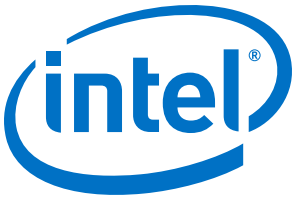


High Bandwidth Memory (HBM2) Interface Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.4**

IP Version: **19.3.0**

Subscribe
Send Feedback

UG-20031 | 2020.03.02
Latest document on the web: **PDF** | **HTML**

Contents

1. About the High Bandwidth Memory (HBM2) Interface Intel® 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. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Core Current Release Information

Item	Description
IP Version	19.3.0
Intel Quartus Prime Version	19.4
Release Date	2019.12.16

2. Introduction to High Bandwidth Memory

High Bandwidth Memory (HBM) is a JEDEC specification (JESD-235) for a wide, high bandwidth memory device. The next generation of High Bandwidth Memory, HBM2, is defined in JEDEC specification JESD-235A. The HBM2 implementation in Intel Stratix® 10 MX devices complies with JESD-235A.

The High Bandwidth Memory DRAM is tightly coupled to the host die with a distributed interface. The interface is divided into independent channels, each completely independent of one another. Each channel interface maintains a 128-bit data bus, operating at DDR data rates.

2.1. HBM2 in Intel Stratix 10 MX Devices

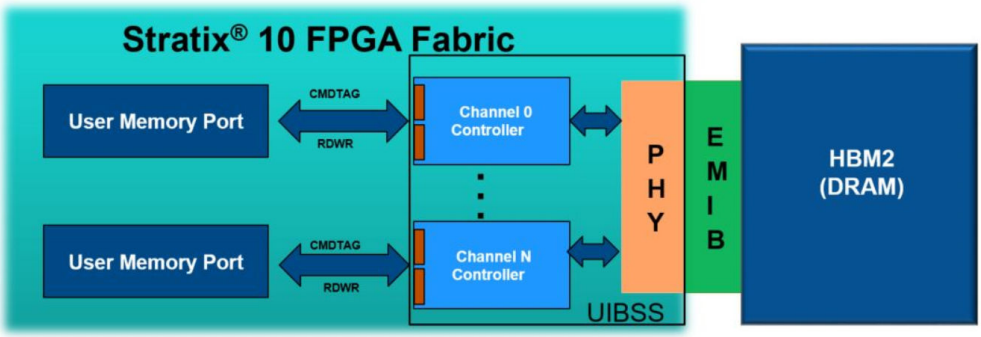
Intel Stratix 10 MX incorporates a high-performance FPGA fabric along with a HBM2 DRAM in a single package. Intel Stratix 10 MX devices support up to a maximum of two HBM2 interfaces.

Intel Stratix 10 MX incorporates Intel's Embedded Multi-Die Interconnect Bridge (EMIB) technology to implement a silicon bridge between HBM2 DRAM memory and the Universal Interface Block Subsystem (UIBSS), which contains the HBM2 controller (HBMC), physical-layer interface (PHY), and I/O ports to interface to the HBM2 stack.

As illustrated below, each Intel Stratix 10 MX device contains a single universal interface bus per HBM2 interface, supporting 8 independent channels.

The user interface to the HBM2 controller is maintained through the AXI4 protocol. Sixteen AXI interfaces are available in the user interface from each HBM2 controller, with one AXI interface available per HBM2 Pseudo Channel. HBM2 DRAM density of 4GB and 8GB are supported.

Figure 1. Intel Stratix 10 MX Device with UIB, EMIB, and HBM2 DRAM

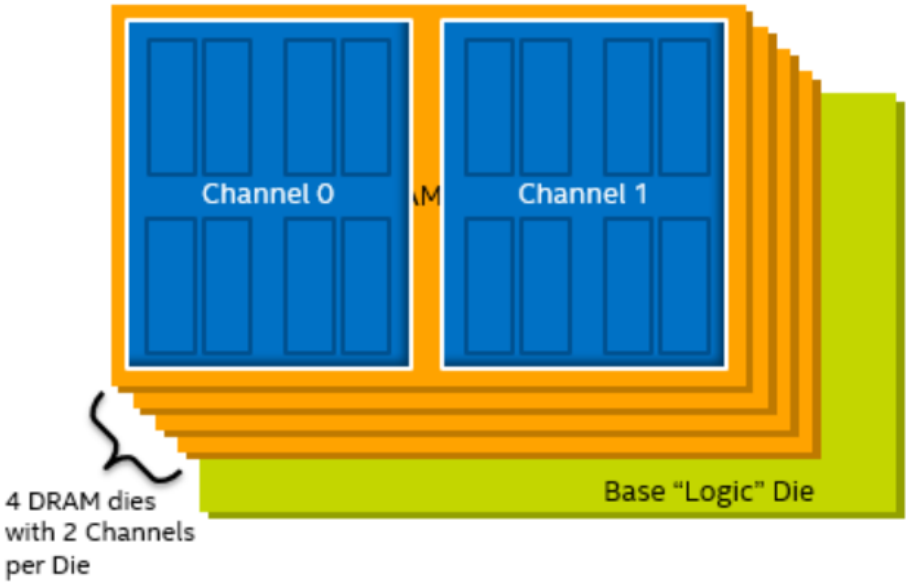


2.2. HBM2 DRAM Structure

The HBM DRAM is optimized for high-bandwidth operation to a stack of multiple DRAM devices across several independent interfaces called channels. Each DRAM stack supports up to eight channels.

The following figure shows an example stack containing four DRAM dies, each die supporting two channels. Each die contributes additional capacity and additional channels to the stack, up to a maximum of eight channels per stack. Each channel provides access to an independent set of DRAM banks. Requests from one channel may not access data attached to a different channel.

Figure 2. High Bandwidth Memory Stack of Four DRAM Dies



2.3. Intel Stratix 10 MX HBM2 Features

Intel Stratix 10 MX FPGAs offer the following HBM2 features.

- Supports one to eight HBM2 channels per HBM2 interface in the Pseudo Channel mode.
- Each HBM2 channel supports a 128-bit DDR data bus, with optional ECC support.
- Pseudo Channel mode divides each channel into two individual 64-bit I/O pseudo-channels. The two pseudo-channels operate semi-independently; they share the channel's row and column command bus as well as CK and CKE inputs, but they decode and execute commands individually. Address BA4 directs commands to either pseudo-channel 0 (BA4 = 0) or pseudo-channel 1 (BA4 = 1), offering unique address space to each pseudo-channel. Pseudo Channel mode requires that the burst length for DRAM transactions is set to 4.
- Data referenced to strobes RDQS_t / RDQS_c and WDQS_t / WDQS_c, one strobe pair per 32 DQs.

- Differential clock inputs (CK_t / CK_c). Unterminated data/address/cmd/clk interfaces.
- DDR commands entered on each positive CK_t and CK_c edge. Row Activate commands require two memory cycles; all other command are single-cycle commands.
- Supports command, write data and read data parity.
- Support for bank grouping.
- Support for data bus inversion.
- 64-bit data per pseudo-channel. Eight additional data bits are available per pseudo-channel; you can use these data bits for any of the following:
  - ECC. The ECC scheme implemented is single-bit error correction with double-bit error detection (SECDEC). This includes 8 bits of ECC code (also known as syndrome).
  - Data mask (DM). The data mask for masking write data per byte.
  - Can be left unused.
- I/O voltage of 1.2V and DRAM core voltage of 1.2V.

2.4. Intel Stratix 10 MX HBM2 Controller Features

Intel Stratix 10 MX FPGAs offer the following controller features.

- User applications communicate with the HBMC using the AXI4 Protocol.
- There is one AXI4 interface per HBM2 Pseudo Channel. Each HBM2 interface supports a maximum of sixteen AXI4 interfaces to the sixteen Pseudo Channels.
- The user interface can operate at a frequency lower than the HBM2 interface frequency. The maximum supported HBM2 interface frequency depends on the FPGA device speed grade. The minimum frequency of the core clock is one quarter of the HBM2 interface frequency.
- Each AXI interface supports a 256-bit Write Data interface and a 256-bit Read Data interface.
- The controller offers 32B and 64B access granularity supporting burst length 4 (BL 4) and pseudo-BL 8 (two back to back BL4).
- The controller offers out-of-order command scheduling and read data reordering.
- The controller supports user-initiated Refresh commands, and access to the HBM2 channel status registers, through the side band Advanced Peripheral Bus (APB) interface.
- The controller supports data mask or error correction code (ECC). When you do not use data mask or ECC, you may use those bits as additional data bits.

Related Information

Clock Signals on page 36

3. Intel Stratix 10 MX HBM2 Architecture

This chapter provides an overview of the Intel Stratix 10 MX HBM2 architecture.

3.1. Intel Stratix 10 MX HBM2 Introduction

Intel Stratix 10 MX devices use the Intel EMIB technology to interface to the HBM2 memory devices.

- The Intel Stratix 10 MX FPGAs offer up to two HBM2 interfaces.
- Each HBM2 device can have a device density of 4GB or 8GB, based on the FPGA chosen.

This system-in-package solution helps to achieve maximum bandwidth and low power consumption in a small footprint.

3.2. Intel Stratix 10 MX UIB Architecture

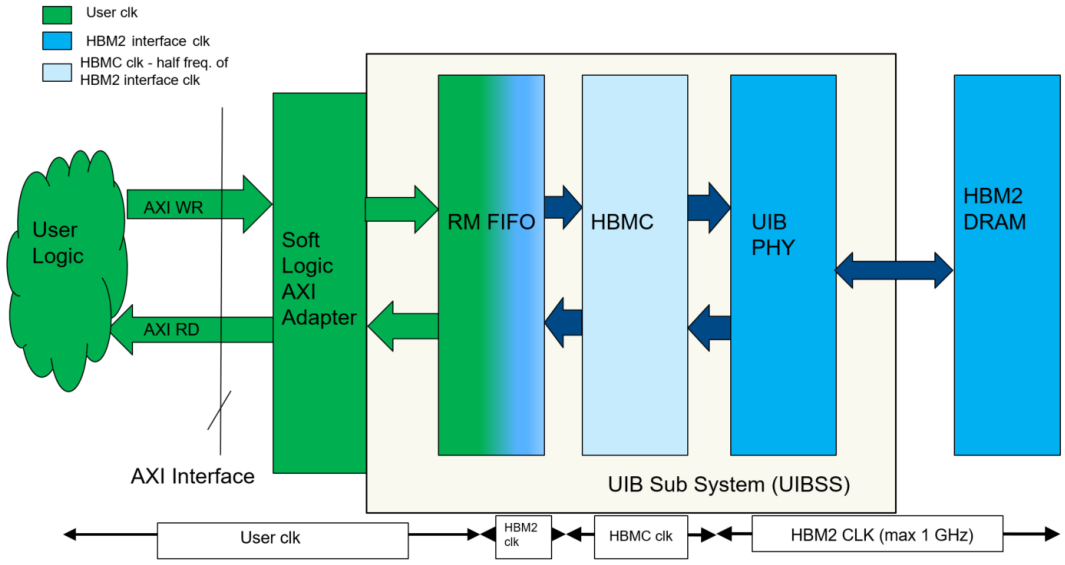
The Intel Stratix 10 MX device architecture includes the universal interface bus (UIB) subsystem (UIBSS) which contains the necessary logic to interface the FPGA core to the HBM2 DRAM.

Each UIB subsystem includes the HBM2 hardened controller and the universal interface bus, consisting of the hardened physical interface and I/O logic needed to interface to each HBM2 DRAM device. The AMBA AXI4 protocol interfaces the core logic with the universal interface bus subsystem. An optional soft logic adapter implemented in the FPGA fabric helps to efficiently interface user logic to the hardened HBM2 controller.

The following figure shows a high-level block diagram of the Intel Stratix 10 HBM2 universal interface bus subsystem. The UIB subsystem includes the following hardened logic:

- Rate-matching FIFOs that transfer logic from the user core clock to the HBM2 clock domain.
- HBM2 memory controller (HBMC).
- UIB PHY, including the UIB physical layer and I/O.

Figure 3. Block Diagram of Intel Stratix 10 MX HBM2 Implementation



The user core clock drives the logic highlighted in green, while the UIB clocks the logic highlighted in blue. The UIB clock also drives the HBM2 interface clock. User logic can run up to one-to-four times slower than the HBM2 interface.

Soft Logic AXI Adaptor

The HBM2 IP also includes a soft logic adaptor implemented in FPGA core logic. The soft logic adaptor gates the user valid signals (`write address valid`, `write data valid`, and `read address valid`) with the corresponding pipelined ready signals from the HBM2 controller. The soft logic adapter also temporarily stores output from the HBM2 controller (AXI write response and AXI read data channels) when the AXI ready signal is absent. You can disable the temporary storage logic if user logic is always ready to accept output from the HBM2 controller through the parameter editor when generating the HBM2 IP.

HBM2 DRAM

The following table lists the HBM2 signals that interface to the UIB. The UIB drives the HBM2 signals and decodes the received data from the HBM2. These signals cannot be accessed through the AXI4 User Interface.

Table 2. Summary of Per-channel Signals

Signal Name	Signal Width	Notes
Data	128	128 bit bidirectional DQ per channel
Column command/address	8	8-bit wide column address bits
Row command/address	6	6-bit wide row address bits
DBI	16	1 DBI per 8 DQs
DM_CB	16	1 DM per 8 DQs. You can use these pins for DM or ECC, but not both.

continued...

Signal Name	Signal Width	Notes
PAR	4	1 parity bit per 32 DQs
DERR	4	1 data error bit per 32 DQs
Strobes	16	Separate strobes for read and write strobes. One differential pair per 32 DQs for read and write.
Clock	2	Clocks address and command signals
CKE	1	Clock enable
AERR	1	Address error

The following table lists the HBM2 signals that are common to all Pseudo Channels in each HBM2 interface. The HBM2 controller interfaces with the following signals; these signals are not available at the AXI4 user interface.

Table 3. Summary of Global HBM2 Signals

Signal Name	Signal Width	Notes
Reset	1	Reset input
TEMP	3	Temperature output from HBM2.
Cattrip	1	Catastrophic temperature sensor.

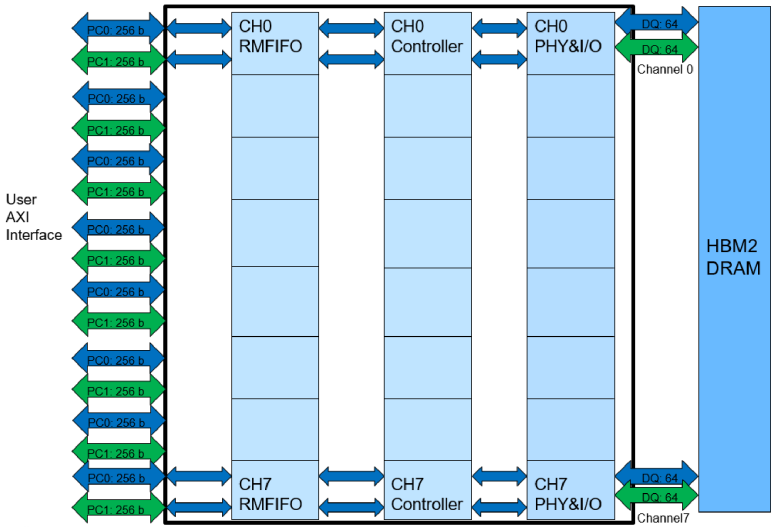
The Intel Stratix 10 MX HBM2 IP supports only the Pseudo Channel mode of the HBM2 specification. Pseudo Channel mode includes the following features:

- Pseudo Channel mode divides a single HBM2 channel into two individual subchannels of 64 bit I/O.
- Both Pseudo Channels share the channel's row and column command bus, CK, and CKE inputs, but decode and execute commands individually.
- Pseudo Channel mode requires a burst length of 4.
- Address BA4 directs commands to either Pseudo Channel 0 (BA4 = 0) or Pseudo Channel 1 (BA4 = 1). The HBM2 controller handles the addressing requirements of the Pseudo Channels.
- Power-down and self-refresh are common to both Pseudo Channels, due to a shared CKE pin. Both Pseudo Channels also share the channel's mode registers.

User AXI Interface

Each Intel Stratix 10 MX HBM2 interface supports a maximum of eight HBM2 channels. Each HBM2 channel has two AXI4 interfaces, one per Pseudo Channel. Each AXI4 interface includes a 256-bit wide Write and Read Data interface per Pseudo Channel. The following figure shows the flow of data from user logic to the HBM2 DRAM through the UIBSS, while selecting HBM2 channels 0 and 7.

Figure 4. Intel Stratix 10 MX HBM2 Interface Using HBM2 Channels 0 and 7 through the UIBSS



There is one AXI interface per Pseudo Channel. Each AXI interface supports a 256-bit wide Write Data interface and a 256-bit wide Read Data interface, to and from the HBM2 controller. The AXI4 protocol can handle concurrent writes and reads to the HBM2 controller. There is also a sideband user port per user channel pair, compliant to the Advanced Peripheral Bus (APB). The sideband provides access to user-controlled features such as refresh requests, ECC status, Power Down status, HBM2 temperature readout, calibration status, and User Interrupt.

For information on the AXI protocol features supported by the Intel Stratix 10 HBM2 controller, refer to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface*.

Related Information

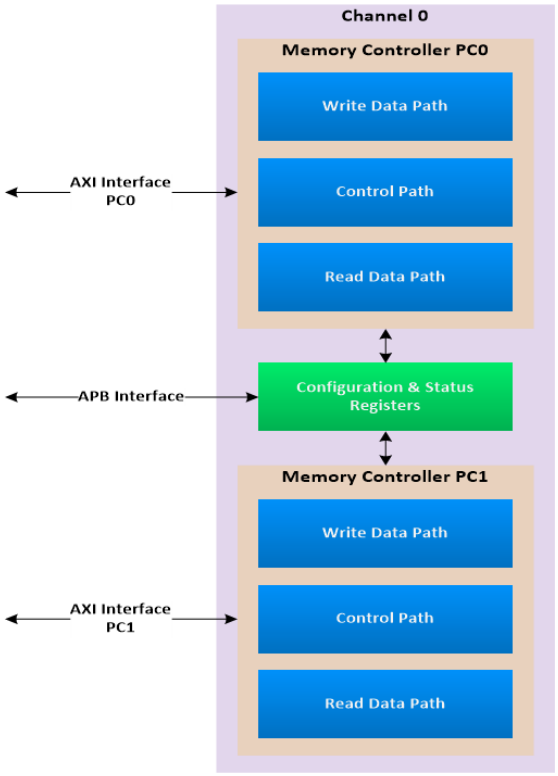
- Intel Stratix 10 MX HBM2 Controller Details on page 13
- High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface on page 35

3.3. Intel Stratix 10 MX HBM2 Controller Architecture

The hardened HBM2 controller provides a controller per Pseudo Channel.

Each controller consists of a write and read data path and the control logic that helps to translate user commands to the HBM2 memory. The HBM2 controller logic accounts for the HBM2 memory specification timing and schedules commands in an efficient manner. The following figure shows a block diagram of the HBM2 controller, corresponding to channel 0. The HBM2 controller's user-logic interface follows the AXI4 protocol. You can find more information about the interface timing details in the *User AXI Interface Timing* section.

Figure 5. Intel Stratix 10 MX HBM2 Controller Block Diagram



3.3.1. Intel Stratix 10 MX HBM2 Controller Details

This topic explains some of the high level HBM2 controller features.

HBM2 burst transactions

The HBM2 controller supports only the Pseudo Channel mode of accessing the HBM2 device; consequently, it can only support BL4 transactions to the DRAM. For improving efficiency, it supports the pseudo-BL8 mode, which helps to provide two back-to-back BL4 data using a given start address, similar to a BL8 transaction.

Each BL4 transaction corresponds to 4*64 bits or 32 bytes and a BL8 transaction corresponds to 64 bytes per Pseudo Channel. You can select the burst transaction mode (32 B vs 64B) through the parameter editor.

The user logic can interface to a maximum of 16 Pseudo Channels (16 AXI ports) per HBM2 interface. Each AXI port has a separate write and read interface, and can handle write and read requests concurrently at the same clock. Each write and read data interface per AXI port is 256 bits wide; that is, each AXI Write or Read Data transaction corresponds to data transfers corresponding to two HBM2 memory clock cycles.

User interface vs HBM2 Interface Frequency

The user interface runs at a frequency lower than the HBM2 interface; the maximum interface frequency depends on the chosen device speed grade and the FPGA core logic frequency. The rate-matching FIFOs within the UIB subsystem handle the data transfer between the two clock domains.

Command Priority

You can set command priority for a write or read command request through the AXI interface, through the qos signal in the AXI write address channel, or in the AXI read address channel. The HBM2 controller supports normal and high priority levels. The system executes commands with the same priority level in a round-robin scheme.

Starvation limit

The controller tracks how long each command waits and leaves no command unserviced in the command queue for a long period of time. The controller ensures that it serves every command efficiently.

Command scheduling

The HBM2 controller schedules the incoming commands to achieve maximum efficiency at the HBM2 interface. The HBM2 controller also follows the AXI ordering model of the AXI4 protocol specification.

Data re-ordering

The controller can reorder read data to match the order of the read requests.

Address ordering

The HBM2 controller supports different address ordering schemes that you can select for best efficiency given your use case. The chosen addressing scheme determines the order of address configurations in the AXI write and read address buses, including row address, column address, bank address, and stack ID (applicable only to the 8H devices). The HBM2 controller remaps the logical address of the command to physical memory address.

Thermal Control

The HBM2 controller uses the TEMP and CATTRIP outputs from the HBM2 device to manage temperature variations in the HBM2 interface.

- Temperature compensated refresh (TEMP): The HBM2 DRAM provides temperature compensated refresh information to the controller through the TEMP[2:0] pins, which defines the proper refresh rate that the DRAM expects to maintain data integrity. Absolute temperature values for each encoding are vendor-specific. The encoding on the TEMP[2:0] pins reflects the required refresh rate for the hottest device in the stack. The TEMP data updates when the temperature exceeds vendor-specified threshold levels appropriate for each refresh rate.
- Catastrophic temperature sensor (CATTRIP): The CATTRIP sensor detects whether the junction temperature of any die in the stack exceeds the catastrophic trip threshold value CATTEMP. The device vendor programs the CATTEMP to a value less than the temperature at which permanent damage to the HBM stack would occur. The CATTEMP value is also the Absolute Max Junction temperature value as specified in the Intel Stratix 10 data sheet for the family of devices that include the HBM2 DRAM. You can find the Intel Stratix 10 data sheet at the following location: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/s10_datasheet.pdf.

  If a junction temperature anywhere in the stack exceeds the CATTEMP value, the HBM stack drives the external CATTRIP pin to 1, indicating that catastrophic damage may occur. When the CATTRIP pin is at 1, the controller stops all traffic to HBM and stalls indefinitely. To resolve the overheating situation and return the CATTRIP value to 0, remove power from the device and allow sufficient time for the device to cool before again applying power.
- Thermal throttling: Thermal throttling is a controller safety feature that helps control thermal runaway if the HBM2 die overheats, preventing a catastrophic failure. You can specify the HBM2 device junction temperature at which the controller begins to throttle input commands, and the throttle ratio that determines the throttle frequency. The controller deasserts the AXI ready signals (`awready`, `wready` and `arready`) when it is actively throttling the input commands and data.

Refresh requests

The HBM2 controller handles HBM2 memory refresh requirements and issues refresh requests at the optimal time, as specified by the JEDEC specification of the HBM2 DRAM. The controller automatically controls refresh rates based on the temperature setting of the memory through the TEMP vector that the memory provides. You can select the HBM2 controller refresh policy, based on the frequency of refresh requests. You can choose to issue refresh commands directly, through the sideband APB interface.

Precharge policy

The HBM2 controller issues precharge commands to the HBM2 memory based on the write/read transaction address. In addition, you can issue an auto-precharge command together with a write and read command, through the AXI write address port and AXI read address port.

There are two auto-precharge modes:

- HINT – You can issue the auto-precharge request. The controller then decides when to issue the precharge command.
- FORCED – You provide auto-precharge requests through the AXI interface and the precharge request executes.

Power down enable

To conserve power, the HBM2 controller can enter power-down mode when the bus is idle for a long time. You can select this option if required.

ECC

The HBM2 controller supports ECC. The ECC scheme implemented is single-bit error correction with double-bit error detection, with 64-bits of data and 8-bits of ECC code (also known as the syndrome).

HBM2 Controller features enabled by default

The HBM2 controller enables the following features by default:

- DBI – The DBI option supports both write and read DBI, and optimizes SI/power consumption by restricting signal switching on the HBM2 DQ bus.
- Parity – Supports command/address parity and DQ parity.

Related Information

- Clock Signals on page 36
- Intel Stratix 10 MX UIB Architecture on page 9

4. Creating and Parameterizing the High Bandwidth Memory (HBM2) Interface Intel FPGA IP

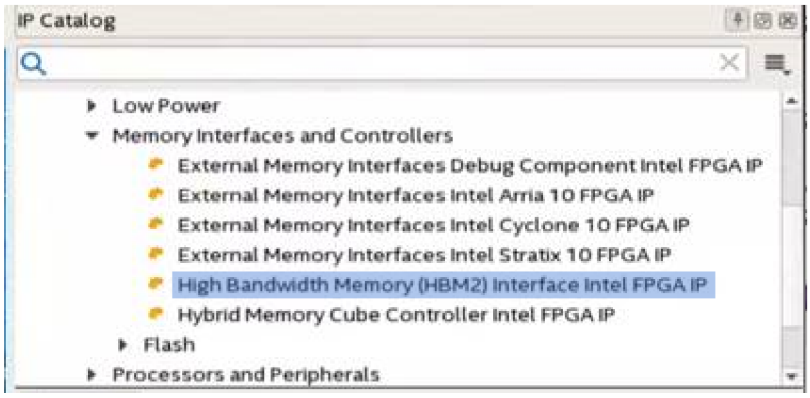
This chapter contains information on project creation, IP parameter descriptions, and pin planning for your High Bandwidth Memory (HBM2) Interface Intel FPGA IP.

4.1. Creating an Intel Quartus Prime Pro Edition Project for High Bandwidth Memory (HBM2) Interface FPGA IP

You can parameterize and generate the High Bandwidth Memory (HBM2) Interface Intel FPGA IP using the Intel Quartus Prime Pro Edition software.

1. Before generating the HBM2 IP, you must create a new project:
   a. Launch the Intel Quartus Prime Pro Edition software.
   b. Launch the New Project Wizard by clicking **File ➤ New Project Wizard**.
   c. Type a name for your project in the **Directory, Name, Top-Level Entity** field.
   d. In the **Project Type** section, select **Empty Project**.
   e. In the **Add Files** section, click **Next**.
   f. In the **Family, Device, and Board Settings** section, select **Stratix 10 MX** as the device family.
   g. Under **Available Devices**, select any MX device and your desired speed grade.
   h. Click **Next** and follow the Wizard's prompts to finish creating the project.
2. In the **IP Catalog**, open **Library ➤ Memory Interfaces and Controllers**.
3. Launch the parameter editor by selecting **High Bandwidth Memory (HBM2) Interface Intel FPGA IP**.

Figure 6. Selecting High Bandwidth Memory (HBM2) Interface Intel FPGA IP in the IP Catalog



4.2. Parameterizing the High Bandwidth Memory (HBM2) Interface Intel FPGA IP

You can parameterize your HBM2 IP with the HBM2 IP parameter editor.

The parameter editor comprises the following tabs, on which you set the parameters for your IP:

- General
- Controller
- Diagnostics
- Example Designs

4.2.1. General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP

The **General** tab allows you to select the channels that you want to implement, and to select the memory and fabric core clock frequency.

Figure 7. General Tab

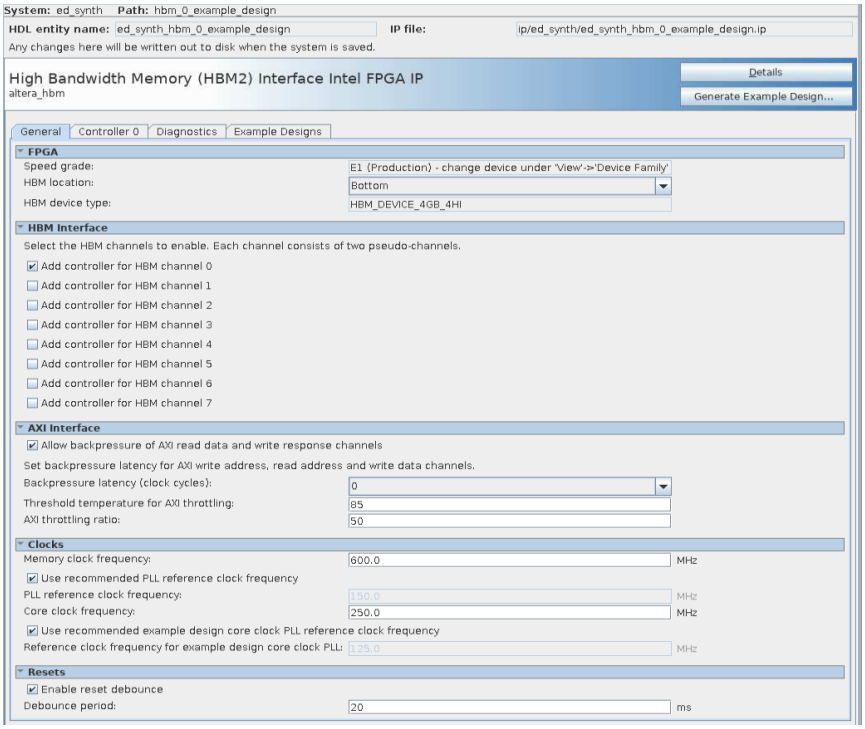


Table 4. Group: General / FPGA

Display Name	Description
Speed Grade	Speed grade of FPGA device and whether it is an engineering sample (ES) or a production device. The value is automatically determined based on the device selected under **View ➤ Device Family**. If you do not specify a

continued...

Display Name	Description
	device, the system assumes a production device of the fastest speed grade. You should always specify the correct target device during IP generation; failure to specify the correct device may result in generated IP that does not work on hardware. Specifies the speed grade of the Intel Stratix 10 FPGA.
HBM2 Location	Determines the location of the HBM2 interface in the Intel Stratix 10 FPGA. The FPGA offers HBM2 interfaces on the top and bottom of the FPGA core.
HBM2 Device Type	The HBM device type. 4GB/4H refers to HBM2 device with a total device density of 4GB in a 4-high Stack, and 8GB8H refers to a total HBM2 device density of 8GB in an 8-high Stack.

Table 5. Group: General / HBM2 Interface

Display Name	Description
Add Controller for HBM Channel 0 ––– 7	Adds a High Bandwidth Memory controller on channel 0 of the Universal Interface Block. Each controller handles two HBM Pseudo Channels. Allows you to select the HBM2 memory channels that you want to implement. Each HBM2 channel supports a 128-bit interface to the HBM2 device, using two 64-bit Pseudo Channels. The user interface to the HBM2 controller uses the AXI4 protocol. Each controller has one AXI4 interface per Pseudo Channel or 2 AXI4 interfaces per channel.

Table 6. Group: General / AXI Interface

Display Name	Description
Allow backpressure of AXI read data and write response channels	Instantiates FIFOs in soft logic to buffer read data and write response on the AXI interfaces. This is required if the RREADY/BREADY signals are ever deasserted. You can disable this option to reduce latency but only if RREADY/BREADY are never used to backpressure the interface.
Backpressure latency (clock cycles)	Select a value from 0 to 2 (0 is the default). Allows the user interface time to react to the controller READY signals (AXI Write Address / Read Address / Write Data) and can also be used to improve timing between the AXI User Interface and the HBM2 IP with no increase in latency. When Backpressure latency is set to 1, a single register stage is added in the AXI interface signals from the core to the HBM2 controller. When Backpressure latency is set to 2, one register stage is added on signals from the core to the HBM2 controller and one register on READY signal from the controller to the core. Beginning with the Intel Quartus Prime software version 19.4, the HBM2 IP itself generates the registers, so there is no need to manually add the registers as was necessary in earlier versions. Refer to *Improving User Logic to HBM2 Controller AXI Interface Timing* for information on using this feature.
Threshold temperature for AXI throttling	This parameter defines the temperature of the HBM2 stack, in degrees Celsius, above which the HBM2 controller throttles AXI interface transactions. The temperature setting applies to all the AXI4 interfaces; however, you must enable this feature on the corresponding controller tab of each

continued...

Display Name	Description
	HBM2 controller. When you enable throttling, the HBM2 controller reduces the amount of traffic on the DRAM channel.
AXI throttling ratio	If AXI interface throttling is enabled based on temperature this parameter defines the throttle ratio as a percentage (0 = no throttling, 100 = full throttling). You can enable or disable throttling on the individual controller tabs.

Table 7. Group: General / Reset

Display Name	Description
Enable reset debounce	If enabled, reset debouncing logic is added on the `wmcrst_n_in` signal
Debounce period	Set the debounce period in ms. The minimum period is 20 ms.

Table 8. Group: General / Clocks

Display Name	Description
Memory clock frequency	The frequency of the memory clock in MHz. Specifies the clock frequency for the HBM2 interface. The maximum supported HBM2 clock frequency depends on the FPGA device speed grade: • -1 Speed grade : 1 GHz • -2 Speed grade : 800 MHz • -3 Speed grade : 600 MHz
Use recommended PLL reference clock frequency	When checked the PLL reference clock frequency is automatically calculated for best performance. Uncheck the check box if you want to specify your own PLL reference clock frequency. Automatically calculates the PLL reference clock frequency for best performance. You should disable this parameter if you want to select a different PLL reference clock frequency
PLL reference clock frequency	PLL reference clock frequency. This is a universal interface bus (UIB) PLL reference clock. You must feed a clock of this frequency to the PLL reference clock input of the memory interface. Enable this parameter only if you disable **Use recommended PLL reference clock frequency**, and want to specify a PLL reference clock frequency. You should use the fastest possible PLL reference clock frequency to achieve best jitter performance.
Core clock frequency	The frequency of the user AXI4 interface in MHz. This clock must be provided from an I/O PLL (not instantiated by the HBM2 IP). The reference clock driving the core I/O PLL must be provided from the same oscillator that supplies the UIB PLL reference clock on the board for a given HBM2 interface, to achieve maximum performance. The maximum supported core frequency depends on the device speed grade and timing closure of the core interface clock within the FPGA. The minimum frequency of the core clock is one-fourth the HBM2 interface frequency.
Use recommended example design core clock PLL reference clock frequency	Automatically calculates the PLL reference clock frequency for best performance. Uncheck the check box if you want to specify your own PLL reference clock frequency.

continued...

Display Name	Description
	Automatically calculates the example design core clock PLL reference clock frequency for best performance. Disable this parameter if you want to select a different reference clock frequency.
Reference clock frequency for example design core clock PLL	The PLL reference clock frequency in MHz for the PLL supplying the core clock. This parameter is used only in the example design PLL. Specify the externally provided reference clock frequency for the core clock PLL.

Related Information

- Clock Signals on page 36
- Non-zero Latency Backpressure on page 51

4.2.2. Controller Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP

The parameter editor contains one **Controller** tab for each memory channel that you specify on the **General** tab. The **Controller** tab allows you to select the HBM2 controller options that you want to enable.

Figure 8. Controller Tab

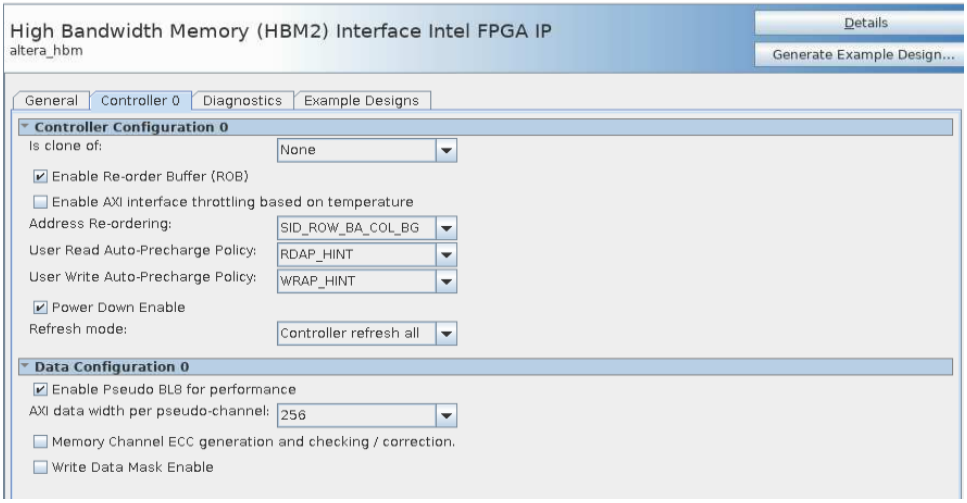


Table 9. Group: Controller/ Controller 0 Configuration

Display Name	Description
Is clone of	Set this option to make this controller a clone of the selected controller. Parameters are copied from the specified controller. This parameter applies when you select more than one HBM2 controller. Set this parameter if you want one controller to have the same settings as another.
Enable Re-order buffer	Specifies that read data be reordered to match the order in which transactions are issued. If you disable this feature, the Read Data provided at the AXI Interface is not expected to follow the same order in which the Read Requests were issued. You can then reorder the data, based on the AXI read ID of the transaction.

continued...

Display Name	Description
	This parameter applies to cases with multiple AXI transaction IDs. By using different AXI read/write IDs, you allow the HBM2 controller to reorder transactions for better efficiency. If you use the same AXI ID for all transactions, the controller issues the commands to memory in the order in which they arrive; in this instance, you need not enable the reorder buffer.
Enable AXI interface throttling based on temperature	Enables temperature-based throttling for this channel. You can specify the threshold temperature and throttling ratio on the General tab. Enables AXI thermal throttling for the specific HBM2 controller. The parameter that sets the trigger temperature for thermal throttling resides on the **General** tab. When the temperature of the HBM2 stack reaches the threshold temperature for HBM2 throttling as set on the *General* tab, the HBM2 controller begins to throttle user requests.

Address reordering	Describes the mapping of AXI address to HBM address. Specifies the pattern for mapping from the AXI interface to the HBM2 memory device. By choosing the right address reordering configuration, you help to improve the efficiency of accesses to the HBM2 memory device, based on user traffic pattern. The HBMC supports three types of address reordering:
	Address order (32B access: pseudo-BL8 disabled):
	`SID-BG-BANK-ROW-COL[5:1]`
	`SID-ROW-BANK-COL[5:1]-BG`
	`ROW-SID-BANK-COL[5:1]-BG`
	`COL[0]=0`
	Address order (64B access (pseudo-BL8 enabled):
	`SID-BG-BANK-ROW-COL[5:1]`
	`SID-ROW-BANK-COL[5:2]-BG-COL[1]`
	`ROW-SID-BANK-COL[5:2]-BG-COL[1]`
	`COL[1:0] = {00})`
	SID applies only to the 8GB/8H HBM2 devices and is not available for 4GB/4H devices.

User Read Auto-Precharge Policy	Describes the policy for determining whether to issue auto-precharge. FORCED indicates that the controller follows the user auto-precharge request exactly. HINT indicates that the controller may override the auto-precharge request by disabling it when a page hit is detected (that is, if it receives two commands, one with auto-precharge and one without auto-precharge to the same page, the controller changes auto-precharge for the first to 0 so that the second can access the page without reopening it). You can issue the request to precharge together with the read command, through the `axi_x_y_aruser` input, where *x* denotes the HBM2 channel number (0-7) and *y* denotes the HBM2 Pseudo Channel number (0/1). You can choose between two values for this parameter: • RDAP_FORCED mode, in which the HBM2 controller implements a user-requested auto-precharge command. • RDAP_HINT mode, in which the controller determines when to issue an auto-precharge command, based on user-issued auto-precharge input and the address specified.
User Write Auto-Precharge Policy	Describes the policy for determining whether to issue auto-precharge. FORCED indicates that the controller follows the user auto-precharge request exactly. HINT indicates that the controller may override the auto-precharge request by disabling it when a page hit is detected (that is, if it receives two commands, one with auto-precharge and one without

continued...

Display Name	Description
	auto-precharge to the same page, the controller changes auto-precharge for the first to 0 so that the second can access the page without reopening it). You can issue a precharge request together with the write command, through the `axi_x_y_awuser` input, where *x* denotes the HBM2 channel number (0-7) and *y* denotes the HBM2 Pseudo Channel number (0/1). You can choose between two values for this parameter: • WRAP_FORCED mode, in which the HBM2 controller implements a user-requested auto-precharge command. • WRAP_HINT mode, in which the controller determines when to issue an auto-precharge command, based on user-issued auto-precharge input and the address specified.
Power Down Enable	Causes the controller to power down when idle.
Refresh mode	Specifies the method of controlling refreshes to the high bandwidth memory. User refresh modes are initiated with an access on the Advanced Peripheral Bus (APB). You can choose one of three values for this parameter: • The default value is **Controller refresh all**, which allows the controller to decide when to issue refresh requests. • The **User refresh all** mode issues commands to all banks that correspond to an HBM2 Pseudo-Channel. **User refresh all** follows the APB protocol; the request must be addressed to one Pseudo-Channel at a time. A user-controlled refresh example is enabled in the design example when you select the **User refresh all** mode. • The **User refresh per-bank** mode issues commands to a specific bank that corresponds to an HBM2 Pseudo-Channel. **User refresh per-bank** follows the APB protocol; the request must be addressed to one Pseudo-Channel at a time.
Enable Pseudo BL8 for performance	If enabled, data access granularity is 64B (64 bits per Pseudo-Channel at BL8) which can achieve greater efficiency. Otherwise data access granularity is 32B (64 bits per Pseudo-Channel at BL4). The controller sets the memory transactions burst length based on this setting and ignores the input provided on the user-driven `axi_awsize` and `axi_arsize` signals on the AXI interface.
AXI data width per Pseudo-Channel	AXI data width per Pseudo-Channel. The ruser_data, wuser_data, and wuser_strb ports can be used for additional data if ECC and DM are disabled.
Memory channel ECC generation and checking/correction	Memory Channel ECC generation and checking / correction. The HBM2 controller supports single-bit error correction and double-bit error detection. The controller does not support write data mask in ECC generation mode.
Write data mask enable	Enables the write data mask (DM) input to the HBM2 DRAM. When you use the DM pins, you cannot use ECC.

4.2.3. Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP

The **Diagnostics** tab allows you to select traffic options and to enable the efficiency monitor that measures HBM2 controller efficiency during functional simulation.

Figure 9. Diagnostics Tab

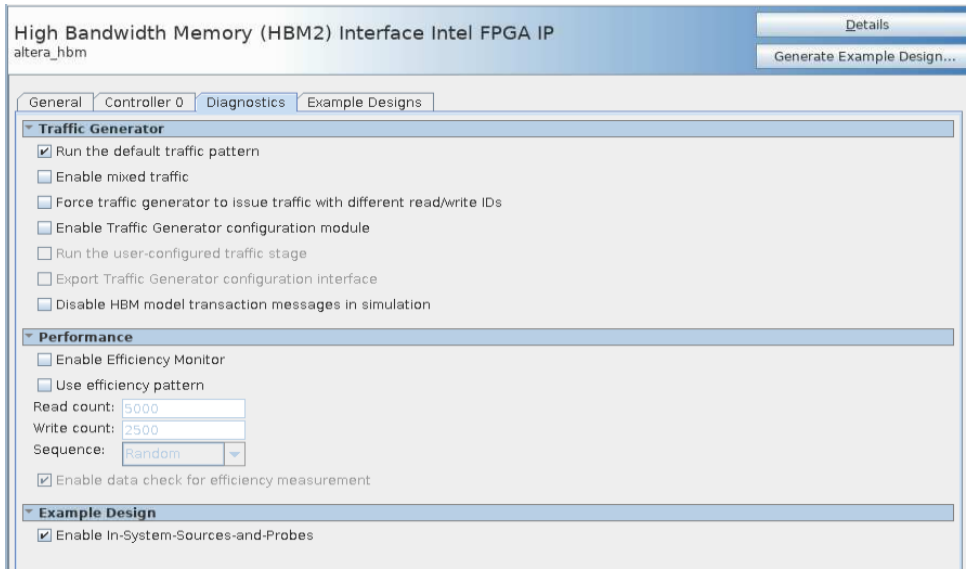


Table 10. Group: Diagnostics / Traffic Generator

Display Name	Description
Run the default traffic pattern	Runs the default traffic pattern after reset. The default traffic pattern consists of multiple stages testing single/block reads and writes using sequential and random addressing.
Enable mixed traffic	Configures the traffic generator to send out a variety of traffic patterns, including single and block reads/writes, using a mix of sequential and random addressing. If you do not enable this parameter, the traffic generator sends block reads/writes with sequential addressing. This parameter can help you understand the HBM2 interface performance over different traffic patterns.
Force traffic generator to issue traffic with different read/write IDs	Forces the traffic generator to issue traffic with different read/write IDs regardless of whether the reorder buffer is on. Using different read/write IDs allows the controller to reorder transactions for higher efficiency, but results in data mismatches if you have disabled the reorder buffer and the user logic does not handle read data returning out-of-order. When you do enable the reorder buffer, the traffic generator automatically generates transactions with different IDs. If you do not enable this parameter, the traffic generator does not issue AXI transactions with different read/write IDs, unless you have enabled the reorder buffer.
Enable Traffic Generator Configuration Module	Enables instantiation of the traffic generator configuration module, which is necessary only if you are creating custom traffic patterns.
Run the user-configured traffic stage	Runs the user-configured traffic pattern after reset. (You can still reconfigure the traffic generator later.) The traffic generator does not assert a pass or fail status until the Avalon configuration interface configures it and signals it to start. You can perform configuration by connecting to the traffic generator via the EMIF Debug Toolkit or by using custom logic connected to the Avalon-MM configuration

continued...

Display Name	Description
	slave port on the traffic generator. You can also simulate configuration with the example testbench provided in the `altera_hbm_tg_axi_tb.sv` file.
Export Traffic Generator Configuration Interface	Exports an Avalon-MM slave port for configuring the traffic generator. This is necessary only if you are configuring the traffic generator with user-configured traffic.
Disable HBM model transaction messages in simulation	If enabled, HBM model transaction messages are not displayed in simulation.

Table 11. Group: Diagnostics / Performance

Display Name	Description
Enable Efficiency Monitor	Adds an Efficiency Monitor component to the AXI interface of the memory controller. The Efficiency Monitor gathers and reports statistics on the efficiency of the interface during simulation.
Use efficiency pattern	The traffic generator generates a high-efficiency concurrent traffic pattern with features integrated in the design example.
Read count	Defines the read count for the traffic generator. The read and write count should be equal, for the validity check to pass.
Write count	Defines the write count for the traffic generator. The read and write count should be equal, for the validity check to pass.
Sequence	Defines the write and read sequence for the traffic generator, with the selection of *Random* or *Sequential*. (For best HBM2 efficiency, select *Sequential* for this parameter.)
Enable data check for efficiency measurement	Enables a data check for soft traffic generator efficiency measurement.

Table 12. Group: Diagnostics / Example Design

Display Name	Description
Enable in-System-Source-and-Probes	Enables In-System-Sources-and-Probes in the design example for common debug signals such as calibration status or example traffic generator per-bit status. You must enable this parameter if you want to do driver margining.

4.2.4. Example Designs Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP

The **Example Designs** tab allows you to configure example design files for simulation and synthesis.

Figure 10. Example Designs tab of HBM2 IP Parameters

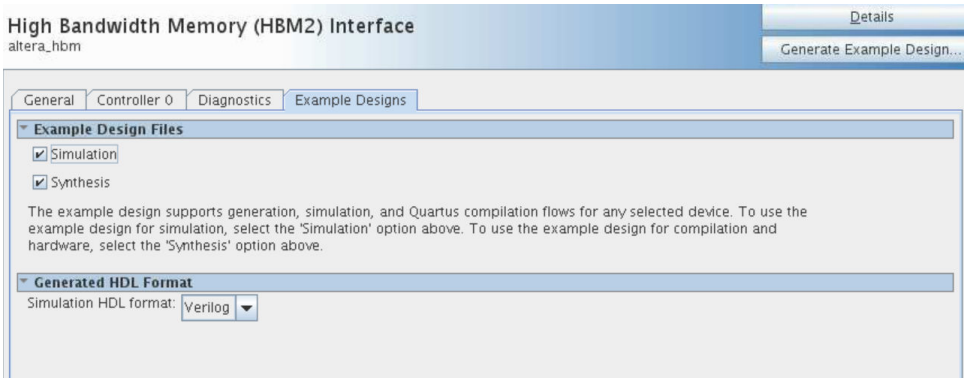


Table 13. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the system generate all necessary file sets for simulation when you click **Generate Example Design**. Expect an additional 1-2 minute delay when generating the simulation fileset. If you do not enable this parameter, the system does not generate simulation file sets. Instead, the output directory contains the `ed_sim.qsys` file which contains details of the simulation example design for Platform Designer, and a `make_sim_design.tcl` file with other corresponding tcl files. You can run the `make_sim_design.tcl` file from a command line to generate a simulation example design. The generated example designs for various simulators reside in the `/sim` subdirectory.
Synthesis	Specifies that the system generate all necessary file sets for synthesis when you click **Generate Example Design**. Expect an additional 1-2 minute delay when generating the synthesis file set. If you do not enable this parameter, the system does not generate synthesis file sets. Instead, the output directory contains the `ed_synth.qsys` file which contains details of the synthesis example design for Platform Designer, and a `make_qii_design.tcl` file with other corresponding tcl files. You can run the `make_qii_design.tcl` file from a command line to generate a synthesis example design. The generated example design resides in the `/qii` subdirectory.

Tip: The example design supports generation, simulation, and Intel Quartus Prime compilation flows for any selected device. To use the example design for simulation, enable the **Simulation** parameter. To use the example design for compilation and hardware, enable the **Synthesis** parameter.

Table 14. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	Format of HDL files generated for simulating the design example.

Related Information

High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example User Guide

4.3. Pin Planning for the High Bandwidth Memory (HBM2) Interface Intel FPGA IP

The High Bandwidth Memory (HBM2) Interface Intel FPGA IP requires the following clock inputs:

- UIB PLL reference clock – Reference clock input for the UIB PLL. There is one UIB PLL reference clock per HBM2 interface.
- Core clock input – Fabric core clock, generated through an I/O PLL.

For maximum performance, the reference clock that drives the core I/O PLL should come from the same oscillator as that supplying the UIB PLL reference clock on the board for a given HBM2 interface.

Table 15. Placement Requirements for PLL Reference Input Clocks

Signal	Description	Pin Placement Guidelines
pll_ref_clk	DC-coupled LVDS differential input clock used by the hardened UIB-HBM2 subsystem.	Place this reference clock input on the UIB_PLL_REF_CLK_00 pins while using the HBM2 device on the bottom of the FPGA, or the UIB_PLL_REF_CLK_01 pins while using the HBM2 on the top of the FPGA.
ext_core_clk	LVDS differential input clock used to generate the fabric core clock. Instantiate the I/O PLL that shall generate the core clock for the HBM2 IP.	Place the reference clock input on *CLK_* pins to access the I/O PLL. You should select pins that are close to the UIB_PLL_REF_CLK input. You must instantiate the I/O PLL in the design flow. The output of the I/O PLL serves as the EXT_CORE_CLK.

Jitter Specifications for the Input Reference Clocks

Both the reference clock inputs should meet and not exceed the following time interval error (TIE) jitter requirements:

- 20ps peak-to-peak
- 1.42ps RMS at 1e-12 BER
- 1.22ps at 1e-16 BER

5. Simulating the High Bandwidth Memory (HBM2) Interface Intel FPGA IP

This section describes how to simulate the generated HBM2 IP.

Simulation Assumptions

The parameter settings that you make on the **Controller** tab affect efficiency during simulation. In the default configuration, with the default parameter settings, the traffic generator issues sequential transactions.

Supported Simulators

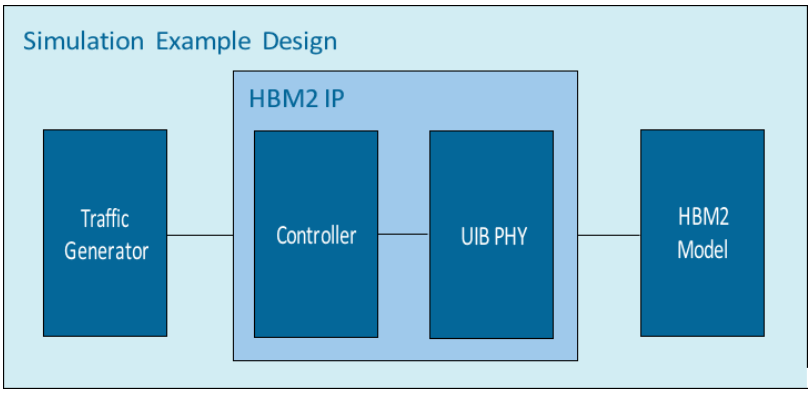
The HMB2 IP supports the following simulators:

- ModelSim*- Intel FPGA Edition
- ModelSim SE
- Questa* Advanced Simulator
- NCSim*
- Aldec Riviera-PRO*
- Synopsys* VCS
- Xcelium* Parallel Simulator

5.1. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Example Design

The following illustration shows a high-level block diagram of the HBM2 example design that provides the simulation environment for the High Bandwidth Memory (HBM2) Interface Intel FPGA IP MX HBM2 IP when generated for simulation.

Figure 11. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Generated for Simulation



The Traffic Generator emulates a real-world application that writes to, and reads back from memory and validates the read data. You can modify the traffic generator logic to fit your traffic pattern or drive the transactions to the HBM2 memory with your own logic.

Simulation incorporates an abstract model of the hardened HBM2 controller and the universal interface block (UIB). The HBM2 controller performs data reordering and enhancement functions, and allows communication between the AXI4 user interface and the UIB PHY. The universal interface block PHY (UIB PHY) is the physical-layer interface that carries low-level signaling.

The HBM2 Model is an abstract generic model representative of the HBM2 DRAM for simulation. This is not a vendor-specific model.

5.2. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP with ModelSim* and Questa*

1. Launch the ModelSim simulator.
2. Select **File ➤ Change Directory**and navigate to: *project_directory*`/sim/ed_sim/sim/mentor`
3. Verify that the **Transcript** window is visible; if it is not, display it by selecting **View ➤ Transcript**.
4. In the **Transcript** window, run `source msim_setup.tcl` at the bottom of the ModelSim tool screen.
5. After the Tcl script finishes running, run `ld_debug` in the **Transcript** window**.** This command compiles the design files and elaborates the top-level design.
6. After `ld_debug` finishes running, the **Objects** window appears. In the **Objects** window, select the signals to simulate by right-clicking and selecting **Add Wave** from the context menu.
   For example, if you want to see the HBM2 interface signals, select the module `mem0_0` from the **Instance** window. With `mem0_0` selected, go to the **Objects** window and select the signals that you want to see. (If the **Objects** window is not visible, you can display it by selecting **View ➤ Objects**.
7. To run the HBM2 simulation, type `run -all`.

If the simulation is not visible, select **View ➤ Wave**. With the **Wave** window open, select **File ➤ Save Format**. Click **OK** to capture your selected waveforms in a `wave.do` file. To display the waveforms, type `do wave.do`, and then type `run -all`.
Whenever you make changes to the design or to the `wave.do`, you must repeat step 7 of this procedure. Alternatively, you can combine the instructions into a script and run that script instead. The following example illustrates a `run.do` script containing the necessary commands:

```
if {[file exists msim_setup.tcl]} {
       source msim_setup.tcl
      ld_debug
       do wave.do
      run -all } else {        error "The msim_setup.tcl script does not
exist.
Please generate the example design RTL and simulation scripts. See ../../
README.txt
for help." }
```

Save the `run.do` script in the same directory as the `msim_setup.tcl` file. Type `do run.do` to run this script from the **Transcript** window.

8. Upon completion of the simulation, the **Transcript** window displays efficiency data and other useful information.

5.3. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP with Synopsys VCS*

1. Navigate to the *project_directory*`/hbm_0_example_design/sim/ed_sim/sim/synopsys/vcs` directory.
2. To run the simulation, type `sh vcs_setup.sh`. To view the simulation results, write the output to a log file. The simulation log provides efficiency data and other useful information.
3. To view the waveform, add `+vcs+dumpvars+test.vcd` to the `vcs` command.
4. To view the waveform, type `dve &` to launch the waveform viewer. Add the necessary signals or module to the waveform view to view the required signals.

5.4. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP with Riviera-PRO*

1. Navigate to: *project_directory*`>/sim/ed_sim/aldec`.
2. Type `rungui` to launch the Riviera-PRO simulator.
3. Type `source rivierapro_setup.tcl`.
4. Type `ld_debug` to compile the design files and elaborate the top-level design.
5. Type `run -all` to run the HBM2 simulation.

5.5. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP with Cadence NCSim*

1. Navigate to: `project_directory>/sim/ed_sim/cadence`.
2. Type `sh ncsim_setup.sh` to launch the NCSim simulator.
3. To view the simulation results, write the output to a log file. The simulation log provides efficiency data and other useful information.

5.6. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP with Cadence Xcelium* Parallel Simulator

1. Navigate to: `project_directory>/sim/ed_sim/xcelium`.
2. Type `sh xcelium_setup.sh` to launch the Xcelium simulator.
3. To view the simulation results, write the output to a log file. The simulation log provides efficiency data and other useful information.

5.7. Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP for High Efficiency

The default traffic pattern can achieve high efficiency by efficiently utilizing the HBM2 memory bandwidth and providing an efficient flow of traffic between the HBM2 controller and AXI user interface.

The main steps to deriving higher efficiency are:

- Turn off **Enable Reorder Buffer** on the **Controller** tab. The Reorder Buffer rearranges the read data in the order of the issued requests.
- Turn on **Force traffic generator to issue different AXI Read/Write IDs** and **Enable Efficiency Test Mode** on the **Diagnostics** tab. In this configuration, the traffic generator issues concurrent read and write transactions; consequently, you may receive data mismatch warnings, which you can ignore.
- Turn on **Use efficiency pattern** and **Enable data check for efficiency measurement** on the **Diagnostics** tab, to have the traffic generator use an efficiency pattern to test both synthesis and simulation designs. Ensure that the **Read count** and **Write count** values are equal, for the validity check to pass. For best HBM2 efficiency, select a **Sequence** value of *Sequential*.

The following sections explain the **General**, **Controller**, and **Diagnostic** tab parameters required to perform high efficiency HBM2 simulation. The following figures illustrate parameter settings for a high-efficiency simulation for a single-channel HBM2 controller.

For more information on improving controller efficiency, refer to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Performance*.

Figure 12. Controller Tab Settings for High Efficiency Simulation

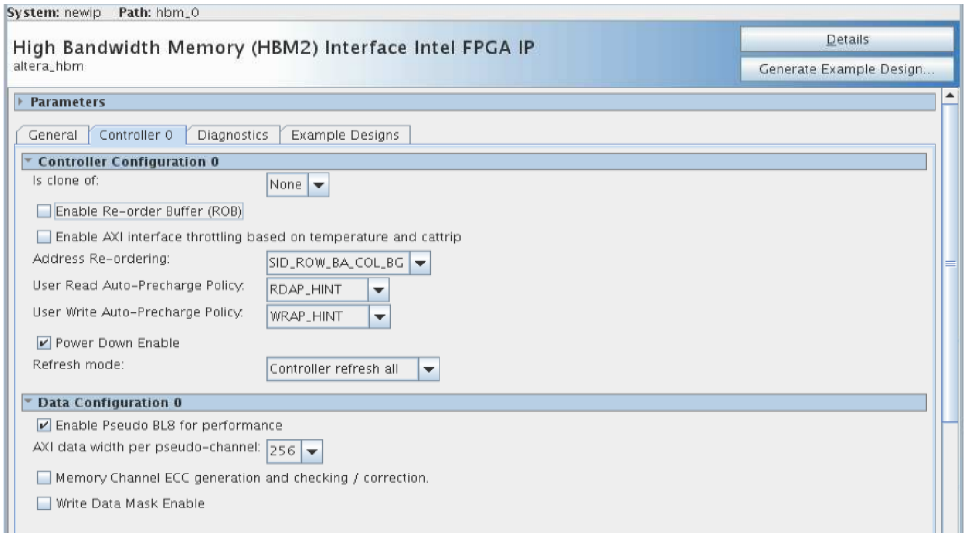
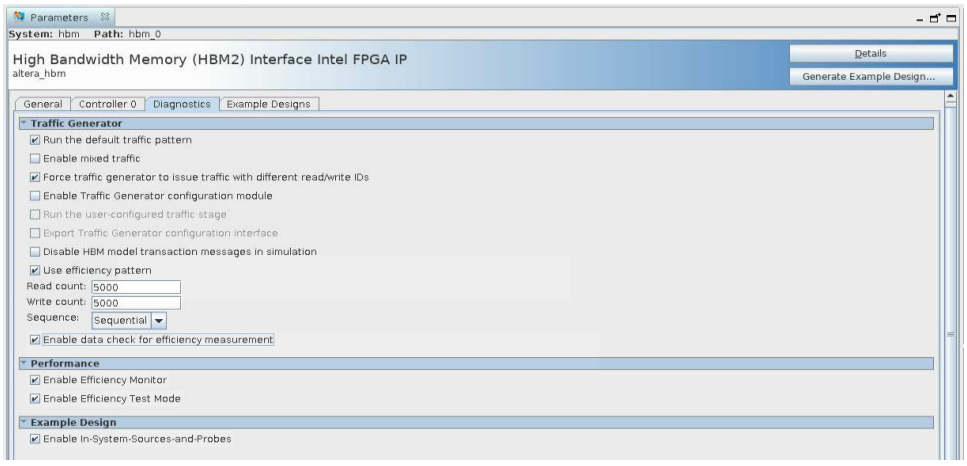


Figure 13. Diagnostics Tab Settings for High Efficiency Simulation



5.8. Simulating High Bandwidth Memory (HBM2) Interface IP Instantiated in Your Project

This topic outlines the flow for simulating the HBM2 IP instantiated in your project, rather than the HBM2 design example.

1. In your simulation project, include <*project_directory*>\<*IP_name*>\sim\<*IP_name*>.v
2. The generated HBM2 IP simulation file set does not include an HBM2 memory model file; you must add a memory model file to the project. Intel recommends that you use the memory model from the design example simulation file set generated from your IP: `hbm_0_example_design\sim\ip\ed_sim\ed_sim_mem\sim\ed_sim_mem.v`

6. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface

This chapter provides an overview of the signals that interface to the HBM2 IP.

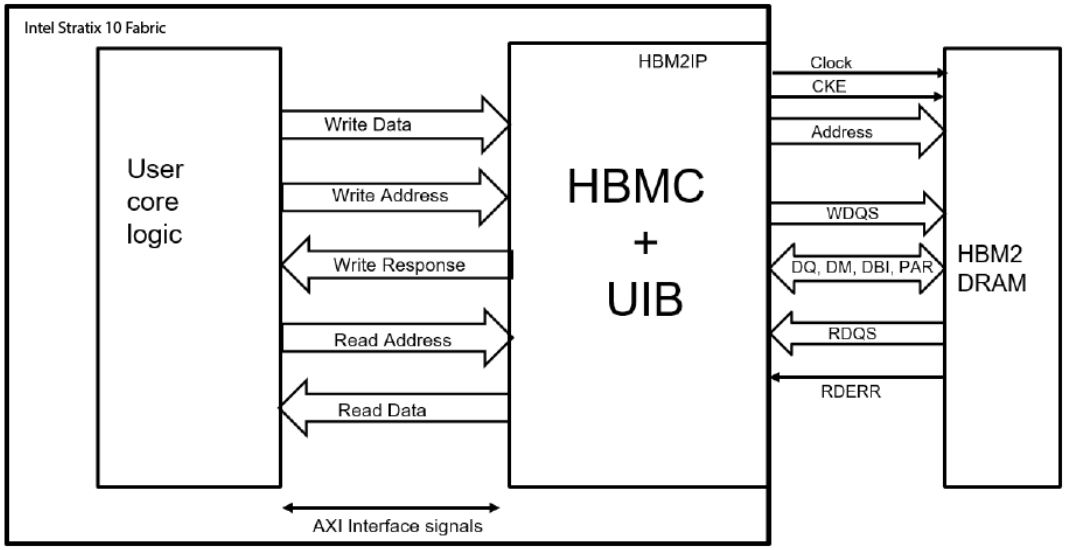
Related Information

Intel Stratix 10 MX UIB Architecture on page 9

6.1. High Bandwidth Memory (HBM2) Interface Intel FPGA IP High Level Block Diagram

The following figure shows a high-level block diagram of the High Bandwidth Memory (HBM2) Interface Intel FPGA IP per Pseudo Channel. The IP communicates with user logic through the AXI protocol.

Figure 14. High Level Block Diagram of HBM2 Implementation



6.2. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Interface Signals

This section lists the signals that connect core logic to the High Bandwidth Memory (HBM2) Interface Intel FPGA IP.

6.2.1. Clock Signals

Figure 15. HBM2 IP Clocking and Reset Diagram

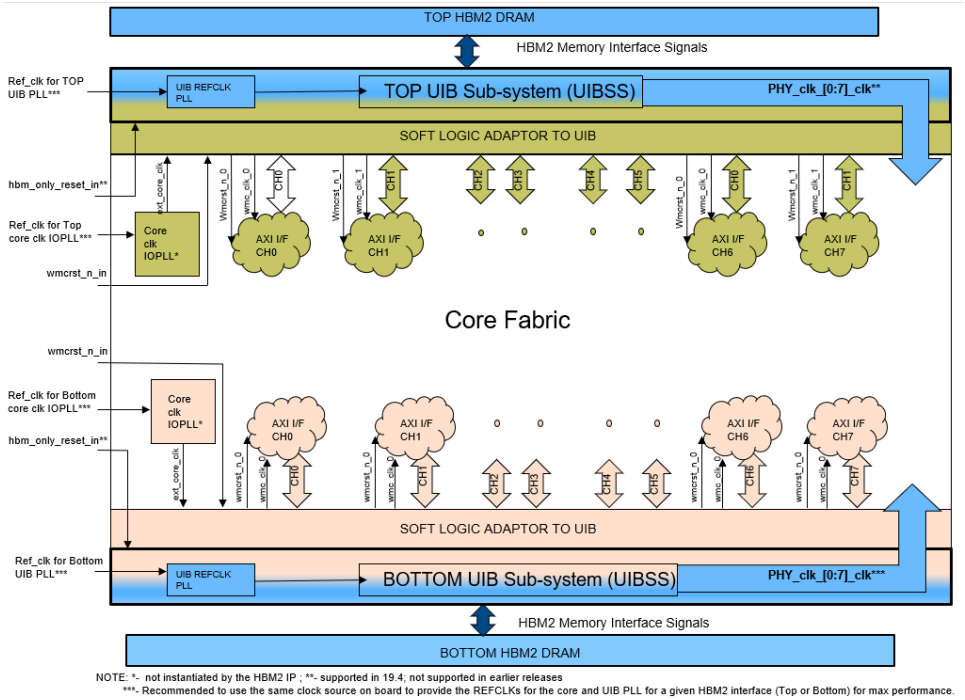


Table 16. Clock Signals

Signal	Direction	Description
ext_core_clk	Input	Core clock. Output of user-instantiated I/O PLL. The HBM2 IP does not instantiate the I/O PLL required to generate the ext_core_clk. For maximum performance, the reference clock that drives the core I/O PLL should come from the same oscillator as that supplying the UIB PLL reference clock on the board for a given HBM2 interface.
ext_core_clk_locked	Input	LOCKED status of I/O PLL driving ext_core_clk, indicating the ext_core_clk is stable. This I/O PLL should achieve LOCKED status within 1ms of device configuration for calibration to be successful. The I/O PLL driving the ext_core_clk cannot be reset after calibration is completed.
pll_ref_clk	Input	LVDS reference clock input for UIB PLL. This reference clock must be provided through dedicated UIB_PLL_REF_CLK_p/n pins available in Intel Stratix 10 MX devices and cannot be provided through an internal I/O PLL. The pll_ref_clk must be stable and free-running at device power-up for successful configuration. Refer to

continued...

Signal	Direction	Description
		Intel Stratix 10 pin connection guidelines for information on how to supply these clocks.
wmc_clk_x_clk	Output	Core clock output feedback from UIB, based on ext_core_clk provided, one per HBM2 Channel (represented by *x*). Intel recommends that the user interface drive the AXI interface for the corresponding channels with this clock.
phy_clk_x_clk	Output	UIB PHY clk output. Not supported. Leave unconnected. This signal appears as `phy_clk_x` conduit in the Platform Designer.

Clocking recommendations for Reliable Calibration of the HBM2 Interface

Observe the following clock guidelines for reliable calibration of the HBM2 interface:

- The UIB PLL reference clocks (one per HBM2 interface) must be provided through an external clock source and must be stable and free running prior to configuration and stable thereafter.
- The I/O PLL driving the core clock (ext_core_clk) should achieve a LOCKED condition within one millisecond of device configuration. This I/O PLL cannot be reset once the interface is calibrated.
- For a given HBM2 interface, Intel recommends that the same clock source provide the reference clocks for both the I/O PLL and the UIB PLL.

Related Information

- Intel Stratix 10 MX HBM2 Controller Features on page 7
- Intel Stratix 10 MX HBM2 Controller Details on page 13
- General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP on page 19
- High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example

6.2.2. Reset Signals

Table 17. Reset Signals

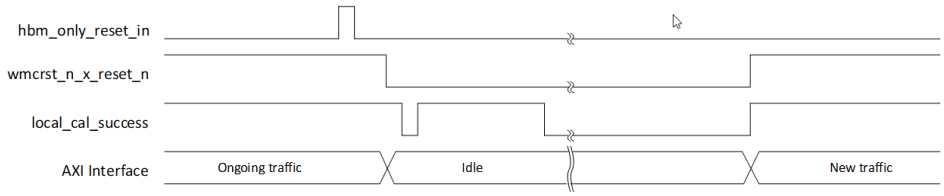
Signal	Direction	Description
wmcrst_n_in	Input	Core logic external input reset, active LOW. Resets all fabric logic, including the Soft Logic Adaptor. This is an asynchronous reset signal. You should assert this reset only when the AXI Interface is idle and there are no pending memory transactions.
hbm_only_reset_in	Input	Reset request for the soft fabric AXI logic as well as the hardened UIBSS, including the HBM Controller and PHY. From the Intel Quartus Prime software version 19.4 onwards, Intel recommends to use only hbm_only_reset_in whenever you need to reset the HBM subsystem, both during active traffic to the DRAM or when the interface is idle. (See additional information immediately following this table.)
wmcrst_n_x_reset_n	Output	Core input reset synchronized to the AXI interface clock domain. One per AXI Channel (represented by x). Intel recommends that the user interface use the respective channel reset outputs in the logic that drives the AXI interfaces for the corresponding channels.

hbm_only_reset_in Timing

The **hbm_only_reset_in** signal requests the reset of the soft AXI logic as well as the hardened UIBSS, without initiating a recalibration. This signal is supported only in the Intel Quartus Prime software version 19.4 and later.

The following waveform illustrates the timing associated with this signal.

Figure 16. hbm_only_reset_in Timing



The following sequence explains the requirements and timing of the hbm_only_reset_in signal:

- The hbm_only_reset_in signal is internally synchronized to the core clock (ext_core_clk). This reset input detects the rising edge. The minimum requirement is to assert this signal for one core clock cycle.
- wmcrst_n_x_reset_n is reset output synchronized to the core clock. Intel recommends that you connect user logic reset to this reset output so that AXI traffic can be stopped during the reset sequence.
- The start of the reset sequence is indicated by wmcrst_n_x_reset_n going low.
- The end of the reset sequence is indicated by wmcrst_n_x_reset_n going high.
- The hbm_only_reset_in can be asserted again after wmcrst_n_x_reset_n goes high (reset completed). Any assertion of hbm_only_reset_in *during* the reset sequence is ignored.
- The local_cal_success signal deasserts for several core clock cycles after the reset sequence begins, however this does not perform a recalibration.

Reset Recommendations for Reliable Calibration of the HBM2 Interface

Observe the following reset signal guidelines for reliable calibration of the HBM2 interface:

- Ensure that the Reset Release Intel FPGA IP is instantiated in your design. This IP helps to hold the FPGA in reset until all registers and core logic are in user mode. Intel recommends that you use the ninit_done output of the Reset Release Intel FPGA IP as one of the initial inputs to your reset circuit. You can find this IP in the IP Catalog under **Basic Functions ➤ Configuration and Programming ➤ Reset Release Intel FPGA IP**. You can find more information on using this IP, in ***AN 891: Using the Reset Release Intel FPGA IP***, available here: https://www.intel.com/content/www/us/en/programmable/documentation/prh1555609801770.html#iuj1556119879221.

  Use the ninit_done signal output of this IP to gate the core input reset (wmcrst_n_in). The ninit_done output is active low; that is, `ninit_done = 0` after FPGA enters user mode.
- Ensure that wmcrst_n_in (active low soft logic reset input) and hbm_only_reset_in (active high reset signal for soft logic as hard logic of the HBM subsystem) are not asserted until after completion of calibration.
- Commencing with the Intel Quartus Prime software version 19.4, Intel recommends using hbm_only_reset_in whenever you need to reset the HBM subsystem, both during active traffic to the DRAM or when the interface is idle.
- The I/O PLL that generates ext_core_clk, the core AXI interface input clock, cannot be reset once the I/O PLL has achieved a locked condition.
- The wmcrst_n_*x*_reset_n signal that is driven as an output from the UIBSS per HBM2 channel (represented by *x*) is to be used by the user AXI interface along with the wmc_clk_*x*_clk provided per channel as shown in Figure 15 on page 36.

Related Information

High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example

6.2.3. Calibration Status Signals

Table 18. Calibration Status Signals

Signal	Direction	Description
local_cal_success	Output	Indicates calibration success.
local_cal_fail	Output	Indicates calibration failure.
cal_lat	Output	Calibrated latency output. Not supported.

6.2.4. Memory Interface Signals

The following HBM2 memory signals are driven by the HBM2 controller through the UIBSS; you do not need to drive these signals.. These signals are provided at the top level, for successful compilation.

Table 19. HBM2 Memory Interface Signals

Signal	Direction	Width	Description
Cattrip	Input	1	HBM2 signals common to each HBM2 interface; these signals must be brought out to the design top level. You do not need to drive these signals and can leave them unconnected. These signals are grouped under a conduit named `m2u_bridge` in the Platform Designer, and must be exported.
Temp	Input	3	
Wso	Input	8	
Reset_n	Output	1	
Wrst_n	Output	1	
Wrck	Output	1	
Shiftwr	Output	1	
Capturewr	Output	1	
Selectwir	Output	1	
Wsi	Output	1	
Ck_t	Output	1	HBM2 signals per HBM2 channel. You do not need to drive these signals and can leave them unconnected. These signals are grouped under a conduit named `mem_x` in the Platform Designer. You should leave this conduit unconnected and ignore any warning message about `mem_x` being unconnected.
Ck_c	Output	1	
Cke	Output	1	
C	Output	8	
R	Output	6	
Dq	Inout	128	
Dm	Inout	16	
Dbi	Inout	16	
Par	Inout	4	
Derr	Inout	4	
Rdqs_t	Input	4	
Rdqs_c	Input	4	
Wdqs_t	Output	4	

continued...

Signal	Direction	Width	Description
Wdqs_c	Output	4	
Rd	Inout	8	
Rr	Output	1	
Rc	Output	1	
Aerr	Input	1	

6.2.5. AXI User-interface Signals

The user interface to the HBM2 controller follows the Amba AXI4 protocol specification. Each AXI port serves the read and write operations for one Pseudo Channel. Each HBM2 channel consists of two Pseudo Channels, therefore each controller has two AXI ports.

AXI Interface Signals

Each AXI port consists of five subchannels:

- Write Address Channel –AXI Write Address that maps to the HBM2 DRAM Write Address.
- Write Data Channel – AXI Write data provided by the core logic corresponding to the Write Address.
- Write Response Channel – Response from the HBM2 controller on the status of the Writes.
- Read Address Channel – AXI Read Address that maps to the HBM2 DRAM Read Address.
- Read Data Channel – AXI Read Data provided from the corresponding HBM2 DRAM Read Address.

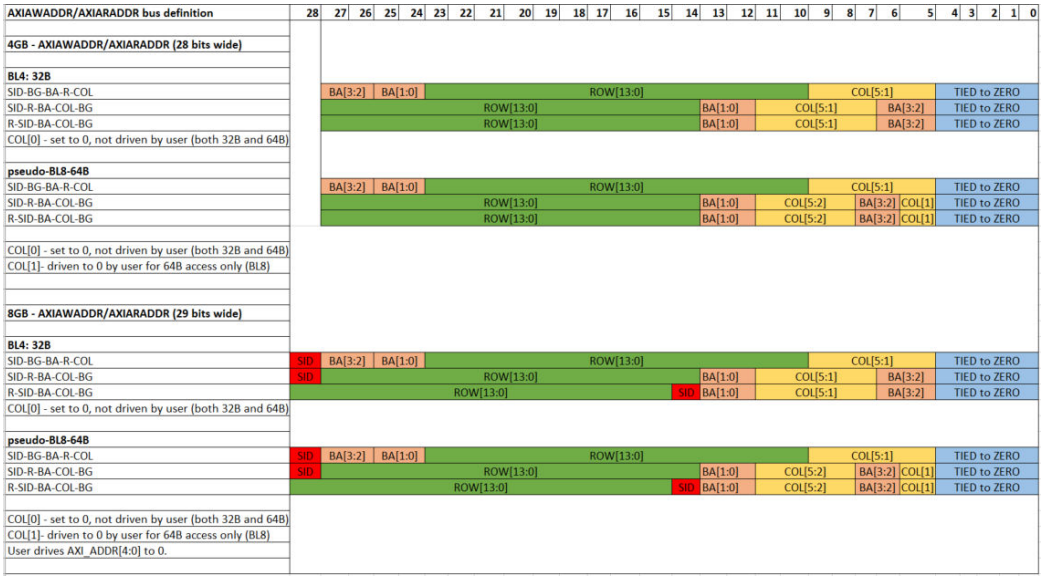
AXI Address Definition

The HBM2 DRAM addressing consists of the following:

- 14-bit wide Row Address.
- 6-bit wide Column Address. The user logic drives Column Address bits COL[5:1], while the controller sets the lower order Column Address bit COL[0] to 0. For BL8 transactions, the user logic sets COL[1] to 0.
- 4-bit wide Bank Address. Bank Group corresponds to the higher order 2 bits of the Bank Address (BA[3:2]).
- 1-bit wide Stack ID(SID) available only in the 8GB configuration.

The following figure illustrates the mapping of the AXI Address bus (28-bit wide for 4GB configurations and 29-bit wide for 8GB configurations) for the various address ordering schemes to address the HBM2 DRAM.

Figure 17. AXI Address Definition



AXI Subchannels Descriptions

The syntax for referencing AXI port signal names is *axi_x_y_portname* where *x* is the channel number and *y* is the Pseudo Channel number. For example, `axi_0_1_awid` refers to the write address ID of the AXI port corresponding to channel 0 and Pseudo Channel 1.

The signals in the following tables refer to the signal names corresponding to a single AXI port: Channel 0, Pseudo Channel 0.

Table 20. User Port 0's AXI4 Write Address (Command) Channel

Port Name	Width	Direction	Description
axi_0_0_awid	9	Input	Write address ID. This signal is the ID tag for the write address group of signals.
axi_0_0_awaddr	28/29	Input	Write address. The write address gives the address of the first transfer in a write burst transaction. This address bus is 28 bits wide for a 4 GB device and 29 bits for an 8 GB HBM2 device.
axi_0_0_awlen	8	Input	Burst Length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. **Value** / **Burst Length** 0b00000000 / 1 0b00000001 / 2 Burst length 1 refers to burst-length-4 transactions at the DRAM interface. This requires one 256-bit data transfer at the AXI Interface.

continued...

<table>
<tr><th>Port Name</th><th>Width</th><th>Direction</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>Burst length 2 refers to burst-length-8 transactions at the DRAM interface. This requires two 256-bit transfers at the AXI interface.
The AXI interface supports only one burst transfer at a time, based on HBM2 burst transaction of 4 or 8 selected through the HBM2 IP parameter editor. The HBM2 controller sets the Burst Length value regardless of the value driven on the axi_awlen port.</td></tr>
<tr><td>axi_0_0_awsize</td><td>3</td><td>Input</td><td>Burst Size. This signal indicates the size of each transfer in the burst.<table><tr><th>Value</th><th>Burst Size (Bytes)</th></tr><tr><td>0b101</td><td>32</td></tr></table>The HBM2 controller supports only 256 bits, or 32 Bytes, of data transfers at the AXI Interface per AXI clock cycle. The HBM2 controller sets the value of axi_awsize and ignores any non-supported value driven through the axi_awsize port.</td></tr>
<tr><td>axi_0_0_awburst</td><td>2</td><td>Input</td><td>The burst type and the size information, determined how the address for each transfer within the burst is calculated.<table><tr><th>Value</th><th>Burst Type</th></tr><tr><td>2’b01</td><td>INCR</td></tr></table>Only INCR is supported. The HBM2 controller sets the value of axi_awburst and ignores any non-supported value driven through the axi_awburst port.</td></tr>
<tr><td>axi_0_0_awprot</td><td>3</td><td>Input</td><td>Protection Type. [Reserved for Future Use]
This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
3'b000 = No protection</td></tr>
<tr><td>axi_0_0_awqos</td><td>4</td><td>Input</td><td>Quality of Service. The Quality of Service identifier sent for each write transaction.
4'b1111 = High priority
4'b0000 = Normal priority</td></tr>
<tr><td>axi_0_0_awuser</td><td>1</td><td>Input</td><td>User Signal for auto-precharge.
1'b0 = No auto-precharge
1'b1 = Auto-precharge
Optional user-defined auto-precharge signal in the write address channel.</td></tr>
<tr><td>axi_0_0_awvalid</td><td>1</td><td>Input</td><td>Write Address Valid. Indicates that the channel is signaling valid write address and control information.</td></tr>
<tr><td>axi_0_0_awready</td><td>1</td><td>Output</td><td>Write Address Ready. Indicates that the slave is ready to accept an address and associated control signals.</td></tr>
</table>

Table 21. User Port 0's AXI4 Write Data Channel

Port Name	Width	Direction	Description
axi_0_0_wdata	256	Input	Write Data.
axi_0_0_wstrb	32	Input	Write Strobes (Byte Enables). Indicates which byte lanes (for axi_0_0_wdata) hold valid data. There is one write strobe bit for every eight bits of write data.
axi_extra_0_0_wuser_data	32	Input	Extra Write Data (AXI WUSER port). Carries additional write data when AXI Data Width of 288-bits data is selected in the HBM2 IP GUI. When 256 bits data is chosen in the GUI, this bus should be left undriven.
axi_extra_0_0_wuser_strb	4	Input	Extra Write Strobes (AXI WUSER port). Indicates which byte lanes (for axi_extra_0_0_wuser_data) hold valid data. This bus is used only when the AXI Data width of 288 bits is chosen in the HBM2 IP GUI. These signals should be left undriven when the data width is selected as 256 bits.
axi_0_0_wlast	1	Input	Write Last. Indicates the last transfer in a write burst.
axi_0_0_wvalid	1	Input	Write Valid. Indicates that valid write data and strobes are valid.
axi_0_0_wready	1	Output	Write Ready. Indicates that the slave (HBM2 controller) can accept write data.

Table 22. User Port 0's Write Response Channel

Port Name	Width	Direction	Description
axi_0_0_bid	9	Output	Response ID Tag. The ID tag of the write response.
axi_0_0_bresp	2	Output	Write response. Indicates the status of the write transaction. • 2'b00 = OKAY; indicates that normal access is successful.
axi_0_0_bvalid	1	Output	Write response valid. Indicates that the channel is signaling a valid write response.
axi_0_0_bready	1	Input	Response ready. Indicates that the master can accept a write response.

Table 23. User Port 0's AXI4 Read Address (Command) Channel

Port Name	Width	Direction	Description
axi_0_0_arid	9	Input	Read address ID. The ID tag for the read address group of signals.
axi_0_0_araddr	28/29	Input	Read address. The address of the first transfer in a read burst transaction. This address bus is 28-bits wide for a 4 GB device and 29-bits wide for an 8 GB device.
axi_0_0_arlen	8	Input	Burst Length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.

continued...

<table>
<tr><th>Port Name</th><th>Width</th><th>Direction</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Burst Length</th></tr>
<tr><td>0b00000000</td><td>1</td></tr>
<tr><td>0b00000001</td><td>2</td></tr>
</table>
Burst length 1 refers to burst-length-4 transactions at the DRAM interface. This requires one 256-bit data transfer at the AXI Interface.
Burst length 2 refers to burst-length-8 transactions at the DRAM interface. This requires two 256-bit transfers at the AXI interface.
The AXI interface supports only one burst transfer at a time, based on HBM2 burst transaction of 4 or 8 selected through the HBM2 IP parameter editor. The HBM2 controller sets the Burst Length value regardless of the value driven on the axi_arlen port.
</td></tr>
<tr><td>axi_0_0_arsize</td><td>3</td><td>Input</td><td>
Burst Size. This signal indicates the size of each transfer in the burst.
<table>
<tr><th>Value</th><th>Burst Size (Bytes)</th></tr>
<tr><td>0b101</td><td>32</td></tr>
</table>
The HBM2 controller supports only 256-bits, or 32 Bytes, of data transfers at the AXI Interface per AXI clock cycle. The HBM2 controller sets the value of axi_arsize and ignores any non-supported value driven through the axi_arsize port.
</td></tr>
<tr><td>axi_0_0_arburst</td><td>2</td><td>Input</td><td>
The burst type and the size information, determined how the address for each transfer within the burst is calculated.
<table>
<tr><th>Value</th><th>Burst Type</th></tr>
<tr><td>2’b01</td><td>INCR</td></tr>
</table>
Only INCR is supported. The HBM2 controller sets the value of axi_arburst and ignores any non-supported value driven through the axi_0_0_arburst port.
</td></tr>
<tr><td>axi_0_0_arprot</td><td>3</td><td>Input</td><td>Protection Type. [Reserved for Future Use] Indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
• 3'b000 = No protection</td></tr>
<tr><td>axi_0_0_arqos</td><td>4</td><td>Input</td><td>Quality of Service. The Quality of Service identifier sent for each write transaction.
• 4’b1111 = High priority
• 4’b0000 = Normal priority</td></tr>
<tr><td>axi_0_0_aruser</td><td>1</td><td>Input</td><td>User Signal for auto-precharge.</td></tr>
</table>

continued...

Port Name	Width	Direction	Description
			• 1'b0 = No auto-precharge • 1'b1 = Auto-precharge
axi_0_0_arvalid	1	Input	Read address valid. Indicates that the channel signals valid read address and control information.
axi_0_0_arready	1	Output	Read address ready. Indicates that the slave is ready to accept an address and associated control signals.

Table 24. User Port 0's Read Data Channel

Port Name	Width	Direction	Description
axi_0_0_rid	ARID_WIDTH	Output	Read ID tag. The ID tag for the read data group of signals generated by the slave.
axi_0_0_rdata	256	Output	Read data.
axi_extra_0_0_ruser_data	32	Output	Extra Read Data (AXI RUSER port). Carries the extra read data when the data width of 288 is chosen in the HBM2 IP GUI. This bus can be left unconnected when a 256 Data option is chosen in the HBM2 IP GUI.
axi_extra_0_0_ruser_err_dbe	1	Output	Double-Bit-Error (AXI RUSER port). Carries DBE information. 1'b1 indicates error.
axi_0_0_rresp	2	Output	Read response. Indicates the status of the read transfer: • 2'b00 = OKAY
axi_0_0_rlast	1	Output	Read last. Indicates the last transfer in a read burst.
axi_0_0_rvalid	1	Output	Read valid. Indicates that the channel is signaling the required read data. All AXI Read data Channel output signals that output from the HBM2 Controller are provided aligned to the axi_0_0_rvalid signal.
axi_0_0_rready	1	Input	Read ready. Indicates that the master can accept the read data and response information.

Related Information

High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example

6.2.6. Sideband APB Interface

The sideband APB interface allows user logic to issue refresh commands and also provides access to the controller status signals. Each HBM2 channel has one APB Interface.

Table 25. APB Interface Signals

Port Name	Width	Direction	Description
ur_paddr	16	Input	APB address bus. You can use the APB address bus to access the MMR register space, to issue specific user-requested commands.
ur_psel	1	Input	Select. The user interface generates this signal to indicate that the channel APB interface is selected and that a data transfer is required. There is a PSEL signal for each HBM2 channel APB interface and this signal can be tied HIGH.
ur_penable	1	Input	Enable. This signal indicates the start of an APB transfer.
ur_pwrite	1	Input	Write/Read access. When this signal is HIGH, it indicates an APB write access. When this signal is LOW, it indicates an APB read access.
ur_pwdata	16	Input	Write data. This signal is driven by user logic during write cycles when PWRITE is HIGH.
ur_pstrb	2	Input	Write strobes (byte enables). This signal indicates which byte lanes to update during a write transfer. There is one write strobe for each eight bits of the write data bus. Write transfers to the HBM2 controller require both the byte enables to be active and hence must be driven to 2'b11. Write strobes must not be active during a read transfer.
ur_prready	1	Output	Ready. This signal indicates the completion of a write or read transaction.
ur_prdata	16	Output	Read data. The read data bus provides useful HBM2 controller information and status signals.

6.3. User AXI Interface Timing

This section explains the interface timing details between user logic and the HBM2 controller. User interface signals follow the AXI4 protocol specification while passing data to and from the HBM2 controller.

The AXI interface consists of the following channels:

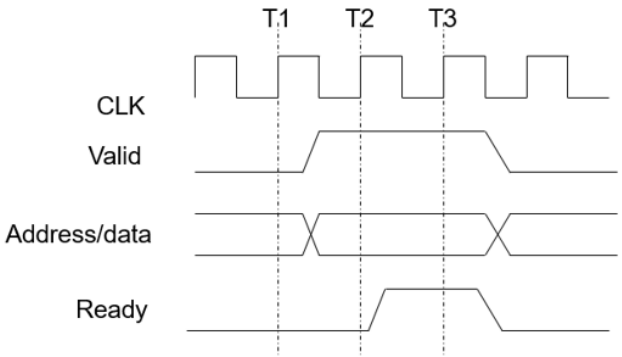
- Write Address channel – Master (user logic) provides relevant signals to issue a write command to the slave (HBM2 controller).
- Write data channel – Master provides the write data signals corresponding to the write address.
- Write response channel – Slave provides response on the status of the issued write command to the master.
- Read Address channel – Master provides relevant signals to issue a read command to the slave.
- Read data channel – Slave provides read data and read data valid signals corresponding to the read command to the master.

All transactions in the five channels use a handshake mechanism for the master and slave to communicate and transfer information.

Handshake Protocol

All five transaction channels use the same VALID/READY handshake process to transfer address, data, and control information. Both the master and slave can control the rate at which information moves between master and slave. The source generates the VALID signal to indicate availability of the address, data, or control information. The destination generates the READY signal to indicate that it can accept the information. Transfer occurs only when both the VALID and READY signals are HIGH.

Figure 18. AXI Protocol Handshake Process



In the figure above, the source presents the address, data or control information after T1 and asserts the VALID signal. The destination asserts the READY signal after T2, and the source must keep its information stable until the transfer occurs at T3, when this assertion occurs. In this example, the source asserts the VALID signal prior to the destination asserting the READY signal. Once the source asserts the VALID signal, it must remain asserted until the handshake occurs, at a rising clock edge at which VALID and READY are both high. Once the destination asserts READY, it can deassert READY before the source asserts VALID. The destination can assert READY whenever it is ready to accept data, or in response to a VALID assertion from the source.

AXI IDs

In an AXI system with multiple masters, the AXI IDs used for the ordering model include the infrastructure IDs that identify each master uniquely. The ordering model applies independently to each master in the system.

The AXI ordering model also requires that all transactions with the same ID in the same direction must provide their responses in the order in which they are issued. Because the read and write address channels are independent, if an ordering relationship is required between two transactions with the same ID that are in different directions, then a master must wait to receive a response to the first transaction before issuing the second transaction. If a master issues a transaction in one direction before it has received a response to an earlier transaction in the opposite direction, there is no ordering guarantee between the two transactions.

AXI Ordering

The AXI system imposes no ordering restrictions between read and write transactions. Read and write can complete in any order, even if the read address AXI ID (ARID) of a read transaction is the same as the write address AXI ID (AWID) of a write transaction. If a master requires a given relationship between a read transaction and a write transaction, it must ensure that the earlier transaction is completed before it issues a subsequent transaction. A master can consider the earlier transaction complete only when one of the following is true:

- For a read transaction, it receives the last of the read data.
- For a write transaction, it receives the write response.

6.3.1. AXI Write Transaction

AXI Write Address

You can initiate an AXI write transaction by issuing a valid Write Address signal on the AXI Write Address Bus, AWADDR. The user logic should provide a valid write address in the AWADDR bus and assert the AWVALID to indicate that the address is valid. The master can assert the AWVALID signal only when it drives valid address and control information.

When the HBM2 controller is ready to accept a Write command transaction, it asserts the AWREADY signal. Address transfer happens when both AWVALID and AWREADY are asserted.

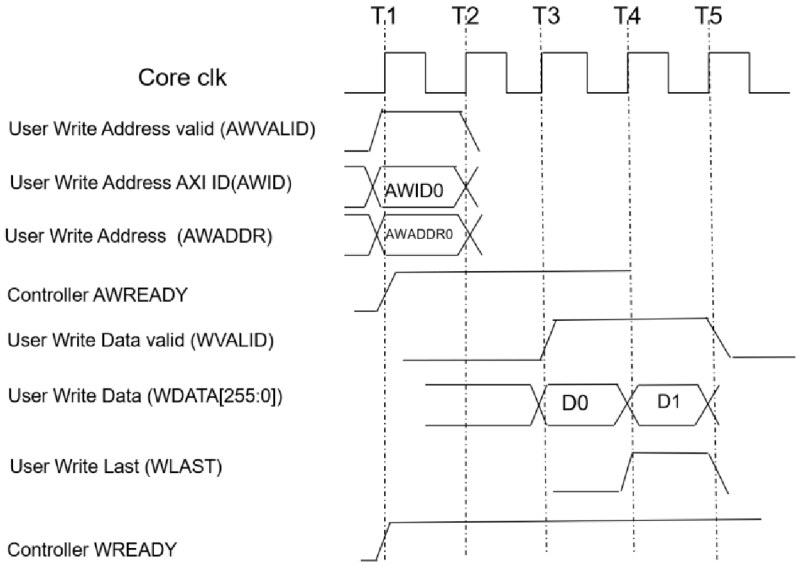
AWUSER, the user signal for auto precharge, also follows the same timing as AWADDR ; that is, AWUSER must be presented with the same timing of AWADDR.

AXI Write Data

During a write burst, the master asserts the WVALID signal only when it drives valid write data. Once asserted, WVALID must remain asserted until the rising clock edge after the slave asserts WREADY. The master must assert the WLAST signal while it is driving the final write transfer in the burst. User logic must issue the write data in the same order in which the write addresses are issued. Write data transfer happens when both WVALID and WREADY are HIGH.

The following diagram illustrates a BL8 Write transaction. The master asserts the Write address (WA0) in T1 using transaction ID AWID0, the HBM2 controller asserts the AWREADY when it is ready to accept write requests. The master asserts the Write data in clock cycle T3. Because the controller WREADY is already asserted, the write data is accepted starting cycle T3. The last piece of the burst 8 transaction is asserted in clock cycle T4.

Figure 19. AXI Write Transaction – Using Pseudo-BL8 Memory Write Transaction



Write Response Channel

The HBM2 controller uses the Write Response channel to respond on successful Write transactions. The slave can assert the BVALID signal only when it drives a valid write response. When asserted, BVALID must remain asserted until the rising clock edge after the master asserts BREADY. The default state of BREADY can be high, but only if the master can always accept a write response in a single cycle.

6.3.2. AXI Read Transaction

Read Address

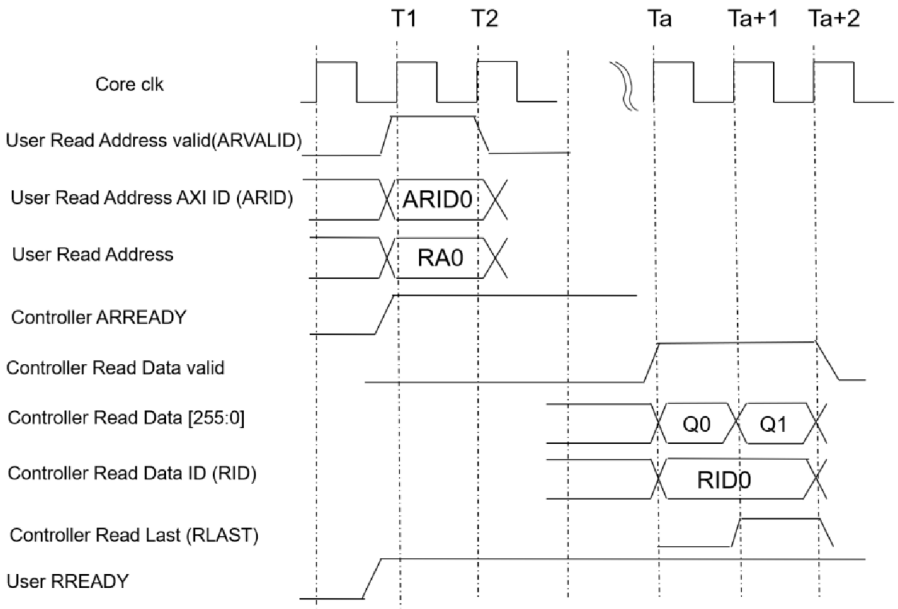
The user logic asserts the ARVALID signal only when it drives valid Read address, ARADDR, information. Once asserted, ARVALID must remain asserted until the rising clock edge after the HBM2 controller asserts the ARREADY signal. If ARREADY is high, the HBM2 controller accepts a valid address that is presented to it. Once calibration is completed and the HBM2 Controller is ready to accept commands, the ARREADY is asserted high. ARUSER, the user signal for auto precharge, also follows the same timing as ARADDR, that is, ARUSER must be presented with the same timing of ARADDR.

Read Data Channel

The HBM2 controller asserts the RVALID signal when it drives valid read data to the user logic. The master interface uses the RREADY signal to indicate that it accepts the data. The state of RREADY can be always held high, if the master is always able to accept read data from the HBM2 Controller. The soft logic first in, first out (FIFO) buffers can be instantiated through the HBM2 parameter editor if the HBM2 controller expects to ever deassert the RREADY signal. The HBM2 controller asserts the RLAST signal when it is driving the final read transfer in the burst.

Figure below describes a BL8 Read transaction. The user logic asserts the Read address (RA0) in T1 using transaction ID ARID0, the HBM2 controller ARREADY is already asserted, the READ command is accepted. The controller provides the Read Data back to the user interface after issuing the READ command to the HBM2 DRAM. The HBM2 controller asserts the Read data in clock cycle TB. The Read transaction ID (RID) provided by the HBM2 controller corresponds to the Read Address ID (ARID). The last piece of the burst 8 transaction (RLAST) is asserted in clock cycle Ta+1.

Figure 20. AXI Read Transaction – Using Pseudo-BL8 Memory Read Transaction



6.3.3. Non-zero Latency Backpressure

The HBM2 IP provides a non-zero cycle latency backpressure feature that can improve core timing and usability of the HBM2 IP from the AXI Write Address, Read Address and Write Data Interface. You can enable this feature through the parameter editor.

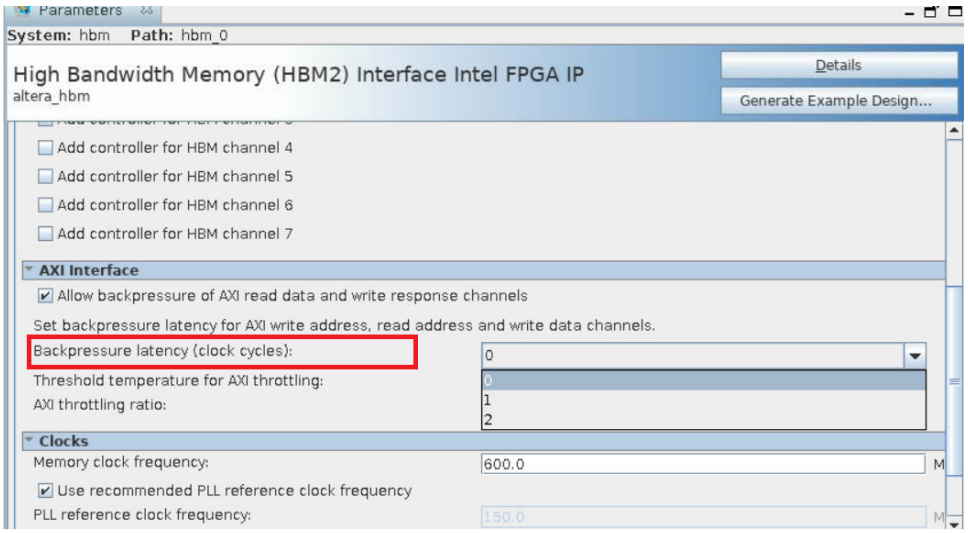
Improving User Logic to HBM2 Controller AXI Interface Timing

When non-zero latency backpressure is enabled, set to *N* cycles (supported clock cycle values for *N* are 1, 2 when the feature is enabled and 0 when disabled), the READY signals provided by the HBM2 Controller (that is, AXI Write Address Ready (AXI_X_Y_AWREADY), AXI Read Address Ready (AXI_X_Y_ARREADY) and AXI Write

Data Ready (AXI_X_Y_WREADY)) are issued *N* clock cycles early. This request allows you *N* clock cycles to react to the controller ready signals (address write, address read and write channels).

Read data and write response channels do not see an increase in latency due to the backpressure registers.

Figure 21. AXI Backpressure Latency Selection



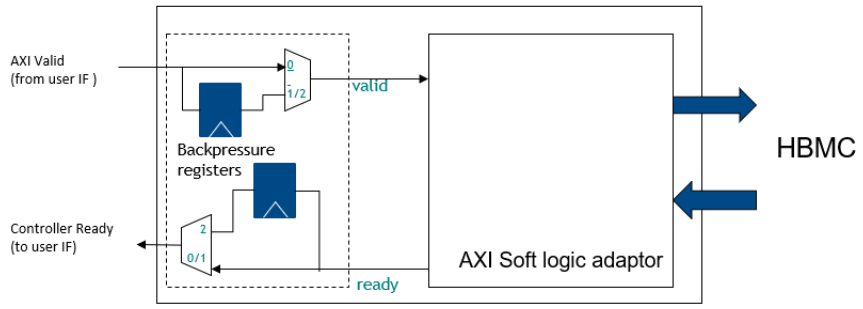
Prior to the Intel Quartus Prime software version 19.4, AXI backpressure registers were implemented only at the traffic generator side in the HBM2 design example. This means that in the HBM2 IP, it was necessary to manually compensate the backpressure latency by adding registers in the user logic.

Beginning in the Intel Quartus Prime software version 19.4, the HBM2 IP adds the following registers based on the **Backpressure latency** setting; it is no longer necessary to manually insert the registers.

- When **Backpressure latency** is set to 1, the controller deasserts the READY signal one clock cycle early. A single register stage is added in signals in the AXI interface Valid path from the core to the HBM2 controller.
- When **Backpressure latency** is set to 2, the controller deasserts the READY signal two clock cycles early. The HBM2 IP adds the following registers:
  - One register stage is added on the AXI interface valid and all related AXI interface signals from the user logic.
  - One register stage is added on READY signals (AWREADY, WREADY, ARREADY) from the controller to the core.

The following figure illustrates the back pressure register placement in the data path between the user core logic and the AXI soft logic adaptor.

Figure 22. Non-zero Latency Backpressure Implementation



Write Response and Read Data Interface Timing

When you enable the read and write response backpressure feature, Write response and Read data interface timing must still be treated at zero cycle latency. The user-interface-issued BREADY and RREADY must be driven HIGH to ensure the controller can provide the outputs when applicable.

When you disable the read and write response backpressure feature, you can add pipeline registers to read and write response outputs, because BREADY and RREADY are always HIGH, indicating that the user interface is ready to accept response at any time.

Related Information

- General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP on page 19
- High Bandwidth Memory (HBM2) Interface Intel FPGA IP Timing on page 71

6.4. User APB Interface Timing

The Advanced Peripheral Bus (APB) interface lets you issue user-controlled refresh signals and access the HBM2 controller's Control and Status registers.

Unlike the AXI4 interface which is used for high-bandwidth, main band operations and where each HBM pseudo-channel is mapped to its own dedicated AXI4 port, the APB interface is intended for relatively low-bandwidth sideband operations. (For example, to provide an alternative means of controlling refreshes without colliding with the main traffic stream, and for issuing custom commands useful for debugging.) Another difference is that the APB bus interface commands can target one or both Pseudo Channels at the same time, as there is one APB interface per HBM2 channel or two HBM2 Pseudo Channels.

6.4.1. Advanced Peripheral Bus Protocol

The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) protocol family. It defines a low-cost interface that is optimized for minimal power consumption and reduced interface complexity.

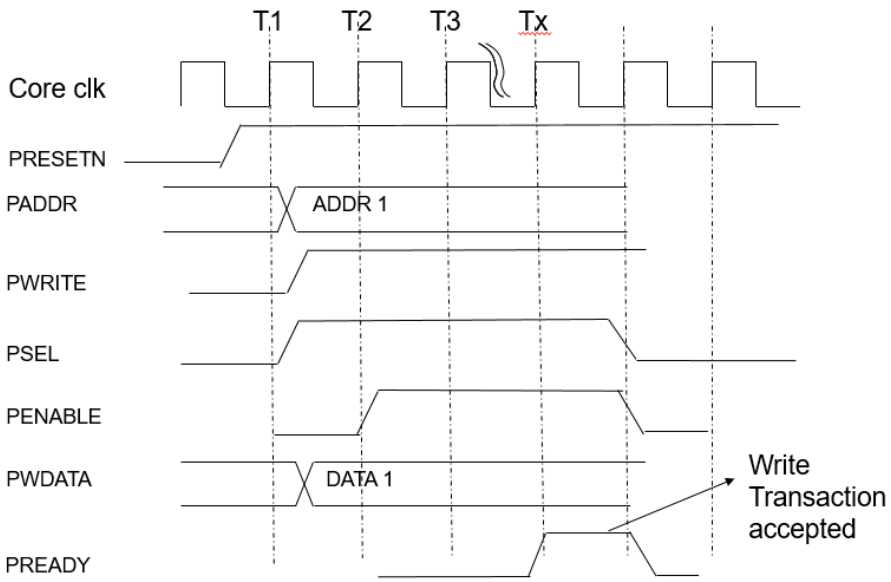
6.4.2. APB Interface Timing

Write Access

Referring to the following diagram, write transactions to the APB interface follow these steps:

1. At T1, a write transfer starts with address ADDR1, write data DATA1, write signal PWRITE and select signal PSEL registered at the rising edge of the core clock. This is the setup phase of the write transfer.
2. At T2, PENABLE is registered at the rising edge of the core clock and held HIGH until the HBM2 controller asserts PREADY HIGH. The values of PADDR, PSEL, PENABLE, PWDATA, PSTRB and PWRITE must remain unchanged while PREADY remains LOW.
3. At Tx, when PREADY goes HIGH, the write transaction completes at the next rising edge of the core clock. This indicates the end of the Write Access Phase. PREADY stays HIGH only for one clock cycle.
4. PENABLE is deasserted at the end of the transfer. The select signal PSEL, is also deasserted unless the transfer is to be followed immediately by another transfer.

Figure 23. APB Write Transaction

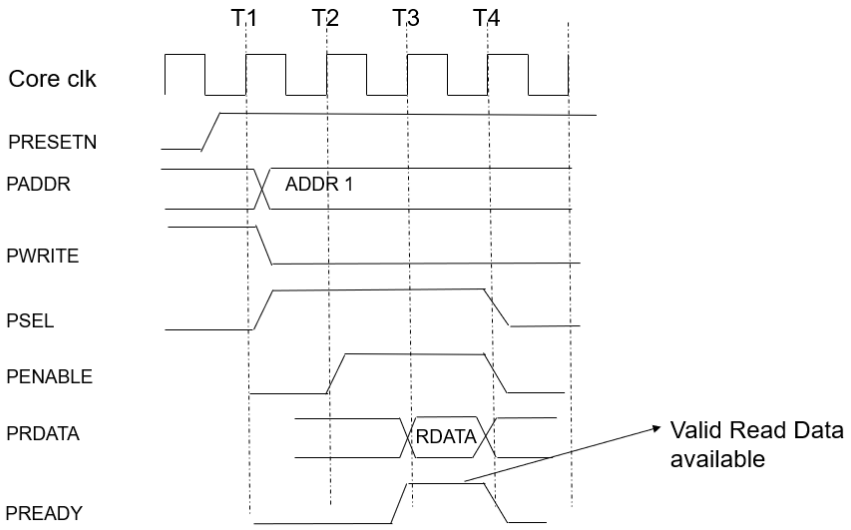


Read Access

Referring to the following diagram, read transactions to the APB interface follow these steps:

1. At T1, a read transfer starts with address ADDR1, PWRITE asserted LOW and select signal PSEL registered at the rising edge of the core clock. This is the setup phase of the write transfer.
2. At T2, PENABLE is registered at the rising edge of the core clock and held HIGH until the HBM2 controller asserts PREADY HIGH. The values of PADDR, PSEL, PENABLE and PWRITE must remain unchanged while PREADY remains LOW.
3. At Tx, when PREADY goes HIGH, Read Data is available on the PRDATA bus. PREADY stays HIGH only for one clock cycle.
4. PENABLE is deasserted at the end of the transfer. The select signal PSEL, is also deasserted unless the transfer is to be followed immediately by another transfer.

Figure 24. APB Read Transaction



6.5. User-controlled Accesses to the HBM2 Controller

You can use the APB interface in applications when you need to directly control the HBM2 Refresh commands and access HBM2 Controller Status registers.

Each physical HBM2 channel is mapped to its own sideband register space. The sideband register map is shared between the two HBM2 Pseudo Channels and the allocation of addresses is organized as follows:

- Registers common to both Pseudo-Channels:
  - Address map – 16'h0000-16'h00FF.
  - Includes Refresh (per-bank, all banks), Self-Refresh, Temperature Readout and Power down status.
- Register Map for individual Pseudo Channels:
  - Address map – Pseudo Channel 0 (16'h0100- 16'h01FF) and Pseudo Channel 1 (0x200-0x2FF).
  - This map is used to access ECC and Interrupt Status Registers for each Pseudo Channel.

6.5.1. User-controlled Refreshes

By default, the HBM2 parameter editor settings allow the HBM2 controller to schedule Refresh commands to the HBM2 DRAM. However, user logic can also issue Refresh commands to memory.

If you want to allow user logic to issue Refresh requests, you must specify so in the Controller Settings of the parameter editor, as follows:

Change the **Controller Settings ➤ REFRESH MODE** value from **Controller Refresh All** to **User Refresh All** or **User Refresh Per-bank**.

Each APB interface corresponds to one HBM2 channel, therefore requests must be addressed to either of the two Pseudo Channels, one Pseudo Channel at a time.

The timing of the refresh interval is specific to the HBM2 DRAM timing. The required refresh rate is provided by the HBM2 DRAM through the TEMP[2:0] vector, which can be read through the APB interface.

Table 27. DRAM Refresh Timing

TEMP[2:0]	Refresh Rate
000	4x t_{REFI}
001	2x t_{REFI}
011	1x t_{REFI}
010	0.5x t_{REFI}
110	0.25x t_{REFI}
111	TBD
101	TBD
100	TBD

Refresh command options available from user logic are as follows:

- User refresh commands issued on a per bank basis. This corresponds to banks corresponding to the HBM2 Pseudo Channel.
- User refresh commands to all banks. This corresponds to all banks corresponding to the HBM2 Pseudo Channel.
- Self-refresh command.

All the user refresh requests follow the APB interface protocol. User logic issues the Refresh request and waits for the acknowledgement signal from the HBM2 controller before issuing another Refresh request.

User Refresh Per Bank

User refresh requests per-bank consist of the following commands issued to the APB Interface

Table 28. APB Write Data Definition for User Refresh Request Per Bank

Write Data Definition for Refresh Request	Description
[0]	Pseudo Channel Number. Set the following values to indicate the Pseudo channel number: 0 - Pseudo Channel 0 1 - Pseudo Channel 1
[5:1]	Bank number for per-bank refresh. Bit[5] - Stack ID in 8GB configuration. Bit[4:1] – 4-bit Bank Address.
[6]	Select Per bank or all bank refresh. Set to 1’b0 for per bank refresh.
[7]	Reserved. Set to 1'b0.
[8]	User initiated refresh request. Set this bit to 1’b1 to initiate refresh command.
[12:10]	Number of bursts for per bank refresh bursting feature. Burst = value[1:0] + 1. The per-bank refresh bursting is a feature supported by the HBM2 controller where the user interface can request refresh commands to be issued to more than one HBM2 DRAM bank.
[15:13]	Increment value for per bank refresh bursting feature; rolls over once it reached the max bank value. Increment = *value*[2:0] + 1, where *value* refers to the current bank address when the write command is issued. This becomes the starting bank address. *Note:* The ratio of maximum increment value to total number of banks per Pseudo Channel divided by 4. (For example, a 4H device has a maximum value of 16/4=4.)

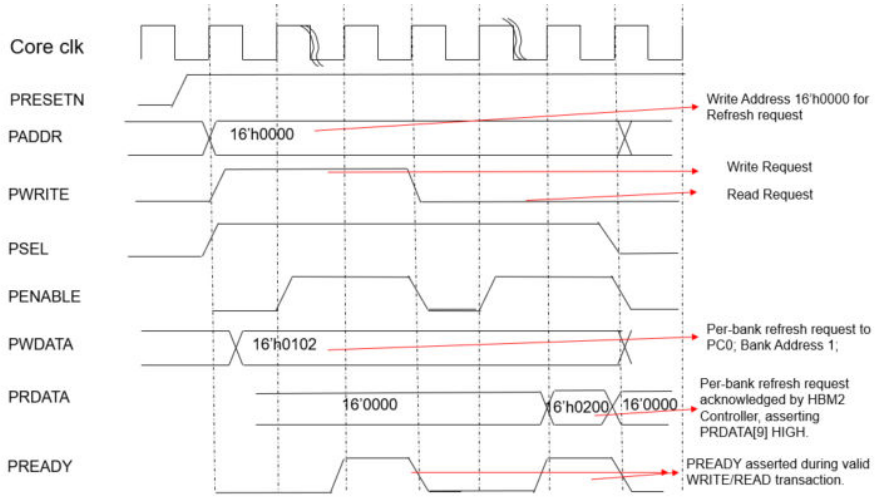
To issue user refresh commands on a per-bank basis, follow these steps:

1. Write to address 16’h0000 with corresponding Write Data.

- For example, Write data of 16'b0000_0001_0000_0010 (Pseudo-channel 0; bank address is located in Bits[5:1] ; Bit[6] set to 0 for per-bank refresh; Bit [8] set to User initiated Refresh Request).
- To issue multiple burst of per-bank refresh, set corresponding bits in PWDATA[12:10] and PWDATA[15:13] with desired settings.

2. Read from address 16'h0000 for Refresh Request Acknowledge.
3. Wait till PRDATA[9] returns 1'b1 to indicate refresh request is accepted by controller. You can ignore the values of all the other bits of PRDATA. The following figure explains the timing of per-bank refresh request to the APB Interface.

Figure 25. Timing of APB Refresh Request Per Bank



Individual requests to different Pseudo Channels must be issued serially. You must wait for the HBM2 controller to issue the Acknowledge signal before issuing another Refresh request.

User Refresh to All Banks

Table 29. APB Write Data Definition for User Refresh Request to All Banks

Write Data Bit Definition for Refresh Request	Description
[0]	Pseudo Channel Number. Set a value of 0 for Pseudo Channel 0, or a value of 1 for Pseudo Channel 1.
[5:1]	Bank number for per-bank refresh. Ignored for All bank refresh.
[6]	Select Per bank or all bank refresh. Set to 1'b1 - all banks refresh.
[7]	Reserved. Set to 1'b0.
[8]	User initiated refresh request. Set this bit to 1'b1 to initiate refresh command.
[15:9]	Ignored.

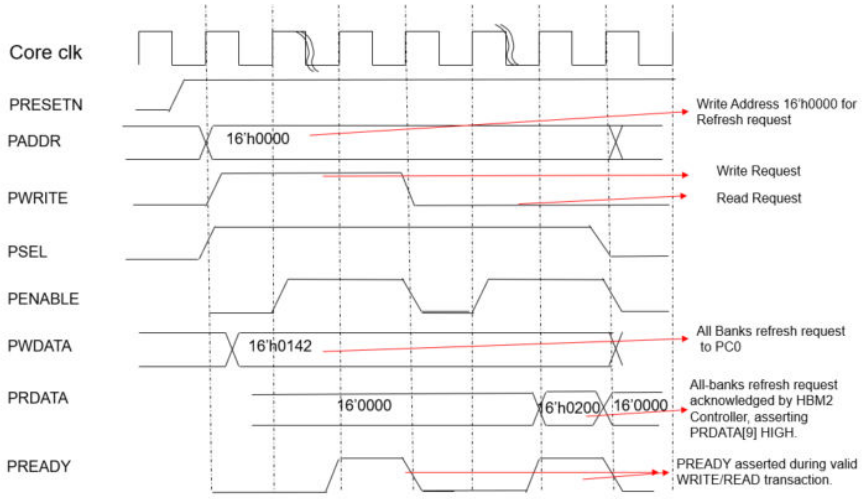
Table 30. APB Read Data Definition for User Refresh Request

Read Data Bit Definition for Refresh Request	Description
[8:0]	Reserved.
[9]	HBM2 controller-issued Refresh Request Acknowledge. • 1'b1 – Read data of 1'b1 indicates that the HBM2 controller Acknowledge of Refresh command. • Once a refresh request has been issued, you must wait for the Acknowledge bit to be asserted before issuing further requests.
[15:10]	Reserved.

To issue user refresh commands to all banks, follow these steps:

1. Write to address 16'h0000 with corresponding Write Data.
   - For example, Write data of 16'b0000_0001_0100_0010 (Pseudo Channel 0; bank address is located in Bits[5:1] ; Bit[6] set to 1 for all-banks refresh; Bit [8] set to User initiated Refresh Request).
2. Read to Address 16'h0000 for Refresh Request Acknowledge.
3. Wait till PRDATA[9] returns a value of 1'b1 to indicate User Refresh request is accepted by the controller. The following figure explains the APB interface timing for refresh request to all banks.

Figure 26. Timing of APB Refresh Request to All Banks



Self Refresh

To enter a Self Refresh state, issue these commands to the APB interface:

1. Write to address 16'h004 with Write Data(PWDATA) of 16'h0001.
2. Read from address 16'h004. Wait until Read Data (PRDATA [1]) returns a value of 1'b1 to indicate that the HBM2 controller is in a Self Refresh state.

To exit a Self Refresh state, issue this command to the APB interface:

- Write to address 16'h0004 with Write Data(PWDATA) of 16'h0000 to exit Self-Refresh. Wait for the Self Refresh Acknowledge to be 1'b0 prior to issuing another Self Refresh request.

The write data bus is used to request entry or exit from self refresh. The Read data bus is used to read the self-refresh status of the HBM2 channel. The following tables provide the Write and Read data definitions for accessing self refresh.

Table 31. APB Write Data Bit Definition for Self Refresh

Write Data Bit Definition	Description
[0]	User-initiated Self-Refresh Request. To enter Self Refresh: • 1'b1: to enter Self Refresh. • When asserted, indicates Self Refresh request. Set this bit to 1'b1 to initiate entry to Self Refresh. To exit Self Refresh: • 1'b0: to exit Self Refresh. • Clear this bit to 1'b0 to exit Self Refresh.
[15:1]	Reserved.

Table 32. APB Read Data Bit Definition for Self Refresh

Read Data Bit Definition	Description
[0]	Reserved.
[1]	Self refresh status: • 1'b1 – HBM Channel is in Self Refresh. • 1'b1 – HBM Channel is in Self Refresh.
[15:2]	Reserved.

6.5.2. Temperature and Calibration Status Readout

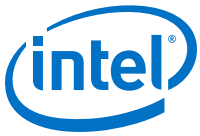
User logic can read the HBM2 TEMP[2:0] signal output value and the CATTRIP signal value from APB address 16'h000C.

Table 33. APB Read Data Temperature and Calibration Status Information

Read Data Bit Definition	Description
[2:0]	TEMP [2:0] value from HBM2 device.
[3]	CATTRIP value from HBM2 device.
[7:4]	Reserved.
[8]	• 1 = calibration in progress. • 0 = calibration complete.
[9]	• 1'b1: Calibration/training successful. • 1'b0: Calibration/training failed.
[15:10]	Reserved.

Related Information

High Bandwidth Memory (HBM2) Interface Intel FPGA IP DRAM Temperature Readout on page 71



6.5.3. Power Down Status

To invoke the Power Down mode in the HBM2 controller, enable the Power Down Enable Mode option in the parameter editor when generating the HBM2 IP.

To read the Power Down Enable status, issue a Read command to APB Address 16'h0008. The following table shows the HBM2 Power Down status information provided by APB Read data.

Table 34. APB Power Down Status Information

Read Data Bit Definition	Description
[0]	Power Down status. • 1'b1 – HBM Channel is in Power Down mode. • 1'b0 – HBM Channel is not in Power Down mode.
[15:1]	Reserved.

6.5.4. ECC Error Status

You can read the status of various registers used by the ECC feature, using the APB interface.

ECC registers provide the following information – and corresponding APB addresses – for Pseudo Channel 0 and Pseudo Channel 1:

- Single-bit error (SBE) counter
- Double-bit error (DBE) counter
- Logical address (AXI address) of the first single-bit error
- Logical address (AXI address) of the first double-bit error
- Read transaction ID (AXI) of the first single-bit error
- Read transaction ID (AXI) of the first double-bit error
- Read Data Parity error counter

The values of the ECC registers are read when the PRREADY output from the HBM2 controller is asserted. When the HBM2 controller is RESET, these register values are reset to zero.

Table 35. ECC Registers

Address	Read Data Bit Location	Description
16'h0120 – PC0 16'h0220 – PC1	[7:0]	Single-Bit-Error Counter. The counter does not overflow when value reaches maximum. Write 0 to clear the counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
	[15:8]	Reserved.
16'h0122 – PC0 16'h0222 – PC1	[7:0]	Double-Bit-Error Counter. The counter does not overflow when value reaches maximum.

continued...

Address	Read Data Bit Location	Description
		Write 0 to clear the counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
	[15:8]	Reserved.
16'h0124 – PC0 16'h0224	[15:0]	First SBE logical address (lower order). This is the logical address of the first single-bit error, corresponding to the lower 16 bits of the AXI Address bus.
16'h0126 – PC0 16'h0226 – PC1	[15:0]	First SBE logical address (higher order). This is the logical address of the first single-bit error, corresponding to the higher 16 bits of the AXI address bus. The higher order 4 bits in 4GB configuration and higher-order 3 bits in 8GB configuration are padded with zeros.
16'h0128 – PC0 16'h0228 - PC1	[15:0]	First DBE logical address (lower order). This is the logical address of the first double-bit error, corresponding to the lower 16 bits of the AXI address bus.
16'h012A – PC0 16'h0228 – PC1	[15:0]	First DBE logical address (higher order). This is the logical address of the first double-bit error, corresponding to the higher 16 bits of the AXI address bus. The higher order 4 bits in 4GB configuration and higher order 3 bits in 8GB configuration are padded with zeros.
16'h012C – PC0 16'h022C – PC1	[8:0]	Read Transaction ID of the first single-bit error.
	[15:9]	Reserved.
16'h012E – PC0 16'h022E – PC1	[8:0]	Read Transaction ID of the first double-bit error.
	[15:9]	Reserved.
16'h0130 – PC0 16'h0230 – PC1	[7:0]	Read Data Parity Error Count. The counter does not overflow when value reaches maximum. Write 0 to clear the counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
	[15:8]	Reserved.

6.5.5. User Interrupt

An interrupt signal occurs when one or more of the Error Status signals are TRUE. You can configure the conditions that help to trigger the interrupt signal.

The various status signals on which you can generate the interrupt include the following:

- Single-bit error (SBE) or double-bit error (DBE) of AXI Read Data or internal RAM used for Write and Read data storage
- Read Data parity error (RDPE) or Write Data parity error (WDPE) of AXI Read Data
- Address Command parity error
- CATTRIP
- Calibration

6.5.5.1. Interrupt Enable and Conditions for Interrupt Generation

If you want interrupts to occur based on certain conditions, you must set the conditions to enable interrupt generation.

To enable interrupt generation, issue a Write command to address location 16'h0100 (for Pseudo Channel 0) and 16'h0200 (for Pseudo Channel 1), with the corresponding Write Data (PWDATA).

- PWDATA[0] – Interrupt enable.
- PWDATA[11:1] - Lists the various status signals that you can use, alone or in combination, to trigger the Interrupt signal.
  - Set the Mask value to 1'b0 to use the corresponding error condition to generate the Interrupt signal.
  - If you set the Mask value to 1'b1, the Interrupt generator ignores that specific error condition. For example, to use the double-bit error condition to generate the Interrupt signal, set PWDATA[2] to 1'b0.

The following table describes the 16-bit Write Data (PWDATA) for setting the Interrupt Enable, and the conditions of the interrupt.

Table 36. Interrupt Conditions

Write Data Bit Definition	Description
[0]	Interrupt Enable: Enables interrupt to the HBM2 controller when the conditions set in PWDATA[11:1] are TRUE. 1 – Enable Interrupt 0 – Disable Interrupt
[1]	SBE Interrupt Mask.
[2]	DBE Interrupt Mask.
[3]	Read DPE Interrupt Mask.
[4]	Write DPE Interrupt Mask.
[5]	Address Command Interrupt Mask.
[6]	CATTRIP Interrupt Mask.
[7]	Calibration Interrupt Mask.
[8]	Write SRAM SBE Interrupt Mask.
[9]	Write SRAM DBE Interrupt Mask.

continued...

Write Data Bit Definition	Description
[10]	Read SRAM SBE Interrupt Mask.
[11]	Read SRAM DBE Interrupt Mask.
[15:12]	Reserved.

6.5.5.2. Interrupt Status

You can read the interrupt status from address 16'h0102 for Pseudo Channel 0 and 16'h0202 for Pseudo Channel 1.The interrupt signal is cleared when the individual error status signals that contribute to the interrupt are cleared and the corresponding error counters are cleared.

Table 37. Read Data Definition for Interrupt Status

Read Data Bit Definition	Description
[0]	Interrupt asserted when one of the following events occurs: • Single-bit error • Double-bit error • Read Data parity error • Write Data parity error • Address Command parity error • CATTRIP assertion • Calibration status failure
[15:1]	Reserved.

6.5.5.3. Error Valid Status

You can read individual status signals from address 16'h0104 for Pseudo Channel 0 and 16'h0204 for Pseudo Channel 1. You can also write to these addresses to clear the status signals.

The above addresses provide the actual status signal for the following error conditions:

- Single bit/Double bit error value from the HBM2 memory
- Read data/Write Data parity error value from the HBM2 memory.
- Address/Command parity value.
- CATTRIP signal.
- Calibration error signal.
- Single-bit error and double-bit error from internal RAM used for Write and Read data storage.

To clear the individual Error Valid Signals, write 1'b1 to clear the Valid signal and the corresponding Error Counter.

Table 38. Read Data Definition for Error Valid Status

Read Data Bit Definition	Definition
[0]	DRAM Single-Bit-Error Valid. Asserted when single-bit error occurs and stays high until cleared. Write 1'b1 to clear.

continued...

Read Data Bit Definition	Definition
	You can retrieve the number of single-bit errors from the single-bit error counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
[1]	DRAM Double-Bit-Error Valid. Asserted when double-bit error occurs and stays high until cleared. Write 1'b1 to clear. You can retrieve the number of double-bit errors from the double-bit error counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
[2]	Read Data Parity Error Valid. Asserted when a Read Data parity error occurs and stays high until cleared. Write 1'b1 to clear. You can retrieve the number of Read Data parity errors from the Read Data parity error counter. Clearing the counter does not clear this valid bit or vice-versa. You must issue a separate write command to valid and counter to clear both registers.
[3]	Write Data Parity Error Valid. Asserted when a Write Data parity error occurs and stays high until cleared. Write 1'b1 to clear.
[4]	Address Command Parity Error Valid. Asserted when an Address Command parity error occurs and stays high until cleared. Write 1'b1 to clear.
[5]	Cat Trip Error Valid. Asserted when a Cat Trip error occurs and stays high until cleared. Write 1'b1 to clear.
[6]	Calibration Error Valid. Asserted when a calibration error occurs and stays high until cleared. Write 1'b1 to clear.
[7]	Write SRAM Single-Bit Error Valid. Asserted when a Write SRAM Single-Bit error occurs and stays high until cleared. Write 1'b1 to clear.
[8]	Write SRAM Double-Bit Error Valid. Asserted when a Write SRAM Double-Bit error occurs and stays high until cleared. Write 1'b1 to clear.
[9]	Read SRAM Single-Bit Error Valid. Asserted when a Read SRAM Single-Bit error occurs and stays high until cleared. Write 1'b1 to clear.
[10]	Read SRAM Double-Bit Error Valid. Asserted when a Read SRAM Double-Bit error occurs and stays high until cleared. Write 1'b1 to clear.
[15:11]	Reserved.

6.5.6. ECC Error Correction and Detection

The HBM2 controller supports single-bit error correction and double-bit error detection. It does not support correction or detection of more than two error bits.

The ECC encoder and decoder blocks reside inside the UIB subsystem and help to efficiently perform the ECC logic without additional latency. The ECC logic performs the following operations:

- When a single-bit error is detected, the error is corrected and passed to the AXI Interface.
- When a double-bit error is detected, a signal is asserted to indicate the error and is passed through the `axi_ruser_err_dbe` signal in the AXI Read Data Channel Interface.
- Error information is available through the APB interface.

You can use the HBM2 controller's Read-Modify-Write feature to correct a single or double-bit error detected in the HBM2 DRAM. A single-bit error or a double-bit error corresponds to an error detected on every 64-bit HBM2 DQ bus; consequently, multiple errors could be detected:

- Multiple single-bit errors are treated as a single single-bit error, and the single-bit error count increases by 1.
- Multiple double-bit errors are treated as a single double-bit error, and the double-bit error count increases by 1.
- A single-bit error and a double-bit error are treated as a double-bit error, because the double-bit error has higher priority. The double-bit error count increases by 1 and the single-bit error count does not change.

Read-Modify-Write

The Read-Modify-Write feature reads from the HBM2 DRAM, modifies the data, and writes back to the HBM2 memory. The HBM2 controller supports the following functions as part of the Read-Modify-Write process:

- Dummy Writes – Corrects HBM2 DRAM data detected to have a single-bit error.
- Partial Writes – Issues partial writes to HBM2 DRAM where not all bytes are enabled.

Dummy Writes

When the ECC decoder logic detects and corrects a single-bit error on the Read data, user logic may correct the corresponding bit in the HBM2 DRAM, using the Dummy Write process. A Dummy Write issues a Read from the memory location and writes the corrected Read data back to the corresponding memory location.

To request a Dummy Write, issue a regular AXI Write transaction with all byte enables deasserted, to the corresponding address.

The HBM2 controller handles the Read-Modify-Write operation internally and corrects the DRAM data without additional user intervention. The Read-Modify-Write operation follows this process:

- The AXI Adaptor within the UIB subsystem decodes all the byte enables deasserted and identifies the Read-Modify-Write request.
- The HBM2 controller then issues a Read to the corresponding address in the HBM2 DRAM.
- The ECC Decoded Read data is used as Write Data – this is the corrected Read data if a single-bit error was detected.
- The HBM2 Controller issues an HBM2 Write transaction and later writes the decoded Read data into the memory.

Partial Writes

The HBM2 controller's Partial Write capability allows the user logic to issue a partial write to the HBM2 DRAM, when not all the byte enables are asserted and only selected DRAM bytes are written to. The Partial Write feature first issues a Read from the memory location, and then merges correct Read data with the correct Write data, to be written back to the memory location. (This process is necessary because Data Mask signals are not available to user logic when ECC is enabled.) The HBM2 controller intelligently supports Partial Writes using the AXI4 interface.

To request a Partial Write, you issue a regular AXI Write transaction, with not all byte enables asserted (that is, not a full Write), and with corresponding Write data to be written to the HBM2 DRAM.

The HBM2 controller handles the issuance of the corresponding memory transactions necessary to complete the Partial Write:

- The AXI Adaptor within the UIB Subsystem decodes the byte enables and identifies the Partial Write request.
- The HBM2 controller then issues a Read to the corresponding address in the HBM2 DRAM.
- The ECC Decoded Read data is merged with the requested Write data based on the byte enables.
- The HBM2 controller issues an HBM2 Write transaction and later writes the merged data into the memory with the updated ECC code.

7. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Performance

This section discusses key aspects of the HBM2 IP controller performance: efficiency, latency, and timing.

7.1. High Bandwidth Memory (HBM2) DRAM Bandwidth

For each HBM2 DRAM in an Intel Stratix 10 MX device, there are eight channels of 128-bits each.The following example illustrates the calculation of bandwidth offered by one HBM2 interface.

Assuming an interface running at 1 GHz: `128 DQ * 1 GHz = 128 Gbps`:

- The interface operates in double data-rate mode, so the total bandwidth per HBM2 is: `128 Gbps * 2 = 256 Gbps`
- The total bandwidth for the HBM2 interface is: `256 Gbps * 8 = 256 GBytes/sec`
- If the HBM2 controller operates at 90% efficiency, the effective bandwidth is: `256 Gbps * 0.9 = ~230 GByte/sec`

7.2. High Bandwidth Memory (HBM2) Interface Intel FPGA IP HBM2 IP Efficiency

The efficiency of the High Bandwidth Memory (HBM2) Interface Intel FPGA IP estimates data bus utilization at the AXI interface.The AXI4 protocol supports independent write and read address and data channel and accepts concurrent write and read transactions. Calculated efficiency values take into consideration that the core clock frequency and the memory clock frequency are different. The HBM2 IP design example includes an efficiency monitor that reports the efficiency and the minimum latency observed in the transactions that are simulated. You can enable the efficiency monitor on the Diagnostics tab of the HBM2 IP parameter editor.

The following equation represents the HBM2 controller efficiency:

```
Efficiency = ((Write transactions + Read transactions accepted by HBM2
controller)/total valid transaction count) * (core clk frequency * 2/
HBM2 interface frequency) * 100
```

- Write transactions – Refers to user write data transactions that the HBM2 controller accepts (user-asserted AXI WVALID and corresponding controller-asserted AXI WREADY).
- Read transactions – Refers to user read data transactions that the HBM2 controller has processed (controller-asserted AXI RVALID and corresponding user-asserted AXI RREADY).
- Total valid transaction count – Total transaction time after first valid transaction has been issued.
- Core frequency (MHz) – The frequency at which user logic operates. The core operates at a lower frequency than the HBM2 interface.
- HBM2 interface frequency (MHz) - The frequency at which the HBM2 interface operates.

The HBM2 controller provides high efficiency for any given address pattern from the user interface. The controller efficiently schedules incoming commands, avoiding frequent precharge and activate commands as well as frequent bus turn-around when possible.

Factors Affecting Controller Efficiency

Several factors can affect controller efficiency. For best efficiency, you should consider these factors in your design:

- User-interface frequency vs HBM2 interface frequency - The frequency of user logic in the FPGA fabric plays an important role in determining HBM2 memory efficiency, as shown in the example above.
- Controller Settings
  - Disable the Reorder Buffer in the Controller Settings to achieve improved efficiency. (However, if the application requires that read data be provided in the same order as the read requests, then it is preferable to enable the Reorder Buffer.)
  - Burst length - The pseudo-BL8 mode helps to ensure shorter memory access timing between successive BL4 transactions, to improve controller efficiency.
- Traffic Patterns - Traffic patterns play an important role in determining controller efficiency.
  - Sequential vs random DRAM addresses: Sequential addresses enable the controller to issue consecutive write requests to an open page and help to achieve high controller efficiency. Random addresses require constant PRECHARGE/ACTIVATE commands and can reduce controller efficiency.
  - Set the User Auto Precharge Policy to FORCED and set the `awuser/aruser` signal on the AXI interface to HIGH to enable Auto Precharge for random transactions. For sequential transactions, set the Auto Precharge Policy to HINT.
  - Sequential Read only or Write Only transactions: Sequential read-only or write-only transactions see higher efficiency as they avoid bus turnaround times of the DRAM bi-directional data bus.
- AXI Transaction IDs – Use of different AXI transaction IDs helps the HBM2 controller schedule the transactions for high efficiency. Use of the same AXI transaction ID preserves command order and may result in lower efficiency.
- Temperature – There are two main temperature effects that reduce the bandwidth available for data transfers in the Intel Stratix 10 HBM2 interface:
  - The HBM2 is exceeding a temperature threshold set in the **General ➤ Threshold temperature for AXI throttling** parameter, along with a value for the **AXI throttling ratio**.
  - Above 85°C, HBM2 refreshes occur more frequently, so there is less bandwidth available for data transfers. At 85°C, the refresh rate is 2×, and at 95°C, 4× more frequent than the standard refresh time interval of 7.8 microseconds.

  The HBM2 TEMP register value can be read back via the AXI APB bus to help with the correlation of memory refreshes, temperature, and data bandwidth.

7.3. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Latency

Read latency measures the number of clock cycles from the time the HBM2 controller receives a valid read address command, to the time that valid read data is available at the user interface.(In other words, from the instant the master asserts the ARVALID signal and the slave asserts the ARREADY signal, until the slave asserts the RVALID signal and the master asserts the RREADY signal.)

Read latency includes the controller command path latency to issue the read command to the HBM2 memory, memory read latency, and the delay in the read data path through the HBMC memory controller. Simulation reports the minimum latency in AXI core clock cycles seen during the simulation time.

7.4. High Bandwidth Memory (HBM2) Interface Intel FPGA IP Timing

The maximum HBM2 memory interface frequency is based on the Intel Stratix 10 MX device speed grade. The maximum core interface frequency is limited by the frequency at which the core logic can meet timing.

For the best HBM2 efficiency, ensure that your user logic follows best design practices. Take care to avoid combinatorial paths between the AXI master and slave input and output signals. Add pipeline registers as necessary and reduce logic levels in timing-critical paths to successfully meet core timing requirements. The Backpressure Latency feature allows you to add up to two register stages between the user interface and the HBM2 IP. Refer to *Improving User Logic to HBM2 Controller AXI Interface Timing* for details on Backpressure Latency.

The following documents provide detailed information on the Intel Stratix 10 device architecture and design techniques that you can adopt to achieve the best core performance:

- The *Intel Stratix 10 High Performance Design Handbook*.
- The *Timing Closure and Optimizations* chapter of the *Design Optimization User Guide: Intel Quartus Prime Pro Edition*.

Related Information

- Non-zero Latency Backpressure on page 51
- Intel Stratix 10 High-Performance Design Handbook
- Design Optimization User Guide: Intel Quartus Prime Pro Edition

7.5. High Bandwidth Memory (HBM2) Interface Intel FPGA IP DRAM Temperature Readout

You can read the temperature of the HBM2 DRAM through the APB interface. Refer to the *Temperature and Calibration Status Readout* topic for more information.

You can also read the temperature using the Intel Stratix 10 Analog to Digital Converter. Refer to the *Intel Stratix 10 Analog to Digital Converter User Guide* for more information.

Related Information

- Temperature and Calibration Status Readout on page 60
- Intel Stratix 10 Analog to Digital Converter User Guide

8. High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide Archives

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

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
19.2.0	High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide
19.1	High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide
18.1.1	High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide
18.0	High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide
17.1	Intel Stratix 10 MX HBM2 IP User Guide

9. Document Revision History for High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.03.02	19.4	19.3.0	In the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface* chapter, in the *AXI User-Interface Signals* topic: • Modified the description of the `axi_0_0_awsize` port in the *User Port 0's AXI4 Write Address (Command) Channel* table. • In the *User Port 0's AXI4 Write Data Channel* table, changed the names and descriptions of the `axi_0_0_wuser_data` and `axi_0_0_wuser_strb` ports. Implemented minor changes to the descriptions of the `axi_0_0_wstrb` and `axi_0_0_wvalid` ports. • Modified the descriptions of the `axi_0_0_arsize` and `axi_0_0_arburst` ports in the *User Port 0's AXI4 Read Address (Command) Channel* table. • Changed the names of the third and fourth entries in the *User Port 0's Read Data Channel* table. Added a sentence to the description of the `axi_0_0_rvalid` port.
2019.12.16	19.4	19.3.0	• In the *Creating and Parameterizing the High Bandwidth Memory (HBM2) Interface Intel FPGA IP* chapter: — In the *General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic, modified the description of the *Backpressure latency* parameter, in the *Group: General / AXI Interface* table. • In the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface* chapter: — Added additional information about the *hbm_only_reset_in* signal, in the *Reset Signals* topic. — Added additional information about clocks and resets in the *Reset Signals* topic. — Retitled the *Improving User Logic to HBM2 Controller AXI Interface Timing* topic to *Non-zero Latency Backpressure*. Implemented extensive changes to the topic content. • In the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Performance* chapter: — Added content about temperature effects, in the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP HBM2 IP Efficiency* topic.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2019.09.19	19.2.0	19.2.0	• In the *Introduction to High Bandwidth Memory* chapter, clarified description of Pseudo Channel mode in the *Intel Stratix 10 MX HBM2 Features* topic. • In the *Intel Stratix 10 MX HBM2 Architecture* chapter, modified the *HBM2 DRAM* section of the *Intel Stratix 10 MX UIB Architecture* topic. • In the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface* chapter: — Modified the descriptions of the `ext_core_clk_locked` and `in thepll_ref_clk sig in the` *Clock Signals* topic. — Modified the descriptions of the `axi_0_0_awlen`, `axi_0_0_awsize`, `axi_0_0_awburst`, `axi_0_0_arlen`, `axi_0_0_arsize`, and `axi_0_0_arburst` ports in the *AXI User-interface Signals* topic. — Minor additions to the *AXI Write Address* section of the *AXI Write Transaction* topic, and to the *Read Address* section of the *AXI Read Transaction* topic.
2019.07.25	19.2.0	19.2.0	• Added *About the High Bandwidth Memory (HBM2) Interface Intel FPGA IP*. • Revised the *Refresh Mode* description in the *Group: Controller/ Controller 0 Configuration* table in the *Controller Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic. • Updated Figure 7 in *General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP*. • Updated Figure 9 in *Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP*. • Added row for *Disable HBM model transaction messages in simulation* to Table 9 in *Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP*. • Updated Figure 13 in *Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP for High Efficiency*. • Updated Figure 20 in *Improving User Logic to HBM2 Controller AXI Interface Timing*. • Updated Width values in Tables 23 and 26 in *AXI User-interface Signals*.
2019.05.03	19.1	19.1	• Updated image of the *Diagnostics* tab, in the *Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* and the *Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP for High Efficiency* topics. • Modified the descriptions of the *PLL reference clock frequency* description in the *Group: General / Clocks* table in the *General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic • Added rows to *Group: Diagnostics / Traffic Generator* table in the *Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic. • Added bullet point to the *Simulating High Bandwidth Memory (HBM2) Interface Intel FPGA IP for High Efficiency* topic.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Minor rewording of the third bullet point in the *Intel Stratix 10 MX HBM2 Controller Features* topic. • Recast the *Jitter Specifications for the Input Reference Clocks* section in the *Pin Planning for the High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic. • Added content to the *Factors Affecting Controller Efficiency* section in the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP HBM2 IP Efficiency* topic. • Added the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide Archives* chapter.
2018.12.24	18.1.1	18.1.1	• Replaced figure 3, *Block Diagram of Intel Stratix 10 MX HBM2 Implementation* in the *Intel Stratix 10 MX HBM2 Architecture* chapter. • Reorganized and changed the title of the *Generating the Intel Stratix 10 MX HBM2 IP* chapter to *Creating and Parameterizing the High Bandwidth Memory (HBM2) Interface Intel FPGA IP.* • Removed the *Generating the Design Example* and *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example* topics from the *Creating and Parameterizing the High Bandwidth Memory (HBM2) Interface Intel FPGA IP.* chapter. • Replaced figure 15, *HBM2 IP Clocking and Reset Diagram*, figure 16, *AXI Address Definition*, figure 18, *AXI Write Transaction*, and figure 19, *AXI Read Transaction*, in the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface* chapter. • Added the topic *Improving User Logic to HBM2 Controller AXI Interface Timing* to the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface* chapter.
2018.05.07	18.0	18.0	• Changed document title from *Intel Stratix 10 MX HBM2 IP User Guide* to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP User Guide*. • Chapter 1, *Introduction to High Bandwidth Memory*: — Added ECC support to *Intel Stratix 10 MX HBM2 Features* topic. — Modified fourth and sixth bullets in *Intel Stratix 10 MX HBM2 Controller Features* topic. • Chapter 2, *Intel Stratix 10 MX HBM2 Architecture*: — Modified both figures in the *Intel Stratix 10 MX HBM2 Architecture* topic. — Added sentence to the paragraph immediately before Figure 4, in the *Intel Stratix 10 MX HBM2 Architecture* topic. — Expanded the last paragraph in the *Intel Stratix 10 MX HBM2 Architecture* topic. — Changed the specified width of write and read data interfaces per AXI port from 128-bits to 256-bits, in the *HBM2 burst transactions* description in the *Intel Stratix 10 MX HBM2 Controller Details* topic. — Removed the last sentence from the *User interface vs HBM2 Interface Frequency* description, in the *Intel Stratix 10 MX HBM2 Controller Details* topic. — Added the ECC description near the end of the *Intel Stratix 10 MX HBM2 Controller Details* topic.

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Chapter 3, *Generating the High Bandwidth Memory (HBM2) Interface Intel FPGA IP*: — Changed chapter title from *Generating the Intel Stratix 10 MX HBM2 IP* to *Generating the High Bandwidth Memory (HBM2) Interface Intel FPGA IP*. — Changed name of the menu selection in step 3 of the procedure and replaced the figure. — Changed the IP name in topic titles throughout the chapter. — Changed the description of the *Core clock frequency* parameter in the *Group: General / Clocks* section of the *General Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic. — In the *Diagnostic Parameters for High Bandwidth Memory (HBM2) Interface Intel FPGA IP* topic, removed two existing graphics and added one new one. Added *Enable mixed traffic* parameter to the *Group: Diagnostics / Traffic Generator* section. — Revised the procedure in the *Generating the Example Design* topic. — Removed the *Intel Stratix 10 MX HBM2 IP Example Design for Synthesis* topic. — Added the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Design Example* and *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Pin Planning* topics. — In the *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Pin Planning* topic, changed *Jitter Specifications for the Input Reference Clocks* from 10ps peak-to-peak to 20ps peak-to-peak.

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Chapter 4, *Simulating the High Bandwidth Memory (HBM2) Interface Intel FPGA IP*: — Changed product name from *Intel Stratix 10 MX HBM2 IP* to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP* in topic titles throughout the chapter. • Chapter 5, *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Interface*: — Changed product name from *Intel Stratix 10 MX HBM2 IP* to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP* in topic titles throughout the chapter. — Revised content of *Clock Signals* and *Reset Signals* topics. — Added *Calibration Status Signals* and *Memory Interface Signals* topics. — Added *AXI Interface Signals* and *AXI Address Definition* sections to the *AXI User-interface Signals* topic. Removed content from the descriptions of the `axi_0_0_awaddr` and `axi_0_0_araddr` ports, in tables 15 and 18, respectively. — Added *User APB Interface Timing* and *User-controlled Access to the HBM2 Controller* topics. — Added figures to the *User Refresh Per Bank* and *User Refresh to All Banks* sections in the *User-controlled Refreshes* topic. • Chapter 6,*High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Performance*: — Changed chapter title from *Intel Stratix 10 MX HBM2 IP Controller Performance* to *High Bandwidth Memory (HBM2) Interface Intel FPGA IP Controller Performance*. — Changed the IP name in topic titles throughout the chapter. — Added *High Bandwidth Memory (HBM2) Interface Intel FPGA IP DRAM Temperature Readout* topic.

Date	Version	Changes
December 2017	2017.12.22	Initial public release.