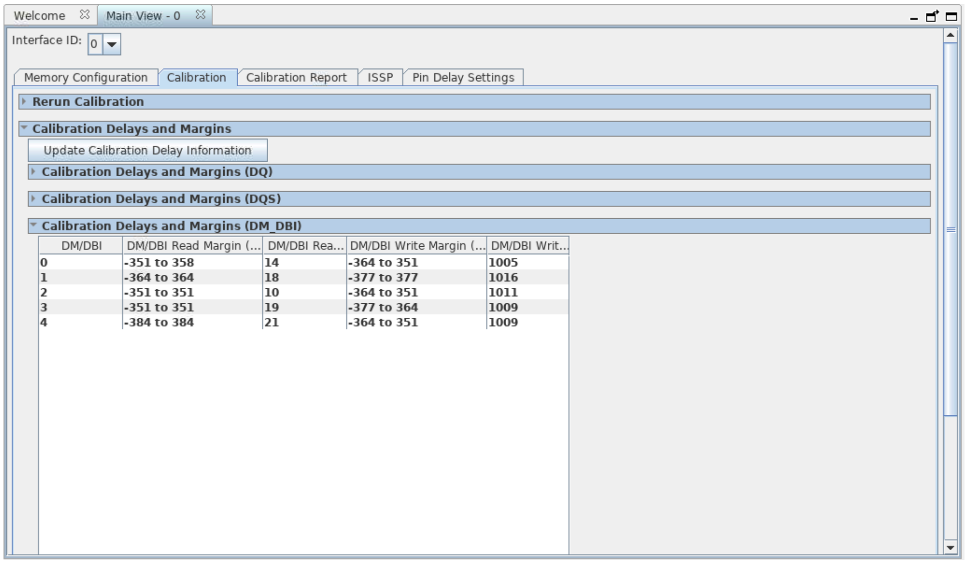
Number of iterations: 1

☑ Enable calibration through ISSP

Delays and Margins

The **Calibration Delays and Margins** section provides detailed information about the margins observed during calibration, and the settings applied to the memory interface during calibration. To review the margins, click on the respective section for DQ, DQS or DM_DBI.

Figure 64. Calibration Delays and Margins

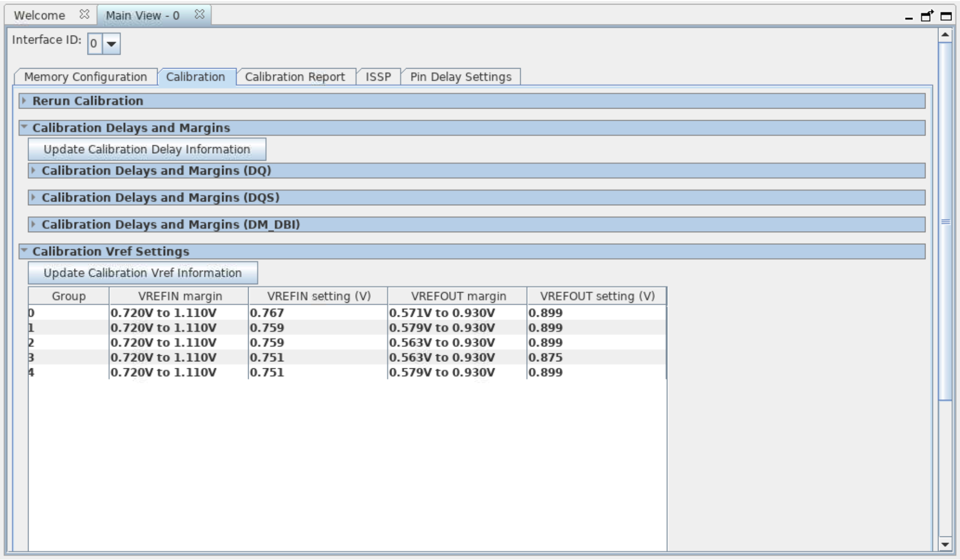

Welcome
Main View - 0
Interface ID: 0
Memory Configuration
Calibration
Calibration Report
ISSP
Pin Delay Settings
Rerun Calibration
Calibration Delays and Margins
Update Calibration Delay Information
Calibration Delays and Margins (DQ)
Calibration Delays and Margins (DQS)
Calibration Delays and Margins (DM_DBI)

DM/DBI	DM/DBI Read Margin (...	DM/DBI Rea...	DM/DBI Write Margin (...	DM/DBI Writ...
0	-351 to 358	14	-364 to 351	1005
1	-364 to 364	18	-377 to 377	1016
2	-351 to 351	10	-364 to 351	1011
3	-351 to 351	19	-377 to 364	1009
4	-384 to 384	21	-364 to 351	1009

V_{REF} Settings Section

You may also choose to see the V_{REF} margins observed during calibration, by opening the **Calibration VREF Settings** section. This section shows the margin at each V_{REF} setting on both the FPGA pin and the memory device pin, obtained by running a full calibration for each V_{REF} setting.

Figure 65. Calibration V_{REF} Settings


Welcome
Main View - 0
Interface ID: 0
Memory Configuration
Calibration
Calibration Report
ISSP
Pin Delay Settings
Rerun Calibration
Calibration Delays and Margins
Update Calibration Delay Information
Calibration Delays and Margins (DQ)
Calibration Delays and Margins (DQS)
Calibration Delays and Margins (DM_DBI)
Calibration Vref Settings
Update Calibration Vref Information

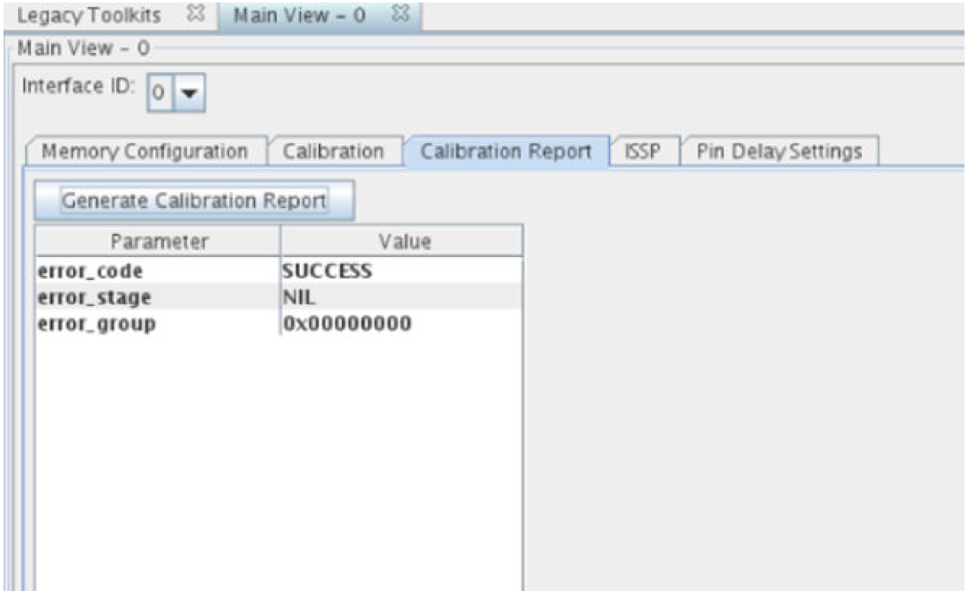
Group	VREFIN margin	VREFIN setting (V)	VREFOUT margin	VREFOUT setting (V)
0	0.720V to 1.110V	0.767	0.571V to 0.930V	0.899
1	0.720V to 1.110V	0.759	0.579V to 0.930V	0.899
2	0.720V to 1.110V	0.759	0.563V to 0.930V	0.899
3	0.720V to 1.110V	0.751	0.563V to 0.930V	0.875
4	0.720V to 1.110V	0.751	0.579V to 0.930V	0.899

9.7.4.3. Calibration Report Tab

The Calibration Report tab shows the results of the latest calibration, including the first failing calibration stage and any DQS groups that failed calibration.

You can click *Generate Calibration Report* to update the result from the latest calibration.

Figure 66. Calibration Report Tab

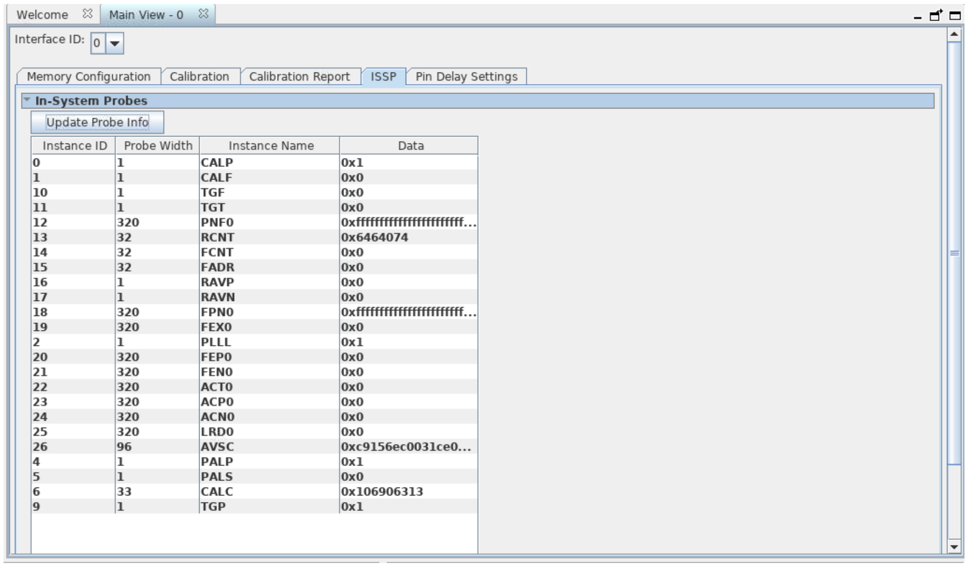


Parameter	Value
error_code	SUCCESS
error_stage	NIL
error_group	0x00000000

9.7.4.4. ISSP Tab

The ISSP tab allows you to read probe data and set source values for the In-System Sources and Probes in the design.

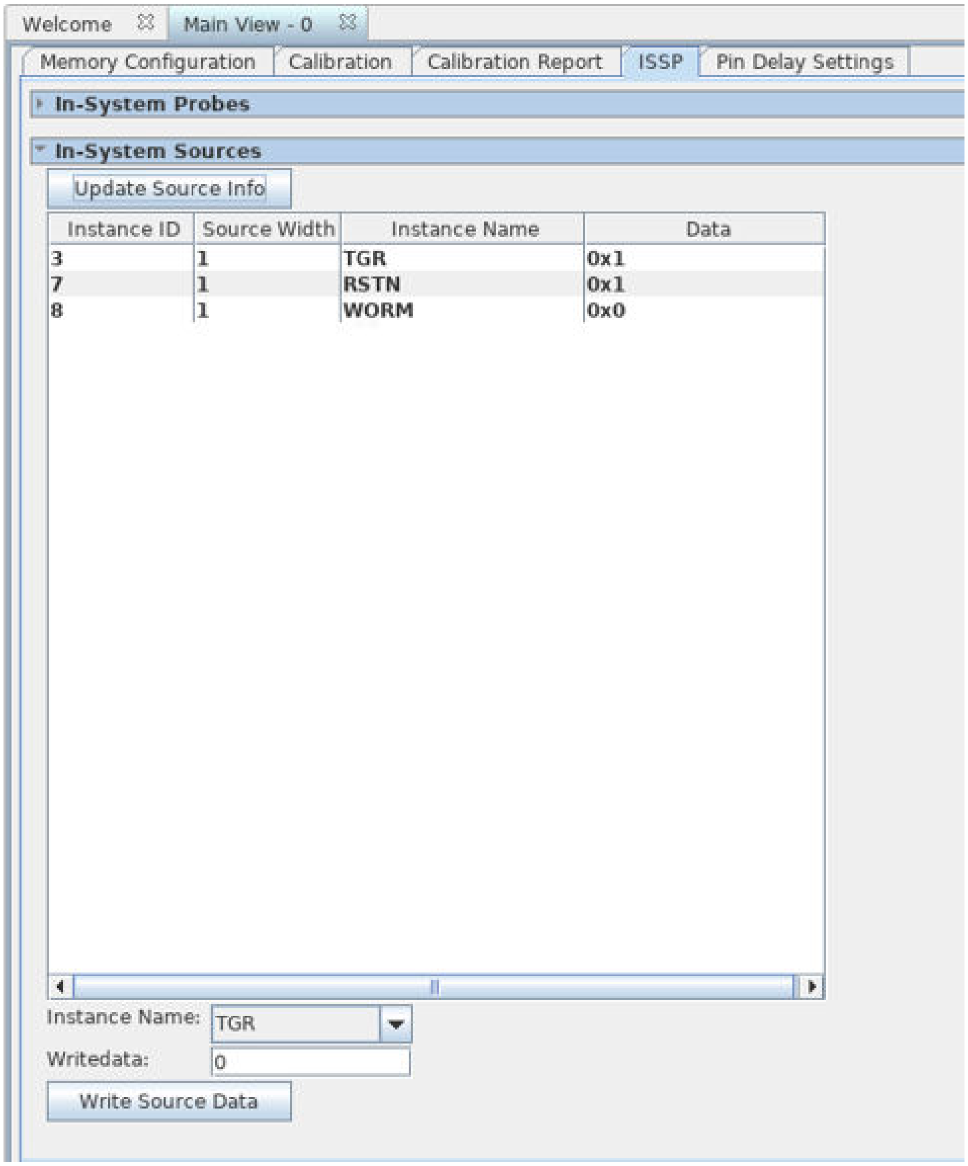
Figure 67. In-System Probes Section of the ISSP Tab



Instance ID	Probe Width	Instance Name	Data
0	1	CALP	0x1
1	1	CALF	0x0
10	1	TGF	0x0
11	1	TGT	0x0
12	320	PNF0	0xfffffffffffffffffffffffff...
13	32	RCNT	0x6464074
14	32	FCNT	0x0
15	32	FADR	0x0
16	1	RAVP	0x0
17	1	RAVN	0x0
18	320	FPN0	0xfffffffffffffffffffffffff...
19	320	FEX0	0x0
2	1	PLLL	0x1
20	320	FEP0	0x0
21	320	FEN0	0x0
22	320	ACT0	0x0
23	320	ACP0	0x0
24	320	ACN0	0x0
25	320	LRD0	0x0
26	96	AVSC	0xc9156ec0031ce0...
4	1	PALP	0x1
5	1	PALS	0x0
6	33	CALC	0x106906313
9	1	TGP	0x1

- To reread the probe data from the ISSPs in the design, click the **Update Probe Info** button.

Figure 68. In-System Sources Section of the ISSP Tab



Instance ID	Source Width	Instance Name	Data
3	1	TGR	0x1
7	1	RSTN	0x1
8	1	WORM	0x0

Instance Name: TGR

Writedata: 0

- To reread the source data from the ISSPs in the design, click **Update Source Info**.

If you want to overwrite the source data, select the *Instance Name* and change the *Writedata* value. The new source data is overwritten when you click the **Write Source Data** button.

Table 84. Definition of ISSPs in EMIF Design Example

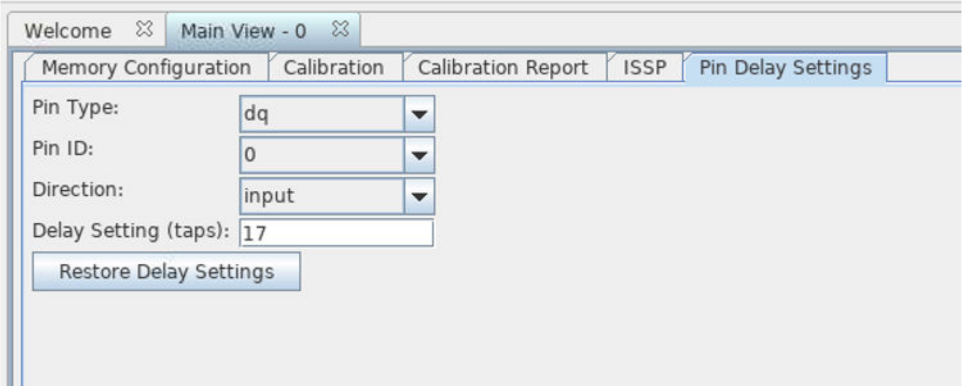
Instance Name	Description
PLLL	PLL Lock signal. A value of 1 means that the PLL is locked, a value of 0 means that the PLL cannot lock to the reference clock.
RCNT	Total read data count.
FCNT	Total fail count (data mismatch count).
FADR	First address where a data mismatch is reported.
RAVP	Read data valid from the data before the first failing address.
RAVN	Read data valid from the data after the first failing address.
PNF#	Persistent Pass Not Fail Flag. A 1 indicates pass, 0 indicates fail.
FPN#	PNF flag for the first data mismatch.
FEX#	The expected read data for the first failing read.
FEP#	The expected read data before the first failing read.
FEN#	The expected read data after the first failing read.
ACT#	The actual read data for the first failing read.
ACP#	The actual read data before the first failing read.
ACN#	The actual read data after the first failing read.
LRD#	The repeated read result. When there is an error, the driver reads again from the first failing address. This is the PNF flag for the repeated read.
AVSC	Avalon Stall Count - a concatenation of the following three 32-bit signals (MSB to LSB): • Count of read/write requests on the ctrl_amm interface. • Count of only read requests on the ctrl_amm interface. • Number of clocks counted since receiving the first read/write request.
PALP	Clock phase alignment lock status.
PALS	Clock phase alignment lock (secondary). *Note:* This is not used in Agilex FPGAs.
CALC	Calibration counter. Highest bit is a *done* signal — a value of 1 means that calibration has completed, and a value of 0 means that calibration is still in progress. The other 32 bits are a clock counter which tracks the number of clocks passed during calibration.
TGP	Traffic Generator Pass Flag. Pass=1.
TGF	Traffic Generator Fail Flag. Fail=1.
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.

Note: A # character in the instance name in the table above indicates that the ISSP spans more than one instance, as the data width exceeds the capacity of one ISSP. Higher values of # represent higher bits.

9.7.4.5. Pin Delay Settings Tab

The Pin Delay Settings tab allows you to view the delay setting on any pin in the selected interface, and to update the setting if necessary.

Figure 69. Pin Delay Settings Tab



Pin Type: dq
Pin ID: 0
Direction: input
Delay Setting (taps): 17

When you select the **Pin Type**, **Pin ID** and **Direction** values of the delay that you are interested in, the delay value appears in the **Delay Setting (taps)** field.

If you make changes on this tab, the delay value will be updated in hardware. After the value has been updated in hardware and no longer matches the setting found in calibration, you will receive a warning message that the margins in the **Calibration** tab are out-of-date.

To restore the delay settings to the values found during the last calibration, click the **Restore Delay Settings** button.

9.8. Using the Traffic Generator with the Generated Design Example

This topic provides tips for using the generated design example with the traffic generator to assist in design evaluation and debugging.

For general information about the generated EMIF design example, refer to the External Memory Interfaces Intel Agilex FPGA IP Design Example User Guide.

1. Create an Intel Quartus Prime project and open your EMIF IP in the parameter editor.
2. In the parameter editor, set the correct parameter values for your memory interface, including correct board timing values for your PCB.
3. On the **Example Designs** tab:

- Select **Synthesis**.
- Select the HDL format that you want.
- Set **Target Development Kit ➤ Select Board** to **None**. This setting ensures that the design example is generated for your specific device part number.

4. At the top right corner of the parameter editor, click **Generate Example Design**. The system generates the design example and populates the top-level project directory with the IP file directories:
   - `ed_synth.qsys` is the design example Platform Designer file and includes the EMIF IP, the traffic generator, RZQ, Reset Release IP, and reset splitter components.
   - `ed_synth.qpf` is the design example project.
5. Open the generated design example project, `ed_synth.qpf`, and verify that the device part number is correct.
6. Open the Calibration IP in the design example and set **Quartus Prime EMIF Debug Toolkit/On-chip Debug Port** to **Add EMIF Debug Interface**. This steps creates connectivity to the EMIF Toolkit for debugging purposes.
7. By default, the traffic generator is configured to run 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:
   a. 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.
   b. Search for `.TEST_DURATION ("SHORT"),` and change it to `.TEST_DURATION ("INFINITE"),`.
   c. Save your change.
8. As generated, the Intel Agilex 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`:

```
    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)
    );
```

9. At this point it is advisable, though not mandatory, to run analysis and elaboration. Doing so helps show project structure and verify assignments.

10. For each of the following top-level signals, either add virtual pins or route them out to external pins connected to test points for monitoring with an oscilloscope or LEDs:

    - local_cal_success
    - local_cal_fail
    - traffic_gen_pass
    - traffic_gen_fail
    - traffic_gen_timeout

11. Add pin location assignments for your PCB.

12. Add Signal Tap to the project. The following are recommended signals to tap:

Pins: All	local_reset_req
	local_reset_done
	local_cal_success
	local_cal_fail
	traffic_gen_pass
	traffic_gen_fail
	traffic_gen_timeout
Signal Tap : pre-synthesis	Pre-synthesis and search for signal names with wildcards as appropriate
Pass-not-fail signals	pnf_per_bit
	pnf_per_bit_persist
Avalon bus signals	amm_read_0
	amm_readdatavalid_0
	amm_ready_0
	amm_write_0
	amm_address_0
	amm_burstcount_0
	amm_byteenable_0
	amm_readdata_0
	amm_writedata_0
For the Signal Tap clock, Signal Tap : Pre-synthesis	emif_usr_clk

13. In the Intel Quartus Prime **Device Settings**, set **Device & Pin Options ➤ Unused Pins** to **As input tri-stated with weak pullup**. Set the default I/O standard as appropriate.

14. Compile your project.

15. Check the Timing Analyzer *Report DDR* report and verify that the project meets timing requirements and that pinouts are as expected.

Information on Traffic Generator status signals

- The pnf_per_bit signals are one bit for each bit on the Avalon interface. For a 32-bit-wide memory interface, this equals 256 bits for a quarter-rate interface.
    - pnf_per_bit[x] is high when the test is working correctly and is a transitory signal going low if incorrect data is seen.
    - pnf_per_bit_persist[x] is the same as pnf_per_bit but once set low , it stays low.
- The mapping of the pnf bits is dependent on the memory bus width and the Avalon interface bus width. The standard DDR4 memory access cycle is a burst length of 8. An example mapping for a 32-bit-wide memory interface is shown below. A similar mapping approach applies to any other supported interface memory bus width.
    - pnf[0] maps to dq[0] for the 1st beat of the memory bus burst
    - pnf[1] maps to dq[1] for the 1st beat of the memory bus burst
    - ...
    - pnf[31] maps to dq[31] for the 1st beat of the memory bus burst
    - pnf[32] maps to dq[0] for the 2nd beat of the memory bus burst
    - pnf[64] maps to dq[0] for the 3rd beat of the memory bus burst
    - pnf[96] maps to dq[0] for the 4th beat of the memory bus burst
    - pnf[128] maps to dq[0] for the 5th beat of the memory bus burst
    - pnf[160] maps to dq[0] for the 6th beat of the memory bus burst
    - pnf[192] maps to dq[0] for the 7th beat of the memory bus burst
    - pnf[224] maps to dq[0] for the 8th beat of the memory bus burst
    - And so forth.
- The traffic_gen_pass signals goes high if there are no bit errors and the test loops for a specific number of cycles. If you have configured the traffic generator to operate with infinite test duration, traffic_gen_pass will never go high.
- traffic_gen_fail goes high whenever a pnf signal goes low, regardless of how many loops the test runs.
- traffic_gen_timeout goes high when there is a timeout due to a problem with the traffic generator; such occurrences are extremely rare.

10. Document Revision History for External Memory Interfaces Intel Agilex FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.10.18	19.3		• In the *Introduction* chapter, revised the *EMIF IP Design Flow* flowchart. • In the *Product Architecture* chapter: — Updated the *Intel Agilex I/O Subsystem* figure in the *EMIF Architecture: I/O Subsystem* topic. — Modified the first paragraph, and updated the connectivity diagrams in the *EMIF Architecture: I/O SSM* topic. — Modified the *Pin Index Mapping* table in the *EMIF Architecture: I/O Lane* topic. — Added an explanation of Quasi-1T to the description of the Arbiter component in the *Main Control Path Components* table in the *Hard Memory Controller Main Control Path* topic. — Removed the note from the beginning of the *Intel Agilex EMIF for Hard Processor Subsystem* topic. • In the *DDR4 Support* chapter: — Removed the *Board Skew Equations* section. — Modified entries for the Clock pin in the *UDIMM, RDIMM, and LRDIMM Pin Options for DDR4* table. — Removed the *Channel Signal Integrity Measurement* and *Package Deskew* sections.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• In the *Timing Closure* chapter: — Modified the *Core to periphery (C2P)* and *Periphery to core (P2C)* descriptions in the *Timing Closure* topic. — Removed references to *Early I/O Timing Estimation*. • In the *Controller Optimization* chapter: — In the *Bank Interleaving* topic, modified the names of the three supported interleaving options. — Added content to the paragraph introducing the examples, in the *Using Auto-precharge to Achieve Highest Memory Bandwidth for DDR4 Interfaces* topic. • In the *Debugging* chapter: — Added the *Debugging with the External Memory Interface Debug Toolkit* section. — Added the *Using the Traffic Generator with the Generated Design Example* topic.
2019.07.31	19.2	1.2.0	• Added the *About the External Memory Interfaces Intel Agilex FPGA IP* chapter. • In the topic *Intel Agilex EMIF for Hard Processor Subsystem*, in the *Product Architecture* chapter, changed the description of *Memory format* in the table from 16GB support to 32GB support. • In the *Intel Agilex FPGA EMIF IP — End-User Signals* chapter: — Removed `emif_usr_reset_n_sec` and `emif_usr_clk_sec` from Table 10, *Interfaces for DDR4*. — In the topic *3.1.1.10, mem for DDR4*, modified the description of `mem_a` in the table. — Removed the *3.1.1.18 emif_usr_reset_n_sec for DDR4* and *3.1.1.19 emif_usr_clk_sec for DDR4* topics. — Removed *sbcfg1*, *sideband2*, *sideband3*, *sideband5*, *sideband8*, *sideband10*, and *sideband15* from the *Intel Agilex EMIF IP Memory Mapped Register (MMR) Tables*. — Changed the *Bit High* value for the second row in the *dramtiming0* MMR table. — Changed the *Field* name and *Description* text for the fourth row in the *caltiming4* MMR table. — Made several changes in the *Description* column of the *sideband13* MMR table. — Changed the *Field*, *Bit High*, *Bit Low*, and *Description* values in the *sideband14* MMR table. • In the *Intel Agilex EMIF IP DDR4 Parameters: Memory* topic of the DDR4 chapter: — Removed the *Enable ALERT#/PAR pins* parameter and recast the description of the *ALERT# pin placement* parameter in the *Group: Memory / Topology* table. • In the *Intel Agilex EMIF IP DDR4 Parameters: Mem I/O* topic, revised the description for the *SPD Byte 145-147 - DB MDQ Drive Strength and RTT* parameter.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• In the *Intel Agilex EMIF IP DDR4 Parameters: Diagnostics* topic: — Added the *Group: Diagnostics / Example Design* table. — Added the *Group: Diagnostics / Traffic Generator* table (marked as future support). — Added the *Group: Diagnostics / Performance* and *Group: Diagnostics / Miscellaneous* tables. • Added the *Intel Agilex EMIF IP DDR4 Parameters: Example Designs* topic. • In the *Intel Agilex FPGA EMIF IP — Timing Closure* chapter, revised the *Optimizing Timing* topic. • Removed occurrences of *Ping Pong PHY* throughout.
2019.04.02	19.1		• Initial release.