Hybrid Memory Cube Controller IP Core User Guide

Last updated for Quartus Prime Design Suite: 16.0
UG-01152
2016.05.02

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Subscribe
Send Feedback

now part of Intel

Contents

About the Altera Hybrid Memory Cube Controller IP Core

1

2016.05.02

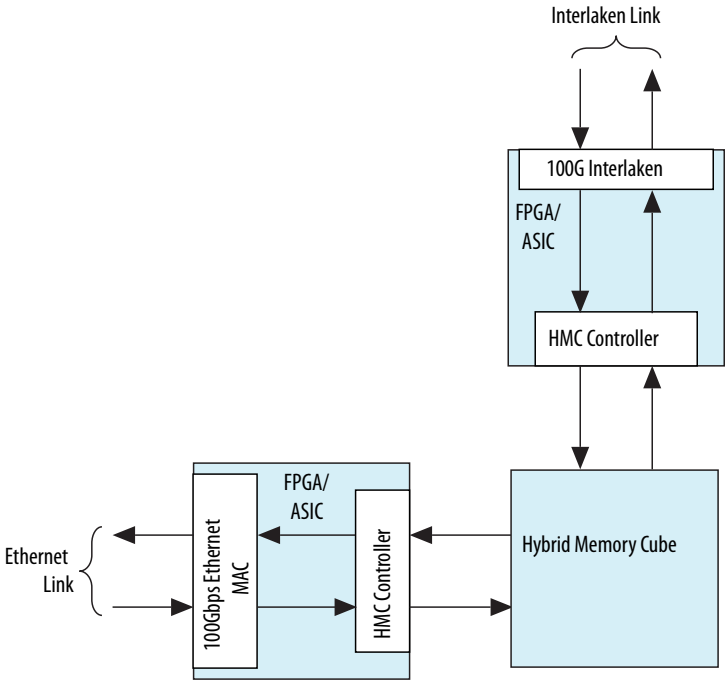
UG-01152

Subscribe Send Feedback

The Hybrid Memory Cube (HMC) specification defines a new type of memory device that provides a significant increase in bandwidth and power efficiency over existing memory architectures. The HMC specification targets high performance computers and next-generation networking equipment and provides scalability for a wide range of applications.

The Altera® HMC Controller MegaCore® IP core enables easy access to external HMC devices. HMC devices provide high bandwidth, reliable access to large amounts of memory with a small form factor, and provide significant system cost savings in high performance, memory intensive applications. The HMC Controller IP core provides a simple user interface through which you can communicate with an external HMC device to incorporate these bandwidth and performance gains in your design.

Figure 1-1: Typical HMC Controller Application



Related Information

- **HMC Controller IP Core User Guide Archives** on page 7-1

- **Hybrid Memory Cube Controller Design Example User Guide**
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.
- **HMC Specification 1.1**
  The HMC specification is available for download from the Hybrid Memory Cube Consortium web page.

HMC Controller IP Core Supported Features

The Altera HMC Controller IP core offers the following features:

- Communicates through Altera high-speed transceivers with an external HMC device compliant with the *HMC Specification 1.1*.
- Communicates with the HMC device at per-lane rates of 10 Gbps or 12.5 Gbps.
- Features Avalon® Memory-Mapped (Avalon-MM) interface to access control and status registers.
- Supports selection of a full-width variation that connects to 16 lanes of an HMC device, or a half-width variation that connects to 8 lanes of an HMC device.
  - Full-width IP core variations feature one to four simple 512-bit client data interfaces. Multiple data interfaces provide increased utilization of the HMC link.
  - Half-width IP core variations feature a single simple 256-bit client data interface.
- Supports memory READ and WRITE transactions with all valid payload sizes.
- Supports posted and non-posted versions of ATOMIC transactions, BIT WRITE transactions, and WRITE transactions.
- Supports MODE READ and MODE WRITE transactions.
- Supports optional response reordering in full-width variations, to ensure the IP core sends responses on each application response interface in the order it received the requests. When you select this option, the IP core manages the tags, which are not visible on the client interfaces.
- Supports Response Open Loop Mode for receive (RX) flow control to decrease device resource requirements.
- Supports token-based transmit (TX) flow control.
- Supports poisoned packets.
- Supports reordering of transceiver lanes for board-design flexibility.
- Supports link training sequence and provides word alignment, lane alignment, and transceiver status information in real time.
- Provides fast simulation support.
- Provides real-time error statistics.
- Provides hardware and software reset control.

- Provides power management control.
- Optionally supports ADME direct access to transceiver registers through the Altera System Console, for debugging or monitoring PHY signal integrity.
- Provides option to include ECC support in all M20K memory blocks configured in the IP core.

To support multi-link connection to the HMC device in your design, you can configure multiple HMC Controllers to communicate with the same HMC device through separate HMC links.

For the detailed HMC specification refer to the *HMC Specification 1.1*.

Related Information

HMC Specification 1.1
The HMC specification is available for download from the Hybrid Memory Cube Consortium web page.

HMC Controller IP Core Supported HMC Transaction Types

The Altera HMC Controller IP core supports all HMC transactions.

HMC Controller To HMC Device Packet Types

The HMC Controller IP core generates the following packet types on the link to the HMC device:

- NULL FLIT
- PRET (single FLIT packet)
- IRTRY (single FLIT packet)
- READ request (single FLIT packet)
- 16-byte WRITE or Posted WRITE request (2-FLIT packet)
- 32-byte WRITE or Posted WRITE request (3-FLIT packet)
- 48-byte WRITE or Posted WRITE request (4-FLIT packet)
- 64-byte WRITE or Posted WRITE request (5-FLIT packet)
- 80-byte WRITE or Posted WRITE request (6-FLIT packet)
- 96-byte WRITE or Posted WRITE request (7-FLIT packet)
- 112-byte WRITE or Posted WRITE request (8-FLIT packet)
- 128-byte WRITE or Posted WRITE request (9-FLIT packet)
- BIT WRITE or Posted BIT WRITE request (2-FLIT packet)
- MODE READ request (single FLIT packet)
- MODE WRITE request (2-FLIT packet)
- Dual 8-byte ADD IMMEDIATE or Posted Dual 8-byte ADD IMMEDIATE request (2-FLIT packet)
- Single 16-byte ADD IMMEDIATE or Posted Single 16-byte ADD IMMEDIATE request (2-FLIT packet)

The HMC Controller IP core operates in the Response Open Loop Mode and therefore does not generate TRET packets.

HMC Device to HMC Controller Packet Types

The HMC Controller IP core can process the following packet types generated by the HMC device:

- NULL FLIT
- PRET (single FLIT packet)
- TRET (single FLIT packet)
- IRTRY (single FLIT packet)
- ERROR response (single FLIT packet)
- WRITE response (single FLIT packet)
- 16-byte READ response (2-FLIT packet)
- 32-byte READ response (3-FLIT packet)
- 48-byte READ response (4-FLIT packet)
- 64-byte READ response (5-FLIT packet)
- 80-byte READ response (6-FLIT packet)
- 96-byte READ response (7-FLIT packet)
- 112-byte READ response (8-FLIT packet)
- 128-byte READ response (9-FLIT packet)
- MODE READ response (2-FLIT packet)
- MODE WRITE response (single FLIT packet)

The HMC Controller IP core does not define or support any vendor specific packet types.

Device Family Support

The following table lists the device support level definitions for Altera IP cores.

Table 1-1: Altera IP Core Device Support Levels

FPGA Device Families
Preliminary support — The core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
Final support — The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

The following table shows the level of support offered by the HMC Controller IP core for each Altera device family.

Table 1-2: HMC Controller IP Core Device Family Support

Device Family	Support
Arria 10	Preliminary
All other device families	No support

IP Core Verification

Before releasing a version of the HMC Controller IP core, Altera runs comprehensive regression tests in the current version of the Quartus® Prime software. The HMC Controller IP core is tested in simulation and hardware to confirm functionality.

Related Information

- **Knowledge Base Errata for HMC Controller IP core**
  Exceptions to functional correctness are documented in the HMC Controller IP core errata.
- **Altera IP Release Notes**
  Changes to the HMC Controller IP core are noted in the Altera IP Release Notes starting from the Quartus II software v15.0.

Simulation

Altera performs the following tests on the HMC Controller IP core in simulation, using the Micron HMC BFM:

- Constrained random tests that cover randomized legal payload sizes and contents
- Assertion based tests to confirm proper behavior of the IP core with respect to the specification
- Extensive coverage of packet retry functionality

Constrained random techniques generate appropriate stimulus for the functional verification of the IP core. Altera monitors line, expression, and assertion coverage metrics to ensure that all important features are verified.

Hardware Testing

Altera performs hardware testing of the key functions of the HMC Controller IP core. The Altera hardware tests of the HMC Controller IP core also ensure reliable solution coverage for hardware related areas such as performance, link initialization, and reset recovery.

Altera performs hardware testing on the Arria 10 GX FPGA Development Kit with an HMC daughter card. A Micron HMC 15G-SR device on the daughter card is connected to the development board through FMC connectors.

Performance and Resource Utilization

Table 1-3: HMC Controller IP Core FPGA Resource Utilization

Typical resource utilization for an HMC Controller IP core configured with a data rate of 10 Gbps, using the Quartus Prime software v16.0, with the following IP core features turned off:

- ADME support
- M20K ECC support

The numbers of ALMs and logic registers are rounded up to the nearest 100. The numbers of ALMs, before rounding, are the **ALMs needed** numbers from the Quartus Fitter Report.

IP Core Variation			Resource Utilization		
Link Width	Response Reordering	Number of Ports	ALMs Needed	Dedicated Logic Registers	M20K Blocks
Full-width	Off	1	24400	48200	51
		2	29200	58400	87
		3	34100	68600	123
		4	38900	78800	158
	On	1	29900	59400	55
		2	37000	76200	93
		3	44200	93100	132
		4	51300	109900	170
Half-width			13400	24000	37

Related Information

- **Fitter Resources Reports in the Quartus Prime Help**
  Information about Quartus Prime resource utilization reporting, including **ALMs needed**.
- **Quartus Prime Standard Edition Handbook, Volume 1: Design and Synthesis**

Device Speed Grade Support

Table 1-4: Minimum Recommended Device Family Speed Grades

Altera recommends that you configure the HMC Controller IP core only in the device speed grades listed in the table, or any faster (lower numbered) device speed grades that are available.

Altera does not support configuration of this IP core in slower (higher numbered) device speed grades.

Device Family	IP Core Variation: Lane Rate	
	10 Gbps	12.5 Gbps
Arria 10	E1, I1, E2, I2	E1, I1

Release Information

Table 1-5: HMC Controller IP Core Current Release Information

Item	Value
Version	16.0
Release Date	May 2016
Ordering Code	Full-width: IP-HMCSR15FW Half-width: IP-HMCSR15HW
Vendor ID	6AF7

Getting Started with the HMC Controller IP Core 2

2016.05.02

UG-01152 Subscribe Send Feedback

The following information explains how to install, parameterize, and simulate the Altera Hybrid Memory Cube Controller IP core.

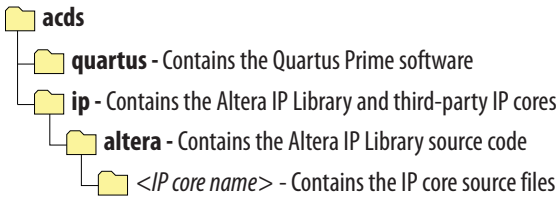
Related Information

- **HMC Controller IP Core Design Example** on page 6-1
  The HMC Controller design example provides an example of how to connect your IP core with an external I^2C master module and an external TX PLL.
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Licensing IP Cores

The Altera IP Library provides many useful IP core functions for your production use without purchasing an additional license. Some Altera MegaCore IP functions require that you purchase a separate license for production use. However, the OpenCore® feature allows evaluation of any Altera IP core in simulation and compilation in the Quartus Prime software. After you are satisfied with functionality and performance, visit the Self Service Licensing Center to obtain a license number for any Altera product.

Figure 2-1: IP Core Installation Path



Note: The default IP installation directory on Windows is **<*drive*>:\altera**<*version number*>; on Linux the IP installation directory is <*home directory*>**/altera/ <*version number*>**.

OpenCore Plus IP Evaluation

Altera's free OpenCore Plus feature allows you to evaluate licensed MegaCore IP cores in simulation and hardware before purchase. You only need to purchase a license for MegaCore IP cores if you decide to take your design to production. OpenCore Plus supports the following evaluations:

- Simulate the behavior of a licensed IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely. This requires a connection between your board and the host computer.

Note: All IP cores that use OpenCore Plus time out simultaneously when any IP core in the design times out.

Related Information

- **Altera Licensing Site**
- **Altera Software Installation and Licensing Manual**

Specifying IP Core Parameters and Options

The HMC Controller parameter editor allows you to quickly configure your custom IP variation. Use the following steps to specify IP core options and parameters in the Quartus Prime software.

1. In the IP Catalog (**Tools** > **IP Catalog**), under **Memory Interfaces and Controllers**, locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>**.qsys**. Click **OK**.
3. Specify the parameters and options for your IP variation in the parameter editor. Refer to the Parameters section for information about specific IP core parameters.
4. Click **Generate HDL**, the **Generation** dialog box appears.
5. To generate a simulation model of the HMC Controller IP core, under **Simulation** > **Create Simulation Model**, select **Verilog** HDL.
6. Specify other output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
7. Click **Finish**. The parameter editor adds the top-level **.qsys** file to the current project automatically. If you are prompted to manually add the **.qsys** file to the project, click **Project** > **Add/Remove Files in Project** to add the file.
8. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

HMC Controller IP Core Parameters

The HMC Controller parameter editor provides the parameters you can set to configure the HMC Controller IP core and simulation testbenches.

The HMC Controller parameter editor includes an **Example Design** tab. For information about that tab, refer to the **Hybrid Memory Controller Design Example User Guide**.

Table 2-1: HMC Controller IP Core Parameters

Parameters for customizing the HMC Controller IP core in the **IP** tab of the HMC Controller parameter editor.

Parameter	Type	Range	Default Setting	Parameter Description
Lanes	Integer	• 8 • 16	16	Selects half-width (8 lanes) or full-width (16 lanes) functionality.
Data rate	String	• 10 Gbps • 12.5 Gbps	10 Gbps	Selects the data rate on each lane.

Parameter	Type	Range	Default Setting	Parameter Description
CDR reference clock	String	• 125 MHz • 156.25 MHz	125 MHz	Selects the frequency of the input reference clock for the RX CDR PLL. You must drive the `rx_cdr_refclk0` input signal at the frequency you specify for this parameter. In addition, your design must derive this clock, the external transceiver TX PLL reference clock, and the `REFCLKP` and `REFCLKN` input signals of the external HMC device from the same clock source.
Ports	Integer	• 1 • 2 (available only for full-width variations) • 3 (available only for full-width variations) • 4 (available only for full-width variations)	1	Number of ports (data path interfaces). This parameter is useful only for full-width variations. Half-width variations have a single port. Increasing the number of ports increases utilization of the Hybrid Memory Cube, increasing efficiency. If you specify more than one port, each port is assigned a range of tags. • If you specify 2 ports, port 0 must use tags in the range 0 to 255, and port 1 must use tags in the range 256 to 511. • If you specify 3 ports, port 0 must use tags in the range 0 to 175, port 1 must use tags in the range 176 to 351, and port 2 must use tags in the range 352 to 511. • If you specify 4 ports, port 0 must use tags in the range 0 to 127, port 1 must use tags in the range 128 to 255, port 2 must use tags in the range 256 to 383, and port 3 must use tags in the range 384 to 511.

Parameter	Type	Range	Default Setting	Parameter Description
Response re-ordering	Boolean	• True • False	False	Specifies whether the IP core ensures that responses appear on each data response interface in the order the original requests arrived on the corresponding request interface. If you turn on this feature, the IP core manages tags internally. In that case tags are not available on the data interfaces. Turning on this feature can increase round-trip latency. This parameter is available only for full-width variations.
RX mapping	64-bit value		0xFEDCBA9876543210	Selects the RX lane mapping. Use caution in modifying this parameter. Refer to **RX Mapping and TX Mapping Parameters** on page 2-7.
TX mapping	64-bit value		0xFEDCBA9876543210	Selects the TX lane mapping. Use caution in modifying this parameter. Refer to **RX Mapping and TX Mapping Parameters** on page 2-7.

Parameter	Type	Range	Default Setting	Parameter Description
Enable ADME and Optional Reconfiguration Logic	Boolean	• True • False	False	Specifies whether the IP core turns on the ADME feature in the embedded Arria 10 Native PHY IP core that configures the transceivers. Turning on this parameter turns on the following Arria 10 PHY features: • **Enable Altera Debug Master Endpoint (ADME)** • **Enable capability registers** • **Enable control and status registers** • **Enable PRBS soft accumulators** **Note:** The **Share reconfiguration interface** PHY parameter is always turned on for this IP core. The ADME feature enables Native PHY register programming with the Altera System Console, and optional reconfiguration logic. For more information, refer to the *Arria 10 Transceiver PHY User Guide*.
Enable M20K ECC support	Boolean	• True • False	False	Specifies whether the IP core supports the ECC feature in the Arria 10 M20K memory blocks that are configured as part of the IP core. You can turn on this parameter to enhance data reliability by enabling single-error correction, double-adjacent-error correction, and triple-adjacent-error detection ECC functionality in the M20K memory blocks configured in your IP core. Turn off this parameter to decrease latency and resource utilization.

Related Information

- **Arria 10 Transceiver PHY User Guide**
  Provides information about the Arria 10 ADME feature.
- **Embedded Memory Blocks in Arria 10 Devices**
  Provides information about the Arria 10 M20K block ECC feature.

RX Mapping and TX Mapping Parameters

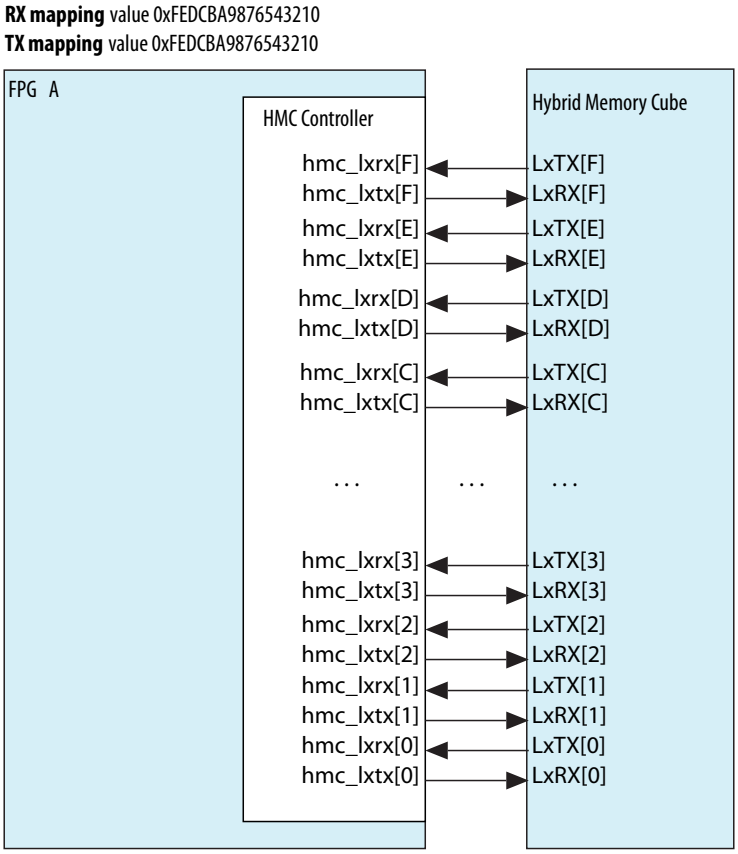
The HMC Controller IP core provides the **RX mapping** and **TX mapping** parameters for flexibility in board design.

The default values of these parameters specify the correct IP core behavior when the HMC device `LxTX[<i>]` output signal connects to the HMC Controller IP core `hmc_lxrx[<i>]` input port, and the `LxRX[<i>]` input signal connects to the HMC Controller IP core `hmc_lxtx[<i>]` output port, for each `<i>`.

However, if your design constraints prevent you from connecting these signals as expected, you can instead modify one or both HMC Controller IP core mapping parameters to accommodate the non-standard connection.

Note: The Quartus Prime Fitter prevents you from mapping the HMC Controller IP core lanes to Arria 10 device transceiver channels out of order. Therefore, these two parameters only compensate for out-of-order connections on the board between the Arria 10 transceiver pins and the HMC device ports.

Figure 2-2: Default RX and TX Mapping Parameter Values



If the HMC device LxTX[<i>] output signal connects to the HMC Controller IP core hmc_lxrx[<k>] input port, you must set the value in bits [(4<i>+3):(4<i>)] (nibble <i>) of the **RX mapping** parameter to 4'h<k>. Therefore, the default value of the **RX mapping** parameter is 0xFEDCBA9876543210, indicating that LxTX[F] connects to hmc_lxrx[F], LxTX[E] connects to hmc_lxrx[E], and so on.

If the HMC device LxRX[<i>] input signal connects to the HMC Controller IP core hmc_lxtx[<k>] input port, you must set the value in bits [(4<i>+3):(4<i>)] (nibble <i>) of the **TX mapping** parameter to 4'h<k>. Therefore, the default value of the **TX mapping** parameter is 0xFEDCBA9876543210, indicating that LxRX[F] connects to hmc_lxtx[F], LxRX[E] connects to hmc_lxtx[E], and so on.

Example: Non-Default RX Mapping Parameter Value

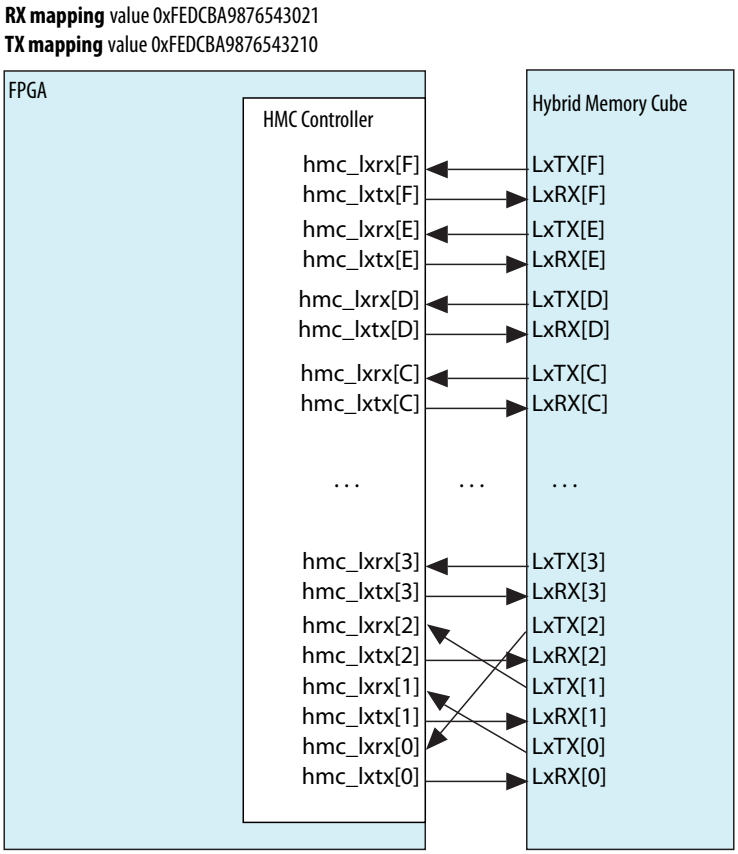
Table 2-2: Non-Default RX Connections

HMC Device Output Signal	IP Core Input Signal
LxTX[2]	hmc_lxrx[0]
LxTX[1]	hmc_lxrx[2]
LxTX[0]	hmc_lxrx[1]

Figure 2-3: Non-Default RX Mapping Parameter Value Example

If you connect the IP core `hmc_lxrx[2:0]` input signals according to the table, and connect all other IP core `hmc_lxrx[<i>]` input ports to the corresponding HMC device `LxTX[<i>]` output ports, you would set the value of the **RX mapping** parameter to 0xFEDCBA9876543**021** to compensate for the non-standard connection.

Note: The **RX mapping** parameter specifies the HMC device lane by position and the IP core lane by value. The figure illustrates a mapping parameter value of 0xFED.....43021 and not a value of 0xFED....43102.



Example: Non-Default TX Mapping Parameter Value

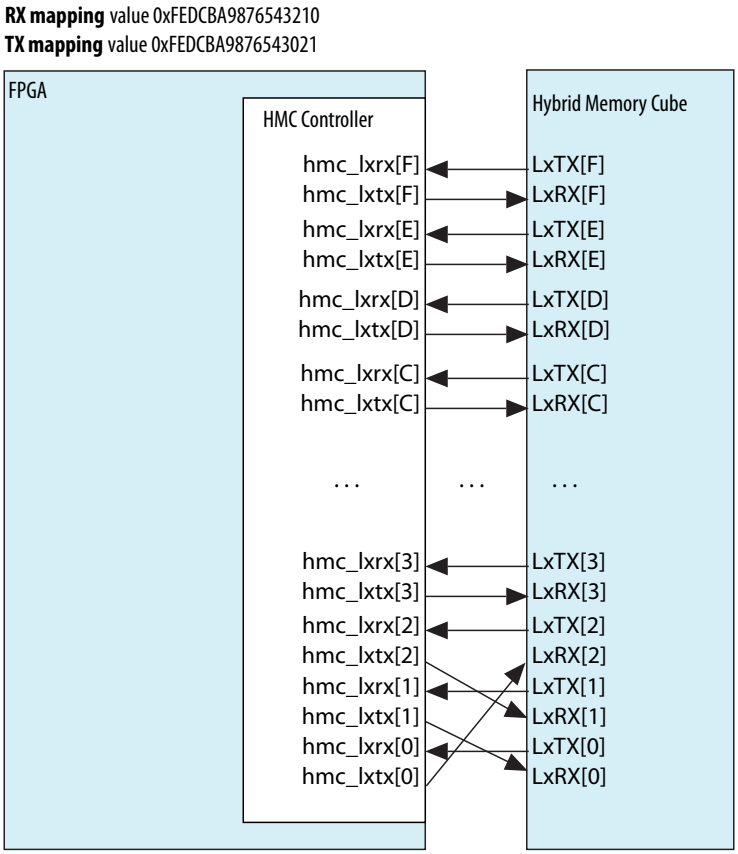
Table 2-3: Non-Default TX Connections

HMC Device Input Signal	IP Core Output Signal
`LxRX[2]`	`hmc_lxtx[0]`
`LxRX[1]`	`hmc_lxtx[2]`
`LxRX[0]`	`hmc_lxtx[1]`

Figure 2-4: Non-Default TX Mapping Parameter Value Example

If you connect the HMC Controller IP core `hmc_lxtx[2:0]` output signals according to the table, and connect all other IP core `hmc_lxtx[<i>]` output ports to the corresponding HMC device `LxRX[<i>]` input ports, you would set the value of the **TX mapping** parameter to 0xFEDCBA9876543**021** to compensate for the non-standard connection.

Note: The **TX mapping** parameter specifies the HMC device lane by position and the IP core lane by value. The figure illustrates a mapping parameter value of 0xFED.....43021 and not a value of 0xFED....43102.

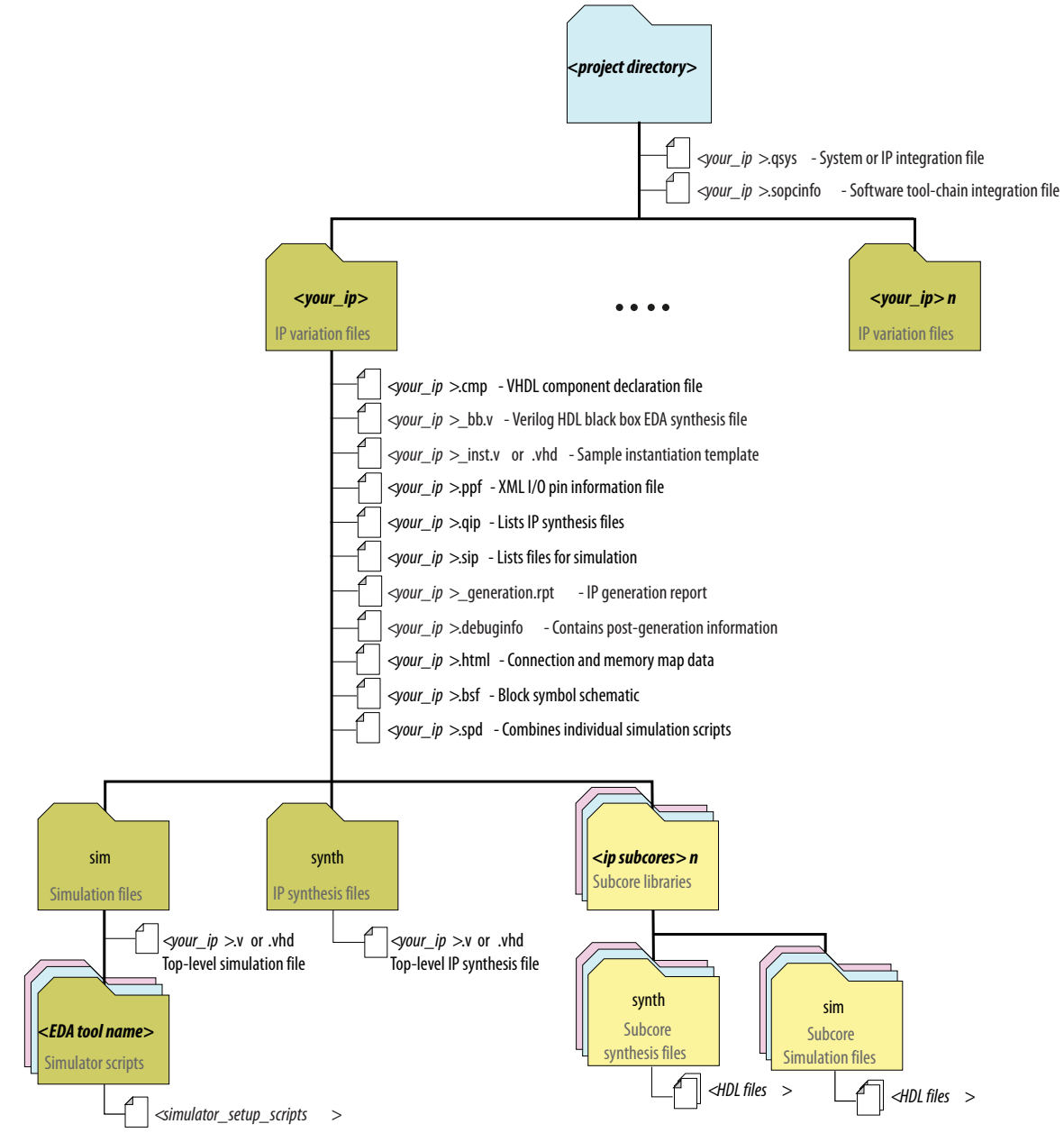


Use caution in modifying these parameters. In loopback configurations, you must ensure the **RX mapping** and **TX mapping** parameters specify reversed mappings. Otherwise, the IP core downstream of the RX lane swapper appears to receive data on the wrong lanes.

Files Generated for Altera IP Cores

The Quartus Prime software generates multiple files during generation of your IP core variation.

Figure 2-5: IP Core Generated Files



Integrating Your IP Core in Your Design

To ensure the HMC Controller IP core functions correctly in hardware, you must connect additional blocks to your IP core and assign device pins in order.

Pin Constraints

When you integrate your HMC Controller IP core instance in your design, you must make appropriate pin assignments. You can create a virtual pin to avoid making specific pin assignments for top-level signals while you are simulating and not ready to map the design to hardware.

When you are ready to map the design to hardware, you must enforce the following constraints:

- Adjacent HMC Controller lanes must map to adjacent Altera device pins. You cannot swap the lane order by mapping lanes to other Altera device pins. Instead, use the **RX mapping** and **TX mapping** parameters to compensate for board design issues.
- The lanes of an HMC Controller IP core must be configured in no more than three transceiver blocks. To enforce this constraint, you must configure IP core lanes in transceiver channels with the following restrictions:
  - Lane 0 of a full-width HMC Controller IP core must map to channel 0, 1, or 2 of a transceiver block.
    - If Lane 0 maps to channel 0, then HMC Controller Lane 1 must map to channel 1 of the same transceiver block (transceiver block N), and Lane 15 maps to channel 3 of the transceiver block N+2.
    - If Lane 0 maps to channel 1, then HMC Controller Lane 1 must map to channel 2 of the same transceiver block (transceiver block N), and Lane 15 maps to channel 4 of the transceiver block N+2.
    - If Lane 0 maps to channel 2, then HMC Controller Lane 1 must map to channel 3 of the same transceiver block (transceiver block N), and Lane 15 maps to channel 5 of the transceiver block N+2.
  - Lane 0 of a half-width HMC Controller IP core can map to any channel. If it maps to any of channels 0, 1, 2, 3, or 4, the IP core lanes are configured in two transceiver blocks.

Required External Blocks

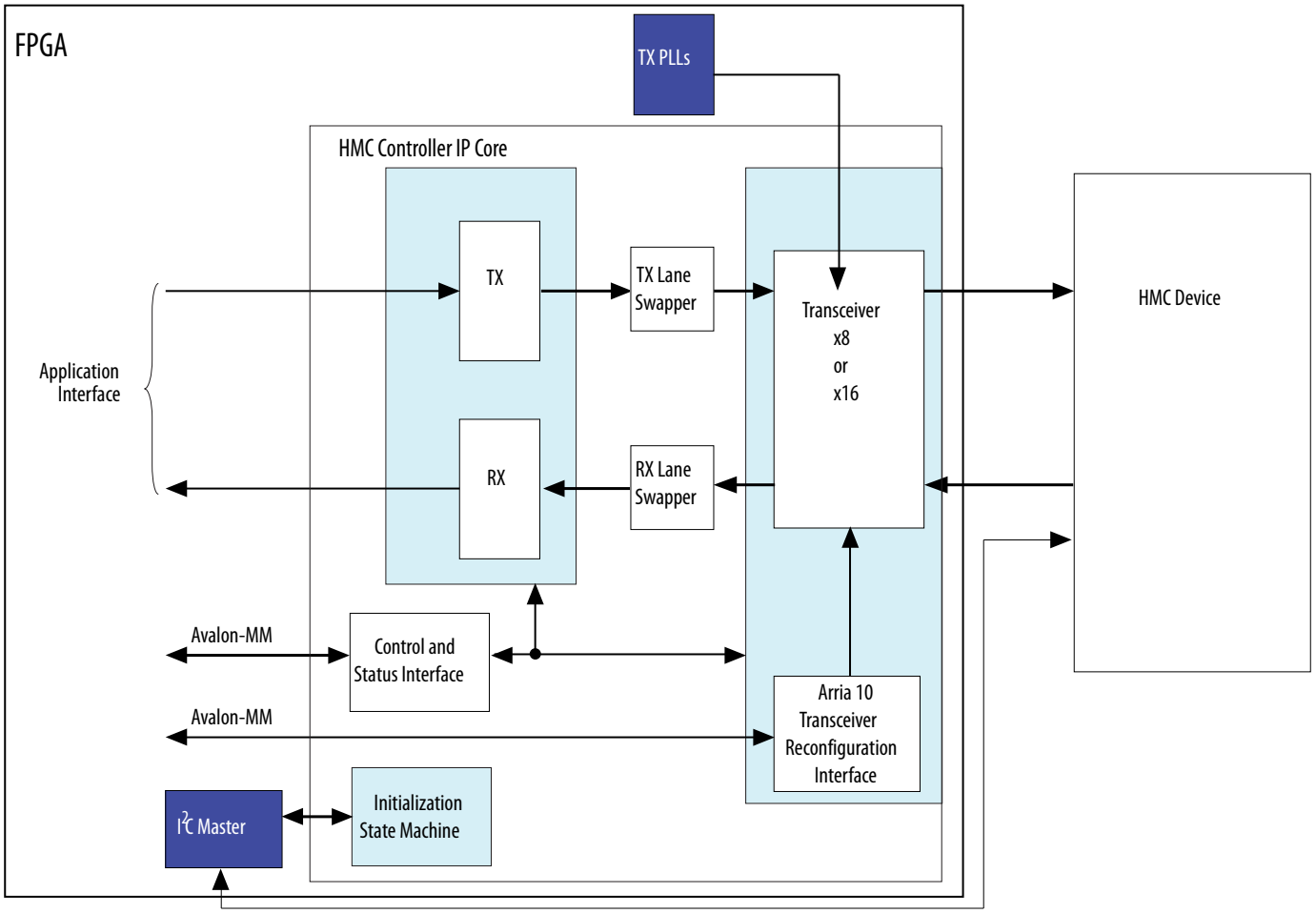
To ensure the HMC Controller IP core functions correctly in hardware, you must connect additional blocks to your IP core.

The HMC Controller IP core requires that you define and instantiate the following additional modules:

- External PLL IP core to configure transceiver TX PLL for all of the HMC lanes. Although the hardware these IP cores configure might physically be part of the device transceiver, you must instantiate them in software separately from the HMC Controller IP core. This requirement supports the configuration of multiple Altera IP cores using the same transceiver block in the device.
- An external I^2C master module in your design. Your design must include this module to initialize the HMC device to which your IP core connects.

Figure 2-6: Required External Blocks

The required external blocks appear darker than the other blocks in the figure. The external TX PLL IP core configures an ATX PLL in the device transceiver or an fPLL in Transceiver mode.



Adding the External PLL

The HMC Controller IP core requires that you generate and connect an external transceiver PLL IP core. You must generate the PLL IP core required to clock the transceiver channels that are configured as HMC Controller IP core lanes. The ATX PLL IP core configures the transceiver PLL in the transceiver in hardware, but you must generate the transceiver PLL IP core separately from the HMC Controller IP core in software. You can also configure an fPLL in transceiver mode. If you do not generate and connect the transceiver PLL IP core, the HMC Controller IP core does not function correctly in hardware.

You can use the IP Catalog to generate the external PLL IP core that configures a transceiver PLL on the device. In the IP Catalog, select **Arria 10 Transceiver ATX PLL** or **Arria 10 fPLL**.

In the transceiver PLL parameter editor, you must follow the instructions in the *Arria 10 Transceiver PHY User Guide* to configure the PLL IP core in the xN bonding configuration. In addition, you must set the following parameter values:

- **PLL output frequency** to one half of the per-lane data rate of the IP core variation. The transceiver performs dual edge clocking, using both the rising and falling edges of the input clock from the PLL. Therefore, this PLL output frequency setting drives the transceiver with the correct clock for the lanes that connect to the HMC device.
- **PMA interface width** to **32**.
- **PLL integer reference clock frequency** (ATX PLL) or **Desired reference clock frequency** (fPLL).

  Note: The HMC Controller IP core does not support PLL feedback compensation bonding.

  Altera recommends that you specify 125 MHz, 156.25 MHz, or 166.67 MHz. You can theoretically specify any reference clock frequency from which the PLL can generate the required output clock frequency. However, you must drive this TX PLL and the RX CDR PLL (`rx_cdr_refclk0` input signal to the HMC Controller IP core) and the HMC device reference clock input signals (`REFCLKP` and `REFCLKN`) from the same clock source.

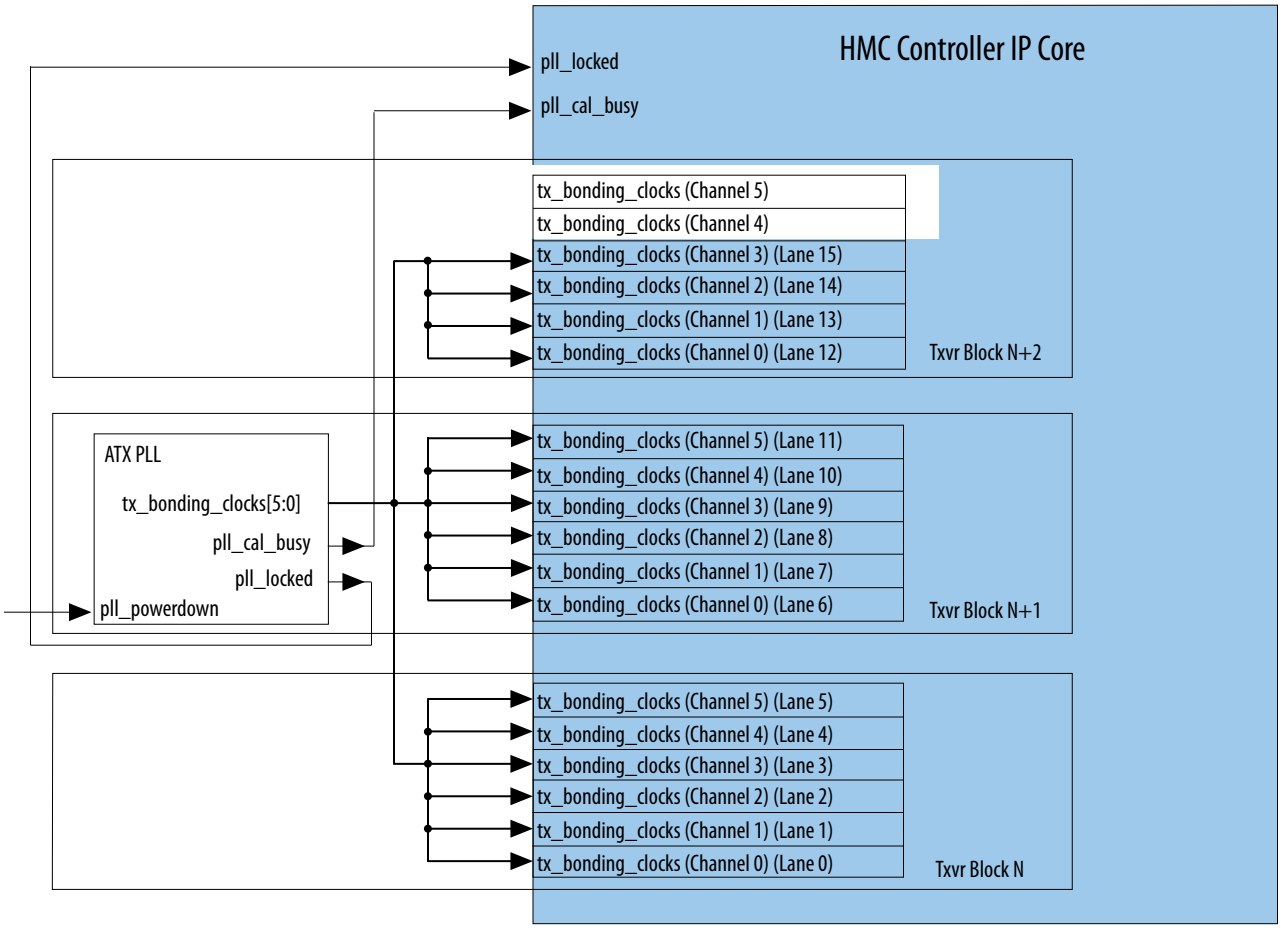
  Note: You must drive the external PLL reference clock input signal at the frequency you specify for this parameter.

In xN bonding mode, a single PLL is sufficient to drive the channels in the configured transceiver blocks. Recall that your HMC link TX serial lanes must be configured in order in adjacent physical transceiver channels so that these lanes configure a maximum of three transceiver blocks. You can view I/O constraints that enforce these requirements in the design example Quartus Settings File **hmcc_example.qsf** provided with the HMC Controller IP core.

The PLL output connects directly to the x6 network for its transceiver block and drives additional transceiver blocks through the xN clock network.

Figure 2-7: Transceiver PLL Connections Example with xN Bonding Scheme

Example connections between a full-width HMC Controller IP core and a single ATX PLL IP core in xN bonding mode.



You must connect the external PLL signals and the HMC Controller IP core transceiver TX PLL interface signals according to the following rules:

HMC Controller Signal	Connects to TX PLL Signal
`tx_bonding_clocks[5:0]` input signal for HMC lane N	`tx_bonding_clocks[5:0]` output vector of PLL IP core for the transceiver block in which lane N is configured. In the case of xN bonding, a single PLL connects to the xN clock network and the `tx_bonding_clocks[5:0]` input pins for HMC lanes in a different transceiver block from the configured PLL receive the clock from the xN clock network.
`pll_locked` input signal	`pll_locked` output signal of the external PLL for all of the HMC lanes.
`pll_cal_busy` input signal	`pll_cal_busy` output signal of the external PLL for all of the HMC lanes.

User logic must provide the AND and OR functions and connections.

Related Information

- **External PLL Interface** on page 3-4
- **Signals on the Interface to the External PLL** on page 4-17
- **HMC Controller IP Core Design Example** on page 6-1
  The HMC Controller design example provides an example of how to connect an external PLL to your HMC Controller IP core.
- **Pin Constraints** on page 2-11
  Describes the requirement that your IP core lanes configure a maximum of three transceiver blocks.
- **Arria 10 Transceiver PHY User Guide**
  Information about the bonding configurations and the correspondence between PLLs and transceiver channels, and information about how to configure an external PLL for your own design. You specify the bonding mode in the PLL parameter editor.

Adding the External I²C Master Module

The HMC Controller IP core requires that you instantiate an external I²C master module in your design. Your design must include this module to initialize the HMC device to which your IP core connects.

The I²C master module in your system must load the HMC device configuration registers according to the initialization requirements of the specific HMC device in your system.

The HMC specification requires that you set the HMC device `REGISTER REQUEST commands` register to the value of `Init Continue` after sending the commands to initialize the HMC. Therefore, the I²C master module must set this register to indicate successful completion of the HMC device configuration register load sequence.

In addition, the I²C master module must provide the following two signals to connect to the HMC Controller IP core:

- An input signal that accepts requests to load the configuration registers of the HMC device. You must connect this signal to the HMC Controller IP core `i2c_load_registers` output signal. If multiple HMC Controller IP cores connect to the same HMC device, you must connect this input signal to the AND of the individual HMC Controller IP core `i2c_load_registers` output signals. You must provide the AND function.
- An output signal that indicates successful completion of the configuration register load sequence. The I²C master must implement this signal with the following behavior:

  1. Deassert this signal when coming out of reset.
  2. Assert this signal after writing `Init Continue` to the HMC device `REGISTER REQUEST commands` register.
  3. Deassert this signal in response to the falling edge of the input signal described above.

  You must connect this signal to the HMC Controller IP core `i2c_registers_loaded` input signal. If multiple HMC Controller IP cores connect to the same HMC device, you must connect this signal to the `i2c_registers_loaded` signals of all of the HMC Controller IP cores.

For information about the required register configuration sequence, you must refer to the data sheet of the HMC device that is connected to your HMC Controller IP core. Recall that the HMC Controller IP core operates in Response Open Loop Mode, and you must configure the HMC device to communicate correctly with the IP core in this mode. In addition, because the IP core does not support the `TGA` field,

you must configure the HMC device to respond to every non-posted Write request with a Write response packet.

Related Information

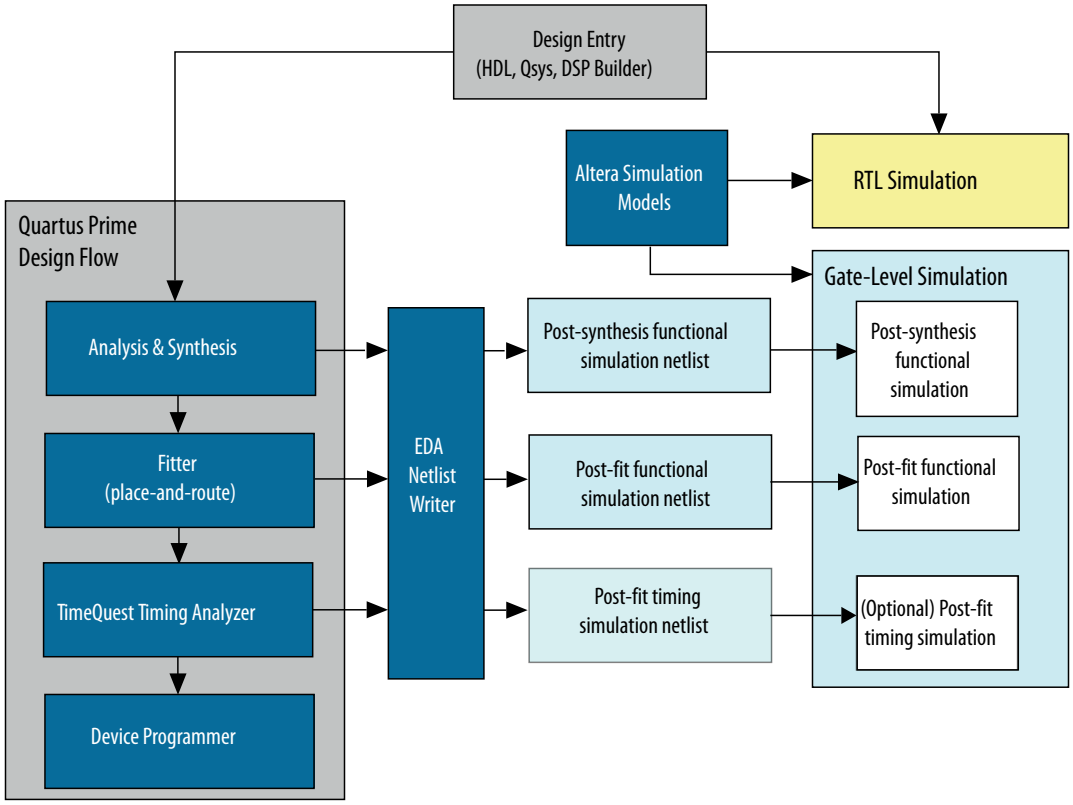
- **HMC Controller IP Core Design Example** on page 6-1
  The HMC Controller design example provides an example I^2C master module and demonstrates how to connect it to your HMC Controller IP core.
- **Interface to External I2C Master** on page 3-3
- **Signals on the Interface to the I2C Master** on page 4-11
  Describes the signals on this interface and the four-way handshaking protocol that the HMC Controller IP core implements and that the I^2C master must implement for correct IP core functionality.
- **HMC Specification 1.1**
  The *Power-On and Initialization* section of the HMC specification describes the initialization sequence requirements.

Simulating Altera IP Cores

The Quartus Prime software supports RTL and gate-level design simulation of Altera IP cores in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

You can use the functional simulation model and the testbench or design example available with your IP core for simulation. When you click the **Generate Example Design** button, the functional simulation model and testbench files are generated in a location you specify. By default, if you do not modify the target location, they are generated in a project subdirectory. This directory includes scripts to compile and run the testbench. For a complete list of models or libraries required to simulate your IP core, refer to the scripts generated with the testbench.

Figure 2-8: Simulation in Quartus Prime Design Flow



Note: Post-fit timing simulation is not supported for 28nm and later device architectures. Therefore, the HMC Controller IP core does not support post-fit timing simulation.

Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model.

Note: Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

If you use an HMC BFM to simulate your HMC Controller IP core, ensure that you set the BFM parameters to match the features of your HMC Controller IP core and design. For example, confirm that you set the BFM memory size (2G or 4G) to match the address space that you expect your design to access, and that you set the BFM to communicate correctly with the HMC Controller IP core in Response Open Loop Mode. You must also set the BFM to send Write response packets for non-posted Write transactions received, because the HMC Controller IP core does not support the TGA field.

Related Information

Simulating Altera Designs

Functional Description 3

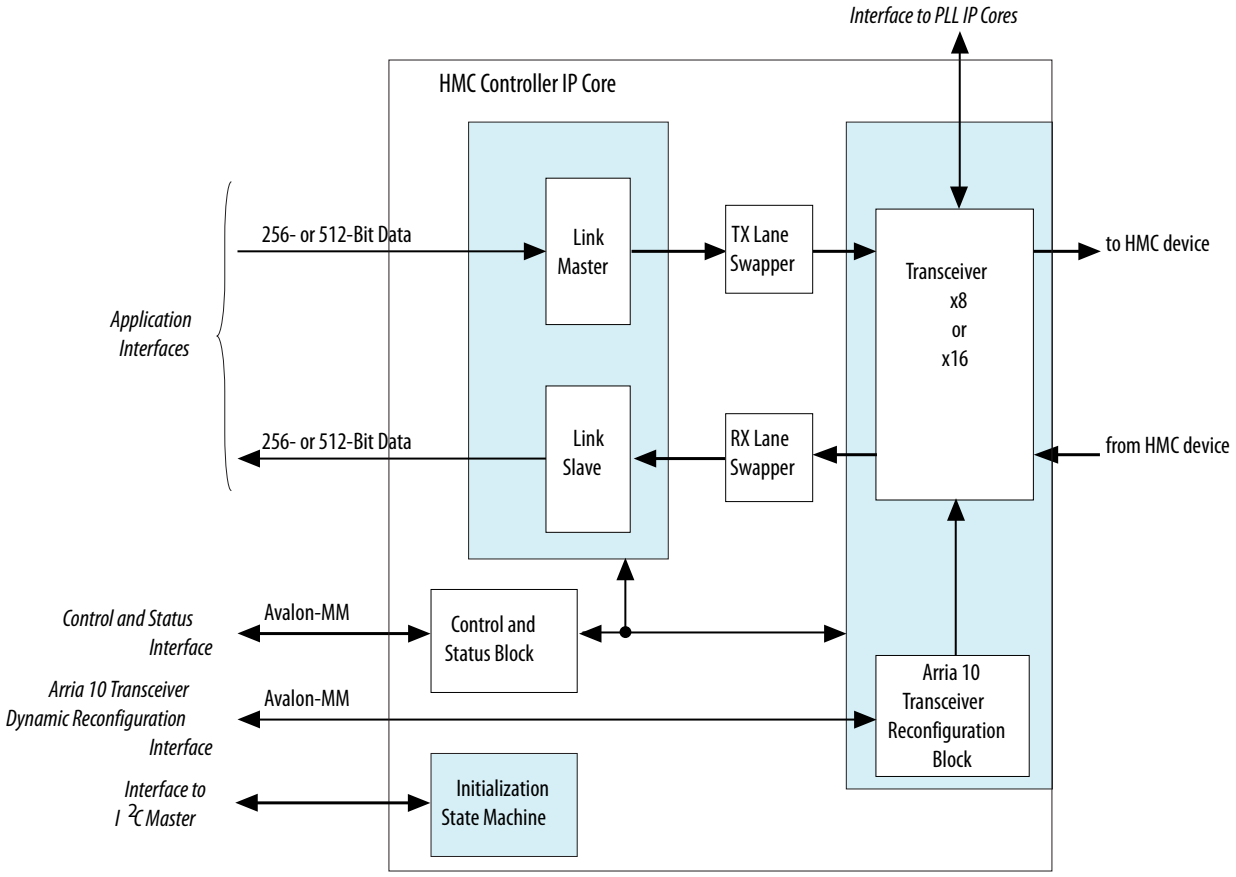
2016.05.02

UG-01152 Subscribe Send Feedback

The Altera HMC Controller MegaCore IP core enables easy access to external HMC devices.

High Level Block Diagram

Figure 3-1: HMC Controller IP Core Block Diagram



The HMC Controller IP core includes the following components:

- Two data paths, an HMC TX path and an HMC RX path. Each path includes a link layer module, a lane swapper, and high-speed transceivers on the HMC link.
- An initialization state machine.
- A register control block.
- An Arria 10 Native PHY dynamic reconfiguration block.

The TX lane swapper remaps the HMC TX lanes to transceiver channels according to the **TX mapping** parameter. The RX lane swapper remaps the HMC RX lanes from transceiver channels according to the **RX mapping** parameter.

Interfaces Overview

The Altera HMC Controller IP core supports multiple external interfaces.

Application Interfaces

The data path request and response interfaces, also called the application request interface and the application response interface, provide a 256-bit or 512-bit data bus and dedicated signals for the application to provide HMC request packet field values and to read HMC response packet field values. Full-width variations support 1, 2, 3, or 4 pairs of data path request and response interfaces. Half-width variations support 1 pair of data path request and response interfaces.

Table 3-1: Application Data Width in Full- and Half-Width IP Core Variations

IP Core Variation	Full-width	Half-width
Memory Interface to HMC Device	16-lane link	8-lane link
Client Data Interface Width	512 bit	256 bit

Related Information

Application Interface Signals on page 4-1

HMC Interface

The HMC interface connects to the external HMC device, and complies with the HMC specification. The interface provides a single 8-lane or 16-lane link, configured in an Altera device in 8 or 16 adjacent transceiver channels.

The HMC Controller IP core operates in Response Open Loop Mode.

Related Information

- **HMC Interface Signals** on page 4-10
The HMC Controller IP core's HMC interface connects to the external HMC device's link interface and main reset signal.
- **HMC Controller IP Core Supported HMC Transaction Types** on page 1-3

Interface to External I²C Master

The HMC Controller IP core requires that you instantiate an external I²C master module in your design. This external I²C master module must coordinate link initialization on the link between the HMC and the HMC Controller. The I²C master coordinates with the HMC Controller internal initialization state machine and programs configuration registers in the HMC device to which your IP core connects.

Separating the HMC Controller IP core from the I²C master module provides design flexibility. Because the IP core does not include the I²C master module, you can instantiate a single I²C master to control link initialization for multiple HMC Controller IP cores. A single I²C master module can also control other I²C slaves.

Related Information

- **Adding the External I2C Master Module** on page 2-16
  Information about how to connect the HMC Controller IP core to the external I2C master module.
- **Signals on the Interface to the I2C Master** on page 4-11
  Describes the signals on this interface and the four-way handshaking protocol that the HMC Controller IP core implements and that the I²C master must implement for correct IP core functionality.

Control and Status Register Interface

The control and status register interface provides access to the HMC Controller IP core internal control and status registers. This interface does not provide access to the transceiver registers.

The control and status interface complies with the Avalon Memory-Mapped (Avalon-MM) specification defined in the *Avalon Interface Specifications*.

The control and status interface provides a 32-bit wide data bus for register content. All HMC Controller control and status registers are 32 bits wide and all register accesses through the control and status interface read or write the full 32 bits of register content.

Related Information

- **Control and Status Interface Signals** on page 4-12
- **HMC Controller IP Core Register Map** on page 5-1
- **Avalon Interface Specifications**

Status and Debug Interface

The status and debug interface provides signals to communicate successful link initalization and to support debugging of your HMC system.

Related Information

Status and Debug Signals on page 4-13

Transceiver Control Interfaces

The HMC Controller IP core supports the following transceiver control interfaces:

External PLL Interface on page 3-4

Transceiver Reconfiguration Interface on page 3-4

External PLL Interface

The HMC Controller IP core requires that you generate an external transceiver PLL IP core and connect it to each HMC Controller IP core lane.

If you do not generate and connect the transceiver PLL IP core, the HMC Controller IP core does not function correctly in hardware.

Related Information

- **Adding the External PLL** on page 2-13
  Describes how to generate an external transceiver PLL IP core, including parameter requirements.
- **Signals on the Interface to the External PLL** on page 4-17
- **HMC Controller IP Core Design Example** on page 6-1
  The HMC Controller design example provides an example of how to connect an external PLL to your HMC Controller IP core.
- **Arria 10 Transceiver PHY User Guide**
  Information about the Arria 10 transceiver PLLs and clock network.

Transceiver Reconfiguration Interface

The transceiver reconfiguration interface provides access to the registers in the embedded Native PHY IP core. This interface provides direct access to the hard PCS registers on the device.

The transceiver reconfiguration interface complies with the Avalon Memory-Mapped (Avalon-MM) specification defined in the *Avalon Interface Specifications*.

Related Information

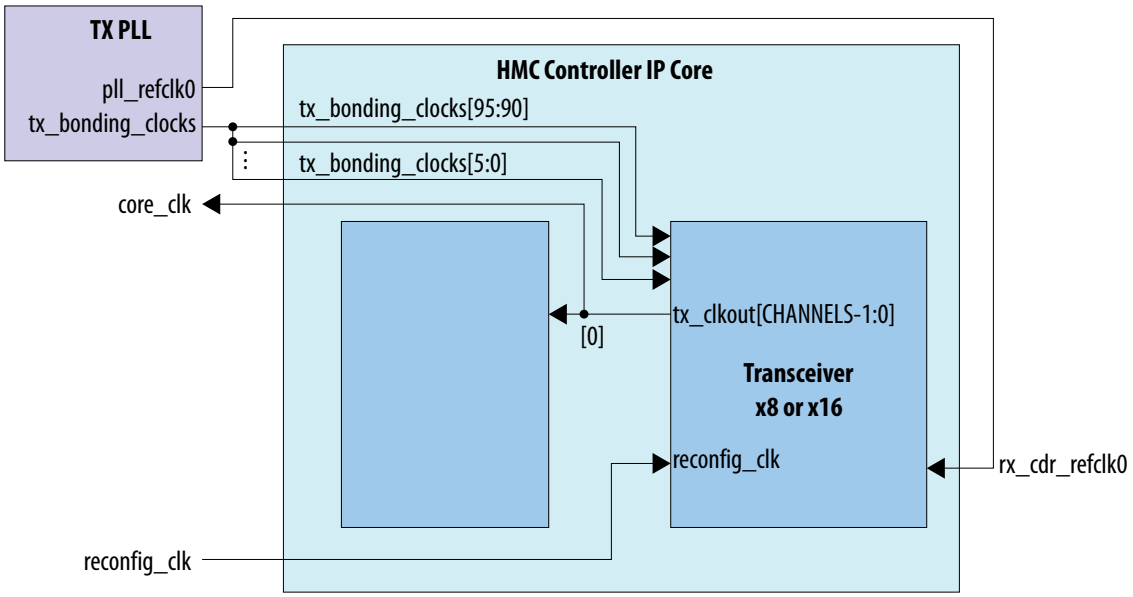
- **Transceiver Reconfiguration Signals** on page 4-15
- **Avalon Interface Specifications**
  Defines the Avalon Memory-Mapped (Avalon-MM) specification.
- **Arria 10 Transceiver PHY User Guide**
  Information about the Arria 10 transceiver reconfiguration interface.
- **Arria 10 Transceiver Registers**
  Detailed information about the Arria 10 transceiver registers.

Clocking and Reset Structure

The HMC Controller IP core has a single core clock domain and multiple transceiver-related clock domains.

Your design must derive the external transceiver TX PLL reference clock, the RX CDR reference clock, and the `REFCLKP` and `REFCLKN` input signals of the external HMC device from the same clock reference source. This requirement ensures a 0 PPM difference between the receive and transmit clocks, as required by the HMC specification.

Figure 3-2: HMC Controller IP Core Clocking Diagram



Related Information

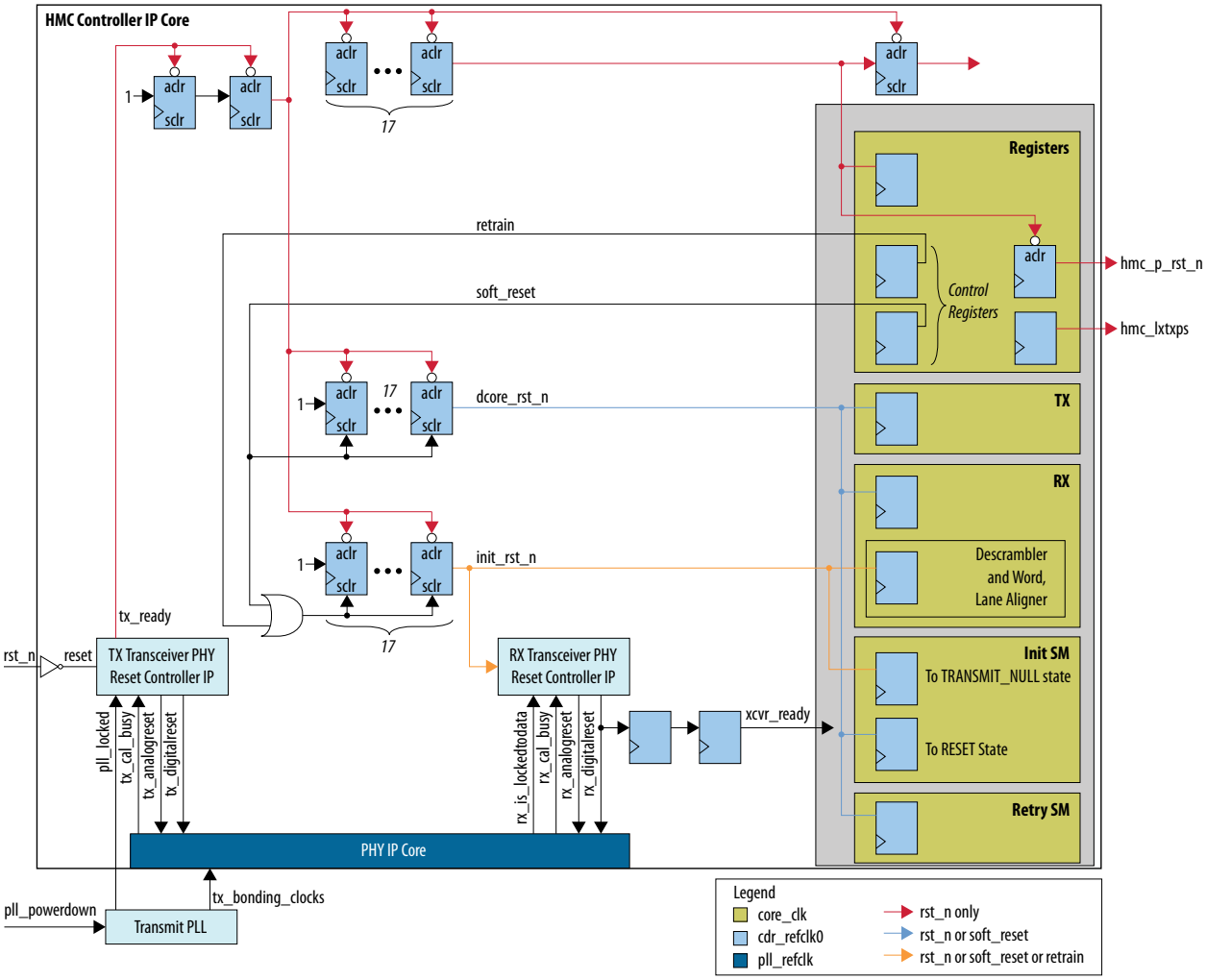
Clock and Reset Signals on page 4-14
Lists the clock and reset signals and the relationships between them.

Initialization and Reset

When you assert the active low `rst_n` signal, you trigger initialization of the HMC Controller IP core, the HMC device, and the HMC link that connects them. To ensure the correct sequence, you must connect the HMC Controller IP core, the I^2C master module, and the HMC device correctly. The IP core provides register fields to support link reinitialization, soft reset, fatal error recovery, and power management sleep and down modes.

Figure 3-3: Initialization and Reset Options in the Full-Width HMC Controller IP Core

The HMC Controller IP core provides several mechanisms to reset various parts of the IP core, including the `rst_n` input signal and the `SoftReset` and `Retrain` fields of the HMC Controller IP core `CONTROL` register. The `SoftReset` and `Retrain` fields of the `CONTROL` register are available only in full-width IP cores.



Initialization

The following signals control HMC Controller IP core, HMC link, and HMC device initialization:

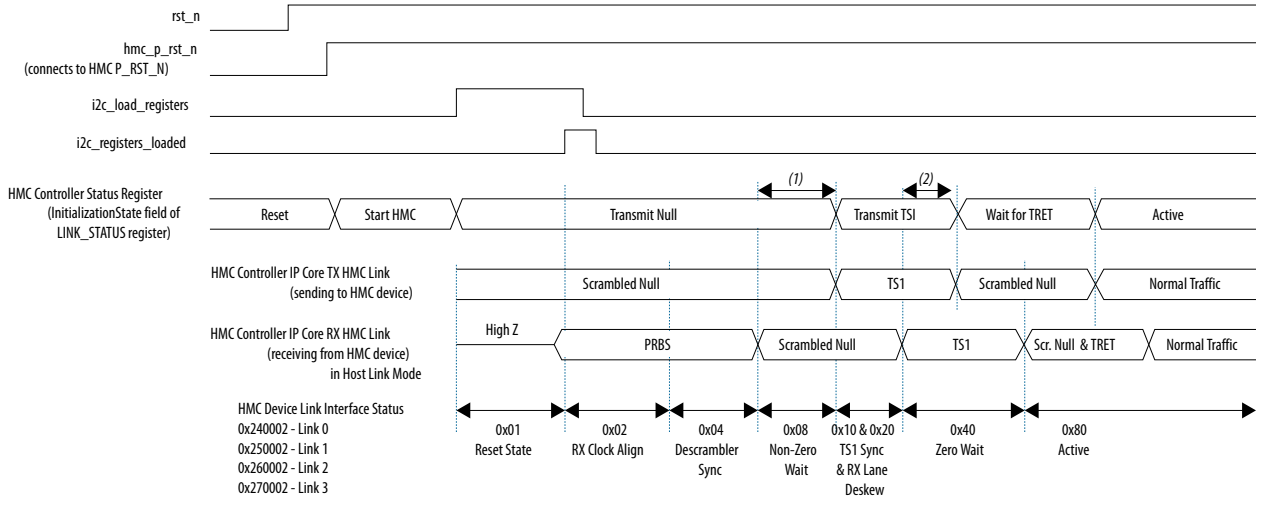
- `rst_n`: Active low HMC Controller IP core input signal that triggers IP core initialization
- `hmc_p_rst_n`: HMC Controller IP core output signal.
  - Signal control: While `rst_n` is asserted, and immediately after it is deasserted, the IP core controls this output signal. However, after the IP core raises this signal when `rst_n` is deasserted, the `P_RST_N` field in bit [17] of the `CONTROL` register drives this signal. Software can modify this bit to force the HMC device to reset.
  - Signal connection: You should connect this signal to the active low HMC device `P_RST_N` input signal. `P_RST_N` triggers HMC device initialization.
- `i2c_load_registers`: HMC Controller IP core output signal. You should connect this signal to the I^2C master module input signal that tells the I^2C master module to load the configuration registers of the HMC device.
- `i2c_registers_loaded`: HMC Controller IP core input signal that indicates the I^2C master module has completed its part in the initialization of the HMC device. You should connect this signal to the I^2C master module output signal that indicates successful completion of the HMC configuration register load sequence.

The IP core reports link initialization status in the `InitializationState` field of the `LINK_STATUS` register at offset 0x10. The HMC device reports its own link initialization status in its own link interface status registers.

When you initialize the HMC link, recall the following HMC Controller IP core requirements:

- The HMC Controller IP core operates in Response Open Loop Mode, and you must configure the HMC device to communicate correctly with the IP core in this mode.
- The IP core does not support the `TGA` field, so you must configure the HMC device to acknowledge every non-posted Write request with a Write response packet.

Figure 3-4: HMC Link Initialization Sequence



Notes:
1. HMC Controller IP core acquires descrambler sync. The Descrambler Sync field of the Lane Status register goes to all ones.
2. HMC Controller IP core does word and lane alignment. The Lanes Aligned bit pf the Link Status register goes to 1 and the Word Lock field of the Lane Status register goes to all ones.

Fatal Error Recovery

If the HMC device declares a fatal error, you must perform a warm reset of the HMC device and a reset of the HMC Controller IP core. To perform the HMC device warm reset, set the appropriate field in the HMC Global Configuration register.

You can perform a soft reset or a hard reset of the HMC Controller IP core. To perform the HMC Controller IP core soft reset, set the `SoftReset` field in bit [2] of the HMC Controller IP core `CONTROL` register To perform the hard reset, assert the `rst_n` input signal.

If the HMC Controller IP core declares a fatal error, set the `ClearFatalError` field in bit [1] of the `CONTROL` register. In some cases, this action is sufficient to force the IP core to resume normal operation. If this action is not sufficient, you must perform a full reinitialization of the IP core.

Power Management

To save power during a period of inactivity on the link, you can put the HMC link in sleep mode. After a certain amount of time in sleep mode, the HMC device automatically moves to down mode.

To put the link to sleep:

1. Stop sending requests on the data path request interfaces, to force the IP core to stop sending requests on the HMC link.
2. Wait for all responses to arrive and tokens to return.
3. Write the value of 0 to the `TXPS` field in bit [16] of the `CONTROL` register.
4. Wait for the IP core to set the `RXPS` field in bit [16] of the `LINK_STATUS` register to the value of 0.

To wake the link up from sleep mode or from down mode:

1. Write the value of 1 to the `Retrain` field in bit [0] of the `CONTROL` register.
2. Write the value of 1 to the `TXPS` field in bit [16] of the `CONTROL` register.
3. Wait for the IP core to set the `RXPS` field in bit [16] of the `LINK_STATUS` register to the value of 0.

Link Reinitialization

To recover from a fatal error, to wake up the link from sleep mode, or to retrain a link exhibiting a high bit error rate, you must reinitialize the HMC link.

After the HMC device exits sleep mode, you must reinitialize the link to resynchronize descramblers and perform word and lane alignment.

To reinitialize the link, write a 1 to the `Retrain` field in bit [0] of the `CONTROL` register.

Related Information

- **Adding the External I2C Master Module** on page 2-16
  Describes the I^2C master module requirements and the module's connections to the HMC Controller IP core.
- **Signals on the Interface to the I2C Master** on page 4-11
  Describes the detailed behavior of the I^2C master module during initialization.
- **HMC Interface Signals** on page 4-10
  This interface includes the `hmc_p_rst_n` signal.
- **Clock and Reset Signals** on page 4-14
- **LINK_STATUS Register** on page 5-4
- **CONTROL Register** on page 5-2

M20K ECC Support

If you turn on **Enable M20K ECC support** in your HMC Controller IP core variation, the IP core takes advantage of the built-in device support for ECC checking in all M20K blocks configured in the IP core on the device. The feature performs single-error correct, double-adjacent-error correct, and triple-adjacent-error detect ECC functionality in the M20K memory blocks configured in your IP core.

The HMC Controller IP core reports ECC error statistics in the registers `RETRY_ECC_COUNT` at offset 0x38 and `RESPONSE_ECC_COUNT` at offset 0x3C.

This feature enhances data reliability but increases request-to-response latency and resource utilization. Enabling this feature might reduce the maximum operating frequency (f_{MAX}) and might increase the difficulty of closing timing in full-width variations.

Related Information

- **Error and Retry Statistics Registers** on page 5-10
  Describes the `RETRY_ECC_COUNT` and `RESPONSE_ECC_COUNT` registers.
- **Embedded Memory Blocks in Arria 10 Devices**
  Information about the built-in ECC feature in Arria 10 devices.

Flow Control

The HMC specification describes two possible flow control schemes for host-to-HMC traffic, token based flow control and Response Open Loop Mode.

In token-passing mode, the device sends information about its buffering capacity to the HMC link partner during transaction layer initialization. In Response Open Loop Mode, the device does not send information about its buffering capacity to the HMC link partner. Instead, it only sends a request packet when it has room to receive the response at any time.

The HMC Controller IP core operates in Response Open Loop Mode. The IP core is designed to have the capacity to accept all response packets from the HMC device.

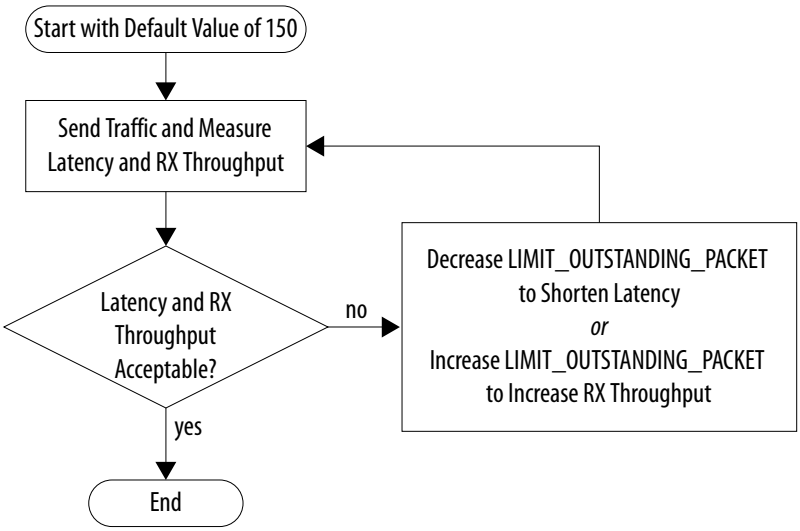
When user requests come in faster than the HMC Controller IP core can send them out on the HMC link, the HMC Controller IP core backpressures the application by deasserting the `dp_req_ready` signal.

In addition, the IP core supports the Limit Outstanding FLITs feature. The client can turn on this feature to ensure that the IP core limits the number of outstanding FLITs in expected read responses. The IP core delays sending a request to the HMC if sending it would increase the number of pending response FLITs from the HMC above a user-specified threshold.

Note: The IP core only supports the Limit Outstanding FLITs feature for full-width variants.

The Limit Outstanding FLITs feature limit the worst-case round-trip latency of packets in the system by avoiding the congestion that can occur when response packets have large payloads. A sequence of RD128 requests, for example, causes congestion in the response path. Designers building read-intensive, latency-sensitive applications can use the Limit Outstanding FLITs feature to improve latency and throughput.

Figure 3-5: Improving Latency and Throughput Flow Chart



Related Information

LIMIT_OUTSTANDING_PACKET Register on page 5-6

Error Detection and Management

The HMC specification defines error detection and recovery processes. The HMC Controller IP core complies with these requirements, and implements the following additional features to support error management:

- Error Response queues to support software handling without dropping Error Responses that arrive in quick succession
- Statistics registers that count the number of packets in various error categories

Table 3-2: HMC Response Packet Field Checking

The HMC Controller checks these HMC packet fields for error indications, and handles errors by entering Error Abort mode to force the HMC device to retransmit the packet. In this mode, the IP core completes transmission of any partially transmitted packet and then submits IRTRY packets, per the HMC specification. The IP core also sets the indicated bit in the `INTERRUPT_STATUS` register and increments the `Local Count` field of the `LOCAL_ERROR_COUNT` register.

Received Packet Field	Error Indication	INTERRUPT_STATUS Register Bit
`LNG` and `DLN`	The two fields have different values, or an invalid value	`LNG/DLN Error`
`CRC`	Incorrect CRC	`CRC Error`
`SEQ`	Unexpected value	`SEQ Error`

The HMC Controller IP core also checks the ERRSTAT field value and treats the response according to the following rules:

- If ERRSTAT has the value of zero, this field indicates no errors or conditions. The IP core processes the response packet as usual.
- If ERRSTAT has a non-zero value in a Read response, Write response, or MODE response packet, the IP core processes the response as usual, but asserts the dp_rsp_error signal on the RX data path interface when passing the response to the application.
- If ERRSTAT has a non-zero value in an Error response packet, the IP core does not forward the Error response packet to the RX data path interface. Instead, the IP core diverts the packet's ERRSTAT and cube ID values to the internal Error Response FIFO. The first element of the internal Error Response FIFO is always readable in the ERROR_RESPONSE register. You can process these packets in software.

The HMC Controller IP core transmits 32 IRTRY packets in every retry sequence.

Note: The IP core expects to receive *at least* 20 IRTRY packets from the HMC device.

Related Information

- **Interrupt Related Registers** on page 5-7
  Describes the INTERRUPT_STATUS interrupt bits CRC Error, SEQ Error, and LNG/DLN Error.
- **Error and Retry Statistics Registers** on page 5-10
  Describes the Local Count field of the LOCAL_ERROR_COUNT register.
- **ERROR_RESPONSE Register** on page 5-5
  Describes the ERROR_RESPONSE register and the Error Response queue.

Testing Features

The HMC Controller IP core supports multiple testing features.

- You can control the following testing features by writing fields in the HMC Controller IP core CONTROL register:
  - Force the HMC Controller IP core to detect an error in the input stream and send a StartRetry request to the HMC device.
  - Inject a single-bit error in the CRC of the next request packet the IP core transmits.
  - Force the Retry State Machine to exit the fatal error state.
  - Force the HMC device and the IP core to reset.
- You can use the testing features that the Native PHY IP core provides. You control these features by writing fields in the hard PCS registers. Write access to these registers is available through the transceiver reconfiguration interface. If you turn on **Enable ADME and Optional Reconfiguration Logic**, write access to these registers is also available through a JTAG master accessible from the Altera System Console.

Related Information

- **Transceiver Reconfiguration Signals** on page 4-15
- **CONTROL Register** on page 5-2

- **Arria 10 Transceiver PHY User Guide**
  Information about configuring the Arria 10 transceiver registers through the Arria 10 transceiver reconfiguration interface and about the ADME reconfiguration featue of the Arria 10 Native PHY IP core.
- **Arria 10 Transceiver Registers**
  Information about the Arria 10 transceiver registers.

HMC Controller IP Core Signals

4

2016.05.02

UG-01152

The HMC Controller IP core communicates with other design components through multiple interfaces. The IP core has the following top-level signals:

Application Interface Signals

The application interface supports easy access to the external HMC device by providing a simple data path interface to specify memory read and write requests and to receive memory read and write responses. This interface is also called the data path interface.

- Full-width IP core variations support 1, 2, 3, or 4 data path interfaces.
- Half-width IP core variations support 1 data path interface.

Related Information

Application Interfaces on page 3-2

Application Request Interface

The data path request interface, or application request interface, provides a 512-bit or 256-bit data bus and dedicated signals for the application to provide HMC request packet field values to the HMC

Controller IP core. Full-width variations have a 512-bit data bus, and half-width variations have a 256-bit data bus. The interface supports Write requests with payload sizes up to 128 bytes. In full-width variations, the maximum payload size limits the interface to data bursts of 2 or fewer `core_clk` clock cycles. In half-width variations, the maximum payload size limits the interface to data bursts of 4 or fewer `core_clk` clock cycles. Write requests and read responses with a payload size that is not a multiple of the bus size carry the end of the payload in the lower order bits of the data bus in the final clock cycle.

Full-width variations can have one, two, three, or four data path request interfaces. The interfaces are numbered and if you turn off the **Response re-ordering** parameter, each is restricted to a set range of tags. If you violate this restriction, the results are undefined.

The application must provide the following routing and control information for every request it sends on the TX data path interface:

- A well-formed HMC destination address.
- A unique 9-bit in-flight tag (if you turn off **Response re-ordering**). This requirement does not apply to posted transaction requests. In the case of multiple ports, each port is assigned a range of tags for its exclusive use. The application can reuse a tag only after the previous transaction with that tag is completely resolved.
- A correct 3-bit cube ID.

In addition, the application must ensure that it sends a request only when it is able to receive the response to the request as soon as that response arrives.

Figure 4-1: Application to HMC Controller IP Core With Single Port

The client acts as a source and the HMC Controller acts as a sink in the transmit direction.

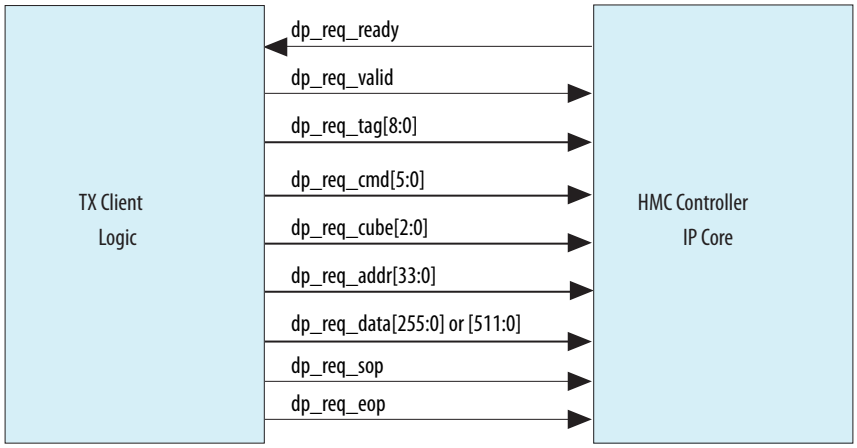


Figure 4-2: Application to HMC Controller IP Core With Multiple Ports

The client acts as a source and the HMC Controller acts as a sink in the transmit direction. Only full-width variations support multiple ports, so all multi-port variations have a 512-bit data interface. N is the number of ports you specify with the **Ports** parameter.

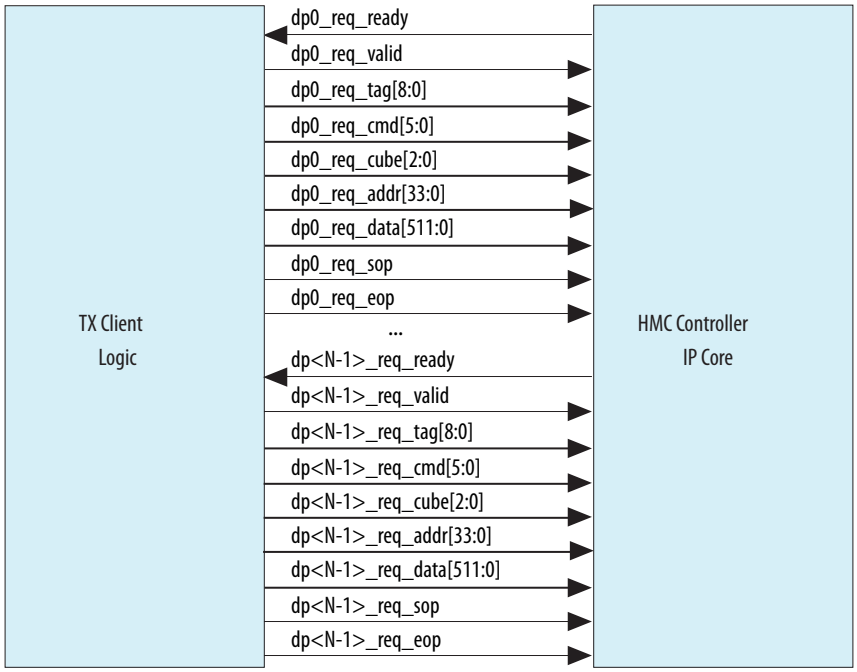


Table 4-1: Signals of Each Data Path Request Interface

All interface signals are synchronous with the `core_clk` clock. If the IP core has multiple ports, the port number is part of the signal name, as shown. If the IP core has a single port, the part number is not included in the signal name, for backward compatibility with previous releases of the IP core. If the IP core has two ports, it has two sets of signals, one with n=0 and one with n=1. If the IP core has four ports, it has four sets of signals, with n=0,1,2,and 3.

Signal Name	Direction	Description
`dp<n>_req_ready`	Output	When the HMC Controller IP core asserts this signal, the IP core is ready to receive data. The HMC Controller IP core accepts data when both `dp<n>_req_ready` and `dp<n>_req_valid` are asserted in the same cycle. The IP core deasserts this signal to back-pressure the application when it cannot currently process any incoming requests. The IP core does not deassert this signal between the cycles of a multi-cycle write data transfer.

Signal Name	Direction	Description
dp<n>_req_valid	Input	Indicates that the transaction is valid—all input signals have valid values. The HMC Controller IP core accepts data on the rising edge of core_clk when both dp<n>_req_ready and dp<n>_req_valid are asserted. The application must maintain this signal asserted during a multi-cycle write data transfer. The application can send back-to-back requests by maintaining the dp<n>_req_valid signal asserted. **Note:** The application must update the value it drives on dp<n>_req_tag for the new request. If the application continues to drive the old values on the input signals, that new request has the same tag as the previous request.
dp<n>_req_tag[8:0]	Input	The user-generated tag associated with this request. The corresponding response returns with the identical tag. This signal is not available if you turn on **Response re-ordering**. In that case the IP core manages tags internally. The value of this signal is a Don't Care for posted transaction requests. Because these requests have no corresponding response that must be identified, they do not require meaningful tag values. You must ensure every tag in use at any given time by a non-posted transaction is unique. After a response returns, the tag is available for re-use. If your IP core has multiple ports, each port is restricted to a specific range of tags: • Two ports: port 0 tag range is 0 to 255, and port 1 tag range is 256 to 511. • Three ports: port 0 tag range is 0 to 175, port 1 tag range is 176 to 351, port 2 tag range is 352 to 511, • Four ports: port 0 tag range is 0 to 127, port 1 tag range is 128 to 255, port 2 tag range is 256 to 383, and port 3 tag range is 384 to 511. Results are undefined if this restriction is violated. The application must maintain the value on this signal during a multi-cycle write data transfer.
dp<n>_req_cmd[5:0]	Input	Indicates the packet command associated with this request. Refer to Table 17 in the HMC Specification v1.1 for the command encodings. The application must maintain the value on this signal during a multi-cycle write data transfer.

Signal Name	Direction	Description
dp<n>_req_cube[2:0]	Input	The CUB ID of the cube to which the request is directed. The application must maintain the value on this signal during a multi-cycle write data transfer.
dp<n>_req_addr[33:0]	Input	Target address in the external HMC device. Current HMC devices ignore the four least significant bits of the address (and assumes they have the value of 4'b0000) in all requests the HMC Controller IP core generates, except for the BIT WRITE request. The application must maintain the value on this signal during a multi-cycle write data transfer.
dp<n>_req_data[511:0] (for full-width IP cores) dp_req_data[255:0] (for half-width IP cores)	Input	Write data. The application must transfer the least significant bytes of the write payload in the first cycle. If the size of the payload is not an integer multiple of the data bus width, then in the final data transfer cycle, the application must transfer the remaining write payload in the least significant bytes of dp<n>_req_data or dp_req_data. For example, the application must: • Transfer a 16-byte payload in dp_req_data[127:0]. • Transfer a 32-byte payload to a full-width or half-width IP core in dp_req_data[255:0]. • Transfer the final (most significant) 16 bytes of a 112-byte payload to a half-width IP core in dp_req_data[127:0] in the fourth data transfer clock cycle. During a read request, the value on this data bus does not matter.
dp<n>_req_sop	Input	Start of packet. The application must assert this signal in the first cycle of all transactions.
dp<n>_req_eop	Input	End of packet. The application must assert this signal in the final cycle of all transactions.

Application Response Interface

The data path response interface, or application response interface, provides a 512-bit or 256-bit data bus and dedicated signals for the IP core to provide HMC response information to the application. Full-width IP cores have one to four 512-bit data buses, and half-width IP cores have a 256-bit data bus. The interface supports Read responses with payload sizes up to 128 bytes. In full-width variations, the maximum payload size limits the interface to data bursts of 2 or fewer core_clk clock cycles. In half-width variations, the maximum payload size limits the interface to data bursts of 4 or fewer core_clk clock cycles. Read responses with a payload size that is not a multiple of the bus size carry the end of the payload in the lower order bits of the data bus in the final clock cycle.

In all half-width variations, and if you turn off **Response re-ordering** in a full-width variation, the HMC Controller returns the 9-bit tag from the original request with every response it sends on the data path

response interface. The application must use the tag to match each response with the corresponding request.

You cannot back-pressure the IP core data path response interface. To ensure the application can process every response it receives, the application must only send requests for which it has the resources to process or buffer the response.

Figure 4-3: HMC Controller IP Core With Single Port to RX Application

The HMC Controller IP core acts as a source and the client acts as a sink in the receive direction.

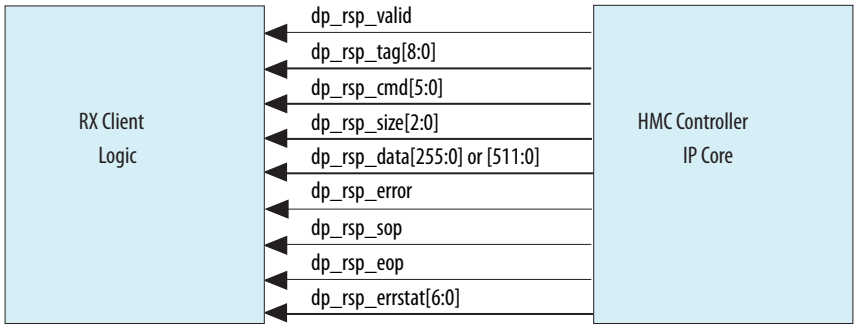


Figure 4-4: HMC Controller IP Core With Multiple Ports to RX Application

The HMC Controller IP core acts as a source and the client acts as a sink in the receive direction. Only full-width variations support multiple ports, so all multi-port variations have a 512-bit data interface. N is the number of ports you specify with the **Ports** parameter.

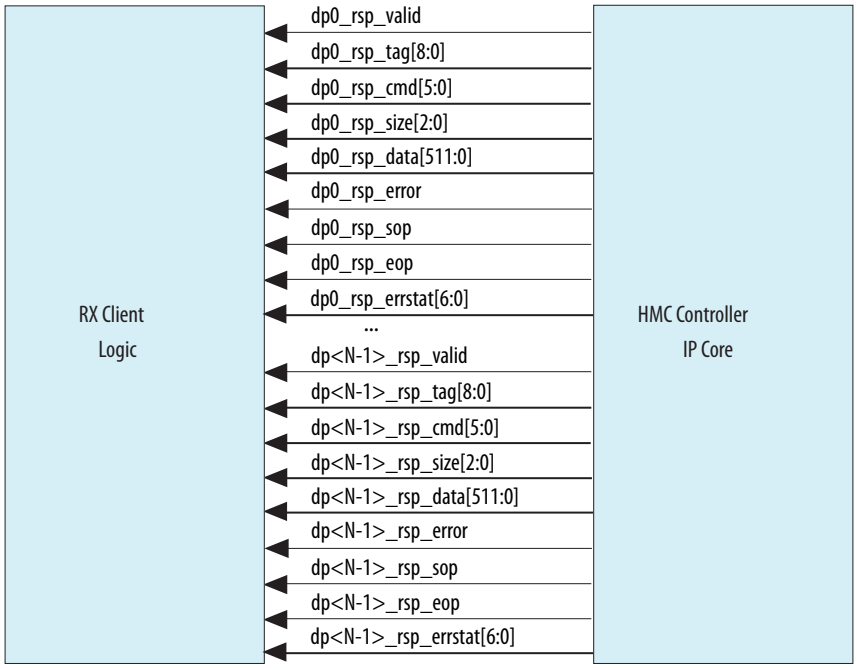


Table 4-2: Signals of Each Data Path Response Interface

All interface signals are clocked by the `core_clk` clock. If the IP core has multiple ports, the port number is part of the signal name, as shown. If the IP core has a single port, the part number is not included in the signal name, for backward compatibility with previous releases of the IP core. If the IP core has two ports, it has two sets of signals, one with n=0 and one with n=1. If the IP core has three ports, it has three sets of signals, with n=0, 1, and 2. If the IP core has four ports, it has four sets of signals, with n=0, 1, 2, and 3.

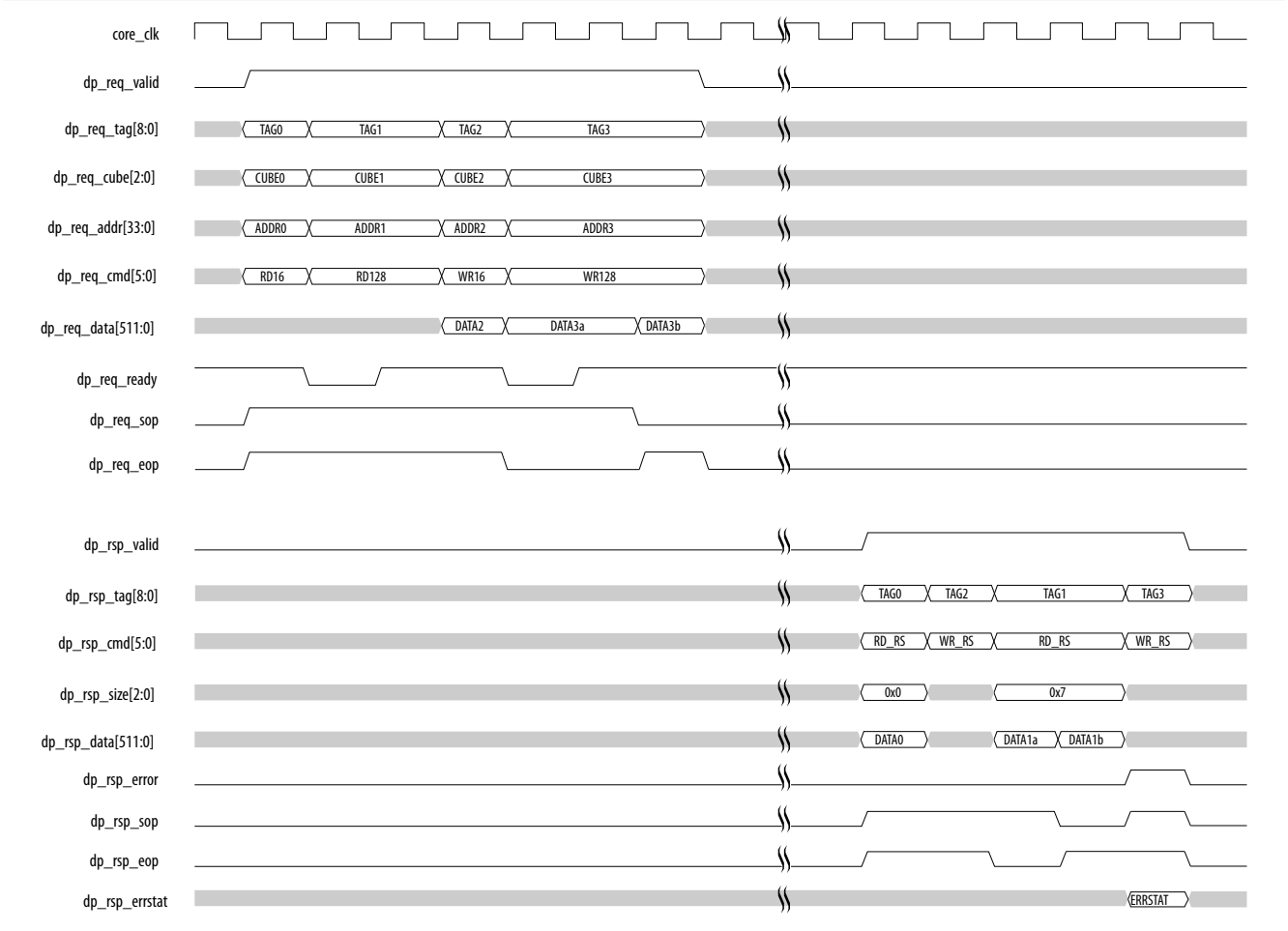
Signal Name	Direction	Description
`dp<n>_rsp_valid`	Output	Indicates that all of the `dp<n>_rsp_tag`, `dp<n>_rsp_cmd`, `dp<n>_rsp_error`, `dp<n>_rsp_sop`, `dp<n>_rsp_eop`, and `dp<n>_rsp_errstat` signals are valid, and in a read response with payload, `dp<n>_rsp_data` and `dp<n>_rsp_size` are valid. The application must accept all valid transactions. You cannot back-pressure the HMC Controller IP core data path response interface. The IP core maintains this signal asserted for the duration of a multi-cycle read data transfer.
`dp<n>_rsp_tag[8:0]`	Output	The tag associated with the original request to which this is a response. After you process this response, the tag is available for re-use. The IP core maintains the value of this signal for the duration of a multi-cycle read data transfer. This signal is not available if you turn on **Response re-ordering**. In that case the IP core manages tags internally.
`dp<n>_rsp_cmd[5:0]`	Output	Indicates the packet command associated with this response. Refer to Table 25 in the HMC Specification v1.1 for the command encodings. This signal holds only non-error response codes; the IP core routes error responses to the registers. The IP core maintains the value of this signal for the duration of a multi-cycle read data transfer.

Signal Name	Direction	Description
dp<n>_rsp_size[2:0]	Output	Indicates the size of the payload associated with this response. If the current response is a Read response, indicates the size of the payload in dp<n>_rsp_data. During a response with an associated payload, the IP core sets this signal to one of the following valid values: • 3'b000 indicates a 16-byte payload or a Write response. • 3'b001 indicates a 32-byte payload. • 3'b010 indicates a 48-byte payload (half-width IP cores only). • 3'b011 indicates a 64-byte payload. • 3'b100 indicates a 80-byte payload (half-width IP cores only). • 3'b101 indicates a 96-byte payload (half-width IP cores only). • 3'b110 indicates a 112-byte payload (half-width IP cores only). • 3'b111 indicates a 128-byte payload. During a response with no associated payload, the value of this signal is undefined. Responses with no associated payload are the responses for which dp<n>_rsp_cmd[0] has the value of 1. The IP core maintains the value of this signal for the duration of a multi-cycle read data transfer.
dp<n>_rsp_data[511:0] (for full-width IP cores) dp_rsp_data[255:0] (for half-width IP cores)	Output	Read response data. During a response with no associated payload, the value of this signal is undefined. Responses with no associated payload are the responses for which dp<n>_rsp_cmd[0] has the value of 1. If the size of the payload is not an integer multiple of the data bus width, then in the final data transfer cycle, the IP core transfers the remaining read payload in the least significant bytes of dp<n>_rsp_data or dp_rsp_data. For example, the IP core: • Transfers a 16-byte payload in dp_rsp_data[127:0]. • Transfers a 32-byte payload (from a full-width or half-width IP core) in dp_rsp_data[255:0]. • Transfers the final (most significant) 16 bytes of a 112-byte payload from a half-width IP core in dp_rsp_data[127:0] in the fourth data transfer clock cycle.

Signal Name	Direction	Description
dp<n>_rsp_error	Output	Indicates that the corresponding request completed with an error and will not be retried automatically. The HMC Controller IP core asserts this signal if it received a Read or Write response packet from the external HMC device with a non-zero ERRSTAT or DINV field. The IP core maintains the value of this signal for the duration of a multi-cycle read data transfer.
dp<n>_rsp_sop	Output	Start of packet. The IP core asserts this signal in the first cycle of all response transactions.
dp<n>_rsp_eop	Output	End of packet. The IP core asserts this signal in the final cycle of all response transactions.
dp<n>_rsp_errstat[6:0]	Output	Error status. The IP core passes this value directly from the external HMC device ERRSTAT field.

HMC Controller IP Core Data Path Example

Figure 4-5: Full-Width HMC Controller IP Core Application Interface Example



In this example, user logic sends four consecutive request packets to a full-width HMC Controller IP core with a single data interface port. The IP core sends the corresponding response packets. The first three requests complete without error. The fourth request completes with an error indication.

When the HMC Controller IP core deasserts the `dp_req_ready` signal, user logic maintains the current values until a full clock cycle after the IP core reasserts `dp_req_ready`. User logic is required to send the values while `dp_req_ready` is asserted, to ensure that the IP core captures them correctly.

The IP core asserts the `dp_rsp_error` signal while sending the response to the WR128 request. This signal indicates that the WR128 request did not complete successfully. The IP core passes this information through from the HMC device; therefore, this signal indicates that the HMC device encountered an error while attempting to implement the request.

HMC Interface Signals

The HMC Controller IP core's HMC interface connects to the external HMC device's link interface and main reset signal.

Table 4-3: Signals of the HMC Interface

Signal Name	Direction	Description
`hmc_lxrx[15:0]` (for full-width IP cores) `hmc_lxrx[7:0]` (for half-width IP cores)	Input	Receiving lanes. Implements HMC specification `LxRXP` and `LxRXN` differential pairs. You must connect this data bus to the HMC device `LxTXP` bus. The Quartus Prime Fitter assigns the correct pin to the negative signal automatically.
`hmc_lxtx[15:0]` (for full-width IP cores) `hmc_lxtx[7:0]` (for half-width IP cores)	Output	Transmitting lanes. Implements HMC specification `LxTXP` and `LxTXN` differential pairs. You must connect this data bus to the HMC device `LxRXP` bus. The Quartus Prime Fitter assigns the correct pin to the negative signal automatically.
`hmc_lxrxps`	Input	Link power reduction input. Implements HMC specification `LxRXPS` signal. The HMC Controller IP core does not use this signal, but provides it to comply with the HMC specification. You should connect this input signal to the HMC device `LxTXPS` output signal.
`hmc_lxtxps`	Output	Link power reduction output. Implements HMC specification `LxTXPS` signal. You must connect this output signal to the HMC device `LxRXPS` input signal.
`hmc_ferr_n`	Input	Active-low fatal error indication from the HMC device. You must connect this signal to the HMC device `FERR_N` signal.

Signal Name	Direction	Description
hmc_p_rst_n	Output	Main reset signal to the HMC device. You must connect this signal to the HMC device P_RST_N signal.

Related Information

HMC Specification 1.1
The HMC specification is available for download from the Hybrid Memory Cube Consortium web page.

Signals on the Interface to the I²C Master

Your design must include an I²C master module that drives the HMC device I²C interface for link initialization. This interface connects to the I²C module.

The I²C module and the IP core together must implement the following four-way handshake with the two interface signals:

1. Resetting the IP core deasserts the i2c_load_registers signal. Resetting the I²C master module should deassert the i2c_registers_loaded signal.
2. When the IP core and the HMC are ready, the IP core asserts i2c_load_registers. In simulation, the IP core assumes the HMC simulation model is ready instantaneously, and in hardware, the IP core waits the required t_{INIT} duration of 20 ms.
3. After the I²C master module detects the assertion of i2c_load_registers, it writes to the HMC device registers to set them up for link initialization (concluding with Init Continue) and then asserts the i2c_registers_loaded signal.
4. The HMC Controller IP core deasserts i2c_load_registers.
5. The I²C master module deasserts i2c_registers_loaded.

Table 4-4: Signals on the Interface to the External I²C Master Module

The IP core i2c_load_registers signal behavior conforms to the four-way handshaking protocol. For correct HMC Controller IP core functionality, you must design the I²C master module in your design to implement i2c_registers_loaded signal behavior that conforms to this four-way handshaking protocol.

Signal Name	Direction	Description
i2c_load_registers	Output	Indicates the HMC Controller IP core is ready for the external I²C master module to load the HMC device configuration registers, as part of the link initialization sequence. You must connect this signal to the I²C master module input port that accepts requests to load the configuration registers of the HMC device.
i2c_registers_loaded	Input	Indicates the external HMC device registers are configured. You must connect this signal to the output port of the I²C master that indicates successful completion of the configuration register load sequence.

If multiple HMC Controller IP cores are connected to different links of the same HMC device, the external I^2C master must wait until all of the HMC Controller IP cores have asserted their `i2c_load_registers` signal, before writing to the HMC device configuration registers. After the external I^2C master completes writing all of the HMC configuration registers, it must assert the `i2c_registers_loaded` signals for all of the HMC Controller IP cores simultaneously.

Related Information

HMC Controller IP Core Design Example on page 6-1
The HMC Controller design example includes an I^2C master module that correctly implements the four-way handshaking protocol with the HMC Controller IP core, and that implements the recommended sequence of register writes to initialize the Micron HMC 15G SR HMC device.

Control and Status Interface Signals

The control and status register interface is an Avalon-MM interface that provides access to the HMC Controller IP core internal control and status registers. This interface does not provide access to the transceiver configuration registers.

The Avalon-MM interface implements a standard memory-mapped protocol. You can connect any Avalon master—for example, an embedded processor or JTAG Avalon master—to this bus to access the IP core control and status registers.

Table 4-5: Control and Status Interface Signals

The `core_clk` clocks the signals on the HMC Controller IP core control and status interface. This interface supports only read operations and write operations with a 32-bit payload (one full register value).

Signal Name	Direction	Description
`csr_address[5:0]`	Input	Byte address for register reads and writes. All HMC Controller control and status registers are 32 bits wide. Therefore, all addresses are 4-byte aligned.
`csr_read`	Input	You must assert this signal to request a read transfer
`csr_write`	Input	You must assert this signal to request a write transfer
`csr_writedata[31:0]`	Input	Write data
`csr_readdata[31:0]`	Output	Read data
`csr_readdatavalid`	Output	Read data is ready for use

Signal Name	Direction	Description
csr_irq	Output	Interrupt request. The value of this signal is not associated with the current values of other signals on this interface. The IP core asserts this interrupt signal asynchronously as soon as an INTERRUPT_STATUS register bit is asserted (if the relevant INTERRUPT_ENABLE bit is set and the GLOBAL_INTERRUPT_ENABLE register's GlobalEnable bit has the value of 1) and maintains the signal asserted until either one of the two relevant enable bits is reset, or the application writes the value of 1 to all currently-asserted INTERRUPT_STATUS register bits.

Related Information

- **Control and Status Register Interface** on page 3-3
- **HMC Controller IP Core Register Map** on page 5-1
- **Interrupt Related Registers** on page 5-7
  Describes the INTERRUPT_STATUS, INTERRUPT_ENABLE, and INTERRUPT_GLOBAL_ENABLE registers.
- **Avalon Interface Specifications**
  For more information about the Avalon-MM protocol, including timing diagrams, refer to the *Avalon Memory-Mapped Interfaces* chapter.

Status and Debug Signals

Table 4-6: Status and Debug Signals

The HMC Controller IP core status and debug interface provides a few extra signals to communicate successful link initalization and to support debugging of your HMC system.

Clock Name	Direction	Description
link_init_complete	Output	The IP core asserts this signal when the link initialization state machine is in the active state.
debug_tx_data[511:0] (for full-width IP cores) debug_tx_data[255:0] (for half-width IP cores)	Output	This data bus shows an unscrambled copy of the striped data before it enters the TX lane swapper. The data on this bus is striped but not scrambled.
debug_rx_data[511:0] (for full-width IP cores) debug_rx_data[255:0] (for half-width IP cores)	Output	This data bus shows the striped and descrambled received data after processing by the RX lane swapper and the descrambler.

Related Information

Status and Debug Interface on page 3-3

Clock and Reset Signals

Table 4-7: HMC Controller IP Core Clock and Reset Signals

The HMC Controller IP core has a single clock domain outside of the transceiver. Your design must derive the external TX PLL reference clock, the RX CDR reference clock, and the HMC device `REFCLKP` and `REFCLKN` input reference clock signals from the same clock reference source.

Clock Name	Direction	Description
`rst_n`	Input	Active low master reset signal for the HMC Controller IP core. When asserted, the signal must remain asserted for at least two `rx_cdr_refclk0` clock cycles.
`core_rst_n`	Output	When asserted, indicates that the HMC Controller IP core is in reset. The IP core deasserts the `core_rst_n` signal only after `core_clk` is stable and the transceiver is ready to transmit data.
`rx_cdr_refclk0`	Input	Reference clock for the RX transceiver CDR PLL. You must drive this clock with the frequency you specify for the **CDR reference clock** parameter. `rx_cdr_refclk0` is not the reference clock for the TX PLL. The reference clock for the TX PLL is an input to the external TX PLL IP cores that you connect to your HMC Controller IP core. The reference clock for the TX PLLs does not drive the HMC Controller IP core directly.
`tx_bonding_clocks[95:0]` (for full-width IP cores) `tx_bonding_clocks[47:0]` (for half-width IP cores)	Input	Clocks for the individual transceiver channels. The input clock to each transceiver channel has six bits. You must connect this input bus to the external transceiver TX PLL IP core. You must parameterize the external TX PLL IP core to specify an output frequency that is 1/2 the per-lane data rate. For a 10 Gbps HMC Controller IP core lane rate, the TX PLL IP core output frequency must be 5 GHz; for a 12.5 Gbps lane rate, the TX PLL IP core output frequency must be 6.25 GHz.

Clock Name	Direction	Description
core_clk	Output	Master clock for the HMC Controller IP core. The transceiver generates core_clk. The frequency of core_clk is the lane rate divided by 32. Lane Rate / core_clk Frequency 10 Gbps / 312.5 MHz 12.5 Gbps / 390.625 MHz core_clk clocks the HMC Controller IP core signals, including the signals on the control and status interface.
reconfig_clk	Input	Clock for the transceiver reconfiguration interface.
reconfig_reset	Input	Reset signal for the transceiver reconfiguration interface.

Related Information

Arria 10 Transceiver PHY User Guide
Information about the Arria 10 dynamic reconfiguration interface clock and reset signals. The HMC Controller IP core reconfig_clk and reconfig_reset input signals pass directly to the Arria 10 Native PHY IP core included in the HMC Controller IP core.

Transceiver Reconfiguration Signals

Altera provides a dedicated Avalon-MM interface, called the transceiver reconfiguration interface, to access the transceiver registers. You access the transceiver registers through this dedicated interface and not through the IP core general purpose control and status interface.

The Avalon-MM interface implements a standard memory-mapped protocol. You can connect an Avalon master to this bus to access the registers of the embedded Arria 10 Native PHY IP core.

Table 4-8: HMC Controller IP Core Arria 10 Transceiver Reconfiguration Interface Signals

The reconfig_clk clocks the signals on the HMC Controller IP core Arria 10 transceiver reconfiguration interface. The reconfig_reset input signal resets the interface.

Signal Name	Direction	Description
reconfig_address[13:0] (full-width IP core) reconfig_address[12:0] (half-width IP core)	Input	Word address for reads and writes. This address has 14 bits in full-width IP core variations and 13 bits in half-width IP core variations.
reconfig_read	Input	You must assert this signal to request a read transfer.
reconfig_write	Input	You must assert this signal to request a write transfer.

Signal Name	Direction	Description
reconfig_writedata[31:0]	Input	Write data
reconfig_readdata[31:0]	Output	Read data The data on reconfig_readdata[31:0] is valid on the rising edge of reconfig_clk following a clock cycle in which reconfig_read is asserted and reconfig_waitrequest is deasserted.
reconfig_waitrequest	Output	Indicates the IP core is not ready. You must maintain the values on the input signals while reconfig_waitrequest is asserted. The data on reconfig_readdata[31:0] is not valid while reconfig_waitrequest is asserted.

Related Information

- **Transceiver Reconfiguration Interface** on page 3-4
- **Testing Features** on page 3-11
- **Avalon Interface Specifications**
  For more information about the Avalon-MM protocol, including timing diagrams, refer to the *Avalon Memory-Mapped Interfaces* chapter.
- **Arria 10 Transceiver PHY User Guide**
  Information about the Arria 10 dynamic reconfiguration interface signals and the Arria 10 Native PHY IP core hard PCS registers that you can program through the Arria 10 transceiver reconfiguration interface. The HMC Controller IP core transceiver reconfiguration signals pass directly to the Arria 10 Native PHY IP core included in the HMC Controller IP core.
- **Arria 10 Transceiver Registers**
  Detailed information about the Arria 10 transceiver registers.

Signals on the Interface to the External PLL

Table 4-9: HMC Controller IP Core External PLL Interface Signals

The HMC Controller IP core requires that you generate and connect a TX PLL IP core to each HMC Controller IP core lane that connects to the HMC device. The HMC Controller IP core external PLL interface connects to these PLL IP core instances.

Signal Name	Direction	Description
`tx_bonding_clocks[95:0]` (for full-width IP cores) `tx_bonding_clocks[47:0]` (for half-width IP cores)	Input	Clocks for the individual transceiver channels. The input clock to each transceiver channel has six bits. You must connect this input bus to a transceiver TX PLL IP core. You must parameterize an external TX PLL IP core to specify an output frequency that is 1/2 the per-lane data rate. For a 10 Gbps HMC Controller IP core lane rate, the TX PLL IP core output frequency must be 5 GHz; for a 12.5 Gbps lane rate, the TX PLL IP core output frequency must be 6.25 GHz.
`pll_locked`	Input	PLL-locked indication from the external TX PLL. User logic must drive this input signal with the `pll_locked` indications from the external TX PLL. `core_clk` can stabilize only after `pll_locked` is asserted. The IP core deasserts the `core_rst_n` signal to indicate that `core_clk` has stabilized.
`pll_cal_busy`	Input	PLL-busy indication from the external TX PLL.

Related Information

- **Adding the External PLL** on page 2-13
  Describes how to generate and connect the external transceiver PLL IP core to your HMC Controller IP core.
- **Arria 10 Transceiver PHY User Guide**
  Information about the Arria 10 transceiver PLLs and clock network.

HMC Controller IP Core Register Map 5

2016.05.02

UG-01152

The HMC Controller IP core internal registers are 32 bits wide and are accessible to you using the control and status interface, an Avalon-MM interface which conforms to the *Avalon Interface Specifications*.

All of these registers are 32 bits wide and the addresses are shown as hexadecimal values. The registers can be accessed only on a 32-bit (4-byte) basis. The addressing for the registers therefore increments by units of 4.

Write accesses to a Reserved or undefined location have no effect. Read accesses to a Reserved or undefined location return an undefined result.

Table 5-1: Register Access Codes

Lists the access codes used to describe the type of register bits.

Code	Description
RW	Read / write
RO	Read only
RW1C	Read / write 1 to clear
RTC	Read to clear
WO	Write only

Table 5-2: Control and Status Register Map

Offset	Register Name	Location of Additional Information
0x00	Reserved	
0x04	`CONTROL`	CONTROL Register
0x08	`XCVR_STATUS`	XCVR_STATUS Register
0x0C	`LANE_STATUS`	LANE_STATUS Register
0x10	`LINK_STATUS`	LINK_STATUS Register
0x14	`ERROR_RESPONSE_CAPTURE`	ERROR_RESPONSE Register
0x18	Reserved	

Offset	Register Name	Location of Additional Information
0x1C	LIMIT_OUTSTANDING_PACKET	LIMIT_OUTSTANDING_PACKET Register
0x20	INTERRUPT_STATUS	Interrupt Related Registers
0x24	INTERRUPT_ENABLE	
0x28	GLOBAL_INTERRUPT_ENABLE	
0x2C	Reserved	
0x30	LOCAL_ERROR_COUNT	Error and Retry Statistics Registers
0x34	REMOTE_ERROR_COUNT	
0x38	RETRY_BUFFER_ECC_COUNT	
0x3C	RESPONSE_BUFFER_ECC_COUNT	

CONTROL Register

Table 5-3: HMC Controller IP Core CONTROL Register at Offset 0x04

Bits	Field Name	Type	Value on Reset	Description
31:18	Reserved	RO	0	
17	P_RST_N	RW	0x1	Software-controlled reset of the HMC. The IP core drives the value of this field on the hmc_p_rst_n output port, which should be connected to the HMC P_RST_N reset signal. For backward compatibility with the IP core v15.0, the IP core forces this output signal low while the rst_n input signal is asserted, and raises it when rst_n is deasserted. After the IP core sets it to the value of 1, the output signal is driven from the P_RST_N register field.
16	TXPS	RW	0x1	Power management field. The IP core drives the value of this field on the hmc_lxtxps output port, which should be connected to the HMC LxRXPS input port. For backward compatibility with the IP core v15.0, the IP core forces this output signal high while the rst_n input signal is asserted, and when rst_n is deasserted. Afterwards, the output signal is driven from the TXPS register field.
15:10	Reserved	RO	0x00	
9	ForceRXError	WO	0x0	Writing the value of 1 to this register field forces the HMC Controller IP core to detect an error in the input stream and send a StartRetry request to the HMC device. This bit is self-clearing.

Bits	Field Name	Type	Value on Reset	Description
8	CRCErrorInject	WO	0x0	Writing the value of 1 to this register field injects a single bit error in the CRC of the next request packet. This bit is self-clearing.
7:3	Reserved	RO	0x00	
2	SoftReset	WO	0x0	Writing the value of 1 to this register field resets all parts of the HMC Controller IP core except the registers and the transmit side of the transceiver. This bit is self-clearing. Reading this bit always returns the value of 0. **Note:** Available in full-width variations only.
1	ClearFatalError	WO	0x0	If the Retry State Machine (RSM) is in fatal error state (RetryFatalError), writing the value of 1 to this register field causes the RSM to resume normal operation. When a RetryFatalError occurs, the RSM halts and waits for external corrective action. Writing the value of 1 to this register field forces the RSM to continue. This bit is self-clearing. It clears whether or not it affects the RSM state. Reading this bit always returns the value of 0.
0	Retrain	WO	0x0	Writing the value of 1 to this register field causes the HMC Controller IP core to restart the link initialization sequence. This bit is self-clearing. Reading this bit always returns the value of 0. **Note:** Available in full-width variations only.

Related Information

- **Testing Features** on page 3-11
- **Initialization and Reset** on page 3-5

XCVR_STATUS Register

Table 5-4: HMC Controller IP Core XCVR_STATUS Register at Offset 0x08

Individual transceiver status in HMC link, ordered by transceiver channel.

Bits	Field Name	Type	Value on Reset	Description
31:16	Reserved	RO	0x0001	

Bits	Field Name	Type	Value on Reset	Description
15:0	CDR Lock	RO	0x0000	Each bit indicates whether the CDR for the corresponding transceiver channel has locked to the received data. **Note:** For half-width variations, bits 15:8 are reserved and set to 0.

LANE_STATUS Register

Table 5-5: HMC Controller IP Core LANE_STATUS Register at Offset 0x0C

Individual lane status in HMC link, ordered by transceiver channel.

Bits	Field Name	Type	Value on Reset	Description
31:16	WordLock	RO	0x00	Each bit indicates whether the corresponding transceiver channel in the HMC link has locked to the TS1 word boundary. **Note:** For half-width variations, bits 31:24 are reserved and set to 0.
15:0	DescramSync	RO	0x00	Each bit indicates whether the descrambler for the corresponding transceiver channel has synchronized to the received data. **Note:** For half-width variations, bits 15;8 are reserved and set to 0.

LINK_STATUS Register

Table 5-6: HMC Controller IP Core LINK_STATUS Register at Offset 0x10

Bits	Field Name	Type	Value on Reset	Description
31:17	Reserved	RO	0x0000	
16	RXPS	RO	0x0	Level of the LxRXPS input from the HMC device.
15:9	Reserved	RO	0x00	
8	LanesAligned	RO	0x0	Indicates whether the received data is aligned across all lanes.
7:6	Reserved	RO	0x0	

Bits	Field Name	Type	Value on Reset	Description
5:0	InitializationState	RO	0x01	Indicates the current state in link initialization. This register field has the following valid values: • 6'b100000: Active • 6'b010000: Transaction Initialization (Wait for TRET) • 6'b001000: Word Synchronization (Transmit TS1) • 6'b000100: Scrambler Synchronization (Transmit NULLs) • 6'b000010: HMC Configuration (by the external I^2C master module) • 6'b000001: Reset

Related Information

Initialization and Reset on page 3-5
Describes the behavior of the InitializationState field during HMC Controller IP core initialization.

ERROR_RESPONSE Register

Table 5-7: HMC Controller IP Core ERROR_RESPONSE Register at Offset 0x14

The HMC Controller IP core stores the ERRSTAT and CUB fields of the Error responses that it receives on the HMC link. The IP core stores these fields in an internal Error Response queue (FIFO buffer). The application can read the relevant information for each Error Response packet from this queue by reading the ERROR_RESPONSE register. Reading from the register advances the queue.

Read, Write, or MODE response packets the HMC Controller IP core receives with a non-zero ERRSTAT field do not route to this queue or register. Instead they are sent to the data path response interface with dp_rsp_error asserted.

Bits	Field Name	Type	Value on Reset	Description
31:17	Reserved	RO	0x0000	
16	Valid	RO	0x0	Indicates the CUB and ERRSTAT fields in the register hold valid values. When the Error Response queue is empty, the CUB and ERRSTAT fields are not valid, and the Valid bit has the value of 0. You can poll the Valid bit to determine if any Error Response packets are waiting to be processed, or you can enable the RX Error Response interrupt in the INTERRUPT_ENABLE register.
15:11	Reserved	RO	0x00	

Bits	Field Name	Type	Value on Reset	Description
10:8	CUB	RO	0x0	The CUB ID extracted from the TAG field of the Error Response packet.
7	Reserved	RO	0x0	
6:0	ERRSTAT	RO	0x00	The ERRSTAT value extracted from the Error Response packet.

Related Information

HMC Specification 1.1

Information about the encoding of the ERRSTAT response packet field is available in Table 16 in the HMC specification.

LIMIT_OUTSTANDING_PACKET Register

Table 5-8: HMC Controller IP Core LIMIT_OUTSTANDING_PACKET Register at Offset 0x1C

Note: Available in full-width variations only.

Bits	Field Name	Type	Value on Reset	Description
31	Enable	RW	0x0	Writing the value of 1 to this register field turns on the Limit Outstanding FLITs feature. When this feature is turned on, the IP core limits the number of FLITs in outstanding response packets, to the threshold specified by the MaxRspPktFlit field. The effect of turning on this feature is that the IP core does not send requests to the HMC that would exceed the threshold in expected responses. Whether the feature is on or off, the IP core sends request packets to the HMC only if retry buffer space and tokens are available. The client is responsible to send requests on the data path request interface only if the client is able to receive the expected response at any time. The IP core can backpressure a data path request interface by deasserting the dp<n>_req_ready signal.
30:10	Reserved	RO	0x000000	
9:2	MaxRspPktFlit	RW	0x00	Specifies the maximum number of FLITs if the Limit Outstanding FLITs feature is turned on. The threshold value is four times the value of this register field. The typical value for this register is 150. However, given your design's traffic pattern and system configuration, you may want to adjust this value for better throughput or latency. A lower value reduces the latency but can degrade the RX throughput. Use the lowest value that results in acceptable throughput.

Bits	Field Name	Type	Value on Reset	Description
1:0	Reserved	RO	0x0	

Related Information

Flow Control on page 3-9

Interrupt Related Registers

THE HMC Controller IP core has three interrupt-related registers.

- `INTERRUPT_STATUS`: Register bits report individual interrupt source status.
- `INTERRUPT_ENABLE`: Register bits individually enable the corresponding interrupts in the `INTERRUPT_STATUS` register to trigger assertion of the IP core `csr_irq` output signal, unless the `GLOBAL_INTERRUPT_ENABLE` register turns off this ability.
- `GLOBAL_INTERRUPT_ENABLE`: Register allows you to disable all interrupt responses or to enable those interrupt sources indicated in the `INTERRUPT_ENABLE` register.

Table 5-9: HMC Controller IP Core INTERRUPT_STATUS Register at Offset 0x20

To clear an interrupt, write the value of 1 to the interrupt bit.

Bits	Field Name	Type	Value on Reset	Description
31:16	Reserved	RO	0x0000	
15	`Response Queue Uncorrectable ECC Error`	W1C	0x0	The IP core sets this bit if it detects an uncorrectable ECC error in the Response Queue memory. The IP core can detect such an error only if you turn on **Enable M20K ECC support** in the parameter editor.
14	`Response Queue ECC Error`	W1C	0x0	The IP core sets this bit if it detects a correctable ECC error in the Response Queue memory. If the IP core sets this bit it also corrects the ECC error. The IP core can detect such an error only if you turn on **Enable M20K ECC support** in the parameter editor.
13	`FERR_N`	W1C	0x0	The IP core sets this bit if the HMC device indicates a fatal error by asserting its active-low FERR_N pin. You must connect the IP core `hmc_ferr_n` input signal to the HMC device `FERR_N` output signal.
12	`Retry Buffer Uncorrectable ECC Error`	W1C	0x0	The IP core sets this interrupt bit if it detects an uncorrectable ECC error in the Retry Buffer memory. The IP core can detect such an error only if you turn on **Enable M20K ECC support** in the parameter editor.
11	`Retry Buffer ECC Error`	W1C	0x0	The IP core sets this interrupt bit if it detects a correctable ECC error in the Retry Buffer memory. The IP core automatically corrects the ECC in this case. The IP core can detect such an error only if you turn on **Enable M20K ECC support** in the parameter editor.

Bits	Field Name	Type	Value on Reset	Description
10	Reserved	RO	0x0	
9	No More Tokens	W1C	0x0	The IP core sets this interrupt bit if it runs out of tokens. Tokens represent available buffer space in the HMC device. While the IP core has no remaining tokens, it does not send any additional requests, per token-based flow control requirements. This situation is not an error condition, but it may indicate a reduction in performance. However, like any interrupt bit, it causes the IP core to assert the csr_irq signal (assuming the global interrupt enable register bit is set). This bit has the value of 0 when the IP core comes out of reset. After link initialization, the HMC device communicates its buffer capacity with a sequence of TRET packets. After the IP core receives the first TRET packet, it begins updating the No More Tokens register field.
8	Retry Buffer Full	W1C	0x0	The IP core sets this interrupt bit if the Retry buffer fills. When the Retry buffer is full, the IP core does not send any additional Read or Write requests. This situation is not an error condition, but it may indicate a reduction in performance.
7	Reserved	RO	0x0	
6	RX Error Response Overflow	W1C	0x0	The IP core sets this interrupt bit if too many Error Response packets are received before they are read from the ERROR_RESPONSE register. If overflow occurs, the IP core drops incoming Error Response packets until space is again available in the Error Response queue.
5	RX Error Response	W1C	0x0	The IP core sets this interrupt bit if the IP core receives an Error Response packet.
4	Fatal Error	W1C	0x0	The IP core sets this interrupt bit if it makes three or more successive retry attempts that are unsuccessful.
3	Remote Error	W1C	0x0	The IP core sets this interrupt bit if it receives a valid IRTRY (StartRetry) sequence, indicating the HMC device detected an error.
2	SEQ Error	W1C	0x0	The IP core sets this interrupt bit if it receives a packet with a SEQ field value that is not a +1 increment from the SEQ field value of the previous packet it received.
1	LNG/DLN Error	W1C	0x0	The IP core sets this interrupt bit if it receives a packet with unequal or invalid values in the LNG (packet length) and DLN (duplicate length) fields.
0	CRC Error	W1C	0x0	The IP core sets this interrupt bit to the value of 1 if it detects an error in the CRC of a packet it receives.

Table 5-10: HMC Controller IP Core INTERRUPT_ENABLE Register at Offset 0x24

Each bit in this register enables the corresponding interrupt in the `INTERRUPT_STATUS` register at offset 0x20. For each register bit:

- If the bit has the value of 0, the interrupt is disabled.
- If the bit has the value of 1, and the `GlobalEnable` bit of the `GLOBAL_INTERRUPT_ENABLE` register at offset 0x28 has the value of 1, the interrupt is enabled.

Bits	Field Name	Type	Value on Reset	Description
31:16	Reserved	RO	0x0000	
15	`Response Queue Uncorrectable ECC Error Enable`	RW	0x0	Enables `Response Queue Uncorrectable ECC Error` interrupt.
14	`Response Queue ECC Error Enable`	RW	0x0	Enables `Response Queue ECC Error` interrupt.
13	`FERR_N Enable`	RW	0x0	Enables `FERR_N` interrupt.
12	`Retry Buffer Uncorrectable ECC Error Enable`	RW	0x0	Enables `Retry Buffer Uncorrectable ECC Error` interrupt.
11	`Retry Buffer ECC Error Enable`	RW	0x0	Enables `Retry Buffer ECC Error` interrupt.
10	Reserved	RO	0x0	
9	`No More Tokens Enable`	RW	0x0	Enables `No More Tokens` interrupt.
8	`Retry Buffer Full Enable`	RW	0x0	Enables `Retry Buffer Full` interrupt.
7	Reserved	RO	0x0	
6	`RX Error Response Overflow Enable`	RW	0x0	Enables `RX Error Response Overflow` interrupt.
5	`RX Error Response Enable`	RW	0x0	Enables `RX Error Response` interrupt.
4	`Fatal Error Enable`	RW	0x0	Enables `Fatal Error` interrupt.
3	`Remote Error Enable`	RW	0x0	Enables `Remote Error` interrupt.

Bits	Field Name	Type	Value on Reset	Description
2	SEQ Error Enable	RW	0x0	Enables SEQ Error interrupt.
1	LNG/DLN Error Enable	RW	0x0	Enables LNG/DLN Error interrupt.
0	CRC Error Enable	RW	0x0	Enables CRC Error interrupt.

Table 5-11: HMC Controller IP Core GLOBAL_INTERRUPT_ENABLE Register at Offset 0x28

Gates the INTERRUPT_ENABLE register.

Bits	Field Name	Type	Value on Reset	Description
31:1	Reserved	RO	0x00000000	
0	GlobalEnable	RW	0x0	Writing the value of 0 to this register field disables all interrupt sources from asserting the csr_irq output signal. Writing the value of 1 to this register field allows the IP core to assert the csr_irq output signal according to the interrupt sources enabled in the INTERRUPT_ENABLE register at offset 0x24. An interrupt source causes the IP core to assert the csr_irq output signal only if the GlobalEnable register field and the relevant INTERRUPT_ENABLE register field both have the value of 1.

Error and Retry Statistics Registers

The HMC Controller IP core has four statistics registers. Counter fields in these registers are all of type RC (Read to Clear).

Table 5-12: HMC Controller IP Core LOCAL_ERROR_COUNT Register at Offset 0x30

Bits	Field Name	Type	Value on Reset	Description
31:16	Reserved	RO	0x0000	
15:0	Local Count	RC	0x0000	Count of received packets with CRC, SEQ, or length errors. The counter saturates at 0xFFFF. Reading this register clears the Local Count field.

Table 5-13: HMC Controller IP Core REMOTE_ERROR_COUNT Register at Offset 0x34

Bits	Field Name	Type	Value on Reset	Description
31:16	Reserved	RO	0x0000	
15:0	`Error Count`	RC	0x0000	Number of times the HMC Controller IP core began the output error recovery process and retransmitted packets. This number indicates the number of errors detected by the external HMC device. This counter saturates at 0xFFFF. Reading this register clears the `Error Count` field.

Table 5-14: HMC Controller IP Core RETRY_BUFFER_ECC_COUNT Register at Offset 0x38

Bits	Field Name	Type	Value on Reset	Description
31:24	Reserved	RO	0x00	
23:16	`Uncorrectable Count`	RC	0x00	Number of uncorrectable ECC errors the IP core detected in the Retry Buffer memory. This counter saturates at 0xFF. This field maintains the value of zero unless you turn on **Enable M20K ECC support** in the parameter editor. Reading this register clears the `Uncorrectable Count` field.
15:8	Reserved	RO	0x00	
7:0	`Correctable Count`	RC	0x00	Number of correctable ECC errors the IP core detected in the Retry Buffer memory (and corrected). This counter saturates at 0xFF. This field maintains the value of zero unless you turn on **Enable M20K ECC support** in the parameter editor. Reading this register clears the `Correctable Count` field.

Table 5-15: HMC Controller IP Core RESPONSE_BUFFER_ECC_COUNT Register at Offset 0x3C

Bits	Field Name	Type	Value on Reset	Description
31:24	Reserved	RO	0x00	

Bits	Field Name	Type	Value on Reset	Description
23:16	`Uncorrectable Count`	RC	0x00	Number of uncorrectable ECC errors the IP core detected in the Response Queue memory. This counter saturates at 0xFF. This field maintains the value of zero unless you turn on **Enable M20K ECC support** in the parameter editor. Reading this register clears the `Uncorrectable Count` field.
15:8	Reserved	RO	0x00	
7:0	`Correctable Count`	RC	0x00	Number of correctable ECC errors the IP core detected in the Response Queue memory (and corrected). This counter saturates at 0xFF. This field maintains the value of zero unless you turn on **Enable M20K ECC support** in the parameter editor. Reading this register clears the `Correctable Count` field.

Related Information

M20K ECC Support on page 3-9

HMC Controller IP Core Design Example

6

2016.05.02

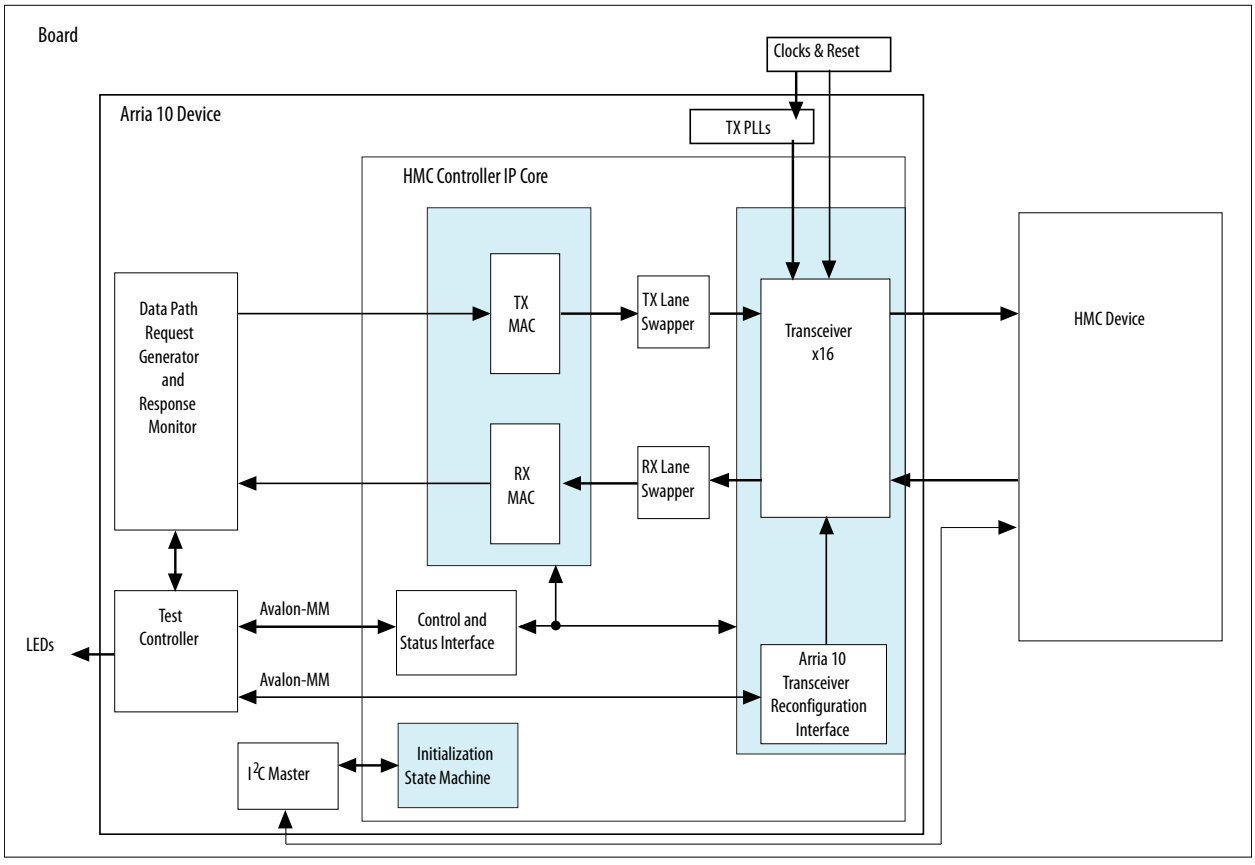
UG-01152

Subscribe Send Feedback

Altera provides a compilation-ready design example with the HMC Controller IP core. This design example targets the Arria 10 GX FPGA Development Kit with an HMC daughter card connected through the FMC connectors. Refer to the *Hybrid Memory Cube Controller Design Example User Guide* for information on using the design example.

Figure 6-1: High Level Block Diagram for the HMC Controller IP Core Design Example

The design example configures a single ATX PLL in xN bonding mode and connects it to the HMC Controller IP core.



Related Information

- **Hybrid Memory Cube Controller Design Example User Guide**
- **All Development Kits web page**
  For information about the Altera development kit that the design examples targets.
- **Quartus Prime Standard Edition Handbook, Volume 3: Verification**
  For information about programming an Altera device, refer to the "Programming Altera Devices" chapter.

HMC Controller IP Core User Guide Archives

A

2016.05.02

UG-01152 Subscribe Send Feedback

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

IP Core Version	User Guide
15.0	Hybrid Memory Cube Controller IP Core User Guide

Additional Information B

2016.05.02

UG-01152 Subscribe Send Feedback

HMC Controller IP Core User Guide Revision History

Table B-1: Document Revision History

Summarizes the new features and changes in the user guide for the HMC Controller IP core.

Date	ACDS Version	Changes

2016.05.02	16.0	• Updated for Quartus Prime software v16.0 release.
		• Added support for multiple (2, 3, or 4) data interfaces in full-width variations.
		  • Added new **Ports** parameter to specify the number of ports (1, 2, 3, or 4). Refer to **HMC Controller IP Core Parameters** on page 2-3.
		  • Added new signals. for the additional interfaces. Signal names on data path interfaces, in the case of more than one port, are `dp<n>_req_*` and `dp<n>_rsp_*` where `<n>` is the port number (0, 1, 2, 3). Refer to **Application Request Interface** on page 4-1 and **Application Response Interface** on page 4-5.
		• Added new **Response re-ordering** parameter for full-width variations, to specify that the IP core should return responses on each data response interface in the order it received the original requests on the corresponding data request interface. When you turn on this new option, the IP core implements tag management internally, and the tags are not visible on the data interfaces. Refer to **HMC Controller IP Core Supported Features** on page 1-2 and **HMC Controller IP Core Parameters** on page 2-3.
		• Expanded list of supported transactions by adding non-power of two payload sizes for READ, WRITE, and Posted WRITE transactions in full-width variations. Full-width variations now support all transaction types that half-width variations support. Refer to **HMC Controller IP Core Supported HMC Transaction Types** on page 1-3.

Date	ACDS Version	Changes

		• Changed name of **Enable Altera Debug Master Endpoint (ADME)** parameter to **Enable ADME and Optional Reconfiguration Logic**. Added list of PHY debugging features the parameter sets. Refer to **HMC Controller IP Core Parameters** on page 2-3.
		• Restricted list of allowed values for the **CDR reference clock** parameter to frequencies that support TX PLL xN bonding mode.
		• Added new data path response interface signal `dp<n>_rsp_errstat[6:0]`. This signal holds the value of the `ERRSTAT` field of the response packet from the external HMC. Refer to **Application Response Interface** on page 4-5.
		• Removed `pll_powerdown` output signal. Refer to **Adding the External PLL** on page 2-13 and **Signals on the Interface to the External PLL** on page 4-17.
		• Updated description of `dp<n>_rsp_error` signal to indicate that the IP core now maintains the value of `dp<n>_rsp_error` for the duration of a multi-cycle transaction. Previously this behavior was not guaranteed. Refer to **Application Response Interface** on page 4-5.
		• Corrected description of `dp<n>_rsp_cmd` signal to clarify that it can hold only the non-error response codes from Table 25 in the HMC specification. Refer to **Application Response Interface** on page 4-5.
		• Added new Limit Outstanding FLITs feature for full-width variations to mitigate read response congestion. Added new `LIMIT_OUTSTANDING_PACKET` register to control the feature. Refer to **Flow Control** on page 3-9 and **LIMIT_OUTSTANDING_PACKET Register** on page 5-6.
		• Added software control for reset, link reinitialization, fatal error recovery, and power management. Added the following fields to the `CONTROL` register:
		  • `P_RST_N` in bit [17]: software controlled reset of the HMC.
		  • `TXPS` in bit [16]: Power management field.
		  • `SoftReset` in bit [2]: software-controlled reset of the IP core.
		  • `Retrain` in bit [0]: Restart IP core link initialization sequence.
		  Refer to **Initialization and Reset** on page 3-5 and **CONTROL Register** on page 5-2.
		• Corrected references to signal direction in description of `hmc_lxtxps` signal in **HMC Interface Signals** on page 4-10.

2015.05.04	15.0	Initial release.

How to Contact Altera

Table B-2: How to Contact Altera

To locate the most up-to-date information about Altera products, refer to this table. You can also contact your local Altera sales office or sales representative.

Contact	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Nontechnical support: general	Email	nacomp@altera.com
Nontechnical support: software licensing	Email	authorization@altera.com

Related Information

- **www.altera.com/support**
- **www.altera.com/training**
- **custrain@altera.com**
- **www.altera.com/literature**
- **nacomp@altera.com**
- **authorization@altera.com**

Typographic Conventions

Table B-3: Typographic Conventions

Lists the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, \ **qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.

Visual Cue	Meaning
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix V Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, <*file name*> and <*project name*> **.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus Prime Help topics. For example, "Typographic Conventions."
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
•	Bullets indicate a list of items when the sequence of the items is not important.
The **Subscribe** button links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.	
The **Feedback** icon allows you to submit feedback to Altera about the document. Methods for collecting feedback vary as appropriate for each document.	

Related Information

Email Subscription Management Center

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Intel ***manufacturer:***

Other Similar products are found below :

RAPPID-567XFSW SRP004001-01 SW163052 SYSWINEV21 Core429-SA WS01NCTF1E W128E13 SW89CN0-ZCC IPS-EMBEDDED IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR LIB-PL-PC-N-1YR-DISKID LIB-PL-A-F SW006026-COV 1120270005 1120270006 MIKROBASIC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR PIC (USB DONGLE LICENSE) MIKROBASIC PRO FOR AVR (USB DONGLE LICEN MIKROBASIC PRO FOR FT90X MIKROC PRO FOR DSPIC30/33 (USB DONGLE LI MIKROPASCAL PRO FOR ARM (USB DONGLE LICE MIKROPASCAL PRO FOR FT90X MIKROPASCAL PRO FOR FT90X (USB DONGLE) MIKROPASCAL PRO FOR PIC32 (USB DONGLE LI SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 2702546 SW006022-DGL 2400303 2701356 VDSP-21XX-PCFLOAT VDSP-BLKFN-PC-FULL 88970111 DG-ACC-NET-CD 55195101-102 SW1A-W1C MDK-ARM PCI-EXP1-E3-US PCI-T32-E3-US SW006021-2NH SW006021-1H SW006021-2