# HMC Performance Model Specifications

Document Revision: 1.25 (hmc_gen2_r14330)

3/22/2013 12:54 PM

## REVISION HISTORY

| Authors | Rev | Description of Revision Change | Date |
|---|---|---|---|
| Junghae Lee | 1.0 | Initial release for “hmc_gen2_r3512” | 11/01/2011 |
| Junghae Lee | 1.01 | Release for “hmc_gen2_r3513”<br>-Fixed Modelsim qverilog options.<br>-Added +link_rate plusarg to simulation command<br>-Added .rst( ) argument in input file for non-posted write | 11/04/2011 |
| Junghae Lee | 1.1 | Release for “hmc_gen2_r3845”<br>- Added +args for memory address map, write posting override, link rate and request injection rate<br>-Fixed early read response from memory controller.<br>-Fixed posted writes causing early read response in link driver. | 12/27/2011 |
| Junghae Lee | 1.2 | Release for “hmc_gen2_r4959”<br>- Added support for BWR, 2ADD8, and ADD16 transactions.<br>- Added CRC generation on request and response packets.<br>- Added plusargs for seq_tags, max_reads, max_writes, max_txn.<br>- Added compatibility with VCS simulator. | 02/03/2012 |
| Junghae Lee | 1.21 | Release for “hmc_gen2_r5539”<br>- Fix for +link_rate=10 and +link_rate=12.5<br>-Added packet accumulation latency for transaction size > 32B | 02/28/2012 |
| Junghae Lee | 1.22 | -Fixed errors and added comments in documents<br>1. Version requirement for Cadence Incisive<br>2.Added default value for “+link_rate”<br>3.Added notes for “+tags” and “+max_txn” usage.<br>4.Added comments on multi-link simulation<br>5. VCS command options | 03/13/2012 |
| Junghae Lee | 1.23 | -Added descriptions on debug signals | 03/22/2012 |
| Kevin Lin | 1.24 | Release for “hmc_gen2_r9165”<br>-Align memory timing to RTL<br>- Default credit count change to align to RTL.<br>`lnk_in_flits`: Previous value 256 → Current value 100 | 6/19/2012 |
| Kevin Lin | 1.25 | Release for “hmc_gen2_r14330”<br>- Update of the link switch and vault switch arbitration algorithm to closer align with RTL | 3/22/2013 |

Contact information

| Name | Email address |
|---|---|
| Kevin Lin | Kevinlin@micron.com |

## Table of Contents

## Table of Figures



## Table of Tables

# 1 INTRODUCTION

## 1.1 Scope

This document describes the architecture, configuration, limitation and user guide for HMC_performance_model ver1.25 (hmc_gen2_r14330).

## 1.2 Simulator compatibility

Mentor Graphics: Questa/Modelsim (6.5b and later)
Cadence: Incisive (11.10-2s20 and later)
Synopsys: VCS (2011.03 and later)

## 1.3 Overview

HMC_performance_model is cycle-based behavioral system veriolog model which provides the performance measures such as memory bandwidth and latency.

The top level architecture of HMC_performance model is illustrated in Figure 1.

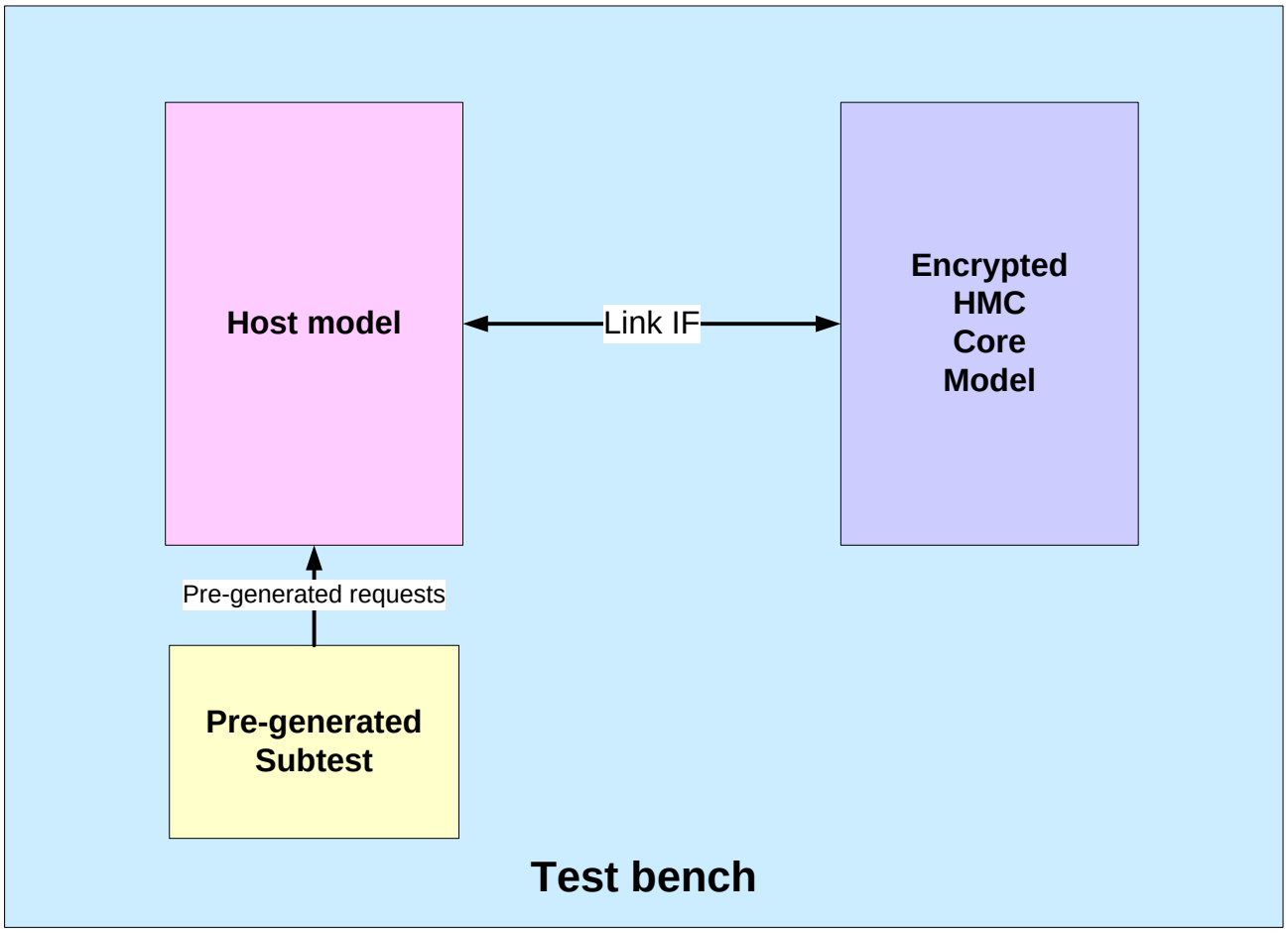


**Figure 1 Top level architecture of HMC performance model**

# 2 CONFIGURATIONS

## 2.1 Link lane speed

10, 12.5 and 15Gbps

## 2.2 HMC cube size

2GB or 4GB

## 2.3 Block size

32, 64 and 128 Bytes

### 2.4 Data size

The model supports READ and WRITE data block accesses with 16-byte granularity from 16B to the value of the maximum block size setting (128B in this version).

### 2.5 Address

**Vault address**: 0x0 – 0x3 for Link0, 0x4-0x7 for Link1, 0x8-0xb for Link2 and 0xc-0xf for Link3
**Bank address:** 0x0 – 0x7 for 2GB cube, 0x0 – 0xF for 4GB cube
**DRAM address:** 0x0 – 0xFFFFF. Note that address will wrap around if "DRAM_address + data_size" crosses the block_size boundary.

## 3 LIMITATIONS

### 3.1 Features not supported

#### 3.1.1 DRAM repair

DRAM repair function is not implemented in the performance model. Performance impact due to this limitation is expected to be negligible.

#### 3.1.2 Link retry

Performance measure for Link retry is not supported in this version. Performance impact on memory bandwidth in case of link retry is expected to be negligible. The worst case latency may be affected by link retry.

#### 3.1.3 ERRSTAT error status for responses

Performance impact in case of error is not supported in this version.

#### 3.1.4 Power-on and initialization sequencing

Link starts running after deassertion of reset. No performance impact is expected due to this limitation.

#### 3.1.5 Lane reversal and polarity

No performance impact is expected due to this limitation.

#### 3.1.6 Mode read and mode write command

No performance impact is expected due to this limitation.

#### 3.1.7 JTAG Interface

No performance impact is expected due to this limitation.

# 4 SIMULATION STEPS

## 4.1 Getting started

1. Unzip the included files to a folder.
   - unzip hmc_gen2_r14330.zip
2. Change the working directory to the top level folder of the simulator user wants to run.
   - cd hmc_gen2_r14330
3. According to user's simulation environment, compile the testbench and source codes using
   - hmc_gen2_cdn.f -- for Cadence
   - hmc_gen2_mgc.f – for ModelSim
   - hmc_gne2_vcs.f – for VCS
4. Simulate testbench with command line options
   - Include the directory where subtest.svh (input_file) is located using the +incdir+ switch.
   - Set simulation parameters using "+plusarg=" based on the system configuration to simulate.

**Table 1 Simulation parameters**

| Simulation parameters | Description | Supported values |
|---|---|---|
| +address_mode | Both host_model and HMC uses this argument to determine the address mapping used in encoding and decoding of 32-bit address field in Link packet header. Also, it determins the address map mode in HMC. Refer to "table8 Default Address Map Mode Table" in section 9.1.3 of "Gen2 Advance Customer Datasheet Rev2.1" for detailed definition of address mapping.<br>For example, "4GB_128B" means "4GB HMC device with 128B block size setting".<br>Note that user must apply the proper constraint on bank address range according to HMC size setting. For example, if user set "+address_mode=2GB_128B" and varies bank address between "0x0-0xf", bank_address[3] bit will be ignored in host model and will not be encoded in 32-bit address filed in Link packet header. | 2GB_32B<br>2GB_64B<br>2GB_128B<br>4GB_32B<br>4GB_64B<br>**4GB_128B:default** |
| +tags | +tags controls the max number of tags used by the host model.* | 1: min<br>**512: max, default** |
| +posted_wr | +posted_wr controls how the host model treats the response field during write | **0: transaction response field** |

| | | |
|---|---|---|
| | requests | **controls write posting, default**<br>1: forces every write to be posted |
| +seq_tags | +seq_tags controls the order of tags issued by the host model.<br>Figure 2 and Figure 3 illustrate the difference between two options.<br>In general, "+seq_tags=1" is suitable for the application where "in-order response" is required and as a result "re-ordering logic" is required on host side. In this case, "+tags=" represents "in-order buffer depth" on host side.<br>"+seq_tags=0" is suitable for the application where requests come from multiple sources (multi cores/threads) and as a result "in-order" response is not required. In this case, "+max_txn=" represents "number of outstanding requests" from host perspective. If posted write is set, it represents number of outstanding read requests in HMC since posted write is retired immediately after issueing. If posted write is disabled, it represents number of total outstanding requests in HMC.<br>In most cases, host doesn't care how many write requests exist in HMC pipeline. In this case, user should use "+max_reads" instead of "+max_txn" to reflect the system limitation on host side.<br>**Examples of usage>**<br>**1. Networking packet buffer: +seq_tags=1 posted_wr=1 +tags=256**<br>**2. High Performance CPU application: +seq_tags=0 +posted_wr=1 +max_reads=32** | **0: tags can be issued out of order, default**<br>1: tags must be issued sequentially. |
| +max_txn | +max_txn controls the max number of outstanding requests issued by the host model.<br>Note that host model assumes "immediate retirement" on posted write after it issues posted write. Hence, real number of outstanding requests in HMC pipeline can | 1: min<br>**512: max, default** |

| | | |
|---|---|---|
| | be larger than "max_txn" count. However, from host perspective, it does not have more than +max_txn responses to expect.* | |
| +max_reads | +max_reads controls the max number of outstanding read requests issued by the host model | 1: min<br>**512: max,default** |
| +max_writes | +max_writes controls the max number of outstanding write requests issued by the host model | 1:min<br>**512:max,default** |
| +ck_mhz | +ck_mhz controls the refrence clock input frequency | 50<br>100<br>**125: default**<br>250<br>500 |
| +link_rate | +link_rate controls the link bit rate @125MHz fREFCLK | 10: 10Gbps<br>12.5: 12.5Gbps<br>**15 :15Gbps,default** |
| +lnk_in_flits | +lnk_in_flits controls number of tokens returned during initialization | 9: min<br>231: max<br>**100: default** |

Usage Note*:
<"+tags" Vs "+max_txn">
Both "+tags" and "+max_txn" can be used to limit the number of outstanding requests.

"+tags" controls the max number of tags used by the host model.
If user sets "+tags=64" for example, host model will generate request with tag index from 1-63 and wrap around to 1 once it reaches 63. If you set "+seq_tags=1" together with "+tags=64", host model stops issuing request after request63 (request with tag_index=63) is sent out and waits until response1 arrives. Once it gets reponse1, it sends out request1. Note that host model retires posted_write_request immediately after issuing it.
This setting is suitable for the case where host has specific size of in-order buffer and needs to do re-ordering on host size to guarantee the in-order response to clients. In this case, you can set "+seq_tags=1 +posted_wr=1 +tags=in-order-buffer-depth" to model the expected behavior most closely.

"+max_txn" controls the max number of outstanding requests issued by the host model. The main difference is it doesn't affect the tag_index generation. The reason "+tags" is used in the above example is because user can model the particular behavior regarding in-order-buffer using "+seq_tags and +tags" together. If user uses "+max_txn" instead of "+tags" in the above example, host model will keep on generating requests after request63 as long as some of requests were returned already although request1 has not been returned yet. "+max_txn" really limits the number of outstanding requests in HMC pipeline except posted_write_requests. Again, note that host model retires posted_write_request immediately after issuing it.

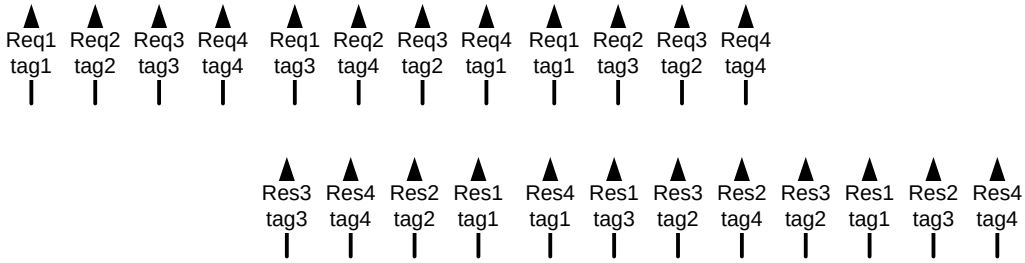


**Figure 2 Reqeust pattern with "+seq_tags=0"**

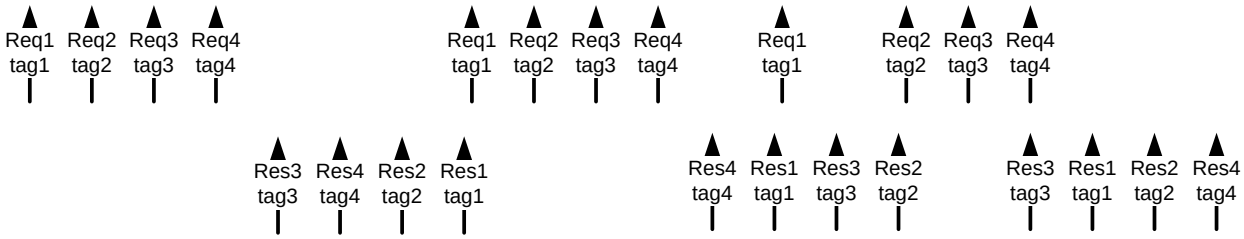


**Figure 3 Request pattern with "+seq_tags=1"**

***Example> Note that command is marked in blue font.***
***ModelSim:***
***qverilog -f hmc_gen2_mgc.f +incdir+../tests/QuadR050I000_64B/ -R +seq_tags=<value> +max_txn=<value> +link_rate=<value> +posted_wr=<value> +address_mode=<value> -novopt***

***Incisive:***
***irun -f hmc_gen2_cdn.f +incdir+../tests/QuadR050I000_64B/ +seq_tags=<value> +max_txn=<value> +link_rate=<value> +posted_wr=<value> +address_mode=<value>***

***VCS:***
***vcs -f hmc_gen2_vcs.f +incdir+../tests/QuadR050I000/ -sverilog***
***simv +seq_tags=<value> +max_txn=<value> +link_rate=<value> +posted_wr=<value> +address_mode=<value>***

## 4.2 Directory hierarchy

**/hmc_gen2_r14330**: Root directory. Includes:
hmc_gen2_cdn.f -- the filelist for Cadence simulation.

hmc_gen2_mgc.f -- the filelist for ModelSim simulation.
hmc_gen2_vcs.f -- the filelist for VCS simulation.
**/hmc_gen2_r14330 /test**: Directory for test files. Refer to 5.1 for detailed descriptions on test files.
**/hmc_gen2_r14330 /doc**: Directory for document. Include HMC_performance_model.pdf.
**/hmc_gen2_r14330 /src**: Directory for (unencrypted) commen source codes and (encrypted) environment specific source codes for Cadence, ModelSim and VCS.
**/mgc**: Directory for Questa/modelsim environment.
**/cdn**: Directory for Incisive environment.
**/ vcs**: Directory for VCS environment

## 4.3 Input file (subtest.svh) format

Subtest.svh is pre-generated iput file to HMC performance model. It defines the sequence of requests via **arr_hm[0].req_que.push_back( )** method.

Users can define their own sequence of requests by modifying and/or adding the below part of subtest.svh file.

**data = '{'hd4713d60, 'h4da5e709, 'hf7c1bd87, 'h7a024204, 'h5ba91faf, 'h9558867f, 'he443df78, 'he87a1613};**
**mask = '{'h0, 'h0, 'h0, 'h0, 'h0, 'h0, 'h0, 'h0};**
**txn=new(); void'(txn.new_txn(.id('{unq:'h0, lnk:'h0, tag:'h0}), .cmd(enu_write), .adrs('{vlt:'h2, bnk:'h2, dram:'h2f190}), .dbytes(32), .lat(0), .nop(0), .data(data), .dmask(mask), .rsp(0)));**
**arr_hm[0].req_que.push_back(txn);**

1. **data:** Write data. Not required for read command.
2. **mask:** Mask for write data. Not required for read command.
3. **txn:** It defines command and address information for each request
    a. **id**
        i. **unq**: identification for this request. Recommendation is increasing this argument sequentially up to 0x1FF and then wrap around to 0. Mainly for debugging.
        ii. **lnk**: link index from 0 to 3.
        iii. **tag**: tag number which will be automatically generated in host model later on. Must be fixed to “0” in subtest.svh file.
    b. **cmd**:
    enu_write: Write request
    enu_read: Read request
    enu_bwrite: BIT write request
    enu_2add8: Atomic request, 2ADD8
    enu_add16: Atomic request, ADD16
    c. **adrs:** Refer to “2GB/4GB Hybrid Memory Cube – Gen2” for detailed definition of each address field.
        i. **vlt**: Vault address (0x0 – 0x3)
        ii. **bnk**: Bank address (0x0 – 0x7 for 2GB cube, 0x0 – 0xF for 4GB cube)

iii. **dram**: DRAM address (0x0 – 0xFFFFF)

d. **dbytes**: data size (unit is byte)
e. **lat**: Must be fixed to "0"
f. **nop**: This field defines the number of null FLITs before sending request. For example, if user set ".nop(5)", host model will wait for 5 FLITs on link before sending the request.
g. **rsp**: User can specify "poster write" per individual request using this filed. "0" means "posted write" and "1" means "non-posted write". If neither this field nor "+posted_wr" is specified, write response is returned by default.

If user doesn't want to specify the data for individual request, he must set the mask field to 0xffffffff at the beginning of file. Then, user doesn't have to specify data and mask field for each write request.

If user wants to enable multiple links, he must specify link_id, in .lnk field and arr_hm[ ]. Please, refer to the following example.

```
txn=new(); void'(txn.new_txn(.id('{unq:'h1, lnk:'h2, tag:'h0}), .cmd(enu_write), .adrs('{vlt:'hc,
bnk:'h6, dram:'h72a8}), .dbytes('h50), .lat(0), .nop(0), .data(data), .dmask(mask)));
arr_hm[2].req_que.push_back(txn);
txn=new(); void'(txn.new_txn(.id('{unq:'h2, lnk:'h1, tag:'h0}), .cmd(enu_read), .adrs('{vlt:'h9,
bnk:'h1, dram:'h5af0}), .dbytes('h40), .lat(0), .nop(0), .data(data), .dmask(mask)));
arr_hm[1].req_que.push_back(txn);
```

Two examples are described below.

```
---------- Start: Example ----------
data = '{'hd4713d60, 'h4da5e709, 'hf7c1bd87, 'h7a024204, 'h5ba91faf, 'h9558867f, 'he443df78,
'he87a1613, 'h37ebdcd9, 'h81332876, 'h23a7711a, 'h48268673, 'h23c6612f, 'hc17c6279,
'h1846d424, 'h9e4d6e3c, 'hcca5a5a1, 'h40212ef7, 'hfcbd04c3, 'he8e5216a, 'h88561712,
'hfb97d435, 'hb4862b21, 'hcf6a659e, 'h9a164106, 'he6f4590b, 'h259f4329, 'h4f65d4d9,
'h19488dec, 'hbad640fb, 'h12e0c8b2, 'he61a441c};
mask = '{'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff,
'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff,
'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff, 'hffffffff,
'hffffffff, 'hffffffff, 'hffffffff};
txn=new(); void'(txn.new_txn(.id('{unq:'h0, lnk:'h0, tag:'h0}), .cmd(enu_write), .adrs('{vlt:'h2,
bnk:'h2, dram:'h2f190}), .dbytes(128), .lat(0), .nop(0), .data(data), .dmask(mask)));
arr_hm[0].req_que.push_back(txn);
txn=new(); void'(txn.new_txn(.id('{unq:'h1, lnk:'h0, tag:'h0}), .cmd(enu_write), .adrs('{vlt:'h1,
bnk:'h2, dram:'h5218}), .dbytes(128), .lat(0), .nop(0), .data(data), .dmask(mask)));
txn=new(); void'(txn.new_txn(.id('{unq:'h2, lnk:'h0, tag:'h0}), .cmd(enu_write), .adrs('{vlt:'h3,
bnk:'h6, dram:'h60010}), .dbytes(128), .lat(0), .nop(0), .data(data), .dmask(mask)));
arr_hm[0].req_que.push_back(txn);
---------- End: Example ----------
```

Note that user must include the directory where subtest.svh is located using "+incdir+" switch during compilation.

## 4.4 Source file list

File list for compile order is specified in "hmc_gen2_xxx.f". "xxx" for "cdn", "mgc" or "vcs"

The detailed information on each file is described in Table 2 .

**Table 2 List of source file**

| File name | Encyption | Description |
|---|---|---|
| Pkg_cad.svp | Yes | Defines basic command/address/data class and queues to store them. |
| Pkg_cke.sv | No | Package that defines a base class and clock counter. Implements the concept of clocks and latency. |
| Pkt.svp | Yes | Defines packet classes for HMC link interface. |
| nextCRC32_D128 | No | Includes CRC-32 function |
| Pkg_vif.sv | No | Defines link interface. |
| Pkg_seq.sv | No | Defines sequencer class for sending and receiving transaction objects. |
| Pkg_drv.sv | No | Defines driver class for moving transaction objects to/from HMC link interface. |
| Pkg_mem.svp | Yes | Defines memory controller class, memory array, and memory timings. |
| Pkg_sw.svp | Yes | Defines a switch class, with scalable number of inputs and outputs. |
| Hmc_gen2.svp | Yes | Hybrid Memory Cube Gen2 Model. |
| Hmc_gen2_tb.sv | No | Tesbench for Hybrid Memory Cube Gen2 Model. Connects a host model running HMC Gen2 Link protocol. Instantiates a Device Under Test (hmc_dut). |

## 4.5 Debug signals

Following debug signals are available in test bench. (hmc_gen2_tb.sv)

```
// debug signals
bit  [0:par_hms-1][31:0] tx_credits;
wire [0:par_hms-1][31:0] lnksw_credits = hmc_dut.lnksw_credits;
wire [0:16-1][31:0] swvlt_credits = hmc_dut.swvlt_credits;
wire [0:16-1][31:0] vltsw_credits = hmc_dut.vltsw_credits;
wire [0:par_hms-1][31:0] swlnk_credits = hmc_dut.swlnk_credits;
```

| Debug signal name | Descriptions |
|---|---|
| Tx_credits[0:3] | This is credit number which is shown to host directly via token return on |

| | |
|---|---|
| | Link IF. It reflects the buffer status of link input buffer (currently 128FLITs but planned to be increased to 256 FLITs)<br>Tx_credit[N] corresponds to link_N. |
| Lnksw_credits[0:3] | This credit number shows the buffer status of link input buffer. However, there is delay between tx_credit and link_to_switch_credit due to internal pipeline. As a result, tx_credit stays at lower value than link_to_switch_credit.<br>Lnksw_credit[N] corresponds to link_N. |
| Swvlt_credits[0:15] | This credit number reflects "16 command credits + 64 write data credit" in each vault controller.<br>Swvlt_credits[N] corresponds to vault_controller_N. |
| Vltsw_credits[0:15] | This credit number reflects the buffer status of vault response buffer in vault switch. |
| Swlnk_credits | This credit number reflects the buffer status of link output buffer. |

# 5 SAMPLE TEST AND RESULTS

## 5.1 Included tests

The tests directory contains examples of 32B, 64B, 96B and 128B transaction sizes.
There is an example test for each transfer size at 0%, 50% and 100% Reads.

- 2GB or 4GB HMC cube size
- Random addressing.
- No idle time between transactions.
- The test name indicates the type of transactions in the test.

For example: **4GB_QuadR050I000_64B_16Vaults_10000**
**4GB** indicates HMC cube size
**Quad** indicates sinlge Link IF is enabled
**R050** indicates 50% read transactions
**I000** indicates 0% idle time between transactions
**64B** indicates 64B transactions
**16Vaults** indicates that requests access all 16 Vuatls. If not specified, requests access 4 local Vaults only.
**10000** indicates 10000 requests

Another sample test is "1ofeach". It includes one request of each command type.

## 5.2 Result for the selected tests

This section includes the simulation results for the selected tests. User can use these simulation results as reference to validate the simulator set-up.

**Table 3 1ofeach**

+seq_tags=1 / +posted_wr=1 / +tags=256 (Networking example)

| Model revision | Memory Bandwidth (GB/s) | Read ratio (%) | Average latency (ns) | Max latency (ns) | Min latency (ns) |
|---|---|---|---|---|---|
| R14330 | 3.26 | 50 | 285.4 | 639.0 | 113.1 |

**Table 4 4GB_QuadR050I000_128B_10000**

+seq_tags=1 / +posted_wr=1 / +tags=256 (Networking example)

| Model revision | Memory Bandwidth (GB/s) | Read ratio (%) | Average latency (ns) | Max latency (ns) | Min latency (ns) |
|---|---|---|---|---|---|
| R14330 | 36.5 | 50 | 222.1 | 868.9 | 60.3 |

**Table 5 2GB_QuadR050I000_128B_1000**

+seq_tags=1 / +posted_wr=1 / +tags=256 (Networking example)

| Model revision | Memory Bandwidth (GB/s) | Read ratio (%) | Average latency (ns) | Max latency (ns) | Min latency (ns) |
|---|---|---|---|---|---|
| R14330 | 34.38 | 50 | 211.3 | 815.5 | 61.4 |

**Table 6 4GB_QuadR050I000_16Vaults_64B_40000**

+seq_tags=0 / +posted_wr=1 / +max_reads=32 (HPC example)

| Model revision | Memory Bandwidth (GB/s) | Read ratio (%) | Average latency (ns) | Max latency (ns) | Min latency (ns) |
|---|---|---|---|---|---|
| R14330 | 38.77 | 50 | 89.9 | 306.7 | 57.1 |

**Table 7 4GB_QuadR050I000_16Vaults_128B_40000**

+seq_tags=0 / +posted_wr=1 / +max_reads=32 (HPC example)

| Model revision | Memory Bandwidth (GB/s) | Read ratio (%) | Average latency (ns) | Max latency (ns) | Min latency (ns) |
|---|---|---|---|---|---|
| R14330 | 46.21 | 50 | 95.9 | 338.2 | 60.3 |