# Hardware System Verification (HSV)
# Vertical Solutions Engineering (VSE)

# Hybrid Memory Cube (HMC)
# Palladium Memory Model
# User Guide

Document Version: 1.9

Document Date: July 2018

# Contents

## List of Figures

## List of Tables

# General Information

The Cadence Memory Model Portfolio provides memory device models for the Cadence Palladium XP, Palladium XP II and Palladium Z1 series systems. Optimizing the acceleration and/or emulation flow on these platforms for MMP memory models may require information outside the scope of the MMP user guides and related MMP documentation.

## 1.1 Related Publications

For basic information regarding emulation and acceleration, please refer to the following documents:

For Palladium XP and Palladium XP II:

UXE User Guide
UXE Library Developer's Guide
UXE Known Problems and Solutions
UXE Command Reference Manual
Palladium XP Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in UXE

For Palladium Z1:

VXE User Guide
VXE Library Developer's Guide
VXE Known Problems and Solutions
VXE Command Reference Manual
Palladium Z1 Planning and Installation Guide
Palladium Target System Developer's Guide
What's New in VXE

# HMC Memory Model

## 1. Introduction

The Cadence Hybrid Memory Cube (HMC) Palladium memory model is based upon *Hybrid Memory Cube Specification 2.1* (October 2015).

## 2. Model Release Levels

All models in the Memory Model Portfolio are graded with a release level. This release level informs users of the current maturity and status of the model. All families in the library are graded at one of these levels.

The different levels give an overall indication of the amount of testing, level of quality and feature availability in the model. For details on supported features check the User Guide for that particular model family.

There are three release levels for models in the MMP release.

| Release Level | | Model Status | Available in Release | Listed in Catalog | Requires Beta Agreement |
|---|---|---|---|---|---|
| Mainstream Release | MR | Fully released and available in the catalog for all customers to use. | Yes | Yes | No |
| Emerging Release | ER | Model has successfully completed Beta engagement(s). Most, but not all features have been tested. Documentation is available. | No | Yes | Yes |
| Initial Release | IR | Model has completed initial development and has been released to Beta customer(s). The model may have missing features, may not be fully tested and may not have documentation. Model may contain defects. | No | Yes | Yes |

Access to Initial Release and Emerging Release versions of the models will require a Beta Agreement to be signed before the model can be delivered.

## 3. Acronyms

**Table 1: HMC Model Acronyms**

| Term | Definition |
|---|---|
| ADR or ADRS | Address |
| CMD | Command |
| CRC | Cyclical Redundancy Checking |
| CUB | Cube |
| DLN | Duplicate length |
| DNU | Do Not Use |
| Downstream | Away from the host |
| ERRSTAT | Error status |
| Forward | From the point of view of the link master, “forward” meaning in the direction that the link master is driving, on this side of the link. |
| FRP | Forward retry pointer |
| HMC | Hybrid memory cube |
| Host link | HMC link configuration that uses its link slave to receive request packets and its link master to transmit response packets. |
| HSS | High-speed SerDes |
| IRTRY | Init retry |
| Lane | A pair of differential signal lines, one in each direction (transmitters and receivers); multiple lanes are combined to form links. |
| LFSR | Linear feedback shift register |
| Link | A fully-duplexed interface connecting two components, implemented with SerDes lanes that carry system commands and data. |
| LNG | Length |
| LSB | Least significant bit |
| MSB | Most significant bit |
| NULL | Null packets |
| Partition | Portion of a memory die that is connected to and controlled by a single vault controller |
| Pass-thru Link | HMC link configuration that uses its link master to transmit the request packet toward its destination cube and its link slave to receive response packets destined for the host processor. |
| PRET | Retry pointer return |
| Quadrant | The four local vaults associated with a given link that do not require routing across the crossbar switch. |
| Requester | Represents either a host processor or an HMC link configured as a pass-thru link. A requester transmits packets downstream to the responder. |
| Responder | Represents an HMC link configured as a host link. A responder transmits packets upstream to the requester. |
| Return | From the point-of-view of the link master, “return” meaning in the direction on the other side of the link that the local link slave is receiving. |
| RRP | Return retry pointer. |
| RTC | Return token counts |
| SBE | Single bit error |
| SEQ | Sequence number |
| TAG | Data tag |
| TRET | Token return |
| Upstream | Toward the host |
| Vault | Independent memory controller and local memory stacked directly above, in a cube |

## 4. Features

The Cadence Hybrid Memory Cube (HMC) Palladium memory model is based upon *Hybrid Memory Cube Specification 2.1* (October 2015). The HMC features and the level of support provided in the MMP memory model are listed below in Table 2: Features List of HMC Model.

**Table 2: Features List of HMC Model**

| Feature | Support | Spec. Section |
|---|---|---|
| Full width link (16 lanes) | Yes | Logical Sub-Block of Physical Layer |
| Half width link (8 lanes) | Yes | Logical Sub-Block of Physical Layer |
| Quarter width link (4 lanes) | No | Logical Sub-Block of Physical Layer |
| Half-width link configuration uses lanes 0-7 or 8-15 | Yes | Logical Sub-Block of Physical Layer |
| Lane scrambling and descrambling | Yes | Logical Sub-Block of Physical Layer |
| Lane reversal | No | Logical Sub-Block of Physical Layer |
| Lane polarity | No | Logical Sub-Block of Physical Layer |
| Chaining (up to 8 Cubes) | Yes | Chaining |
| Link layer initialization | Yes | Power-On and Initialization |
| Power state management | Yes | Power State Management |
| Address mapping | Yes | Memory Addressing |
| Tagging | Yes | Tagging |
| CRC error checking | Yes | Packet Integrity |
| Sequence number checking | Yes | Request Packets |
| Packet length checking | Yes | Request Packets |
| Command validity checking | Yes | Request Packets |
| Error status report | Yes | Response Packets |
| Flow packets | Yes | Flow Packets |
| Transaction layer initialization | Yes | Transaction Layer Initialization |

| Feature | Support | Spec. Section |
|---|---|---|
| Link retry | Yes | Link Retry |
| Poisoned data feature in WRITE/READ request command | Yes[1] | |
| Warm reset | No | Warm Reset |
| Retraining | No | Retraining |
| Vault ECC and Reference Error Detection | No | Functional Characteristics |
| Refresh | No | Functional Characteristics |
| Scrubbing | No | Functional Characteristics |
| Response open loop mode | No | Functional Characteristics |
| JTAG interface | No | JTAG Interface |
| I2C interface | No | I2C Interface |
| **Commands** | | |
| 16-byte .. 256-byte WRITE request | Yes | Request Commands |
| 16-byte .. 256-byte POSTED WRITE request | Yes | Request Commands |
| MODE WRITE request | Yes | Request Commands |
| All Arithmetic Atomics requests | Yes | ATOMIC Request Commands |
| All Boolean Atomics requests | Yes | ATOMIC Request Commands |
| All Comparison Atomics requests | Yes | ATOMIC Request Commands |
| All Bitwise Atomics requests | Yes | ATOMIC Request Commands |
| 16-byte .. 256-byte READ request | Yes | Request Commands |
| MODE READ request | Yes | Request Commands |
| READ response | Yes | Response Commands |
| WRITE response | Yes | Response Commands |
| MODE READ response | Yes | Response Commands |
| MODE WRITE response | Yes | Response Commands |
| ERROR response | Yes | Response Commands |
| Null (NULL) | Yes | Flow Commands |
| Retry pointer return (PRET) | Yes | Flow Commands |

| Feature | Support | Spec. Section |
|---|---|---|
| Token return (TRET) | Yes | Flow Commands |
| Init retry (IRTRY) | Yes | Flow Commands |

Notes:

1. Poison code is 128'hEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE; For the READ request with pb=1, the response only set the ERRSTAT to 7'b0000011, no ECC function.

## 2. Verilog Macro Defines

The following table, Table 3: Verilog Macro Defines, lists the Verilog macro `define(s) required by the MMP HMC model. There is an example in section 11 Compile and Emulation to show how to define these macros. It is not necessary for the user to define all the LINK and CUBE macros; the usage of these macros depends on the users' requirements. <u>But, at least one LINK and one CUBE MUST be defined.</u>

**Table 3: Verilog Macro Defines**

| Marco Name | Purpose |
|---|---|
| MMP_HMC_ENABLE_HOST_LINK_0 | Enable host link 0 [1] |
| MMP_HMC_ENABLE_HOST_LINK_1 | Enable host link 1 |
| MMP_HMC_ENABLE_HOST_LINK_2 | Enable host link 2 |
| MMP_HMC_ENABLE_HOST_LINK_3 | Enable host link 3 |
| MMP_HMC_ENABLE_CUBE_0 | Enable cube 0 [2] |
| MMP_HMC_ENABLE_CUBE_1 | Enable cube 1 |
| MMP_HMC_ENABLE_CUBE_2 | Enable cube 2 |
| MMP_HMC_ENABLE_CUBE_3 | Enable cube 3 |
| MMP_HMC_ENABLE_CUBE_4 | Enable cube 4 |
| MMP_HMC_ENABLE_CUBE_5 | Enable cube 5 |
| MMP_HMC_ENABLE_CUBE_6 | Enable cube 6 |
| MMP_HMC_ENABLE_CUBE_7 | Enable cube 7 |
| MMP_HMC_INSERT_CRC_ERROR | Inject CRC error in the packet sent to host controller [3] |
| MMP_EN_SCRAMBLE | Enable the TX scramble and RX descramble configuration |
| MMP_HMC_STATISTIC_ENABLE | Enable statistic report [4] |

Notes:

1. Enable specific host link. If the user wants to use all the links as host links, the user needs to define all of the macros --- MMP_HMC_ENABLE_HOST_LINK_X. The number of defined LINK macros shall match with the parameter "SOURCE_LINK_NUM" described in section 7 Model Parameter Descriptions. For example, if the user enables host link 0 and host link1 by defining the appropriate macros, then the corresponding parameter SOURCE_LINK_NUM needs to be '2'.

2. Enable specific cubes by setting the corresponding Verilog macro if the user expects to use chaining.

3. If the user defines this macro, the HMC model will inject CRC errors every 100 packets.

4. If the user defines this macro, statistic reporting will be enabled. This is part of HMC Debug Display, for the details, please read section 14 of this user guide.

## 3. Model Block Diagram

The following diagram, Figure 1: HMC Model Block Diagram shows the HMC memory model. The number of host links and cubes are defined by users. The different delays between host and cubes can be set by parameters.

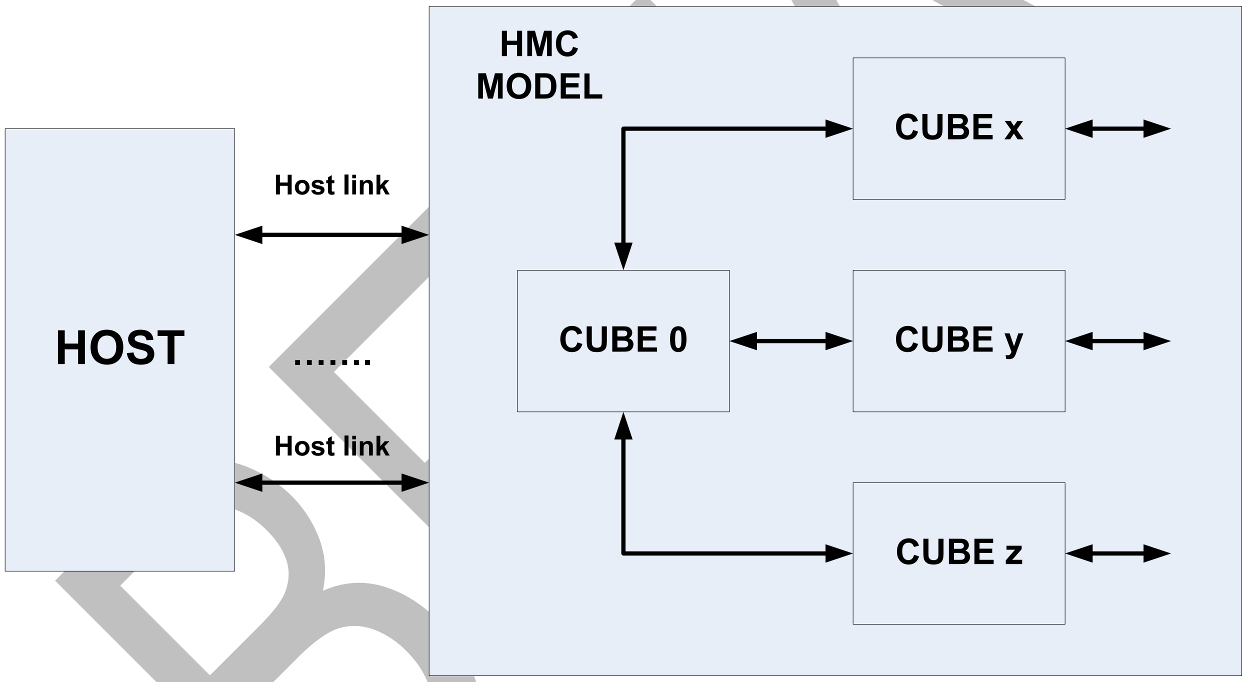

**Figure 1: HMC Model Block Diagram**

## 4. Input/Output (I/O) Diagram

The IO signals shown in Figure 2: HMC Model I/O Diagram include common model signals (clock and reset) and standard HMC link interfaces.

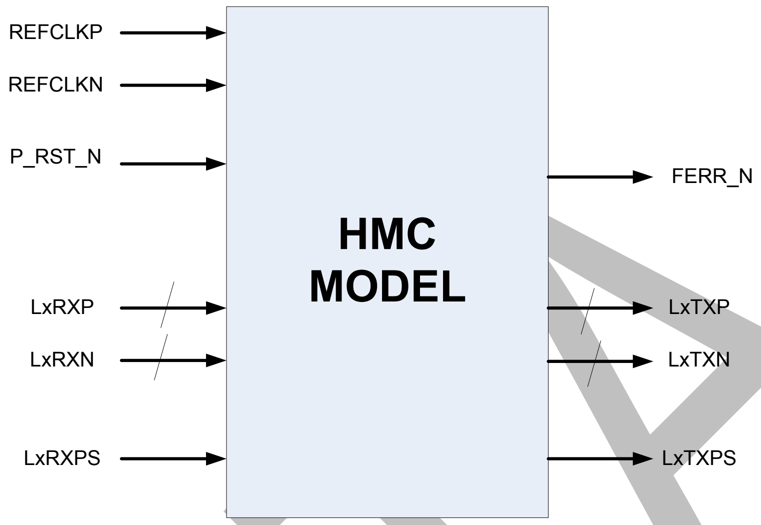

**Figure 2: HMC Model I/O Diagram**

## 5. I/O Signal Description

The table below, Table 4: HMC Model I/O Signals, lists and describes the model I/O signals. Some ports listed in the spec are not supported in the Palladium HMC model, they are listed in the table as “Not supported”.

**Table 4: HMC Model I/O Signals**

| NAME | TYPE | DESCRIPTION |
|---|---|---|
| **Link Interface** [1] | | |
| LxRXP[n:0] | input | Receiving lanes [2] |
| LxRXN[n:0] | | |
| LxTXP[n:0] | output | Transmitting lanes [2] |
| LxTXN[n:0] | | |
| LxRXPS | input | Power-reduction input |
| LxTXPS | output | Power-reduction output |
| FERR_N | output | Fatal error indicator. HMC drives this signal LOW if a fatal error occurs, otherwise the pull down is turned off and floats HIGH |
| **Clocks and Reset** | | |
| REFCLKP | input | Reference clock for all links, it is used as the UI clock for input data LxRxP and output data LxTxP. |
| REFCLKN | | |
| P_RST_N | input | System reset, active LOW |

| NAME | TYPE | DESCRIPTION |
|---|---|---|
| flit_clk | output | The flit clock generated and used by MMP HMC model.<br>NOTE: The user needs check this signal with the flit clock of the host controller, and set the parameter 'FLIT_CLK_SHIFT_NUM' to make the two flit clock aligned. Refer to section 7 for detail. |
| **Unsupported I/O** | | |
| REFCLKSEL | input | Not supported |
| CUB[2:0] | input | User-assigned HMC identification to enable the host to map the unique location of an HMC in a system.<br><br>NOTE: For the Palladium HMC model, the user enables the chaining mode by setting specific macros (section 2 Verilog Macro Defines) and delay parameters (section 7 Model Parameter Descriptions). The CUB[2:0] signal is therefore not valid for Palladium HMC model. The port exists but the input information is not handled in any way. |
| All the JTAG interfaces | | Not supported |
| All the I2C interfaces | | Not supported |
| Bootstrapping pins | | Not supported |
| All the analog interfaces | | Not supported |

Notes:
1. 'x' represents a specific link number and will be within 0 .. 3 depending upon the user's configuration.
2. In full-width mode n=15. In half-width mode n=7, lanes 0-7 or 8-15 are used depending on the parameter LANE_MAP_MODE.

## 6. Example Instantiation of HMC Model

The following is an instantiation example that uses a chain which consists of 3 host links and 8 cubes. The cube delays shown here are default values; see [Table 5: User Adjustable Parameters]. In this example the link width is 16 lanes and address mapping is 4GB-32byte.

```
mmp_hmc_top    #(
    .SOURCE_LINK_NUM(1),
    .LANE_NUM       (16),
    .VAULT_WIDTH    (5),
    .BANK_WIDTH     (3),
    .LANE_MAP_MODE  (0)
)
hmc_pd  (
    .REFCLKP     (CLK),
    .REFCLKN     (),
    .L0RXP       (LINK_RX[0]),
    .L0RXN       (),
    .L1RXP       (LINK_RX[1]),
    .L1RXN       (),
    .L2RXP       (LINK_RX[2]),
    .L2RXN       (),
    .L3RXP       (LINK_RX[3]),
    .L3RXN       (),
    .L0TXP       (LINK_TX[0]),
    .L0TXN       (),
    .L1TXP       (LINK_TX[1]),
    .L1TXN       (),
    .L2TXP       (LINK_TX[2]),
    .L2TXN       (),
    .L3TXP       (LINK_TX[3]),
    .L3TXN       (),
    .P_RST_N     (P_RST_N),
    .FERR_N      (FERR_N),
    .L0RXPS      (RXPS[0]),
    .L1RXPS      (RXPS[1]),
    .L2RXPS      (RXPS[2]),
    .L3RXPS      (RXPS[3]),
    .L0TXPS      (TXPS[0]),
    .L1TXPS      (TXPS[1]),
    .L2TXPS      (TXPS[2]),
    .L3TXPS      (TXPS[3]),
    .flit_clk    ()
 );
```

## 7. Model Parameter Descriptions

The following table, Table 5: User Adjustable Parameters, provides details on the user adjustable parameters for the Palladium HMC Memory Model. These parameters may be modified when instantiating a HMC wrapper or, if necessary, by modifying the HDL parameter declarations and default values which are exposed in the RTL for access and debug visibility. Note that all the numbers related to timing are measured by REFCLK, NOT ns/us/ms.

**Table 5: User Adjustable Parameters**

| User Adjustable Parameter | Default Value | Description |
|---|---|---|
| SOURCE_LINK_NUM | 1 | Number of source links which are enabled in user's configuration. Value can be 1 to 4. The value shall be compatible with MMP_HMC_ENABLE_LINK_X |
| LANE_NUM | 16 | Number of the lanes which are enabled in user's configuration. Value can be 8 or 16. |
| VAULT_WIDTH | 5 | The number of the vaults is (1<<VAULT_WIDTH). The value is related to address mapping (section 8 of this User Guide) |
| BANK_WIDTH | 3 | The number of the banks is (1<<BANK_WIDTH). The value is related to address mapping (section 8 of this UG) |
| LANE_MAP_MODE | 0 | 0: full-width link configuration, use all 0~15<br>1: half-width link configuration, use 0~7<br>2: half-width link configuration, use 8~15 |
| INPUT_BUFFER_ADDR_WIDTH | 10 | The depth of link input buffer is (1<<INPUT_BUFFER_ADDR_WIDTH) |
| SOURCE_CUBE_ID | 0 | Specify the cube ID which is connected to the host directly. Typically the value is '0'. If the user wants to specify another cube as the source cube, this parameter is set.<br><br>NOTE: The parameter CUBEx_RESPONSE_DELAY shall be compatible with this parameter. The source cube MUST have the smallest delay. |
| CUBE0_RESPONSE_DELAY | 0 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE1_RESPONSE_DELAY | 100 | The delay between current cube and the source cube. This |

| User Adjustable Parameter | Default Value | Description |
|---|---|---|
| | | value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE2_RESPONSE_DELAY | 100 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE3_RESPONSE_DELAY | 100 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE4_RESPONSE_DELAY | 200 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE5_RESPONSE_DELAY | 200 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE6_RESPONSE_DELAY | 200 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| CUBE7_RESPONSE_DELAY | 300 | The delay between current cube and the source cube. This value is measured by REFCLK, all the 8 cubes make a star topology |
| DELAY_POINTER_WIDTH | 9 | (1<<DELAY_POINTER_WIDTH) MUST be larger than the value of the maximum cube delay. For example, the maximum default cube delay is 300, (1<<DELAY_POINTER_WIDTH) must be larger than 300. |
| FLIT_CLK_SHIFT_NUM | 0 | The number of UI clock (REFCLKP) cycles to shift to make the flit clock in the MMP HMC model aligned with the controller's. '+' for left-shift and '-' for right-shift. The value range is 0 to 7 for full-width and |

| User Adjustable Parameter | Default Value | Description |
|---|---|---|
| | | 0 to 15 for half-width configuration. |
| STATISTIC_PERIOD | 100000 | Indicates the period of STATISTIC display, measured by "REFCLK". For example, clock frequency is 100MHz, default value of parameter is 100000, so statistic will be printed every 1ms |

The following table Table 6: Visible Non-User-Adjustable Parameters & Localparams provides some information about exposed parameters and localparams that are NOT user adjustable. On rare occasion the user may find one of these parameters or localparam needs adjusting for their configuration. If this case arises, please contact Cadence emulation or MMP support. Note that all the numbers related to timing are measured by REFCLK, NOT ns/us/ms.

**Table 6: Visible Non-User-Adjustable Parameters & Localparams**

| Parameter / Localparam | Default Value | Description |
|---|---|---|
| hmc_lane_sync.vp | | |
| TS1_OK_NUM | 8 | The number of scrambled TS1 patterns which aligns; after receiving these patterns, step to the stage that is waiting for at least 32 null packets. |
| hmc_link_sync.vp | | |
| NULL_RECEIVE_AT_LEAST_NUM | 32 | After finishing TS1 sync, the memory model needs to receive at least 32 null packets before starting normal transaction flow |
| NULL_SEND_NUM | 36 | After finishing TS1 sync, the model needs to send 36 null packets before sending normal flow packets |
| hmc_link.vp | | |
| tPST | 8 | Power state transition timing: represents the delay from the transition of a link's LxRXPS signal to the transition of its LxTXPS signal |

Note that there are additional exposed localparams in the model HDL that are not described here nor intended to be described here. These additional localparams are exposed for debugging purposes only and will not be described herein.

## 8. Address Mapping

A request packet header includes an address field of 34 bits for internal memory addressing within the HMC. This includes vault, bank, and DRAM address bits. The configurations currently defined will provide a total of up to 8GB of addressable memory locations within one HMC. This requires use of the lower 33 address bits of the 34-bit field; the upper 1 bits is reserved for future expansion and are ignored by the HMC.

**Table 7: Addressing Definitions**

| Address | Description | Comments |
|---|---|---|
| Byte address | Bytes within the maximum supported block size | The 4 LSBs of the byte address are ignored for READ and WRITE requests with the exception of BIT WRITE command |
| Vault address | Addresses vaults within the HMC | Lower 3 bits of the vault address specifies 1 of 8 vaults within the logic chip quadrant;<br>Upper 2 bits of the vault address specifies 1 of 4 quadrants |
| Bank address | Addresses banks within a vault | 4GB HMC: addresses 1 of 8 banks in the vault;<br>8GB HMC: addresses 1 of 16 banks in the vault |
| DRAM address | Addresses DRAM rows and column within a bank | The vault controller breaks the DRAM address into row and column addresses, addressing 1Mb blocks of 16 bytes each |

The host can choose to use one of the default address map modes offered in the HMC (See

Table 8: Default Address-Map Mode Table). The default address mapping is based on the maximum block size chosen in the address map mode register. It maps the vault address to the least significant address bits, with the bank address field lying just above the vault address. This address mapping algorithm is referred to as “low interleave,” and forces sequential addressing to be spread across different vaults and then across different banks within a vault, thus avoiding bank conflicts. A request stream that has a truly random address pattern is not sensitive to the specific method of address mapping. This function is related to the mode register address_mapping_mode in Table 9: Configuration and Status Registers.

**Table 8: Default Address-Map Mode Table**

| Request Address Bit | 4GB | | | |
|---|---|---|---|---|
| | 32-Byte Max Block Size | 64-Byte Max Block Size | 128-Byte Max Block Size | 256-Byte Max Block Size |
| 33 | Ignored | Ignored | Ignored | Ignored |
| 32 | Ignored | Ignored | Ignored | Ignored |
| 31 | DRAM[19] | DRAM[19] | DRAM[19] | DRAM[19] |
| 30 | DRAM[18] | DRAM[18] | DRAM[18] | DRAM[18] |
| 29 | DRAM[17] | DRAM[17] | DRAM[17] | DRAM[17] |
| 28 | DRAM[16] | DRAM[16] | DRAM[16] | DRAM[16] |
| 27 | DRAM[15] | DRAM[15] | DRAM[15] | DRAM[15] |
| 26 | DRAM[14] | DRAM[14] | DRAM[14] | DRAM[14] |
| 25 | DRAM[13] | DRAM[13] | DRAM[13] | DRAM[13] |
| 24 | DRAM[12] | DRAM[12] | DRAM[12] | DRAM[12] |
| 23 | DRAM[11] | DRAM[11] | DRAM[11] | DRAM[11] |
| 22 | DRAM[10] | DRAM[10] | DRAM[10] | DRAM[10] |
| 21 | DRAM[9] | DRAM[9] | DRAM[9] | DRAM[9] |
| 20 | DRAM[8] | DRAM[8] | DRAM[8] | DRAM[8] |
| 19 | DRAM[7] | DRAM[7] | DRAM[7] | DRAM[7] |
| 18 | DRAM[6] | DRAM[6] | DRAM[6] | DRAM[6] |
| 17 | DRAM[5] | DRAM[5] | DRAM[5] | DRAM[5] |
| 16 | DRAM[4] | DRAM[4] | DRAM[4] | DRAM[4] |
| 15 | DRAM[3] | DRAM[3] | DRAM[3] | Bank[2] |
| 14 | DRAM[2] | DRAM[2] | Bank[2] | Bank[1] |
| 13 | DRAM[1] | Bank[2] | Bank[1] | Bank[0] |
| 12 | Bank[2] | Bank[1] | Bank[0] | Vault[4] |
| 11 | Bank[1] | Bank[0] | Vault[4] | Vault[3] |
| 10 | Bank[0] | Vault[4] | Vault[3] | Vault[2] |
| 9 | Vault[4] | Vault[3] | Vault[2] | Vault[1] |
| 8 | Vault[3] | Vault[2] | Vault[1] | Vault[0] |
| 7 | Vault[2] | Vault[1] | Vault[0] | Byte[7]=DRAM[3] |
| 6 | Vault[1] | Vault[0] | Byte[6]=DRAM[2] | Byte[6]=DRAM[2] |
| 5 | Vault[0] | Byte[5]=DRAM[1] | Byte[5]=DRAM[1] | Byte[5]=DRAM[1] |
| 4 | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] |
| 3 | Byte[3] = ignored | Byte[3] = ignored | Byte[3] = ignored | Byte[3] = ignored |
| 2 | Byte[2] = ignored | Byte[2] = ignored | Byte[2] = ignored | Byte[2] = ignored |
| 1 | Byte[1] = ignored | Byte[1] = ignored | Byte[1] = ignored | Byte[1] = ignored |
| 0 | Byte[0] = ignored | Byte[0] = ignored | Byte[0] = ignored | Byte[0] = ignored |

| **Request Address Bit** | **8GB** | | | |
|---|---|---|---|---|
| | **32-Byte Max Block Size** | **64-Byte Max Block Size** | **128-Byte Max Block Size** | **256-Byte Max Block Size** |
| 33 | Ignored | Ignored | Ignored | Ignored |
| 32 | DRAM[19] | DRAM[19] | DRAM[19] | DRAM[19] |
| 31 | DRAM[18] | DRAM[18] | DRAM[18] | DRAM[18] |
| 30 | DRAM[17] | DRAM[17] | DRAM[17] | DRAM[17] |
| 29 | DRAM[16] | DRAM[16] | DRAM[16] | DRAM[16] |
| 28 | DRAM[15] | DRAM[15] | DRAM[15] | DRAM[15] |
| 27 | DRAM[14] | DRAM[14] | DRAM[14] | DRAM[14] |
| 26 | DRAM[13] | DRAM[13] | DRAM[13] | DRAM[13] |
| 25 | DRAM[12] | DRAM[12] | DRAM[12] | DRAM[12] |
| 24 | DRAM[11] | DRAM[11] | DRAM[11] | DRAM[11] |
| 23 | DRAM[10] | DRAM[10] | DRAM[10] | DRAM[10] |
| 22 | DRAM[9] | DRAM[9] | DRAM[9] | DRAM[9] |
| 21 | DRAM[8] | DRAM[8] | DRAM[8] | DRAM[8] |
| 20 | DRAM[7] | DRAM[7] | DRAM[7] | DRAM[7] |
| 19 | DRAM[6] | DRAM[6] | DRAM[6] | DRAM[6] |
| 18 | DRAM[5] | DRAM[5] | DRAM[5] | DRAM[5] |
| 17 | DRAM[4] | DRAM[4] | DRAM[4] | DRAM[4] |
| 16 | DRAM[3] | DRAM[3] | DRAM[3] | Bank[3] |
| 15 | DRAM[2] | DRAM[2] | Bank[3] | Bank[2] |
| 14 | DRAM[1] | Bank[3] | Bank[2] | Bank[1] |
| 13 | Bank[3] | Bank[2] | Bank[1] | Bank[0] |
| 12 | Bank[2] | Bank[1] | Bank[0] | Vault[4] |
| 11 | Bank[1] | Bank[0] | Vault[4] | Vault[3] |
| 10 | Bank[0] | Vault[4] | Vault[3] | Vault[2] |
| 9 | Vault[4] | Vault[3] | Vault[2] | Vault[1] |
| 8 | Vault[3] | Vault[2] | Vault[1] | Vault[0] |
| 7 | Vault[2] | Vault[1] | Vault[0] | Byte[7]=DRAM[3] |
| 6 | Vault[1] | Vault[0] | Byte[6]=DRAM[2] | Byte[6]=DRAM[2] |
| 5 | Vault[0] | Byte[5]=DRAM[1] | Byte[5]=DRAM[1] | Byte[5]=DRAM[1] |
| 4 | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] | Byte[4]=DRAM[0] |
| 3 | Byte[3] = ignored | Byte[3] = ignored | Byte[3] = ignored | Byte[3] = ignored |
| 2 | Byte[2] = ignored | Byte[2] = ignored | Byte[2] = ignored | Byte[2] = ignored |
| 1 | Byte[1] = ignored | Byte[1] = ignored | Byte[1] = ignored | Byte[1] = ignored |
| 0 | Byte[0] = ignored | Byte[0] = ignored | Byte[0] = ignored | Byte[0] = ignored |

## 9. Configuration and Status Registers

The HMC model supports the following configuration and status registers.

**Table 9: Configuration and Status Registers**

| Address | Name | Reset value | Type | Description |
|---|---|---|---|---|
| 0x2B0000 | External Data Register 0 | 0 | RW | Multipurpose register used to access configuration and status registers |
| 0x2B0001 | External Data Register 1 | 0 | RW | Multipurpose register used to access configuration and status registers |
| 0x2B0002 | External Data Register 2 | 0 | RW | Multipurpose register used to access configuration and status registers |
| 0x2B0003 | External Data Register 3 | 0 | RW | Multipurpose register used to access configuration and status registers |
| 0x2B0004 | External Request Register | 0 | RW | Bit[31]: Start<br>Bit[30:26]: External request status<br>Bit[25:22]: Size<br>Bit[21:16]: Target location<br>Bit[15:8]: Type<br>Bit[7:0]: Register Request commands |
| 0x280000 | Global Configuration | 0 | RW | Bit[31:7]: Vendor specific<br>Bit[6]: Warm Reset<br>Bit[5]: Clear Error<br>Bit[4]: Stop on Fatal Error<br>Bit[3:0]: Vendor specific |
| 0x240000 | Link Configuration (Link0)[1] | 0x00000879 | RW | Bit[31:12]: Vendor specific<br>Bit[11]: Error Response Packet<br>Bit[10]: Link Scramble Enable<br>Bit[9]: Link Descramble Enable<br>Bit[8]: Inhibit Link Down Mode<br>Bit[7]: Packet Output Enable<br>Bit[6]: Packet Input Enable<br>Bit[5]: Link Duplicate Length Detection<br>Bit[4]: Link CRC Detection<br>Bit[3]: Link Packet Sequence Detection<br>Bit[2]: Link Response Open Loop Mode<br>Bit[1:0]: Link Mode |
| 0x250000 | Link Configuration (Link1)[1] | 0x00000879 | RW | Bit[31:12]: Vendor specific<br>Bit[11]: Error Response Packet<br>Bit[10]: Link Scramble Enable<br>Bit[9]: Link Descramble Enable<br>Bit[8]: Inhibit Link Down Mode<br>Bit[7]: Packet Output Enable<br>Bit[6]: Packet Input Enable<br>Bit[5]: Link Duplicate Length Detection<br>Bit[4]: Link CRC Detection<br>Bit[3]: Link Packet Sequence Detection<br>Bit[2]: Link Response Open Loop Mode<br>Bit[1:0]: Link Mode |
| 0x260000 | Link Configuration (Link2)[1] | 0x00000879 | RW | Bit[31:12]: Vendor specific<br>Bit[11]: Error Response Packet<br>Bit[10]: Link Scramble Enable<br>Bit[9]: Link Descramble Enable<br>Bit[8]: Inhibit Link Down Mode |

| | | | | |
|---|---|---|---|---|
| | | | | Bit[7]: Packet Output Enable<br>Bit[6]: Packet Input Enable<br>Bit[5]: Link Duplicate Length Detection<br>Bit[4]: Link CRC Detection<br>Bit[3]: Link Packet Sequence Detection<br>Bit[2]: Link Response Open Loop Mode<br>Bit[1:0]: Link Mode |
| 0x270000 | Link Configuration (Link3)[1] | 0x00000879 | RW | Bit[31:12]: Vendor specific<br>Bit[11]: Error Response Packet<br>Bit[10]: Link Scramble Enable<br>Bit[9]: Link Descramble Enable<br>Bit[8]: Inhibit Link Down Mode<br>Bit[7]: Packet Output Enable<br>Bit[6]: Packet Input Enable<br>Bit[5]: Link Duplicate Length Detection<br>Bit[4]: Link CRC Detection<br>Bit[3]: Link Packet Sequence Detection<br>Bit[2]: Link Response Open Loop Mode<br>Bit[1:0]: Link Mode |
| 0x240003 | Link Run Length Limit (Link0) | 0 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Transmit Run Length Limit<br>Bit[15:0]: Vendor specific |
| 0x250003 | Link Run Length Limit (Link1) | 0 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Transmit Run Length Limit<br>Bit[15:0]: Vendor specific |
| 0x260003 | Link Run Length Limit (Link2) | 0 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Transmit Run Length Limit<br>Bit[15:0]: Vendor specific |
| 0x270003 | Link Run Length Limit (Link3) | 0 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Transmit Run Length Limit<br>Bit[15:0]: Vendor specific |
| 0x0C0000 | Link Retry(Link0) | 0x00100856 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Init Retry Packet Receive Number<br>Bit[15:14]: Vendor specific<br>Bit[13:8]: Init Retry Packet Transmit Number<br>Bit[7]: Vendor specific<br>Bit[6:4]: Retry Timeout Period<br>Bit[3:1]: Retry Limit<br>Bit[0]: Retry Status |
| 0x0D0000 | Link Retry(Link1) | 0x00100856 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Init Retry Packet Receive Number<br>Bit[15:14]: Vendor specific<br>Bit[13:8]: Init Retry Packet Transmit Number<br>Bit[7]: Vendor specific<br>Bit[6:4]: Retry Timeout Period<br>Bit[3:1]: Retry Limit<br>Bit[0]: Retry Status |
| 0x0E0000 | Link Retry(Link2) | 0x00100856 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Init Retry Packet Receive Number<br>Bit[15:14]: Vendor specific<br>Bit[13:8]: Init Retry Packet Transmit Number<br>Bit[7]: Vendor specific |

| | | | | |
|---|---|---|---|---|
| | | | | Bit[6:4]: Retry Timeout Period<br>Bit[3:1]: Retry Limit<br>Bit[0]: Retry Status |
| 0x0F0000 | Link Retry(Link3) | 0x00100856 | RW | Bit[31:24]: Vendor specific<br>Bit[23:16]: Init Retry Packet Receive Number<br>Bit[15:14]: Vendor specific<br>Bit[13:8]: Init Retry Packet Transmit Number<br>Bit[7]: Vendor specific<br>Bit[6:4]: Retry Timeout Period<br>Bit[3:1]: Retry Limit<br>Bit[0]: Retry Status |
| 0x040000 | Input Buffer Token Count(Link0) | 0x00000064 | RW | Bit[31:8]: Vendor specific<br>Bit[7:0]: Link Input Buffer Max Token Count |
| 0x050000 | Input Buffer Token Count(Link1) | 0x00000064 | RW | Bit[31:8]: Vendor specific<br>Bit[7:0]: Link Input Buffer Max Token Count |
| 0x060000 | Input Buffer Token Count(Link2) | 0x00000064 | RW | Bit[31:8]: Vendor specific<br>Bit[7:0]: Link Input Buffer Max Token Count |
| 0x070000 | Input Buffer Token Count(Link3) | 0x00000064 | RW | Bit[31:8]: Vendor specific<br>Bit[7:0]: Link Input Buffer Max Token Count |
| 0x2C0000 | Address Configuration | 0x00000001 | RW | Bit[31:14]: Vendor specific<br>Bit[13:9]: User-defined Bank Register Address<br>Bit[8:4]: User-defined Vault Register Address<br>Bit[3:0]: Address Mapping Mode<br>(0x0: 32-byte Max block size<br>0x1: 64-byte Max block size<br>0x2: 128-byte Max block size<br>0x3: 256-byte Max block size) |
| 0x108000 | Vault Control Register | 0x00001A78 | RW | Bit[31:13]: Vendor specific<br>Bit[12]: Enable SBE Report<br>Bit[11]: Hard SBE Repair<br>Bit[10]: Vendor specific<br>Bit[9]: Data ECC Correction<br>Bit[8:6]: Command/Address Retry Count<br>Bit[5]: packet CRC Detection<br>Bit[4]: Patrol Scrubbing<br>Bit[3]: Demand Scrubbling<br>Bit[2]: Hard SBE Detect<br>Bit[1:0]: DRAM Initialization Mode |
| 0x2C0003 | Features | 4GB:<br>0x00000011<br>8GB:<br>0x00000112 | RO | Bit[31:16]: Vendor specific<br>Bit[15:12]: PHY<br>Bit[11:8]: Number of Banks per Vault<br>Bit[7:4]: Number of Vaults<br>Bit[3:0]: Cube Size |
| 0x2C0004 | Revisions and Vendor ID | 0x02020000 | RO | Bit[31:24]: PHY revision<br>Bit[23:16]: Protocol revision<br>Bit[15:8]: Product revision<br>Bit[7:0]: Vendor ID |

Notes:

1. The reset value of Link Configuration register will be 0x00000E79 when defined the macro 'MMP_EN_SCRAMBLE'.

## 10. HMC Initialization Sequence

The HMC initialization sequence is comprised of two portions: a link layer startup sequence and a transaction layer startup sequence. The host controller should follow the sequences described below to ensure that initialization is completed. The default value of 'Link descramble enable' and 'Link scramble enable' bits in the Link Configuration register are 0 (disabled). By defining the macro 'MMP_EN_SCRAMBLE', the default value of 'Link descramble enable' and 'Link scramble enable' bits will be 1 (enabled).

### a. Link Layer Initialization

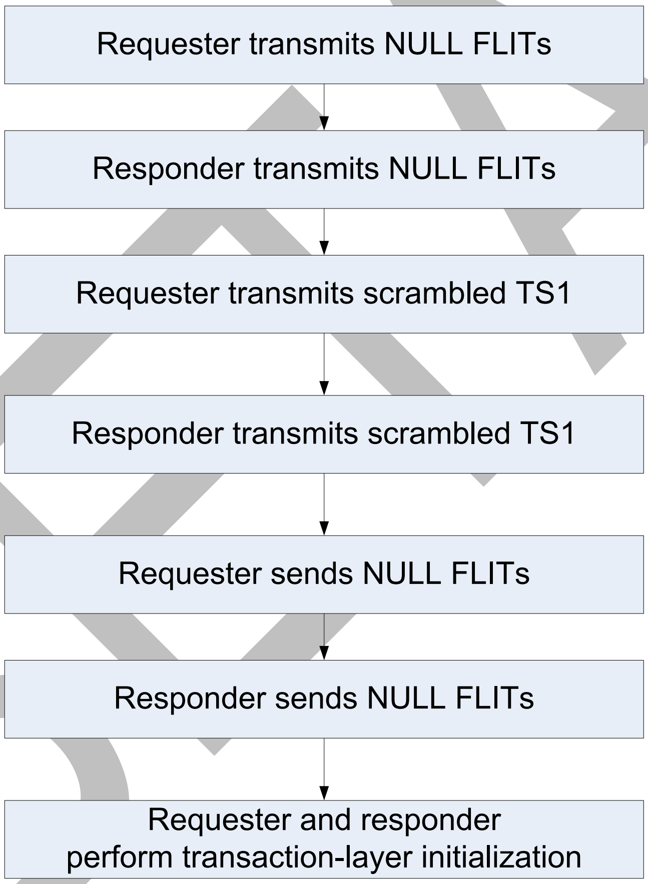
Requester transmits NULL FLITs
Responder transmits NULL FLITs
Requester transmits scrambled TS1
Responder transmits scrambled TS1
Requester sends NULL FLITs
Responder sends NULL FLITs
Requester and responder perform transaction-layer initialization

**Figure 3: Link Layer Initialization Sequence**

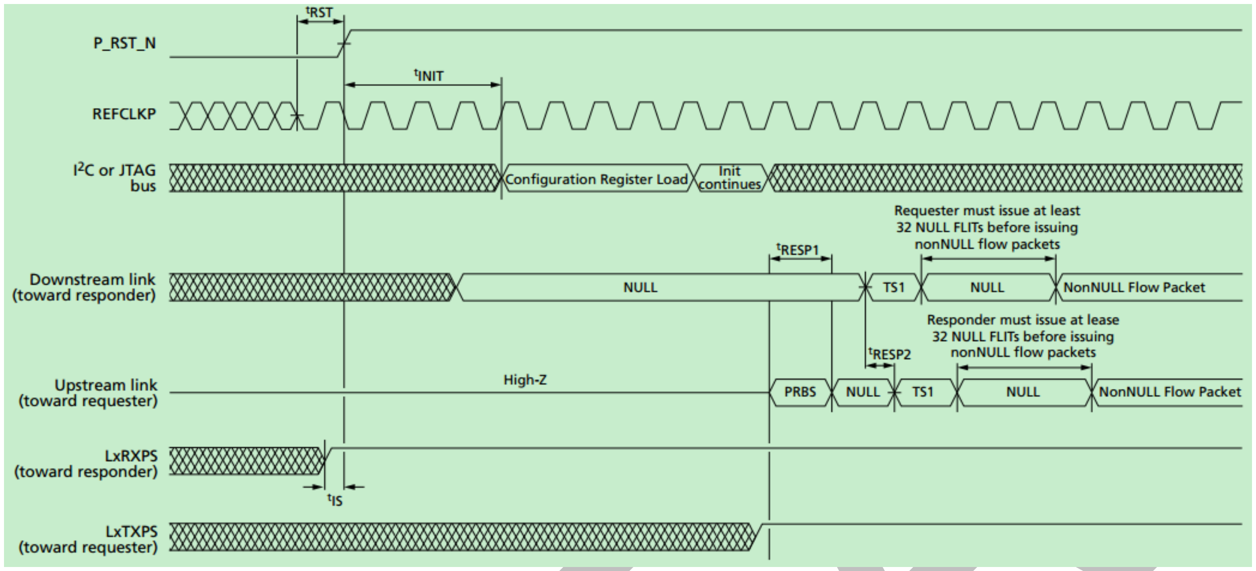
tRST
P_RST_N
tINIT
REFCLKP
I2C or JTAG bus
Configuration Register Load
Init continues
Requester must issue at least 32 NULL FLITs before issuing nonNULL flow packets
tRESP1
Downstream link (toward responder)
NULL
TS1
NULL
NonNULL Flow Packet
Responder must issue at lease 32 NULL FLITs before issuing nonNULL flow packets
tRESP2
Upstream link (toward requester)
High-Z
PRBS
NULL
TS1
NULL
NonNULL Flow Packet
LxRXPS (toward responder)
tIS
LxTXPS (toward requester)

**Figure 4: Link Layer Initialization Timing**

## b. Transaction Layer Initialization

Transaction layer initialization continues with the following steps:

1. After link layer initialization is complete, the responder will send one or more TRET packets that will transfer the number of its link input buffer tokens to the requester. Note that each TRET packet can transmit a count of 31 tokens, therefore there will be multiple TRETs required to transfer the entire number of tokens representing the link input buffer's available space.

2. Once the requester receives at least one TRET packet from the responder, the requester should transmit a series of TRET packets back to the responder carrying the tokens representing the available space in the requester's link input buffer. The HMC can support up to 1023 tokens from the host on each link.
After the TRET packets are transmitted from the requester to the responder, the requester can now start sending transaction packets. There is no required timeframe for when the transaction packets can start.

## 11. Compile and Emulation

The model is provided as protected RTL files (*.vp). These files need to be synthesized prior to the back-end Palladium compile. An example of the command for compilation (including synthesis) and run of this model in the IXCOM flow is shown below. The rtl_file_list_vp.f lists all the protected RTL files that are shown in the table Table 10: HMC Model RTL File List below. All the RTL files reside in the release folder “hmc_model”.

```
ixcom -64bit +sv –ua \
   +dut+mmp_hmc_top \
   +libext+.v+.vp \
   -incdir hmc_model \
   -f rtl_file_list_vp.f \
   ./hmc_model/mmp_hmc_top.vp \
   +define+MMP_HMC_ENABLE_HOST_LINK_0 \
   +define+ MMP_HMC_ENABLE_CUBE_0 \
   +define+ MMP_HMC_ENABLE_CUBE_1 \
   -incdir ../../../utils/cdn_mmp_utils/sv \
   ../../../utils/cdn_mmp_utils/sv/cdn_mmp_utils.sv \
   ……

xeDebug -64 --ncsim \
   -sv_lib ../../../utils/cdn_mmp_utils/lib/64bit/libMMP_utils.so -- \
   -input auto_xedebug.tcl
```

The scripts below show two examples for Palladium classic ICE synthesis:

1)
```
hdlInputFile -i ./hmc_model
hdlInputFile -f rtl_file_list_vp.f
hdlInputFile -add ./hmc_model/mmp_hmc_top.vp
+define+MMP_HMC_ENABLE_HOST_LINK_0+MMP_HMC_ENABLE_CUBE_0+MMP_HMC_ENABLE
_CUBE_1
hdlImport -full -2001 -l qtref
hdlOutputFile -add -f Verilog mmp_hmc_top.vg
hdlSynthesize –memory -keepRtlSymbol -keepAllFlipFlop mmp_hmc_top
……
```

2)
```
vavlog       -incDir ./hmc_model \
             -f rtl_file_list_vp.f \
             ./hmc_model/mmp_hmc_top.vp \
             -define MMP_HMC_ENABLE_HOST_LINK_0 \
             -define MMP_HMC_ENABLE_CUBE_0 \
             -define MMP_HMC_ENABLE_CUBE_1
vaelab -keepRtlSymbol -keepAllFlipFlop -outputVlog mmp_hmc_top.vg
mmp_hmc_top
```

**Table 10: HMC Model RTL File List**

| FILE NAME | DESCRIPTIONS |
|---|---|
| mmp_hmc_top.vp | HMC top level module |
| hmc_link.vp | The sub top level module for individual link |
| hmc_memory.vp | Memory core of HMC |
| hmc_lane_sync.vp | Operate lane synchronization |

| | |
|---|---|
| hmc_link_sync.vp | Operate link synchronization |
| hmc_pwr_mng.vp | HMC power management |
| hmc_cube_delay.vp | Delay data for individual cube and decode packets |
| hmc_link_proc.vp | Link/transaction flow. |
| hmc_statistic.vp | Operate statistic report. |

## a. Memory Load/Dump

Users can load/dump data into/from the HMC memory arrays. For example, if users create a 4-cubes chain by defining the macros “MMP_HMC_ENABLE_CUBE_0”, “MMP_HMC_ENABLE_CUBE_1”, “MMP_HMC_ENABLE_CUBE_2”, “MMP_HMC_ENABLE_CUBE_3”, there will be 4 memory arrays named “memcore_0”, “memcore_1”, “memcore_2”, and “memcore_3”. Users can load/dump data into/from each cube they want. The memcore width is 128bits.

The address mapping for each cube is {VAULT, BANK, DRAM_ADDR}. For example, if users want to preload the address --- “**vault=3, bank=4, dram_addr=5**” of a **2GB** (see address mapping of section11 in this UG) **cube[1],** they shall configure their DAT file like: “@1c00005 128bits data”, then load this file into memcore_1.

Below is a scripting example snippet showing the preloading of the dat files into the HMC model. Users can change the HMC model instance name “mmp_hmc_top” as needed and they need to create the DAT files by their own.

```
xc memory -load %readmemh mmp_hmc_top.mem_model.memcore_0 -file
user_cube0.dat
xc memory -load %readmemh mmp_hmc_top.mem_model.memcore_1 -file
user_cube1.dat
xc memory -load %readmemh mmp_hmc_top.mem_model.memcore_2 -file
user_cube2.dat
xc memory -load %readmemh mmp_hmc_top.mem_model.memcore_3 -file
user_cube3.dat
```

# 12. HMC Debugging

The HMC model is complex and therefore the associated problem of debugging any potential issues is likewise complex. Below is a list of recommended debugging techniques and tips that the user may use in isolating a problem.

- For issues that may not be HMC specific please review the *Memory Model Portfolio FAQ for All Models User Guide.*

- **Waveform debugging:** signal and sequences
  - Check that the clock and reset signals are correctly driven.
  - Check that the link layer initialization sequence finished successfully. Related signals for checking: *step1_ok, step2_ok, step3_ok, link_init_done* in module *hmc_pd.link0.link_sync*.
  - Check that the transaction layer initialization sequence finished successfully. Related signals for checking: *init_tret_start, init_tret_pulse, token_cnt_reg* in module *hmc_pd.link0.link_proc*.
  - Check normal packet receiving. Related signals for checking: *cmd_processing, proc_cmd, proc_cube, proc_tag, proc_rtc, proc_rrp, proc_frp* in module *hmc_pd.link0.link_proc*
  - Check normal packet transmitting. Related signals for checking: *drive_resp, resp_cmd, resp_tag, resp_rtc, resp_rrp, resp_frp* in module *hmc_pd.link0.link_proc*
  - Check link retry. Related signals for checking: *err_abort_mode, insert_crc_error, send_start_irtry, send_clear_irtry* in module *hmc_pd.link0.link_proc*

- **Golden waveform:** A package with a reference waveform is available which shows the following command sequence:
  - (1) synchronization with scramble/discramble disabled and link layer initialization
  - (2) transaction layer initialization by TRET packets
  - (3) normal command processing including READ, WRITE, BIT WRITE, ATOMIC WRITE, MODE REG READ/WRITE, POSTED COMMANDS, FLOW COMMANDS

- **Debug Display:** The Palladium HMC memory model has available a built-in debug methodology called MMP Debug Display that is based on the Verilog system task $display. Please see the *Palladium Memory Model Debug Display User Guide* in the release docs directory for additional information. Some of the debug information displayed when the Debug Display macro options are enabled includes:

  - HMC command sequence received
  - HMC command sequence transmitted
  - HMC flow commands sequence
  - HMC mode registers operation
  - HMC error report including CRC error, sequence number error, and others

  There is a specific Debug Display macro --- MMP_DEBUG_DISPLAY_STATISTIC, which is NOT introduced in *Palladium Memory Model Debug Display User Guide.* When this option is enabled, the following statistic information will be displayed periodically. The period is controlled by parameter STATISTIC_PERIOD.

```
/////////// IN 120000 CYCLES; WRITE 10203 FLITs(16BYTEs); READ 10944 FLITs(16BYTEs)
/////////////
****** RUN TIME -- 120000 CYCLES; STATISTIC of SOURCE LINK[0] ******
TOTAL COMMANDs:           4001
POSTED COMMANDs:          605
FLOW COMMANDs:            110
EXPECTED RESPONSEs:       3286
DEVICE SIDE ERROR ABORTs: 0
HOST SIDE ERROR ABORTs:   0
NORMAL/MODE read/write:   WRITE--480; POST_WRITE--492; MODE_WRITE--0; READ--1942;
MODE_READ--0
ARITHMETIC ATOMICs:       DUAL_8B--23; SINGLE_16B--22; POST_DUAL_8B--29;
POST_SINGLE_16B--21; RETURN_DUAL_8B--51; RETURN_SINGLE_16B--44; INCREMENT_8B--23;
POST_INCREMENT_8B--17
BOOLEAN ATOMICS:          SUM--245
COMPARISON ATOMICS:       SUM--253
BITWISE ATOMICS:          8BIT_WR--36; POST_8BIT_WR--46; RETURN_8BIT_WR--106;
SWAP_16B--61
FLOW COMMANDs:            PRET--52; TRET--58
```

## Revision History

The following table shows the revision history for this document

| Date | Version | Revision |
|---|---|---|
| July 2015 | 1.0 | Initial release |
| July 2015 | 1.1 | Update Cadence naming on front page |
| September 2015 | 1.2 | Modify compile notes to reflect *.vp as sole model format |
| January 2016 | 1.3 | Update for Palladium-Z1 and VXE |
| March 2016 | 1.4 | Update parameter definition |
| April 2016 | 1.5 | Update for HMC2.1 |
| July 2016 | 1.6 | Remove hyphen in Palladium naming |
| October 2017 | 1.7 | Update module name to be much unique |
| January 2018 | 1.8 | Modify header and footer |
| July 2018 | 1.9 | Update for new utility library |