Hybrid Memory Cube Specification 1.0

©2013 Altera Corporation, ARM Ltd., Hewlett-Packard Company, International Business Machines Corporation, Micron Technology, Inc., Open-Silicon, Inc., Samsung Electronics Co., Ltd., SK Hynix, Inc., and Xilinx, Inc.
All rights reserved. Hybrid Memory Cube Consortium

Hybrid Memory Cube

HMC Memory Features

- Configured with independent memory vaults
- Built-in memory controller for each vault
  - Automatic refresh control over all temperatures
- Internal ECC data correction
- Advanced RAS features including data scrubbing
- Post-assembly repair capability
- In-field repair for ultimate reliability

HMC Interface Features

- 10 Gbps, 12.5 Gbps, or 15 Gbps SerDes I/O interface
- Multiple 16-lane, full-duplex serialized links
  - Half-width link (8-lane) configuration also supported
- Packet-based data/command interface
- Supports 16, 32, 48, 64, 80, 96, 112, and 128 byte references per request
- Error detection (cyclic redundancy check [CRC]) for packets with automatic retry
- Power management supported per link
- Through-silicon via (TSV) technology
- Built-in self-test (BIST)
- JTAG interface (IEEE 1149.1-2001, 1149.6)

Description

A hybrid memory cube (HMC) is a single package containing multiple memory die and one logic die, all stacked together using through-silicon via (TSV) technology.

Scope of Specification

- Section 1: HMC Architecture Overview
- Sections 2-16: Common Protocol
- Section 17-19: Electrical and Physical Link Specifications for HMC-15G-SR (short reach)
- Section 20-21: Electrical and Physical Link Specifications for HMC-10G-USR (ultra short reach)

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.
©2012 Altera Corporation, ARM Ltd., Hewlett-Packard Company, International Business Machines Corporation, Micron Technology, Inc., Open-Silicon, Inc., Samsung Electronics Co., Ltd., SK Hynix, Inc., and Xilinx, Inc. All rights reserved.
Hybrid Memory Cube Consortium

Table of Contents

1 HMC Architecture 8
1.1 HMC Architecture 8
1.2 Logic Base Architecture 9
2 Pin Descriptions 11
3 Link Data Transmission 13
4 Logical Sub-Block of Physical Layer 14
4.1 Link Serialization 14
4.2 Scrambling and Descrambling 15
4.3 Lane Run Length Limitation 17
4.4 Lane Reversal 18
4.5 Lane Polarity 18
5 Chaining 19
6 Power-On and Initialization 21
7 Power State Management 24
8 Link Layer 27
9 Transaction Layer 28
9.1 Memory Addressing 29
9.1.1 Memory Addressing Granularity 30
9.1.2 Memory Address-to-Link Mapping 30
9.1.3 Default Address-Map Mode Table – 4-link Devices 32
9.1.4 Default Address-Map Mode Table – 8-link Devices 33
9.1.5 Address Mapping Mode Register 34
9.1.6 DRAM Addressing 34
9.2 Packet Length 34
9.3 Packet Flow Control 35
9.4 Tagging 35
9.5 Packet Integrity 36
9.6 Request Packets 37
9.6.1 Request Header Layout 37
9.6.2 Request Tail Layout 38
9.7 Response Packets 38
9.8 Flow Packets 43
9.9 Poisoned Packets 43
9.10 Request Commands 44
9.10.1 READ and WRITE Request Commands 45
9.10.2 POSTED WRITE Request Commands 45
9.10.3 ATOMIC Request Commands 45
9.10.3.1 Dual 8-Byte Add Immediate 46
9.10.3.2 Single 16-Byte Add Immediate 46
9.10.4 MODE READ and WRITE Request Commands 47
9.10.5 BIT WRITE Command 47
9.11 Response Commands 48
9.11.1 READ and MODE READ Response Command 48
9.11.2 WRITE and MODE WRITE Response Command 49
9.11.3 ERROR Response Command 49
9.12 Flow Commands 49
9.12.1 NULL Command 50
9.12.2 RETRY POINTER RETURN (PRET) Command 50
9.12.3 TOKEN RETURN (TRET) Command 50
9.12.4 INIT RETRY (IRTRY) Command 50
9.13 Valid Field Command Summary 50

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.14 Transaction Layer Initialization . . . 51
10 Configuration and Status Registers . . . 52
11 Link Retry . . . 61
11.1 Retry Pointer Description . . . 62
11.2 Link Master Retry Functions . . . 63
11.2.1 Forward Retry Pointer Generation . . . 64
11.2.2 Packet Sequence Number Generation . . . 64
11.2.3 Forward Retry Pointer Reception and Embedding . . . 65
11.2.4 Return Retry Pointer Reception . . . 65
11.2.5 Link Master Sequences . . . 65
11.2.5.1 StartRetry Sequence . . . 65
11.2.5.2 LinkRetry Sequence . . . 67
11.3 Link Slave Retry Functions . . . 69
11.3.1 Packet CRC/Sequence Check . . . 69
11.3.1.1 FRP and RRP Extraction . . . 69
11.3.2 Error Abort Mode . . . 69
11.3.3 IRTRY Packet Operation . . . 69
11.3.4 Resumption of Normal Packet Stream after the Retry Sequence . . . 71
11.4 Retry Pointer Loop Time . . . 71
11.5 Link Flow Control During Retry . . . 72
11.6 Example Implementation Link Error and Retry Sequence . . . 72
12 Warm Reset . . . 74
13 Retraining . . . 74
13.1 Retraining a Link with High Error Rate . . . 74
13.2 Host Recovery after Link Retry Fails . . . 75
14 Functional Characteristics . . . 76
14.1 Packet Ordering and Data Consistency . . . 76
14.2 Data Access Performance Considerations . . . 76
14.3 Vault ECC and Reference Error Detection . . . 77
14.4 Refresh . . . 77
14.5 Scrubbing . . . 77
14.6 Response Open Loop Mode . . . 78
15 JTAG Interface . . . 79
15.1 Disabling the JTAG Feature . . . 79
15.2 Test Access Port (TAP) . . . 79
15.2.1 Test Clock (TCK) . . . 79
15.2.2 Test Mode Select (TMS) . . . 79
15.2.3 Test Reset (TRST_N) . . . 79
15.2.4 Test Data-In (TDI) . . . 79
15.2.5 Test Data-Out (TDO) . . . 80
15.3 TAP Controller . . . 80
15.3.1 Test-Logic-Reset . . . 80
15.3.2 Run-Test/Idle . . . 80
15.3.3 Select-DR-Scan . . . 80
15.3.4 Capture-DR . . . 80
15.3.5 Shift-DR . . . 80
15.3.6 Exit1-DR, Pause-DR, and Exit2-DR . . . 80
15.3.7 Update-DR . . . 80
15.3.8 Instruction Register States . . . 80
15.4 Performing a TAP RESET . . . 81
15.5 TAP Registers . . . 81
15.5.1 Instruction Register . . . 82
15.5.2 Bypass Register . . . 82

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

15.5.3 Identification (ID) Register . . . 82
15.5.4 Boundary Scan Register . . . 82
15.5.5 CADATA Register . . . 82
15.6 TAP Instruction Set . . . 82
15.6.1 Overview . . . 82
15.6.2 EXTEST . . . 83
15.6.3 EXTEST_PULSE & EXTEST_TRAIN . . . 83
15.6.4 HIGH-Z . . . 83
15.6.5 CLAMP . . . 83
15.6.6 SAMPLE/PRELOAD . . . 83
15.6.7 IDCODE . . . 83
15.6.8 BYPASS . . . 83
15.6.9 CFG_RDA . . . 84
15.6.10 CFG_WRA . . . 84
15.6.11 Reserved for Future Use . . . 84
15.7 JTAG Configuration Register Write and Status Register Read . . . 84
16 I2C Interface . . . 89
17 HMC-15G-SR Electrical Specifications . . . 92
17.1 Absolute Maximum Ratings . . . 92
17.2 DC Operating Conditions . . . 93
18 HMC-15G-SR Physical Link Specifications . . . 95
18.1 Physical Link Electrical Interface . . . 95
18.1.1 Termination Configurations . . . 95
18.2 Equalization Schemes . . . 98
18.3 Link Bit Rate . . . 98
18.4 High Speed Signaling Parameters . . . 99
18.5 Non-High Speed Link Parameters . . . 102
18.6 Impedance Calibration . . . 104
19 Packages for HMC-15G-SR Devices . . . 105
20 HMC-10G-USR Electrical Specifications . . . 107
20.1 Absolute Maximum Ratings . . . 107
20.2 DC Operating Conditions . . . 108
21 HMC-10G-USR Physical Link Specifications . . . 109
21.1 Physical Link Electrical Interface . . . 109
21.1.1 Termination Configuration . . . 109
21.2 Link Bit Rate . . . 111
21.3 High Speed Signaling Parameters . . . 111
21.3.1 HMC-USR TX Signaling Specifications . . . 111
21.3.1.1 HMC-USR TX Return Loss . . . 111
21.3.1.2 HMC-USR TX Jitter . . . 111
21.3.2 HMC-USR Supported Channels . . . 113
21.3.3 HMC-USR RX Signaling Specifications . . . 115
21.3.3.1 Jitter Model . . . 116
21.4 Non-High Speed Link Parameters . . . 117
21.4.1 PLL Reference Clock . . . 118
22 Appendix A: Glossary of Terms . . . 120
23 Revision History . . . 121

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

List of Figures

Figure 1: Example HMC Organization 8
Figure 2: HMC Block Diagram Example Implementation (4-link HMC configuration)....9
Figure 3: Link Data Transmission Implementation Example 13
Figure 4: Scrambler and Descrambler Paths from Requester to Responder 15
Figure 5: Scrambler Logic 16
Figure 6: Example of Chaining Topology 19
Figure 7: HMC Initialization Flowchart 22
Figure 8: Initialization Timing 23
Figure 9: Sleep Mode Entry and Exit (Single Link Only) 25
Figure 10: Simultaneous Transition of Four Host Links to Sleep Mode, Entry into Down Mode and Return to Active Mode (Single HMC, Four Link Example) 26
Figure 11: Packet Layouts....28
Figure 12: Request Packet Header Layout 37
Figure 13: Request Packet Tail Layout....38
Figure 14: Response Packet Header Layout....38
Figure 15: Response Packet Tail Layout 39
Figure 16: Configuration and Status Register Access....53
Figure 17: Implementation Example of Link Retry Block Diagram 62
Figure 18: Implementation Example of a Retry Buffer 63
Figure 19: Warm Reset Flow Chart....74
Figure 20: Host Recovery after Link Retry Fails....75
Figure 21: TAP Controller State Diagram 81
Figure 22: TAP Controller Block Diagram....81
Figure 23: JTAG Configuration Register Write Flow Chart 84
Figure 24: JTAG Status Register Read Flow Chart....85
Figure 25: JTAG Operation – Loading Instruction Code and Shifting Out Data....86
Figure 26: TAP Timing....87
Figure 27: I^2C Block Diagram 89
Figure 28: I^2C State Diagram 90
Figure 29: I^2C Register Accessibility....90
Figure 30: I2C Packet Protocol Key 91
Figure 31: Configuration Register WRITE and READ Commands....91
Figure 32: VDDPLLX Filtering Requirement 94
Figure 33: HMC Tx Driving Host Rx - Example #1 95
Figure 34: HMC Tx Driving Host Rx - Example #2 96
Figure 35: HMC Tx Driving Host Rx with External AC Coupling Present 96
Figure 36: Host Tx Driving HMC Rx Without External AC Coupling 97
Figure 37: Host Tx Driving HMC Rx With External AC Coupling....97
Figure 38: Receiver Sinusoidal Jitter Tolerance.... 102
Figure 39: HMC Package Drawing (4-Link HMC-15G-SR Device) 105
Figure 40: HMC Ballout for 4-link HMC-15G-SR 106
Figure 41: HMC-USR Tx Driving Host Rx.... 110
Figure 42: Host Tx Driving HMC-USR Rx.... 110
Figure 43: Example HMC-USR Implementation.... 113
Figure 44: Example Insertion Loss Plots for Recommended HMC-10G-USR Applications 114
Figure 45: Illustration of Determining Insertion Loss Deviation (ILDhb).... 115

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

List of Tables

Table 1: HMC Configurations 10
Table 2: Pin Descriptions 11
Table 3: Unit Interval FLIT Bit Positions for Full-Width Configuration 14
Table 4: Unit Interval FLIT Bit Positions for Half-Width Configuration 14
Table 5: Scrambler Logic Seed Values 17
Table 6: TS1 Definition 17
Table 7: TS1 Training Sequence Example 17
Table 8: Link Power States and Conditions 25
Table 9: Addressing Definitions 31
Table 10: Default Address Map Mode Table 32
Table 11: Default Address Map Mode Table 33
Table 12: Request Packet Header Fields 37
Table 13: Request Packet Tail Fields 38
Table 14: Response Packet Header Fields 39
Table 15: Response Packet Tail Fields 39
Table 16: ERRSTAT[6:0] Bit Definitions 40
Table 17: Transaction Layer – Request Commands 44
Table 18: Operands from DRAM 46
Table 19: Immediate Operands Included in Atomic Request Data Payload 46
Table 20: Operand from DRAM 46
Table 21: Immediate Operand Included in Atomic Request Data Payload 46
Table 22: Mode Request Addressing 47
Table 23: Valid Data Bytes 47
Table 24: Request Packet Bytes to be Written 48
Table 25: Transaction Layer – Response Commands 48
Table 26: Flow Commands 50
Table 27: Valid Field and Command Summary Table 50
Table 28: External Data Register 0 54
Table 29: External Data Register 1 54
Table 30: External Data Register 2 54
Table 31: External Data Register 3 54
Table 32: External Request Register 54
Table 33: Global Configuration 56
Table 34: Link Configuration 56
Table 35: Link Run Length Limit 56
Table 36: Link Retry 57
Table 37: Input Buffer Token Count 57
Table 38: Address Configuration 58
Table 39: Vault Control Register 58
Table 40: Features 59
Table 41: Revisions and Vendor ID 60
Table 42: Retry Pointer Loop Time Implementation Example 71
Table 43: CADATA Register 82
Table 44: JTAG Voltage and Timing Parameters 87
Table 45: Identification Register Definitions 87
Table 46: Scan Register Sizes 88
Table 47: Instruction Codes 88
Table 48: Absolute Maximum Ratings 92
Table 49: Maximum Current Conditions 92
Table 50: DC Electrical Characteristics 93
Table 51: Synchronous Link Bit Rate Specifications 98

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 52: TX Signaling Parameters 99
Table 53: RX Signaling Parameters 101
Table 54: Initialization Timing Parameters 102
Table 55: Link Power Management Parameters 103
Table 56: Reference Clock Parameters 103
Table 57: Absolute Maximum Ratings 107
Table 58: Maximum Current Conditions 107
Table 59: DC Electrical Characteristics 108
Table 60: Source Synchronous Link Bit Rate Specification 111
Table 61: HMC-USR TX Signaling Parameters 112
Table 62: HMC-USR Channel Parameters 115
Table 63: HMC-USR RX Signaling Parameters 116
Table 64: Initialization Timing Parameters 117
Table 65: Link Power Management Parameters 117
Table 66: Reference Clock Parameters 119
Table 67: Glossary of Terms 120

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

1 HMC Architecture

1.1 HMC Architecture

An HMC consists of a single package containing multiple memory die and one logic die, stacked together, using through-silicon via (TSV) technology (see example in Figure 1). Within an HMC, memory is organized into vaults. Each vault is functionally and operationally independent.

Figure 1: Example HMC Organization

Each vault has a memory controller in the logic base (called a vault controller) that manages all memory reference operations within that vault. Each vault controller determines its own timing requirements. Refresh operations are controlled by the vault controller, eliminating this function from the host memory controller.

Each vault controller may have a queue that is used to buffer references for that vault's memory. The vault controller may execute references within that queue based on need rather than order of arrival.Therefore, responses from vault operations back to the external serial I/O links will be out of order. However, requests from a single external serial link to the same vault/bank address are executed in order. Requests from different external serial links to the same vault/bank address are not guaranteed to be executed in a specific order and must be managed by the host controller.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 2: HMC Block Diagram Example Implementation (4-link HMC configuration)

1.2 Logic Base Architecture

The logic base manages multiple functions for the HMC (see example implementation in Figure 2):

- All HMC I/O, implemented as multiple serialized, fully duplexed links
- Memory control for each vault; Data routing and buffering between I/O links and vaults
- Consolidated functions removed from the memory die to the controller
- Mode and configuration registers
- BIST for the memory and logic layer
- Test access port compliant to JTAG IEEE 1149.1-2001, 1149.6
- Some spare resources enabling field recovery from some internal hard faults.

The collective internally available bandwidth from all of the vaults is made accessible to the I/O links. A crossbar switch is an example implementation of how this could be achieved. The external I/O links consist of multiple serialized links, each with a default of 16 input lanes and 16 output lanes for full duplex operation. A half-width configura-

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

tion is also supported, comprised of 8 input lanes and 8 output lanes per link. Each lane direction in each link has additional power-down signals for power management. The following configurations are those supported within this specification.

Table 1: HMC Configurations

	4-link	8-link
Link speed (Gb/s)	10, 12.5, 15	10
Maximum aggregate link bandwidth[1]	160 GB/s, 200 GB/s, 240 GB/s	320 GB/s

Notes: 1. Link bandwidth includes data as well as packet header and tail. Full-width (16 input and 16 output lanes) configuration is assumed.

All in-band communication across a link is packetized. Packets specify single, complete operations. For example, a READ reference of 64 data bytes. There is no specific timing associated with memory requests, and responses can generally be returned in a different order than requests were made because there are multiple independent vaults, each executing independent of the others. The vaults generally reorder their internal requests to optimize bandwidths and to reduce average latencies.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

2 Pin Descriptions

Table 2: Pin Descriptions
"P" pins are the true side of differential signals; "N" pins denote the complement side

Signal Pin Symbol	Type	Description
Link Interface[1]		
LxRXP[n:0] LxRXN[n:0]	Input	Receiving lanes[2]
LxTXP[n:0] LxTXN[n:0]	Output	Transmitting lanes[2]
LxRXPS	Input	Power-reduction input
LxTXPS	Output	Power-reduction output
FERR_N	Output	Fatal error indicator. HMC drives LOW if fatal error occurs, otherwise the pull-down is turned off and floats HIGH. FERR_N operates in V_{DDK} domain. Requires external pull-up resistor. FERR_N can be wire-OR'd among multiple HMC devices. R_{ON} = 300Ω; R_{OFF} = 10kΩ
Clocks and Reset		
REFCLKP REFCLKN	Input	Reference clock for all links
REFCLKSEL	Input	Determines on-die termination for REFLCLKP/N. Set DC HIGH (≥V_{DDK} - 0.5V) if REFCLKP/N are AC-coupled. Set DC LOW (≤V_{SS} + 0.5V) if REFCLKP/N are DC-coupled.
P_RST_N	Input	System reset. Active LOW CMOS input referenced to V_{SS}. The P_RST_N is a rail-to-rail signal with DC HIGH ≥ (V_{DDK} - 0.5V) and DC LOW ≤ (V_{SS} + 0.5V).
JTAG Interface		
TMS	Input	JTAG test mode select
TCK	Input	JTAG test mode clock
TDI	Input	JTAG test data-in
TDO	Output	JTAG test data-out
TRST_N	Input	JTAG test reset (active LOW)
I^2C Interface		
SCL	Input	I^2C clock
SDA	Bidirectional	I^2C data
Bootstrapping Pins		
CUB[2:0]	Input	User-assigned HMC identification to enable the host to map the unique location of an HMC in a system. The value that the CUB[2:0] bits are tied to will be used as the I^2C slave address. This value will also be used as the default value for the CUB field of the Request packet header. DC HIGH ≥ (V_{DDK} – 0.5V) and DC LOW ≤ (V_{SS} + 0.5V).
REFCLK_BOOT[1:0]	Input	Bootstrapping pins that enable autonomous PLL configuration. Tie pins DC HIGH or DC LOW to match reference clock frequency being used. DC HIGH ≥ (V_{DDK} – 0.5V) and DC LOW ≤ (V_{SS} + 0.5V) **REFCLK_BOOT[1:0]** **Reference Clock Frequency** 00 125 MHz 01 156.25 MHz 10 166.67 MHz

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 2: Pin Descriptions (Continued)
"P" pins are the true side of differential signals; "N" pins denote the complement side

Signal Pin Symbol	Type	Description
Analog Pins		
EXTRESTP	–	Pins used to connect upper (top) precision 200Ω resistor used for termination impedance auto-calibration
EXTRESTN	–	
EXTRESBP	–	Pins used to connect lower (bottom) precision 200Ω resistor used for termination impedance auto-calibration
EXTRESBN	–	
Reserved Pins		
DNU	–	Do not use; must not be connected on the board
Supply Pins		
V_{DD}	Supply	Logic supply: 0.9V ±0.027V
$V_{DDPLLA[3:0]}$, $V_{DDPLLB[3:0]}$, V_{DDPLLR}	Supply	Filtered PLL supplies: 1.2V ±0.06V
V_{TT}, V_{TR}	Supply	Link transmit termination and link receive termination supplies: 1.2V ±0.06V supply
V_{DDM}	Supply	Memory supply: V_{DDM} = 1.2V ±0.06V
V_{CCP}	Supply	DRAM wordline boost supply 2.5V ±0.125V
V_{DDK}	Supply	NVM, I^2C, JTAG, and power management pin supply: 1.5 V (–0.05V/+0.20V)
V_{SS}	Supply	Ground

Notes:
1. x represents specific link number and will be 0 to 7 depending upon configuration.
2. In full-width mode n=15. In half-width mode n=7 and lanes 8-15 are considered DNU.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

3 Link Data Transmission

Commands and data are transmitted in both directions across the link using a packet-based protocol where the packets consist of 128-bit flow units called "FLITs." These FLITs are serialized, transmitted across the physical lanes of the link, then re-assembled at the receiving end of the link. Three conceptual layers handle packet transfers:

- The physical layer handles serialization, transmission, and deserialization.
- The link layer provides the low-level handling of the packets at each end of the link.
- The transaction layer provides the definition of the packets, the fields within the packets, and the packet verification and retry functions of the link.

Two logical blocks exist within the link layer and transaction layer:

- The link master, is the logical source of the link where the packets are generated and the transmission of the FLITs is initiated.
- The link slave, is the logical destination of the link where the FLITS of the packets are received, parsed, evaluated, and then forwarded internally.

The nomenclature below is used throughout the specification to distinguish the direction of transmission between devices on opposite ends of a link. These terms are applicable to both host-to-cube and cube-to-cube configurations.

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.

Figure 3: Link Data Transmission Implementation Example

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

4 Logical Sub-Block of Physical Layer

The transfer of information across the links consists of 128-bit FLITs. Each FLIT takes the following path through the serial link:

1. 128-bit FLITs are generated by the link master and sent in parallel to the transmitting logical sub-block in the physical layer.
2. The transmitting logical sub-block serializes each FLIT and drives it across the link interface in a bit-serial form on each of the lanes.
3. The receiving logical sub-block deserializes each lane and recreates the 128-bit parallel FLIT. Receive deserializer FLIT alignment is achieved during link initialization.
4. The 128-bit parallel FLIT is sent to the link layer link slave section.

4.1 Link Serialization

Link serialization occurs with the least-significant portion of the FLIT traversing across the lanes of the link first. During one unit interval (UI) a single bit is transferred across each lane of the link. For the full-width configuration, 16 bits are transferred simultaneously during the UI, so it takes 8 UIs to transfer the entire 128-bit FLIT. For the half-width configuration, 8 bits are transferred simultaneously, taking 16 UIs to transfer a single FLIT. The following table shows the relationship of the FLIT bit positions to the lanes during each UI for both full-width and half-width configurations.

Table 3: Unit Interval FLIT Bit Positions for Full-Width Configuration

	Lane															
UI	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
7	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112

Table 4: Unit Interval FLIT Bit Positions for Half-Width Configuration

	Lane							
UI	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	7	6	5	4	3	2	1	0
1	15	14	13	12	11	10	9	8
2	23	22	21	20	19	18	17	16
…	…	…	…	…	…	…	…	…
15	127	126	125	124	123	122	121	120

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

4.2 Scrambling and Descrambling

Figure 4: Scrambler and Descrambler Paths from Requester to Responder

Use the following polynomial for the scrambler and descrambler logic: 1 + x-14 + x-15. To scramble the data, XOR the LSB of the linear feedback shift register (LFSR) with the data, as shown in Figure 5 on page 16. Seeding values vary per lane, as shown in Table 5 on page 17. Designers should include a mode register bit to bypass scrambling for debug purposes.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 5: Scrambler Logic

```
// Title  :  scr_x15_serial.v
//
// This module implements a serial test circuit for a scrambler based on the
// polynomial 1+ x^(-14) + x^(-15).
//
// Such Scrambler is typically shown as a 15 bit Linear Feedback Shift Register
// (LFSR) with bits shifting from register 1 on the left to register 15 on the
// right, with register 14 and 15 combining to shift into register 1.
//
// The HMC Serializer outputs data[0] first from  parallel tx data[n:0],
// so if data[n:0] is to be bitwise scrambled with LFSR[n:0], we need the LFSR
// to shift from n -> 0, the opposite direction from the typical illustration.
// This implementation shifts data from LFSR[14] on the left to LFSR[0] on the
// right, with LFSR[1] and [0] combining to shift into LFSR[14].  This way
// LFSR[14:0] can bitwise scramble data[14:0] and be compatible with serializ-
// ation that shifts out on the data[0] side.
//
// Put otherwise: Polynomial 1+ x^(-14) + x^(-15) is equiv to  x^15 + x^1 + x^0

`timescale 100ps/100ps

module scr_x15_serial
(   input         CLK           ,
    input         I_ASYNC_RESETN,
    input  [14:0] I_SCRAM_SEED  , // unique per lane
    input         I_DATA        , // input  serial data
    output        O_DATA          // output serial data
);

reg  [14:0] LFSR; // LINEAR FEEDBACK SHIFT REGISTER

// SEQUENTIAL PROCESS
always @ (posedge CLK or negedge I_ASYNC_RESETN)
begin
    if (~I_ASYNC_RESETN)  LFSR[14:0] <= I_SCRAM_SEED;
    else                  LFSR[14:0] <= { (LFSR[1] ^ LFSR[0]) , LFSR[14:1] };
end                                     // serial shift right with left input

// SCRAMBLE
assign O_DATA = I_DATA ^ LFSR[0];

endmodule
```


The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 5: Scrambler Logic Seed Values

Lane	Seed Value
0	15'h4D56
1	15'h47FF
2	15'h75B8
3	15'h1E18
4	15'h2E10
5	15'h3EB2
6	15'h4302
7	15'h1380

Lane	Seed Value
8	15'h3EB3
9	15'h2769
10	15'h4580
11	15'h5665
12	15'h6318
13	15'h6014
14	15'h077B
15	15'h261F

Notes: 1. Scrambler logic seed values for lanes 0–7 are the same with full-width and half-width configurations.

Table 6: TS1 Definition

TS1 Bit Position															
(Sent Last) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	(Sent First) 0
1	1	1	1	0	0	0	0	L	L	L	L	S	S	S	S
0xF				0x0				Dependent on lane[1]				Sequence[2]			

Notes:
1. Lane field = 0xC for lane 15, or lane 7 in half width configuration, 0x3 for lane 0, and 0x5 for all other lanes.
2. Sequence[3:0] is a four bit counter that increments from 0 to 15 with each successive TS1 sent on a lane. Sequence value will roll back to 0 after 15 is reached.
3. The half width link configuration utilizes Lanes 0–7.
4. The two byte TS1 character establishes the framing that identifies byte pairs that belong to the same FLIT after training completes in half-width mode. The transition from sending TS1s to sending NULL FLITs can occur on any byte boundary within the entire 32 byte TS1 sequence, including mid-TS1 character, so this transition boundary to NULL bytes should NOT be used to infer the FILT boundary in half-width configuration.

Table 7: TS1 Training Sequence Example

TS1 Ordering	Lane 0	Lane 1–14	Lane 15
0	16'hF030	16'hF050	16'hF0C0
1	16'hF031	16'hF051	16'hF0C1
2	16'hF032	16'hF052	16'hF0C2
3	16'hF033	16'hF053	16'hF0C3
...	...	...	...
14	16'hF03E	16'hF05E	16'hF0CE
15	16'hF03F	16'hF05F	16'hF0CF

4.3 Lane Run Length Limitation

For each HMC RX lane, the scrambled data pattern must meet a run length limit of 85 UI. This is the maximum number of consecutive identical digits allowed. A run of 86 or more consecutive ones (or zeroes) must not be generated by the transmitter on any of the downstream lanes at any time. This restriction is in place so that the clock recovery

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

circuit at each HMC RX lane meets its minimum required transition density to assure correct data alignment. This restriction does not require a particular implementation of the transmitter logic, as long as the run length limit is met at the HMC RX lane inputs.

Although a given host may not have a run length limit on the scrambled upstream data, the HMC is designed to meet run length limitations of 85 UI on each lane at all times. This is implemented through the HMC's ability to monitor the scrambled data at each TX lane. If the scrambled TX data on any lane exceeds 85 consecutive digits, the TX scramble logic forces at least one transition. Forcing a transition corrupts the upstream response and results in a link retry (see 11 "Link Retry" on page 61). The probability of such an event is approximately 2^{-80} for random payload data. The HMC run length limitation feature may be disabled within the mode register.

4.4 Lane Reversal

In order to accommodate different external link routing topologies, the RX logical sub-block has the capability to reverse the lane bit number assignment. Lane reversal at the HMC is detected during initialization when receiving the TS1 training sequence and is automatically compensated for by logic in each receiving logical sub-block that reassigns external Lane 0 to connect to Lane 15 internally, Lane 1 to connect to Lane 14 internally, and so on. Once lane ordering is detected and set by the HMC logical sub-block, it remains unchanged thereafter. The lane reversal mode does not have to be the same for both directions of the link. An HMC component may have one or more links with lane reversal enabled at their respective receivers. The host is responsible for any implementation of lane reversal from the upstream lanes.

4.5 Lane Polarity

In order to accommodate different external Link routing topologies, the RX logical sub-block has the capability to logically invert the received data on a lane-by-lane basis. The polarity is determined during link initialization and remains unchanged thereafter. The training sequence is used to infer polarity at the HMC receiver. The requester is responsible for any implementation of lane polarity inversion from upstream lanes.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

5 Chaining

Multiple HMC devices may be chained together to increase the total memory capacity available to a host. A network of up to 8 HMC devices is supported. Each HMC in the network is identified through the value in its CUB field, located within the request packet header. The host processor must load vendor-specific routing configuration information into each HMC. This routing information enables each HMC to use the CUB field to route request packets to their destination. Supported multi-cube topologies are vendor-specific.

Each HMC link in the cube network is configured as either a host link or a pass-thru link, depending upon its position within the topology. See Figure 6 on page 19 for an example.

Figure 6: Example of Chaining Topology

A host link uses its link slave to receive request packets and its link master to transmit response packets. After receiving a request packet, the host link will either propagate the

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

packet to its own internal vault destination (if the value in the CUB field matches its programmed cube ID) or forward it towards its destination in another HMC via a link configured as a pass-thru link.

A pass-thru link uses its link master to transmit the request packet towards its destination cube, and its link slave to receive response packets destined for the host processor.

The HMC link connected directly to the host processor must be configured as a host link in source mode. The link slave of the host link in source mode has the responsibility to generate and insert a unique value into the source link identifier (SLID) field within the tail of each request packet. The unique SLID value is used to identify the source link for response routing. The SLID value does not serve any function within the request packet other than to traverse the cube network to its destination vault where it is then inserted into the header of the corresponding response packet. The host processor must load vendor-specific routing configuration information into each HMC. This routing information enables each HMC to use the SLID value to route response packets to their destination. Only a host link in source mode will generate a SLID for each request packet. On the opposite side of a pass-thru link is a host link that is NOT in source mode. This host link operates with the same characteristics as the host link in source mode except that it does not generate and insert a new value into the SLID field within a request packet. All link slaves in pass-thru mode use the SLID value generated by the host link in source mode for response routing purposes only. The SLID fields within the request packet tail and the response packet header are considered 'Don't Care' by the host processor.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

6 Power-On and Initialization

The HMC must be powered up and initialized in a predefined manner. All timing parameters specified in the following sequence can be found in the Table 54, "Initialization Timing Parameters," on page 102. The HMC does not include or require time-out mechanisms during any step within the initialization routine. Registers indicating the status of the HMC during initialization can be accessed via the I^2C or JTAG bus.

The following sequence is required for power-up:

1. Apply power (all supplies). V_{CCP} must ramp to full rail before the V_{DDM} ramp begins. All supplies must ramp to their respective minimum DC levels within ^tDD, but supply slew rates must not exceed V_{DVDT}.
2. Ensure that P_RST_N is below $0.2 \times V_{DDK}$ during power ramp to ensure that outputs remain disabled (High-Z) and on-die termination (ODT) is off. Link transmit signals (LxTXP[n:0] and LxTXN[n:0]) will remain High-Z until Step 5 below. All other HMC inputs can be undefined during the power ramp with the exception of TRST_N, which must be held LOW. TRST_N will track the subsequent transition of P_RST_N if using the JTAG port.
3. After the voltage supplies and reference clocks (REFCLKP and REFCLKN) are within their specified operating tolerance, P_RST_N must be LOW for at least ^tRST to begin the initialization process.
4. After P_RST_N goes HIGH ($\geq 0.5 \times V_{DDK}$),HMC initialization begins. P_RST_N must meet min slew rate of $V_{RSTDVDT} = 0.1V/ns$ as it transitions from LOW to HIGH. Active mode (full-power/high bandwidth power state) is assumed unless stated otherwise.
5. HMC PLL lock occurs within ^tINIT. After ^tINIT has passed, the I^2C or JTAG bus is used to load the configuration registers. When complete, the Init Continue register must be set to allow internal configuration to continue.
6. After the Init Continue register is set, the HMC continues with internal initialization. The responder enters its start state and the transmit signals (LxTXP[n:0] and LxTXN[n:0]) begin to transmit a nonNULL PRBS stream to the requester. This stream is intended to prohibit the requester from acquiring descrambler sync, which is not desired until step 8.
7. To initialize the responder's descramblers, the requester enters the idle state and issues continuous scrambled NULL FLITs to the responder (see 9.12.1 "NULL Command" on page 50 for NULL description). The responder descrambler sync should occur within $^tRESP1$ of the PLL locking. Upon descrambler sync, the responder will also enter the IDLE state and begin to transmit scrambled NULL FLITs.
8. The responder issues continuous scrambled NULL FLITs, and the requester waits for these at its descrambler outputs. The purpose of this step is to achieve synchronization of the requester's descramblers.
9. Once the requester has synchronized its descramblers, it enters the TS1 state and issues the scrambled TS1 training sequence continuously to the responder to achieve FLIT synchrony (see Table 6 on page 17 for TS1 sequence). Responder link lock should occur within $^tRESP2$.
10. After responder link lock occurs, it will enter the TS1 state and issue a series of scrambled TS1 training sequence back to the requester. The requester waits for one or more valid TS1 training sequence(s) at the local descrambler output. The requester must then align per-lane receiver timing to achieve host FLIT synchronization.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

11. The requester stops sending TS1 training sequences to the responder after it has achieved link lock, enters the active state, and starts sending scrambled NULL FLITs. Upon realization that the requester is no longer sending TS1 sequences, the responder stops sending TS1 sequences and enters the active state, also sending scrambled NULL FLITs back to the requester.
12. The HMC is ready for operational packets. Both sides of the link will perform transaction layer initialization (see 9.14 “Transaction Layer Initialization” on page 51). Host-commanded power-state transitions are supported at this point.

Figure 7: HMC Initialization Flowchart

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 8: Initialization Timing

Notes: 1. Data on links is scrambled.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

7 Power State Management

Each link can independently be set into a lower power state through the usage of the power state management pins, LxRXPS and LxTXPS. Each of the links can be set into a low-power state, sleep mode, as well as a minimum power state called down mode.

Power is reduced when sleep mode is entered from normal operation (active mode) through the disablement of the link's high-speed SerDes circuitry. After a link has been initialized and is in active mode, the requester can transition its power state management pin, LxTXPS, from HIGH to LOW to put the opposite side of the link (responder) into sleep mode. As it begins to enter sleep mode, the responder will transition it's LxTXPS from HIGH to LOW within the ^{t}PST specified timing. Values for ^{t}PST and all other power state management related timing can be found in Table 55, "Link Power Management Parameters," on page 103. Transition of the responder's LxTXPS will initiate the requester's Rx and Tx lanes to enter into sleep mode as well. It is the responsibility of the host to quiesce traffic (ensure closure on all in-flight transactions) prior to exiting active mode link state. The responder's link logic is not responsible for in-flight transactions when LxRXPS goes from 1 to 0.

Down mode allows a link to go to an even lower power state than sleep mode by disabling both the high-speed SerDes circuitry as well as the link's PLLs. A link enters down mode if its corresponding LxRXPS signal is LOW when the P_RST_N signal transitions from LOW to HIGH either during initialization or a reset event. The HMC can be configured so that all of its links will enter down mode after the last link in active mode transitions to sleep mode. In this setting, any link already in sleep mode when the last link exits active mode will be transitioned to down mode to further reduce power consumption. Transitioning any links from sleep mode to down mode will take up to 150µs (represented by tSD specification).

When all links exit active mode and transition to down mode, the HMC enters a self refresh state. Although not accessible from the links, data stored within the memory is still maintained . Upon entering self refresh the HMC must stay in this state for a minimum of 1ms (^{t}SREF). This is measured from the time the HMC's last link transitions its LxRXPS LOW (entering self refresh) to its first LxRXPS to transition HIGH (entering active mode).

All links will independently respond to power state control; however, in the event of simultaneous power state transitions, the HMC will stagger transitions between active mode and sleep modes in order to avoid supply voltage shifts. The amount of stagger time incurred is defined by the ^{t}SS specification. Multiply ^{t}SS by n - 1 to determine when the final link will transition states, where n equals the number of simultaneous link transitions. For example, if the requester sets its L0TXPS, L1TXPS, L2TXPS, and L3TXPS to 0 at the same time, it will take 1.5µs for the HMC to transition all four links into sleep mode (^{t}SS × 3). This is also true for links exiting sleep mode and entering active mode. As an example, if L0RXPS and L1RXPS transition from LOW to HIGH at the same time, the second links will not start the transition to active mode until up to 500ns later (^{t}SS × 1).

The transition of a link to active mode from either sleep mode or down mode requires a re-initialization sequence as illustrated in Figure 9.

Bringing a link up from down mode into active mode requires additional time to complete the HSS PLL self-calibration as is specified by ^{t}PSC (see Note 3 of Figure 9).

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

The transition of a pass-thru link's LxTXPS signal is dependent on the power state transition of the host links within the same HMC:

- A pass-thru link's LxTXPS will transition LOW after the last host link in the same cube enter sleep mode.
- A pass-thru link's LxTXPS will transition HIGH as soon as the LxTXPS of any host link in same cube transitions HIGH.

Table 8: Link Power States and Conditions

Mode	Method of Entry	HMC Tx Lane Status	HMC Rx Lane Status	Link PLL Status
Active	Standard initialization	Enabled	Enabled	Enabled
Sleep	LxRXPS asserted LOW while in active mode	High-Z	Disabled	Enabled
Down	1) LxRXPS asserted LOW when P_RST_N transitions from LOW to HIGH or 2) After the last link in active mode enters sleep mode (all other links already in sleep or down mode)	High-Z	Disabled	Disabled

Notes:
1. Data from all vaults available through any link in active mode, even if other links are in sleep or down mode.
2. Self refresh is entered when all links exit active mode whereby data in HMC memory is retained but unavailable.

Figure 9: Sleep Mode Entry and Exit (Single Link Only)

Notes:
1. PRBS transmitted by HMC pass-thru link; NULLs or PRBS may be transmitted from host.
2. If NULLs transmitted by host, ^{t}RXD_NULL must be met to allow responder clock data recovery to be achieved before starting descramble sync.
3. ^{t}PSC must be added to this delay when exiting down mode.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 10: Simultaneous Transition of Four Host Links to Sleep Mode, Entry into Down Mode and Return to Active Mode (Single HMC, Four Link Example)

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

8 Link Layer

The link layer handles communication across the link not associated with the transaction layer. This consists of the transmission of NULL FLITs and flow packets.

All FLITs of a packet must be transmitted sequentially, without interruption, across the link. NULL FLITs are generated at the link layer when no other packets are being transmitted. A NULL FLIT is an all-zeros pattern that is scrambled prior to transmission. Any number of packets can be streamed back-to-back across the link, or they can be separated by NULL FLITs, depending upon system traffic and transaction layer flow control. There is no minimum or maximum requirement associated with the number of NULL FLITs transmitted between packets. NULL FLITs are not subject to flow control. The first nonzero FLIT following a NULL FLIT is considered to be the first FLIT of a packet. Note that data FLITs within a packet may be all zeros, but these lie between a header FLIT and a tail FLIT.

Flow packets are generated by the link master to pass flow control and retry control commands to the opposite side of the link. Flow packets are sent when no other link traffic is occurring or when a link retry sequence is to be initiated. Flow packets are single-FLIT packets with no data payload and are not subject to flow control. Flow packets utilize the same header and tail format as request packets, but are not considered requests, and do not have corresponding response commands. See Section 8.12, “Flow Commands” for specifics on the commands transmitted with flow packets.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9 Transaction Layer

As noted previously, all information transmitted across the links is included in packets that consist of one or more FLITs. A packet always includes an 8-byte header at the beginning of the packet and an 8-byte tail at the end of the packet. The header includes the command field that identifies the packet type, other control fields, and when required, the addressing information. The tail includes flow and link-retry control fields along with the CRC field. There are three different packet types. The first two, request and response packets, are used at the transaction layer. The third type, the flow packet, is only used at the link layer, however it is included below so that all packet types can be described within a single section.

1. Request packets are issued by the requester (host or HMC configured as a pass-thru link). The request packet header includes address information to perform a READ or WRITE operation. Write request packets include data FLITs.
2. Response packets are issued by the responder (HMC configured as a host link). Response headers do not include address information. READ responses include data FLITs, and can optionally include write response tags embedded in the header. Write response packets do not include data FLITs.
3. Flow packets are not part of the transaction layer protocol and are not subject to flow control. They are only used at the link to pass flow control and retry control information back to the link master when normal transaction layer packets are not flowing in the return direction. Flow packets are generated at the link master and are decoded and then dropped at the link slave; they are not forwarded and do not pass through the HMC.

Each packet type has its own defined header and tail formats, described in Sections 9.6 “Request Packets” on page 37, 9.7 “Response Packets” on page 38, and 9.8 “Flow Packets” on page 43.

Packets that consist of multiple FLITs are transmitted across the link with the least-significant FLIT number first. The first FLIT includes the header and the least significant bits (LSBs) of the data. Subsequent data FLITs progress with more significant data following. Figure 11 illustrates the layout for packets with and without data.

Figure 11: Packet Layouts

Notes: 1. Each numbered data field represents a byte with bit positions [7(msb): 0(lsb)].

A CRC field is included in the tail of every packet that covers the entire packet, including the header, all data, and the nonCRC tail bits. The link retry logic retransmits packets across the link if link errors are detected, as described in Section 11 “Link Retry” on page 61. Upon successful transmission of packets across the link, the packets are transmitted through the logic base all the way to the destination vault controller. CRC

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

provides command and data through-checking to the destination vault controller. ECC provides error coverage of the data from the vault controller to and from the DRAM arrays. Thus, data is protected along the entire path to and from the memory device. If the vault controller detects a correctable error, error correction is executed before the data is returned. If an uncorrectable error is detected, an error status is returned in the response packet back to the requester.

The CRC algorithm used on the HMC is the Koopman CRC-32K. This algorithm was chosen for the HMC because of its balance of coverage and ease of implementation. The polynomial for this algorithm is:

$$x^{32} + x^{30} + x^{29} + x^{28} + x^{26} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{11} + x^{10} + x^{7} + x^{6} + x^{4} + x^{2} + x + 1$$

The CRC calculation operates on the LSB of the packet first. The packet CRC calculation must insert 0s in place of the 32-bits representing the CRC field before generating or checking the CRC. For example, when generating CRC for a packet, bits [63: 32] of the Tail presented to the CRC generator should be all zeros. The output of the CRC generator will have a 32-bit CRC value that will then be inserted in bits [63:32] of the Tail before forwarding that FLIT of the packet. When checking CRC for a packet, the CRC field should be removed from bits [63:32] of the Tail and replaced with 32-bits of zeros, then presented to the CRC checker. The output of the CRC checker will have a 32-bit CRC value that can be compared with the CRC value that was removed from the tail. If the two compare, the CRC check indicates no bit failures within the packet.

Overall packet lengths are from 1 to 9 FLITs. Write request packets and read response packets will contain from 16B to 128B of data between the header and the tail.

All requests are operationally closed as follows:

- Request packets are uniquely identified with a tag field embedded in the packet.
- All read request packets have a corresponding read response packet returned to the requesting link that includes the corresponding request tag and requested data.
- Normal write request packets are acknowledged either with an explicit write response packet that includes the write request's tag, or optionally with the write request tag embedded in the TGA field of a read response packet.
  - This positive acknowledgment provides an indication to the requesting processor that the request has been executed without error.
- As the HMC does not use the TAG field of posted write requests, the host can populate this field with any value.
- If unrecoverable errors occur in the request path, they are captured in the error status that is included in the response packet.
- If unrecoverable errors occur in the request path for posted write requests, an error response packet is returned to the requester indicating an error.

Flow packets are not considered a request, they do not have a valid tag, and they do not have a corresponding response packet.

The flow of request packets and response packets are managed with tokens, described in Section 9.3 "Packet Flow Control" on page 35. Flow packets have no flow control and can be sent at any time by the link master.

9.1 Memory Addressing

A request packet header includes a address field of 34 bits for internal memory addressing within the HMC. This includes vault, bank, and DRAM address bits. The 4-link configurations currently defined will provide a total of up to 4GB of addressable

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

memory locations within one HMC. This requires the lower 32 address bits of the 34-bit field; the upper 2 bits are for future expansion and are ignored by the HMC. The 8-link configurations currently defined will provide a total of up to 8GB of addressable memory locations within one HMC. This requires the lower 33 address bits of the 34-bit field; the upper bit is for future expansion and are ignored by the HMC.

The user can select a specific address mapping scheme so that access of the HMC vaults and banks is optimized for the characteristics of the request address stream. If the request address stream is either random or generally sequential from the low-order address bits, the host can choose to use one of the default address map modes offered in the HMC (See Table 10, "Default Address Map Mode Table," on page 32). The default address mapping is based on the maximum block size chosen in the address map mode register. It maps the vault address toward the less significant address bits, followed by the bank address 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.

9.1.1 Memory Addressing Granularity

The HMC supports READ and WRITE data block accesses with 16-byte granularity from 16B to the value of the maximum block size setting (32B, 64B, or 128B). The maximum block size is set with the Address Configuration register. See 14.2 "Data Access Performance Considerations" on page 76 for information on how to select maximum block size such that it optimizes performance in a given system. The maximum block size specified in the address map mode register determines the number of bits within the byte address field, of which the four LSBs are ignored due to the 16-byte granularity (an exception is the BIT WRITE command, explained on page 47). The 4 LSBs of the byte address may be used for future density expansion if 16B block aligned access is preserved. Details for address expansion are outside the scope of this specification version. The remaining upper bits of the byte address indicate the starting 16-byte boundary when accessing a portion or all of the block. The vault controller uses these upper bits of the byte address as part of its DRAM column addressing. If the starting 16-byte address in conjunction with the data access length of the request causes the access to go past the end of the maximum block size boundary, the DRAM column addressing will wrap within the maximum block size and continue. For example, if the maximum block size is 64 bytes and a 48-byte READ command is received starting at byte 32, the DRAM access will occur starting at byte 32 up to byte 63, then continue at byte 0 up to byte 15.

9.1.2 Memory Address-to-Link Mapping

Each link is associated with four local vaults that are referred to as a quadrant. Access from a link to a local quadrant may have lower latency than to a link outside the quadrant. Bits within the vault address designate both the specific quadrant and the vaults within that quadrant. See Table 9 for the specific vault addressing of 4-link and 8-link devices.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 9: Addressing Definitions

Address	Description	4-Link Configuration	8-Link Configuration
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; See 9.10.5 "BIT WRITE Command" on page 47 for details)	The 4 LSBs of the byte address are ignored for READ and WRITE requests (with the exception of BIT WRITE command; See 9.10.5 "BIT WRITE Command" on page 47 for details)
Vault address	Addresses vaults within the HMC	Lower 2 bits of the vault address specifies 1 of 4 vaults within the logic chip quadrant	Lower 2 bits of the vault address specifies 1 of 4 vaults within the logic chip quadrant
		Upper 2 bits of the vault address specifies 1 of 4 quadrants	Upper 3 bits of the vault address specifies 1 of 8 quadrants
Bank address	Addresses banks within a vault	4GB HMC: addresses 1 of 16 banks in the vault	8GB HMC: addresses 1 of 16 banks in the vault
		2GB HMC: addresses 1 of 8 banks in the vault	4GB HMC: addresses 1 of 8 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 vault controller breaks the DRAM address into row and column addresses, addressing 1Mb blocks of 16 bytes each

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.1.3 Default Address-Map Mode Table – 4-link Devices

Table 10: Default Address Map Mode Table

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

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.1.4 Default Address-Map Mode Table – 8-link Devices

Table 11: Default Address Map Mode Table

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

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.1.5 Address Mapping Mode Register

The address mapping mode register provides user control of mapping portions of the ADRS field within the header of request packets to the vault and bank addresses. When using default address-mapping, the address mapping mode field should be set based on specific system requirements. See Section 14.2 “Data Access Performance Considerations” on page 76 for details.

In addition to the default address mapping algorithms, user-defined fields are provided for the user to specify a unique mapping algorithm. There is a user-defined field for the vault address and the bank address. Each field holds a 5-bit encoded value pointing to the corresponding bit position in the ADRS field of the request packet header. This represents the LSB that will be used for the vault or bank address, with the more significant bits of the subaddress coming from the bit positions immediately above it in the request header ADRS field. When operating with the user-defined modes, it is the user’s responsibility to set the user-defined encoded fields so that the vault and bank addresses do not overlap each other or overlap the byte address within the request header ADRS field. Note that the byte address is always fixed at the LSBs of the request header ADRS field. The user must also be aware that the number of bits in the bank address is dependent upon the size of the HMC. The remaining bits of the request ADRS field not specified as the vault and bank addresses become the DRAM address. See Section 10 “Configuration and Status Registers” for more details on user-defined address mapping.

The address map mode register is writable via the MODE WRITE command or via the maintenance interface (I^2C or JTAG bus). Note that because the MAX block size is defaulted to 64 bytes, the host boot block code must either be compatible (issue requests of 64 bytes or smaller) or issue a MODE WRITE command to change the MAX block size.

9.1.6 DRAM Addressing

Each bank in the HMC contains 16,777,216 memory bytes (16 MB). A WRITE or READ command to a bank accesses 32 bytes of data for each column fetch. The bank configuration is identical in all specified HMC configurations.

9.2 Packet Length

As noted previously, packets are always multiples of 16-byte FLITs. Within each packet header there is a length (LNG) field that indicates the total number of FLITs in the packet. A packet that contains no data FLITs would have LNG = 1; this represents the single FLIT holding the 8-byte header and 8-byte tail. When generating response packets, the necessary data size is determined from the CMD field in the corresponding request packet. Specific packet lengths are provided in the Tables 17, 25, and 26.

A Duplicate Length (DLN) field is included within each packet header to provide additional packet integrity. The value of DLN must equal that of LNG within a packet. The DLN field is compared to the LNG field prior to the CRC being checked. Note that LNG enables the packet parser to determine the location of the CRC field (where a value of 1 points to the current FLIT). A miscompare of DLN and LNG is treated like a CRC check and will trigger the link retry mechanism. If LNG does not equal DLN when the packet is generated at the host, link retries will be unsuccessful and a fatal error will result.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.3 Packet Flow Control

Flow control occurs in both directions on each link. The flow of transaction layer packets is managed with token counts. Each token represents the storage to hold one FLIT (16 bytes). The link slave input buffer temporarily stores transaction layer packets as they are received from the link. (Note that flow packets are not stored in the input buffer, and therefore, are not subject to flow control.) The minimum size of this buffer must be equal to or greater than the number of FLITs in a single packet of the largest supported length, but in general can hold several of the largest packets to enable a constant flow across the link. The available space in this input buffer is represented in a token count register at the link master. This gives the link master knowledge of the available buffering at the other end of the link and the ability to evaluate whether there is enough buffer space to receive every FLIT of the next packet to be transmitted.

The token count register is loaded during the initialization sequence with the maximum number of tokens representing the available buffer space at the link slave when it is empty. As the link master sends each transaction layer packet across the link, it must decrement the token count register by the number of FLITs in the transmitted packet. As the packet is received and stored in the input buffer, its FLITs use up the space represented by the decremented value of token count register at the link master. As each packet is read out of the input buffer, it is forwarded to a destination, and its FLIT location is freed up. This requires a mechanism to return packet tokens to the link transmitter, using the opposite link direction. The input buffer control logic sends a count (representing the number of FLITs that were read out of the input buffer when the packet was forwarded) to the local (adjacent) link master. This count is returned in the return token count (RTC) field of the next possible packet traveling in the opposite direction on the link. At the link slave, the RTC field is extracted from the incoming packet and sent back to the original link transmitter where its value is added to the current value of the token count register. If no transaction layer packet traffic is occurring in the return direction, the link master must create a flow packet known as a token return (TRET) to return the token. TRET is described in 9.12.3 “TOKEN RETURN (TRET) Command” on page 50.

Appropriate sizing of the link slave input buffer requires weighing the trade-offs between latency, throughput, silicon real-estate, and routability. If there is a temporary pause in the packet forwarding priority at the output of the input buffer (internal switch conflicts, or destination vault flow control), a packet might not be emptied from the input buffer. This would cause a pause in the return of the tokens. As long as this is infrequent and the input buffer is sized to accommodate it, the packet flow will not be disrupted unless the link is running at 100% busy. If the input buffer is empty, a specific implementation may choose to bypass the input buffer and forward a packet immediately to its destination. In this case, the tokens for the packet would be immediately returned to the link master.

As noted previously, the link master must not transmit a packet if there is insufficient space in the input buffer at the other end of the link. The LNG field within the header of each outgoing packet must be compared to the token count register to determine whether the packet should be transmitted or the packet stream should be paused.

9.4 Tagging

Operational closure for requests is accomplished using the tag fields in request and response packets. The tag value remains associated with the request until a response is received indicating that the request has been executed. Tags in READ requests are returned with the respective read data in the read response packet header. Write

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

response tags are returned either within a write response packet, or optionally within the return tag A (TGA) field of any other response packet being returned to the same source link. If the feature is not supported, the corresponding bits within response packets will be specified as reserved. In the case of posted write requests, because no response is returned and the HMC does not use the TAG field of posted write requests, the host can populate this field with any value. Note that when multiple host links are connected to the HMC, each response packet will be returned on the same link as its associated request. This keeps the tag range independent for each host link.

Tag fields in packets are nine bits long, which is enough space for 512 tags. All tags are available for use. Tag assignment and reassignment are managed by the host. There is no required algorithm to assign tag values to requests. HMC does not use the tag for internal control or identification, only for copying the tag from the request packet to the response packet.

Tags are assigned by control logic at the host link master and must not be used in another request packet until a response tag with the same tag number is returned to that host link. Other host links will use the same tag range of 512 tags, but they are uniquely identified by their association with each different host link. Thus, the total maximum number of unique tags is 512 times the number of host links. For example, if four links are connected from an HMC to the host, there are 2048 usable tags for requests to the HMC. As an implementation example, a host control logic might decide to provide a time-out capability for every transaction to determine whether a request or response packet has been lost or corrupted, preventing the tag from being returned. The timer in this implementation example would consider link retry periods that occur to account for response packets that are delayed, but still returned.

9.5 Packet Integrity

The integrity of the packet contents is maintained with the CRC field in the tail of every packet. Because the entire packet (including header and tail) is transmitted from the source link all of the way to the destination vault, CRC is used to detect failures that occur not only on transmission across the link, but along the entire path. Note that CRC may be regenerated along the path if flow control fields within the header or tail change. CRC regeneration is done in an overlapped fashion with respect to a CRC check to ensure that no single point of failure will go undetected.

There may be cases when the first FLITs of a packet are forwarded before the tail is received and the CRC is checked. This may occur in the HMC's link slave after a request packet crosses a link and is done to avoid adding latency in the packet path. If the packet is found to have a CRC error as it passes through the link slave, the packet is "poisoned," meaning that a destination, in this case a vault controller, will recognize it as nonfunctional and will not use it. The link slave poisons the packet by inverting a recalculated value of the CRC and inserting that into the tail in place of the errored CRC. Another example of a forwarded poisoned packet is the case in which DRAM errors occur and are internally retried. If a parity error occurs on the command or address from the vault controller to the DRAM, a response packet may be generated and transmission back to the requester could be started before the parity error is detected. In this case, the CRC in the tail of the response packet will be poisoned, meaning the vault controller will invert the CRC and insert it into the tail of the packet. Consequently, the requester may receive the poisoned packet and must drop it. The vault controller will retry the DRAM request, and upon a successful DRAM access, another response packet will be generated to replace the poisoned one.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Note that if a packet travels across a link after it is poisoned, a link master will still be able to embed flow control fields in the packet by recalculating the CRC with the embedded values, then inverting the recalculated CRC so that the poisoned state will be maintained. Because the flow control fields are valid in a poisoned packet, it is stored in the retry buffer. In this case, the link slave on the other end of the link will recognize the poisoned CRC and will still extract the flow control fields. Whenever a packet is poisoned, the CMD, ADRS, TAG, TGA, and ERRSTAT fields are not valid, but the flow control fields (FRP, RRP, RTC) are valid.

9.6 Request Packets

Request packets carry request commands from the requester (host or HMC link configured as pass-thru link) to the responder (HMC link configured as host link). All request packets are subject to normal packet flow control.

9.6.1 Request Header Layout

Figure 12: Request Packet Header Layout

63 61	60 58	57 24	23 15	14 11	10 7	6	5 0
CUB[2:0]	RES[2:0]	ADRS[33:0]	TAG[8:0]	DLN[3:0]	LNG[3:0]	RES	CMD[5:0]

Request packet headers for request commands and flow commands contain the fields shown in the table below. Specific commands may require some of the fields to be 0. These are provided in Table 27 on page 50.

Table 12: Request Packet Header Fields

Name	Field Label	Bit Count	Bit Range	Function
Cube ID	CUB	3	[63:61]	CUB field used as an HMC identifier when multiple HMC devices are chained together. See Section 11 for more information. The HMC ignores bits in this field in non-chained topologies, except for including them in the CRC calculation.
Reserved	RES	3	[60:58]	Reserved: These bits are reserved for future address or Cube ID expansion. The responder will ignore bits in this field from the requester except for including them in the CRC calculation. The HMC can use portions of this field range internally.
Address	ADRS	34	[57:24]	Request address. For some commands, control fields are included within this range.
Tag	TAG	9	[23:15]	Tag number uniquely identifying this request.
Duplicate length	DLN	4	[14:11]	Duplicate of packet length field.
Packet length	LNG	4	[10:7]	Length of packet in FLITs (1 FLIT is 128 bits). Includes header, any data payload, and tail.
Reserved	RES	1	[6]	Reserved: The responder will ignore this bit from the requester except for including it in the CRC calculation. The HMC can use this field internally.
Command	CMD	6	[5:0]	Packet command. (See Table 17 on page 44 for the list of request commands.)

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.6.2 Request Tail Layout

Figure 13: Request Packet Tail Layout

63 – 32	31 – 27	26 – 24	23 – 19	18 – 16	15 – 8	7 – 0
CRC[31:0]	RTC[4:0]	SLID[2:0]	RES[4:0]	SEQ [2:0]	FRP[7:0]	RRP[7:0]

Table 13: Request Packet Tail Fields

Name	Field Label	Bit Count	Bit Range	Function
Cyclic redundancy check	CRC	32	[63:32]	The error-detecting code field that covers the entire packet.
Return token count	RTC	5	[31:27]	Return token count for transaction-layer flow control. In the request packet tail, the RTC contains the tokens that represent available space in the requester's input buffer.
Source Link ID	SLID	3	[26:24]	Used to identify the source link for response routing. The physical link number is inserted into this field by the HMC. The host can ignore these bits (except for including them in the CRC calculation) or alternatively, use the value in this field to validate the response packet is routed to the correct link source.
Reserved	RES	5	[23:19]	Reserved: The responder will ignore bits in this field from the requester except for including them in the CRC calculation. The HMC can use portions of this field range internally.
Sequence number	SEQ	3	[18:16]	Incrementing value for each packet transmitted, except for PRET and IRTRY packets (see 11.2.2 "Packet Sequence Number Generation" on page 64).
Forward retry pointer	FRP	8	[15:8]	Retry pointer representing this packet's position in the retry buffer (see 11.1 "Retry Pointer Description" on page 62).
Return retry pointer	RRP	8	[7:0]	Retry pointer being returned for other side of link (see 11.1 "Retry Pointer Description" on page 62).

9.7 Response Packets

Response packets carry response commands from the responder (HMC link configured as host link) to the requester (host or HMC link configured as pass-thru link). All response packets are subject to normal packet flow control.

Figure 14: Response Packet Header Layout

Notes: 1. Support of TGA[8:0] is optional. Bits are reserved if function not supported.

63 – 42	41 – 39	38 – 33	32 – 24	23 – 15	14 – 11	10 – 7	6	5 – 0
RES[21:0]	SLID[2:0]	RES[5:0]	TGA[8:0][1]	TAG[8:0]	DLN[3:0]	LNG[3:0]	RES	CMD[5:0]

Response packet headers for response commands contain the fields shown in Table 14. Specific commands may require some of the fields to be 0. These are noted in Table 27 on page 50.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 14: Response Packet Header Fields

Name	Field Label	Bit Count	Bit Range	Function
Reserved	RES	22	[63:42]	Reserved: The host will ignore bits in this field from the HMC except for including them in the CRC calculation.
Source Link ID	SLID	3	[41:39]	Used to identify the source link for response routing. The physical link number (0–3) is inserted into this field by the HMC. The host can ignore these bits (except for including them in the CRC calculation) or alternatively, use the value in this field to validate the response packet is routed to the correct link source.
Reserved	RES	6	[38:33]	Reserved: The host will ignore bits in this field from the HMC except for including them in the CRC calculation.
Return tag (optional)	TGA	9	[32:24]	Field that can contain a write response tag (see 9.4 "Tagging" on page 35). Bits are reserved if function not supported.
Tag	TAG	9	[23:15]	Tag number uniquely associating this response to a request.
Duplicate length	DLN	4	[14:11]	Duplicate of packet length field
Packet length	LNG	4	[10:7]	Length of packet in 128-bit FLITs
Reserved	RES	1	[6]	Reserved: The host will ignore this bit from the HMC except for including it in the CRC calculation.
Command	CMD	6	[5:0]	Packet command (see Table 25 on page 48 for response commands)

Figure 15: Response Packet Tail Layout

63 … 32	31 … 27	26 … 20	19	18 … 16	15 … 8	7 … 0
CRC[31:0]	RTC[4:0]	ERRSTAT[6:0]	DINV [0]	SEQ [2:0]	FRP[7:0]	RRP[7:0]

Table 15: Response Packet Tail Fields

Name	Field Label	Bit Count	Bit Range	Function
Cyclic redundancy check	CRC	32	[63:32]	Error-detecting code field that covers the entire packet.
Return token counts	RTC	5	[31:27]	Returns token count for transaction-layer flow control. In the response packet tail, the RTC contains the tokens that represent available space in the HMC input buffer.
Error status	ERRSTAT	7	[26:20]	Error status bits (see Table 16 on page 40)
Data Invalid	DINV	1	[19]	Indicates validity of packet payload. Data in packet is valid if DINV = 0 and invalid if DINV = 1.
Sequence number	SEQ	3	[18:16]	Incrementing value for each packet transmitted. See 11.2.2 "Packet Sequence Number Generation" on page 64).
Forward retry pointer	FRP	8	[15:8]	Retry pointer representing this packet's position in the retry buffer (see 11.1 "Retry Pointer Description" on page 62).
Return retry pointer	RRP	8	[7:0]	Retry pointer being returned for the other side of link (see 11.1 "Retry Pointer Description" on page 62).

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

The ERRSTAT field included in the response tail is valid for all read, write, and error responses. All bits in the ERRSTAT field being zero indicates a normal response with no errors or flags. There are 6 different error and flag categories defined by ERRSTAT[6:4] bits. The unique error or flag descriptions within each category indicate a specific error or item that requires attention. Additional status on errors or flags can be obtained by accessing internal status registers using the MODE READ command or sideband (I^2C or JTAG). Table 16 shows the error categories and descriptions along with the method of response (read/write response packet or error response packet).

Table 16: ERRSTAT[6:0] Bit Definitions

ERRSTAT Bit							Description	Response Packet Type
6	5	4	3	2	1	0		
0	0	0	0	0	0	0	No errors/conditions	
Warnings								
0	0	0	0	0	0	1	Operational temperature threshold: The internal temperature sensors in the HMC have reached the upper limits of operational temperature.	Error response packet (TAG = CUB number)
0	0	0	0	0	1	0	Host token overflow warning: The host sent more than 1023 tokens to the HMC on a link.	
0	0	0	0	1	0	0	Repair information (optional): Remaining repair resources have fallen below a minimum threshold.	
0	0	0	0	1	0	1	Repair warning (optional): There are one or more regions of memory that have no available repair resources left.	
0	0	0	0	1	1	0	Repair alarm: A request has encountered a hard SBE or MUE, and there are no repair resources available.	
DRAM Errors								
0	0	1	0	0	0	0	SBE occurred (this status is normally disabled) - Data is corrected before being returned to the requester. the SBE is scrubbed and tested to determine if it is a hard error, in which case it is dynamically repaired.	Read or write response packet (TAG = tag of corresponding request)
0	0	1	1	1	1	1	MUE has occurred - this indicates that a multiple uncorrectable error has occurred in the DRAM for read data. The DINV (data invalid) flag will also be active. Can also be unsolicited if MUE encountered during patrol scrub.	
Link Errors								

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 16: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type
6	5	4	3	2	1	0		
0	1	0	0	0	0	0	Link 0 retry successful: Link 0 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	Error response packet (TAG = CUB number)
0	1	0	0	0	0	1	Link 1retry successful: Link 1 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	0	1	0	Link 2 retry successful: Link 2 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	0	1	1	Link 3 retry successful: Link 3 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	1	0	0	Link 4 retry successful: Link 4 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	1	0	1	Link 5 retry successful: Link 5 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	1	1	0	Link 6 retry successful: Link 6 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	
0	1	0	0	1	1	1	Link 7 retry successful: Link7 has successfully recovered from a link error by executing one or more link retries (within the retry attempt limit).	Error response packet (TAG = CUB number)
Protocol Errors								
0	1	1	0	0	0	0	Invalid command: An unsupported command existed in a request packet.	Write response packet (TAG = tag of corresponding request)
0	1	1	0	0	0	1	Invalid length: An invalid length was specified for the given command in a request packet.	Read response packet (TAG = tag of corresponding request)
Vault Critical Errors								

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 16: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type
6	5	4	3	2	1	0		
1	1	0	0	0	0	0	Vault 0 critical error: Command/Address parity error has not cleared after the command/address retry count threshold has been met. Vault 0 command execution is halted until a reset is performed.	Error response packet (TAG = CUB number)
1	1	0	0	0	0	1	Vault 1 critical error	
1	1	0	0	0	1	0	Vault 2 critical error	
1	1	0	0	0	1	1	Vault 3 critical error	
1	1	0	0	1	0	0	Vault 4 critical error	
1	1	0	0	1	0	1	Vault 5 critical error	
1	1	0	0	1	1	0	Vault 6 critical error	
1	1	0	0	1	1	1	Vault 7 critical error	
1	1	0	1	0	0	0	Vault 8 critical error	
1	1	0	1	0	0	1	Vault 9 critical error	
1	1	0	1	0	1	0	Vault 10 critical error	
1	1	0	1	0	1	1	Vault 11 critical error	
1	1	0	1	1	0	0	Vault 12 critical error	
1	1	0	1	1	0	1	Vault 13 critical error	
1	1	0	1	1	1	0	Vault 14 critical error	
1	1	0	1	1	1	1	Vault 15 critical error	
1	0	1	0	0	0	0	Vault 16 critical error (8-link device only)	
1	0	1	...	...	...	...	Vault 17–30 (8-link device only); Increment ERRSTAT[3:0] sequentially for each incrementing vault.	
1	0	1	1	1	1	1	Vault 31 (8-link device only)	
Fatal Errors								
1	1	1	0	0	0	0	Link 0 retry failed and was unsuccessful after the retry limit was reached.	Error response packet (TAG = CUB number)
1	1	1	0	0	0	1	Link 1 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	0	1	0	Link 2 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	0	1	1	Link 3 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	1	0	0	Link 4 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	1	0	1	Link 5 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	1	1	0	Link 6 retry failed and was unsuccessful after the retry limit was reached.	
1	1	1	0	1	1	1	Link 7 retry failed and was unsuccessful after the retry limit was reached.	

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 16: ERRSTAT[6:0] Bit Definitions (Continued)

ERRSTAT Bit							Description	Response Packet Type
6	5	4	3	2	1	0		
1	1	1	1	0	0	0	Input buffer overrun: The host has issued too many FLITs and has overrun the link input buffer, or has issued a mode command before the previous mode command has returned a response packet.	Error response packet (TAG = CUB number)
1	1	1	1	0	0	1	Reliability temperature threshold: The internal temperature reliability has been reached. Any additional temperature increase may be harmful.	
1	1	1	1	0	1	1	Recoverable fatal error: HMC has identified a fatal error that will be repaired or recovered after a reset is performed.	
1	1	1	1	1	0	0	Internal fatal error: This group includes but not limited to CRC error at vault controller, internal state machine errors, and internal buffer underrun/overruns.	

9.8 Flow Packets

Flow packets are generated at the link master to convey information for link flow control and link retry control to the link slave on the other end of the link. They are not part of the transaction layer. The link slave extracts the control information from the flow packets and then removes the flow packets from the packet stream without forwarding them to the link input buffer. For this reason flow packets are not subject to normal packet flow control. The link master can generate and transmit flow packets without requiring tokens, and it will not decrement the token count when it transmits flow packets.

Flow packets use the request packet header and tail layouts described in 9.6 “Request Packets” on page 37. The flow packet commands are described in 9.12 “Flow Commands” on page 49.

Some header fields and tail fields within the flow packets are not used and should be 0, as specified in Table 27 on page 50.

9.9 Poisoned Packets

A poisoned packet is uniquely identified by the fact that its CRC is inverted from the correct CRC value. All packet CRC verification logic should include checking for an inverted value of good CRC. A poisoned packet starts out as a good packet (as identified by its good CRC) but for various reasons can become poisoned. The most common cause of a poisoned packet is a link error, as described in 11 “Link Retry” on page 61. Other examples of how a packet becomes poisoned are described in 9.5 “Packet Integrity” on page 36. There are both valid and invalid fields within the header and tail of poisoned packets as is described in the Table 27 on page 50. Note that receiving a poisoned packet does not trigger a link retry.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.10 Request Commands

All request commands are issued by the requester (host or HMC link configured as pass-thru link) to the responder (HMC link configured as host link), using request packets described in 9.6 “Request Packets” on page 37. Every request must have a valid tag included in the request packet header. A corresponding response packet (one that includes the same tag) may be issued by the responder unless it is impossible for the responder to determine the correct tag number of the request due to a request packet CRC error.

All request packets are saved in the link retry buffer as they cross a link. This means that these packets are retried if an error occurs on their transfer.

Table 17: Transaction Layer – Request Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs	Corresponding Response Returned
		5	4	3	2	1	0		
Vendor Specific									
Vendor specific	–	0	0	0	x	x	x	–	–
WRITE Requests									
16-byte WRITE request	WR16	0	0	1	0	0	0	2	WR_RS
32-byte WRITE request	WR32	0	0	1	0	0	1	3	WR_RS
48-byte WRITE request	WR48	0	0	1	0	1	0	4	WR_RS
64-byte WRITE request	WR64	0	0	1	0	1	1	5	WR_RS
80-byte WRITE request	WR80	0	0	1	1	0	0	6	WR_RS
96-byte WRITE request	WR96	0	0	1	1	0	1	7	WR_RS
112-byte WRITE request	WR112	0	0	1	1	1	0	8	WR_RS
128-byte WRITE request	WR128	0	0	1	1	1	1	9	WR_RS
MODE WRITE, BIT WRITE, and ATOMIC Requests									
MODE WRITE request	MD_WR	0	1	0	0	0	0	2	MD_WR_RS
BIT WRITE request	BWR	0	1	0	0	0	1	2	WR_RS
Dual 8-byte add immediate	2ADD8	0	1	0	0	1	0	2	WR_RS
Single 16-byte add immediate	ADD16	0	1	0	0	1	1	2	WR_RS
POSTED WRITE Requests									
16-byte POSTED WRITE request	P_WR16	0	1	1	0	0	0	2	None
32-byte POSTED WRITE request	P_WR32	0	1	1	0	0	1	3	None
48-byte POSTED WRITE request	P_WR48	0	1	1	0	1	0	4	None
64-byte POSTED WRITE request	P_WR64	0	1	1	0	1	1	5	None
80-byte POSTED WRITE request	P_WR80	0	1	1	1	0	0	6	None
96-byte POSTED WRITE request	P_WR96	0	1	1	1	0	1	7	None
112-byte POSTED WRITE request	P_WR112	0	1	1	1	1	0	8	None
128-byte POSTED WRITE request	P_WR128	0	1	1	1	1	1	9	None
POSTED BIT WRITE, and POSTED ATOMIC Requests									
Posted BIT WRITE request	P_BWR	1	0	0	0	0	1	2	None
Posted dual 8-byte add immediate	P_2ADD8	1	0	0	0	1	0	2	None
Posted single 16-byte add immediate	P_ADD16	1	0	0	0	1	1	2	None
Read Requests									
16-byte READ request	RD16	1	1	0	0	0	0	1	RD_RS
32-byte READ request	RD32	1	1	0	0	0	1	1	RD_RS
48-byte READ request	RD48	1	1	0	0	1	0	1	RD_RS

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 17: Transaction Layer – Request Commands (Continued)

Command Description	Symbol	CMD Bit						Packet Length in FLITs	Corresponding Response Returned
		5	4	3	2	1	0		
64-byte READ request	RD64	1	1	0	0	1	1	1	RD_RS
80-byte READ request	RD80	1	1	0	1	0	0	1	RD_RS
96-byte READ request	RD96	1	1	0	1	0	1	1	RD_RS
112-byte READ request	RD112	1	1	0	1	1	0	1	RD_RS
128-byte READ request	RD128	1	1	0	1	1	1	1	RD_RS
Mode Read Requests									
MODE READ request	MD_RD	1	0	1	0	0	0	1	MD_RD_RS
Vendor Specific									
Vendor specific	–	1	1	1	x	x	x	–	–

Notes: 1. All register values not defined in this table are reserved.

9.10.1 READ and WRITE Request Commands

READ and WRITE request commands are issued by the requester to access the DRAM memory. They are block commands that access a contiguous number of memory-addressable bytes with 16-byte granularity, meaning they can begin and end on any 16-byte boundary up to a maximum length of 128 bytes. Due to the 16-byte granularity of READ and WRITE request commands, the four LSBs of the request address are ignored. As shown in Table 17, there are unique commands for each supported memory access length.

READ and WRITE operations will not cross a row (page) boundary within the DRAM array. The maximum block size selected in the address map mode register determines the boundary at which the addressing will wrap if the block access goes beyond the end of the maximum block size. All read requests are acknowledged with a read response packet that includes the tag of the request and the data payload. All write requests (nonposted) are acknowledged either with an explicit write response packet that includes the tag of the write request or, if supported, with the write request tag embedded in the TGA field of some other response packet. If a READ or WRITE request command specifies a data payload length that is longer than the configured maximum block size, it will be treated as an invalid command.

9.10.2 POSTED WRITE Request Commands

POSTED WRITE request commands operate similarly to standard write requests, except that there is no response returned to the requester. As the HMC does not use the TAG field of posted write requests, the host can populate this field with any value. If an error occurs during the execution of the write request, an Error Response packet will be generated indicating the occurrence of the error to the host. Note that in this case, the write request's tag will not be included in the Error Response packet.

9.10.3 ATOMIC Request Commands

Atomic requests involve reading 16 bytes of data from DRAM (as accessed by the request address field), performing an operation on the data through the use of a 16-byte operand (also included in request packet), and then writing the results back to the same location in DRAM. The read-update-write sequence occurs atomically, meaning that no other request can access the same bank until the write of the atomic request is complete. Atomic requests are similar to a 16-byte write request in that they have a 16-byte data

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

payload in the request packet, and a write response may or may not be returned (dependent on posted or nonposted request). The data resulting from the ATOMIC operation is not returned in a response command. The following two sections describe the currently defined ATOMIC operations supported by the HMC.

9.10.3.1 Dual 8-Byte Add Immediate

This atomic request performs two parallel ADD IMMEDIATE operations. Each add operation uses an 8-byte memory operand from DRAM and a 4-byte two's complement signed integer immediate operand from the atomic request data payload. If the result exceeds 2^{63} the carry-out of the most significant bit is dropped and the result rolls over. The results are written back to DRAM, overwriting the original memory operands.

Table 18: Operands from DRAM

Byte	15 (MSB)	14	13	12	11	10	9	8 (LSB)	7 (MSB)	6	5	4	3	2	1	0 (LSB)[1]
Field	Memory operand 2								Memory operand 1							

Notes: 1. Corresponds to the start of the 16-byte aligned address provided in the request address field.
2. Each byte field contains bit positions [7(msb):0(lsb)].

Table 19: Immediate Operands Included in Atomic Request Data Payload

Byte	15	14	13	12	11 (MSB)	10	9	8 (LSB)	7	6	5	4	3 (MSB)	2	1	0 (LSB)
Field	4 bytes of zeros				Imm operand 2				4 bytes of zeros				Imm operand 1			

Notes: 1. Each byte field contains bit positions [7(msb):0(lsb)].

9.10.3.2 Single 16-Byte Add Immediate

This request performs a single ADD IMMEDIATE operation. It uses a 16-byte memory operand from DRAM and an 8-byte two's complement signed integer immediate operand from the atomic request data payload. If the results exceeds 2^{127} the carry-out of the most significant bit is dropped and the result rolls over. The result is written back to the DRAM, overwriting the original memory operand.

Table 20: Operand from DRAM

Byte	15 (MSB)	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0 (LSB)[1]
Field	Memory operand 1															

Notes: 1. Corresponds to the start of the 16-byte aligned address provided in the request address field.
2. Each byte field contains bit positions [7(msb):0(lsb)].

Table 21: Immediate Operand Included in Atomic Request Data Payload

Byte	15	14	13	12	11	10	9	8	7 (MSB)	6	5	4	3	2	1	0 (LSB)
Field	8 bytes of zeros								Immediate operand 1							

Notes: 1. Each byte field contains bit positions [7(msb):0(lsb)].

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.10.4 MODE READ and WRITE Request Commands

MODE READ and WRITE request commands access the internal hardware mode and status registers within the logic layer of the HMC. Each mode request accesses up to 32 contiguous bits of the register.

The address field within the mode request packet is redefined to provide a register address, a start field, and a size field as shown in Table 22. MODE READ and WRITE request commands must include address values that point to the registers defined in Section 9. A MODE READ request command that contains an address that points to a non-existent register will get a MODE READ response back that includes all 0's within the data payload. A MODE WRITE request that points to a non-existent register will not be performed.

Table 22: Mode Request Addressing

Field	Bits	Number of Bits	Description
Unused	33:32	2	Unused bits
Start	31:27	5	**Start field:** Encoded to provide a value from 0 to 31 that points to the starting bit position within the 32 bits of data, 0 being the LSB position and 31 being the most significant bit (MSB) position.
Size	26:22	5	**Size field:** Indicates the number of contiguous bits to read or write (from 1 to 32), and is encoded as follows: 0x0 = 32 bits, 0x1–0x1F = 1–31 bits. This implies that the minimum size is 1 bit
Register address	21:0	22	**Register address field:** Points to a specific mode or status register. All registers are 32 bits or less in size.

Notes: 1. A full 32-bit access would be 0 in both the start and size fields.

The valid data bytes within the data payload in the mode write request packet and the mode read response packet are right-justified at the four least significant bytes as shown in Table 23.

Table 23: Valid Data Bytes

Byte	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Unused												Data			

If less than 32 bits are being accessed, as determined by the start and size fields within the address, the valid data bits are right-justified to the LSBs within the 4-byte data field.

The requester can only issue one mode command at a time and must wait for the corresponding response before issuing another mode command. Because of this requirement, the HMC does not support posted MODE WRITE requests.

9.10.5 BIT WRITE Command

The BIT WRITE request is a 2-FLIT request packet that provides a 0- to 8-byte write capability as indicated by write data mask bits included in the data payload. The vault controller performs a READ-UPDATE-WRITE operation to the DRAM to merge the write data bytes with the existing bytes in the memory. The three LSBs of the byte address field within the address in the request packet header must be 000 when addressing an 8-byte block. The specific bits to be written within the 8-byte block are identified by the data mask bits in bytes[15:8] of the 16-byte data payload within the request packet as shown in Table 24.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 24: Request Packet Bytes to be Written

Byte	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Data mask								Write data							

Notes: 1. Data mask: Each bit inhibits writing the corresponding write data bit when 1.
2. Write data: Up to eight bytes of byte-aligned write data.

The bit(s) to be written must be positioned in the correct bit position within the write data field, as identified with the cleared bit(s) in the data mask field. For example, if the data mask = 0x00FFFFFFFFFF00FF, then bytes 7 and 1 of the write data field will be written into the same bytes in the memory. If the data mask = 0xFFFFFFFFFFFFFFFF, the READ-UPDATE-WRITE operation will be performed with the original value of the data being rewritten into the memory.

9.11 Response Commands

All response commands are issued by the responder (HMC link configured as host link) to the requester (host or HMC link configured as pass-thru link), using response packets described in 9.7 “Response Packets” on page 38. Every response other than the error response includes a valid tag that matches the tag in its corresponding request. All response packets are saved in the link retry buffer as they cross a link, which means they are retried if an error occurs on the link.

Table 25: Transaction Layer – Response Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs
		5	4	3	2	1	0	
READ response	RD_RS	1	1	1	0	0	0	1 + data FLITs
WRITE response	WR_RS	1	1	1	0	0	1	1
MODE READ response	MD_RD_RS	1	1	1	0	1	0	2
MODE WRITE response	MD_WR_RS	1	1	1	0	1	1	1
ERROR response	ERROR	1	1	1	1	1	0	1
Reserved	–	1	1	1	1	1	1	-

9.11.1 READ and MODE READ Response Command

A READ or MODE READ response command is issued by the responder to return requested data to the requester. A read response packet always includes data payload along with the header and tail, and it includes the same tag as its corresponding read request. If an error occurred such that the data is not valid, as indicated in the DINV field in the response tail, the data payload FLITs are still present in the response packet but they hold invalid data. For this reason, the response packet lengths are always consistent with the corresponding request. If supported, a read response packet could also include a tag from a write request embedded in the TGA field of the response packet tail. This acts as a write response, providing a positive acknowledgment of the completion of a successful write request with no errors. The write response tag will always be returned to the same link where the original write request was received.

A MODE READ command must include address values that point to the registers defined in Section 10. A MODE READ request command that contains an address that points to a non-existent register will get a MODE READ response back that includes all 0s within the data payload, and the DINV bit will not be set.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

9.11.2 WRITE and MODE WRITE Response Command

The WRITE or MODE WRITE response command provides an explicit write response packet that returns the tag for the original write request. If errors occurred during the transfer or execution of the WRITE request command, status will be included in the ERRSTAT field of the write response packet. A lack of error status bits indicates that the write was successful.

If supported, any write response tag that is included in a TGA field of another response indicates that there are no associated errors to be reported for the corresponding write request and that the write was successful. Mode write responses cannot be embedded within the TGA field and are sent through the MODE WRITE response command only.

A MODE WRITE command must include address values that point to the registers defined in Section 10. A MODE WRITE request command that contains an address that points to a non-existent register will get a MODE WRITE response back, however the request will not be performed.

9.11.3 ERROR Response Command

The HMC uses the READ, MODE READ, WRITE, and MODE WRITE response commands to report errors or flags that are associated with a specific request and tag. These error reports are included in the ERRSTAT field of the response packet.

The ERROR response command is used by the HMC to report a critical event when the request type and/or the original request tag are not known. The three LSBs of the ERROR response command packet tag field contain the values the CUB[2:0] signals are tied to. TAG[2:0] = CUB[2:0] and TAG[8:3] = 0. This gives the host visibility to which HMC the Error Response packet originated from in a topology with multiple HMCs. The ERROR response command is unsolicited and can be configured to transmit ERROR response commands on any or all links. The following situations cause the HMC to transmit an ERROR response command packet:

- The responder receives a request packet from the requester that is successfully transferred across a link but is internally corrupted before it reaches its destination. In this case the original command and tag are lost and the responder cannot generate a proper response packet or the necessary tag to close the transaction.
- The HMC has an internal error not associated with a transaction.
- The HMC reports a condition or status to the host.

9.12 Flow Commands

Flow commands facilitate retry and flow control on the link and are not part of the transaction layer. They are generated at the link master to send information to the other end of the link when no other link traffic is occurring or when a link retry sequence is to be initiated. Flow commands are not forwarded from that point.

Flow commands use the request packet header and tail layouts described in “Request Packets” on page 37, but are not considered requests, and do not have corresponding response commands. Flow commands are not subject to token flow control. The link master is allowed to send a flow command even if there are no tokens available.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 26: Flow Commands

Command Description	Symbol	CMD Bit						Packet Length in FLITs
		5	4	3	2	1	0	
Null	NULL	0	0	0	0	0	0	NA
Retry pointer return	PRET	0	0	0	0	0	1	1
Token return	TRET	0	0	0	0	1	0	1
Init retry	IRTRY	0	0	0	0	1	1	1

9.12.1 NULL Command

A NULL command field of all zeros, along with all zeros in the rest of the FLIT, define a NULL FLIT. NULL FLITs are driven on the link when there are no packets to be transmitted. (Null FLITs are also used during link initialization.) Note that a FLIT of all zeros will have a correct CRC. A poisoned version of the NULL FLIT is not supported.

9.12.2 RETRY POINTER RETURN (PRET) Command

This command is issued by the link master to return retry pointers when there is no other link traffic flowing at the time the pointer is to be returned. It is a single-FLIT packet with no data payload. PRET packets are not saved in the retry buffer, so their SEQ and FRP fields should be set to 0. Tokens should not be returned in these packets, so the RTC field should be set to 0.

9.12.3 TOKEN RETURN (TRET) Command

This command is issued by the link master to return tokens when there is no other link traffic flowing at the time the token is to be returned. It is a single-FLIT packet with no data payload. Token return packets are saved in the retry buffer, so their SEQ and FRP fields are valid.

9.12.4 INIT RETRY (IRTRY) Command

A programmable stream of INIT RETRY (IRTRY) command packets is issued by the link master when it is in the Retry_Init state. This is a single-FLIT packet with no data payload. IRTRY command packets are not saved in the retry buffer, so their SEQ field should be set to 0. Tokens should not be returned in these packets, so the RTC field should also be set to 0 (see 11.2.5.2.1.2 “ LinkRetry_Init State” on page 68).

9.13 Valid Field Command Summary

Table 27: Valid Field and Command Summary Table

Symbol	Request		Response			Flow				Poisoned[2]
	READ	WRITE	READ	WRITE	ERROR	NULL	PRET	TRET	IRTRY	All
ADRS	V	V	n/a	n/a	n/a	0	0	0	0	NV
CMD	V	V	V	V	V	0	V	V	V	NV
CRC	V	V	V	V	V	0	V	V	V	V[2]
DLN	V	V	V	V	V	0	V	V	V	V[3]
ERRSTAT	n/a	n/a	V	V	V	n/a	n/a	n/a	n/a	NV
FRP	V	V	V	V	V	0	0	V	V[4]	V
LNG	V	V	V	V	V	0	V	V	V	V[3]

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 27: Valid Field and Command Summary Table

Symbol	Request		Response			Flow				Poisoned[2]
	READ	WRITE	READ	WRITE	ERROR	NULL	PRET	TRET	IRTRY	All
RRP	V	V	V	V	V	0	V	V	V	V
RTC	V	V	V	V	V	0	0	V	0	V
SEQ	V	V	V	V	V	0	0	V	0	V
TAG	V	V[5]	V	V	V[6]	0	0	0	0	NV
TGA (optional)	n/a	n/a	V	V	V	n/a	n/a	n/a	n/a	NV

Notes:
1. V: Valid field
   NV: Invalid field - Do not use
   n/a: Not applicable, field doesn't exist
   0: Field must be 0
2. A poisoned packet is defined by an inverted CRC.
3. DLN and LNG are valid as long as they match each other.
4. FRP[0] = "Start Retry" Flag
   FRP[1] = "Clear Error Abort" Flag
5. The TAG field is not used within POSTED WRITE requests, whereby the host can use any value for TAG, including zero.
6. TAG[2:0] = CUB[2:0] and TAG[8:3] = 0

9.14 Transaction Layer Initialization

During link initialization, after the responder stops sending TS1 patterns on the link, the link layer is considered active. Both ends of the link are in the active state, sending NULL FLITs, as described in Step 11 in Section 6 “Power-On and Initialization” on page 21. At this point, the HMC will continue internal initialization.

Transaction layer initialization continues with the following steps:

1. After internal initialization is complete, the responder will send one or more TRET packets that will transfer the 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.
3. 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 when the transaction packets can start.

Within an HMC, all links that are configured as pass-thru links must complete their transaction layer initialization before the links configured as host links begin to send TRET packets upstream. This ensures that all links within a multi-cube topology are initialized once the link configured as a host link in source mode has finished its transaction layer initialization.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

10 Configuration and Status Registers

There are two methods supported for accessing the HMC configuration and status registers:

Direct access: This type of access uses the in-band MODE WRITE and MODE READ request commands or the sideband (JTAG/I2C) to directly read or write the configuration and status registers. Tables 33–41 contain the configuration and status registers accessible by direct access that are supported by this specification. All other register functions are considered vendor specific.

Indirect access through the external data interface (ERI): This type of access utilizes four External Data Registers (see Tables 28–31) and an External Request Register (see Table 32) to indirectly access the HMC configuration and status registers. The external data and request registers are accessible through MODE WRITE and MODE READ requests or sideband (JTAG/I2C). Specific configuration and status registers accessible through the ERI are not contained within this specification, only the framework to enable this type of register access. See Figure 16 on page 53 for the steps required to access the configuration and status registers through the ERI.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 16: Configuration and Status Register Access

The variables in the Attributes column in Tables 28–41 are defined as the following:

- RW: Register can be read from and written to
- RO: Register can only be read from
- RWS: Register self clears after being written to

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 28: External Data Register 0
Register address = 0x2B0000

Name	Start Bit	Size	Type	Reset Value	Description
External Data Register 0	0	32	RW	Register dependent	Multipurpose register used to access configuration and status registers

Table 29: External Data Register 1
Register address = 0x2B0001

Name	Start Bit	Size	Type	Reset Value	Description
External Data Register 1	0	32	RW	Register dependent	Multipurpose register used to access configuration and status registers

Table 30: External Data Register 2
Register address = 0x280002

Name	Start Bit	Size	Type	Reset Value	Description
External Data Register 2	0	32	RW	Register dependent	Multipurpose register used to access configuration and status registers

Table 31: External Data Register 3
Register address = 0x2B0003

Name	Start Bit	Size	Type	Reset Value	Description
External Data Register 3	0	32	RW	Register dependent	Multipurpose register used to access configuration and status registers

Table 32: External Request Register
Register address = 0x2B0004

Name	Start Bit	Size	Type	Reset Value	Description
REGISTER REQUEST commands	0	8	RW	0x00	0x00: NO OPERATION 0x01: REGISTER WRITE 0x02: REGISTER WRITE NEXT[1] 0x03: REGISTER READ 0x04: REGISTER READ NEXT[2] 0x05 - 0xFE: Vendor specific 0xFF: Init continue. This bit loaded by host after it has completed its register configuration. Setting the bit allows for internal HMC configuration to continue.
Type	8	8	RW	0x00	Vendor specific
Target location	16	6	RW	0x00	0x00–0x3E: Target location of command (including but not limited to links and vaults) 0x3F: Used to load global registers. Also used to broadcast to all links.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 32: External Request Register (Continued)
Register address = 0x2B0004

Name	Start Bit	Size	Type	Reset Value	Description
Size	22	4	RW	0x0	0x0: No data registers associated with request 0x0 - 0x4: Number of data registers associated with request 0x5 - 0x1F: Vendor specific
External request status	26	5	RO	0x00	0x00: Request was successful 0x01: Request caused an internal error 0x02: Request was invalid 0x03: Operation complete. The HMC sets this bit when the last group of data registers for a status request is sent to the host. Bit also set by the HMC after the last group of data registers expected from a configuration request. 0x04 - 0x1F: Vendor specific
Start	31	1	RW	0x0	Host sets to '1' to inform HMC there is a valid external request. HMC sets to '0' after request is complete. Host polls bit and when it is '0', it can issue subsequent EXTERNAL REQUEST commands as well as read the External Request Status bits to determine the success of the pending EXTERNAL REQUEST command.

Notes:
1. Used for loading register configuration types that include more than four external data registers. The first four external data registers loaded will be followed by a REGISTER WRITE command and all subsequent external data registers loaded will be followed by a REGISTER WRITE NEXT command.
2. Used for reading Register Status Types that include more than four external data registers. A REGISTER READ command is loaded to read the first four external data registers. A REGISTER READ NEXT command is issued to read all subsequent external data registers.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 33: Global Configuration
Register address = 0x280000

Name	Start Bit	Size	Type	Reset Value	Description
Vendor Specific	0	4	–	–	Vendor specific
Stop on Fatal Error	4	1	RW	0x0	When set to 1, an internal fatal error will cause the HMC to stop returning response packets back to the host as well as stop transaction execution at the vaults.
Clear Error	5	1	RWS	0x0	Writing a 1 clears all internal captured errors and error status registers. This bit auto-clears.
Warm Reset	6	1	RWS	0x0	Writing a 1 initiates a warm reset sequence. This bit auto-clears.
Vendor Specific	7	24	–	–	Vendor specific

Table 34: Link Configuration
Register address: Link 0 = 0x240000, Link 1 = 0x250000, Link 2 = 0x260000, Link 3 = 0x270000

Name	Start Bit	Size	Type	Reset Value	Description
Link Mode	0	2	RW	0x1	0x0: Link is not used 0x1: Link is a Host Link and a Source Link 0x2: Link is a Host Link and NOT a Source Link 0x3: Link is a Pass-thru Link
Link Response Open Loop Mode	2	1	RW	0x0	0x0: Response Open Loop Mode is off 0x1: Response Open Loop Mode is on
Link Packet Sequence Detection	3	1	RW	0x1	0x0: Packet sequence detection is off 0x1: Packet sequence detection is on
Link CRC Detection	4	1	RW	0x1	0x0: Link CRC error detection is off 0x1: Link CRC error detection is on
Link Duplicate Length Detection	5	1	RW	0x1	0x0: DLN detection is off 0x1: DLN detection is on
Packet Input Enable	6	1	RW	0x1	0x0: Decode and parsing of incoming packets is disabled 0x1: Decode and parsing of incoming packets is enabled
Packet Output Enable	7	1	RW	0x0	0x0: Transmission of outgoing packets disabled 0x1: Transmission of outgoing packets enabled
Inhibit Link Down Mode	8	1	RW	0x0	When set to 1, the HSS PLLs will remain on regardless the state of LxRXPS signals (applies only to links not reset in down mode).
Link Descramble Enable	9	1	RW	0x1	0x0: Receiver scramblers are disabled 0x1: Receiver descramblers are enabled
Link Scramble Enable	10	1	RW	0x1	0x0: Transmit scramblers are disabled 0x1: Transmit scramblers are enabled
Error Response Packet	11	1	RW	0x1	When set to 1, the HMC will send Error Response packets on this link.
Vendor Specific	12	20	–	–	Vendor specific

Table 35: Link Run Length Limit
Register address: Link 0 = 0x240003, Link 1 = 0x250003, Link 2 = 0x260003, Link 3 = 0x270003

Name	Start Bit	Size	Type	Reset Value	Description
Vendor Specific	0	16	–	–	Vendor specific

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 35: Link Run Length Limit (Continued)
Register address: Link 0 = 0x240003, Link 1 = 0x250003, Link 2 = 0x260003, Link 3 = 0x270003

Name	Start Bit	Size	Type	Reset Value	Description
Transmit Run Length Limit	16	8	RW	0x00	Sets the run length limit allowed on each lane (that is, the maximum number of zeros or ones that can be sent consecutively before a limiting circuit inserts a transition in the data stream to assure proper clock/data recovery at the far end receiver). A value of 50 (0x32) or more is required in this register to assure proper run limiting operation. The default value is 0x00 (no limiting).
Vendor specific	24	8	–	–	Vendor specific

Table 36: Link Retry
Register address: Link 0 = 0x0C0000, Link 1 = 0x0D0000, Link 2 = 0x0E0000, Link 3 = 0x0F0000

Name	Start Bit	Size	Type	Reset Value	Description
Retry Status	0	1	RW	0x0	0x0: Retry is disabled 0x1: Retry is enabled
Retry Limit	1	3	RW	0x3	Controls the number of attempts before Link Error in ERRSTAT field indicates retry attempt limit has been met.
Retry Timeout Period	4	3	RW	0x5	0x0: 154ns 0x1: 205ns 0x2: 307ns 0x3: 384ns 0x4: 614ns 0x5: 820ns 0x6: 1229ns 0x7: 1637ns
Vendor Specific	7	1	–	–	Vendor specific
Init Retry Packet Transmit Number	8	6	RW	0x08	0x2 – 0x3F: The number of IRTRY packets transmitted from Link Master during LinkRetry_Init is approximately 4 times the number specified in this field.
Vendor Specific	14	2	–	–	Vendor specific
Init Retry Packet Receive Number	16	6	RW	0x10	The number of IRTRY packets that the link slave will detect before it clears Error Abort Mode.
Vendor Specific	24	8	–	–	Vendor specific

Table 37: Input Buffer Token Count
Register address: Link 0 = 0x040000, Link 1 = 0x050000, Link 2 = 0x060000, Link 3 = 0x070000

Name	Start Bit	Size	Type	Reset Value	Description
Link Input Buffer Max Token Count	0	8	RW	0x64	Value represents the buffer space available in the HMC's link input buffer when it is empty.
Vendor Specific	8	24	–	–	Vendor specific

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 38: Address Configuration
Register address = 0x2C0000

Name	Start Bit	Size	Type	Reset Value	Description
Address Mapping Mode	0	4	RW	0x2	0x0: 32-byte MAX block size. Low-interleave address mapping with the byte address set as the lowest 5 bits of the ADRS field in the request header along with the default vault and bank mapping. 0x1: 64-byte MAX block size. Low-interleave address mapping with the byte address set as the lowest 6 bits of the ADRS field in the request header along with the default vault and bank mapping. 0x2: 128-byte MAX block size. Low-interleave address mapping with the byte address set as the lowest 7 bits of the ADRS field in the request header along with the default vault and bank mapping. 0x3–0x7: Vendor specific. 0x8: User-defined 32-byte MAX block size. Byte address is defined as lowest 5 bits of the ADRS field in the request header along with user defined vault and bank mapping. The user defined vault and bank address start bits and size fields must be nonzero. 0x9: User-defined 64-byte MAX block size. Byte address is defined as lowest 6 bits of the ADRS field in the request header along with user defined vault and bank mapping. The user defined vault and bank address start bits and size fields must be nonzero. 0xA: User-defined 128-byte MAX block size. Byte address is defined as lowest 7 bits of the ADRS field in the request header along with user-defined vault and bank mapping. The user-defined vault and bank address start bits and size fields must be nonzero. 0xB–0xF: Vendor specific
User-defined Vault Register Address	4	5	RW	0x00	Encoded field that specifies corresponding bit of ADRS to be LSB of vault address.
User-defined Bank Register Address	9	5	RW	0x00	Encoded field that specifies corresponding bit of ADRS to be LSB of bank address.
Vendor Specific	14	18	–	–	Vendor specific

Table 39: Vault Control Register
Register address = 0x108000

Name	Start Bit	Size	Type	Reset Value	Description
DRAM Initialization Mode	0	2	RW	0x0	0x0: DRAM initialization is disabled (patrol scrubbing must also be disabled). 0x1: Initialize DRAM to poisoned ECC code (or nonwritten ECC code if implemented) 0x2: Initialize DRAM to user pattern. 0x3: Reserved
Hard SBE Detect	2	1	RW	0x0	Set to 0 to disable re-read to determine if a corrected SBE failure is hard.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 39: Vault Control Register (Continued)
Register address = 0x108000

Name	Start Bit	Size	Type	Reset Value	Description
Demand Scrubbing	3	1	RW	0x1	Set to 0 to disable a scrub write to DRAM of corrected data when an SBE is detected on a transaction read request.
Patrol Scrubbing	4	1	RW	0x1	Set to 0 to disable a scrub write to DRAM of corrected data when an SBE is detected on a patrol read request.
Packet CRC Detection	5	1	RW	0x1	Set to 0 to disable the CRC check performed on incoming packets to the vault controllers. This allows the vault controller to execute request packets that have CRC errors.
Command/Address Retry Count	6	3	RW	0x1	When a command or address parity error is detected on DRAM commands, the vault controllers will perform up to the number of DRAM command retries specified in this field. If this field is 0, DRAM command retry is disabled.
Data ECC Correction	9	1	RW	0x1	Set to 1 to disable correction of a SBE if detected by the vault controller. Data from the DRAM will be returned unmodified in the response packet. Demand and patrol scrubbing are also disabled.
Vendor Specific	10	1	–	–	Vendor specific
Hard SBE Repair	11	1	RW	0x1	Set to 1 to invoke a dynamic repair operation when any hard SBE is detected.
Enable SBE Report	12	1	RW	0x1	Set to 1 to report any SBEs that occurred on a read request within the ERRSTAT field of the read response.
Vendor Specific	13	19	–	–	Vendor specific

Table 40: Features
Register address = 0x2C0003

Name	Start Bit	Size	Type	Reset Value	Description
Cube Size	0	4	RO	Product specific	0x00: 2GB 0x01: 4GB 0x02: 8GB 0x03: 16GB 0x4–0xF: Vendor specific
Number of Vaults	4	4	RO	Product specific	0x0: 16 vaults 0x1: 32 vaults 0x2–0xF: Vendor specific
Number of Banks per Vault	8	4	RO	Product specific	0x0: 8 banks 0x1: 16 banks 0x2–0xF: Vendor specific
PHY	12	4	RO	Product specific	0x0: HMC-15G-SR 0x1: HMC-10G-USR 0x2–0xF: Vendor specific
Vendor Specific	16	16	–	–	Vendor specific

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 41: Revisions and Vendor ID
Register address = 0x2C0004

Name	Start Bit	Size	Type	Reset Value	Description
Vendor ID	0	8	RO	Vendor specific	Vendor ID
Product Revision	8	8	RO	Vendor specific	Product revision
Protocol Revision	16	8	RO	Vendor specific	Protocol revision
PHY Revision	24	8	RO	Vendor specific	PHY revision

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

11 Link Retry

The link transmits data at a very high speed. The data transmission bit-failure rate requires a mechanism for correcting failures to increase system reliability and availability. Link retry is a fault-tolerant feature that recovers the link when link errors occur. Link packet integrity is insured with a CRC code that resides in the tail of every packet. It covers all bits in the packet, including the header, any data payload, and the tail. CRC is generated at the link master before the packet bits are serialized and transmitted, and is checked at the link slave after the packet bits are received and deserialized. A mismatch indicates that some of the data within the packet has changed since the original CRC code was generated at the transmitting end of the link. An additional check is provided using an incrementing sequence number included in every packet. If the CRC check is successful upon receipt of the packet, the link slave will verify that the sequence number has a +1 value compared to the last successful packet received.

The link master saves a copy of each packet transmitted across the link. Each packet is held until an acknowledge occurs indicating that the packets have been received without errors. This acknowledgment comes from a retry pointer (included in every transmitted packet); it is returned to the link master embedded in packets traveling in the opposite direction on that link.

If the link slave detects a packet error, it enters error abort mode. This results in the failed packet being poisoned or dropped and all subsequent packets being dropped. Error Abort mode triggers the local link master to send a StartRetry flag back to the transmitting end using the other side of the link. The poisoned and dropped packets are then retransmitted by the link master during the retry sequence.

Figure 17 on page 62 illustrates an example implementation of the link retry blocks on both ends of a link. The forward retry pointer (FRP) and return retry pointer (RRP) extract and embed blocks are color coded to show the path the retry pointer takes for each link direction.

Note that the link is symmetrical and in the following discussions the requester can be either device A or device B. Similarly, the responder can be either device A or device B.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 17: Implementation Example of Link Retry Block Diagram

11.1 Retry Pointer Description

The retry pointer represents the packet's position in the retry buffer. The retry pointer transmitted with the packet points to the retry buffer location + 1 (to which the tail is being written). This is the starting address of the next packet to be transmitted.

There are two retry pointer versions:

- The FRP as it travels in the forward direction on the link
- The RRP, as the same retry pointer travels back on the other side of the link, completing the round trip and eventually getting back to the retry buffer control logic in the link master

The FRP is sent with every packet that is part of the normal packet flow (other than IRTRY packets and PRET packets described later). Note that the retry buffer is addressed using FLIT addresses, and the retry pointer represents a FLIT position in the retry buffer. Even though a packet may be multiple FLITs, the retry pointer will always point to the header FLIT position of a packet.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

There are two pointer fields in the tail of a packet:

- FRP field: This is the forward retry pointer as the packet is traveling in the forward direction on the link. The FRP field is ignored in PRET packets and is redefined to contain two retry control flags in IRTRY packets.
- RRP field: This is the return retry pointer embedded in any packet traveling in the return direction of the link. If the return direction has no normal packet traffic, when an FRP is extracted in the local link slave and sent to the link master, a PRET packet is then created in the link master and the FRP is loaded into the RRP field. The RRP is not associated with the packet in which it is embedded, other than to "ride along" to get back to the other end of the link. The link is symmetrical in both directions, so RRPs are embedded in packet traffic moving in both directions. The RRP field of the packet is always valid, including IRTRY packets and PRET packets.

11.2 Link Master Retry Functions

Figure 18: Implementation Example of a Retry Buffer

In this implementation example, a retry buffer is used to save packets for potential retransmission if a link error occurs. The retry buffer relies on a write pointer and read pointer to track the status of the FLITs stored within it. The write pointer is advanced with valid outgoing FLITs as determined by the packet stream traveling out of the link master. As FLITs are written into the retry buffer, they are considered to be pending FLITs, because they may need to be re-transmitted if an error occurs. They remain pending FLITs until the read pointer is advanced past them, indicating that one or more returned retry pointers were received, representing a successful packet transmission.

The read pointer in this example is used to read data from the retry buffer during a retry sequence. If no errors occur, retry pointers are returned (RRPs) and used to advance the read pointer. As RRPs are received, the read pointer will rotate around the retry buffer address space, following the write pointer, and turn pending FLITs into retired FLITs.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Retired FLITs are unneeded residue FLITs that will eventually be overwritten by new pending FLITs as the write pointer advances and wraps around the retry buffer address space.

During a retry sequence, the read pointer is incremented until it equals the write pointer, indicating that all pending FLITs have been pulled out of the retry buffer and retransmitted across the link. All RRPs received during this period are saved in a read pointer hold register until all pending FLITs have been read out of the retry buffer. At the end of the LinkRetry_FLIT state (see 10.2.7.2.1.3 “LinkRetry_FLIT State”), as the retry control switches back to a flow of new packets, the latest value of the read pointer hold register is copied to the read pointer. This action may retire many of the pending FLITs. After the copy is made, any newly arrived RRPs are used to advance the read pointer directly, thus retiring more FLITs.

A threshold is implemented in this example to detect a near-full condition of the retry buffer. The near-full condition is detected by comparing the value of the write pointer to the read pointer. This threshold leaves enough room remaining in the retry buffer for the current packet in transmission to be completely saved in the retry buffer, without overflow, when the near-full detection occurs. At this point, the link master will stop the packet flow to the link.

11.2.1 Forward Retry Pointer Generation

The link master generates the retry pointer and includes it in the FRP field in the tail of the packet being transmitted. The retry pointer is created by using the value where the tail FLIT of the packet is to be saved for potential retransmission, incrementing by +1, and loading it into the FRP field within the tail of the packet. This value points to the location where the header of the next packet to be transmitted will be saved. This FRP value must be included in the CRC generation, which is also loaded into the tail of the packet being transmitted. Note that the FRP field is valid for all packets that are saved for potential retransmission. All packets that are transmitted other than IRTRY, NULL, and PRET packets are saved until the success of their transmission has been acknowledged. If the original packet stream on the link included NULL FLITs between some of the packets, those NULL FLITs are not saved for potential retransmission. However, during a retry sequence, as the retried packet sequence is transmitted, NULL FLITs may be transmitted between packets.

11.2.2 Packet Sequence Number Generation

The link master generates a sequence number for every outgoing packet that is part of the normal packet flow and loads it into the SEQ field within the tail of the packet. The sequence number should be incremented by + 1 for every normal packet. The sequence number should increment by +1 for every packet that is saved for retransmission. It will not be generated for PRET or IRTRY packets, as these are not saved for retransmission. Any packets replayed during a retry sequence already have their sequence number embedded and are re-sent without updating the SEQ field.After the last retried packet is transmitted, the sequence number generator should increment for the next new packet that follows the retried packet stream. This accommodates a sequence checking across a retry sequence, verifying that all packets are sequential before, during, and after the retry sequence. The sequence number register in the link slave should be set to zero upon a cold or warm reset. The first packet sent from the link master after a cold or warm reset should contain a sequence number of 1.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

11.2.3 Forward Retry Pointer Reception and Embedding

One link master function is to return retry pointers that represent packets traveling in the opposite direction on the other side of the link. The link master captures the FRP from the link slave and then embeds it in the RRP field in the tail of the current or next packet being transmitted. At that point it becomes the return retry pointer. Transmitted packets may consist of multiple FLITs that span multiple core cycles, so one or more new FRPs could be received from the local link slave before the previous one has been embedded. In this case, the latest FRP overwrites the previous one. The link master does not have to embed every retry pointer received from the link slave; only the latest one, which supersedes any previous pointer.

If there is no forward traffic being transmitted by the link master when a retry pointer arrives from the link slave, the link master generates a single-FLIT retry pointer return (PRET) packet where it embeds the latest value of the retry pointer into the RRP field. The PRET packet generated for this purpose is not saved in the retry buffer.

If there are no FRPs being sent from the link slave to the link master, the last valid FRP value received from the link slave will be embedded in subsequent packets sent from the link master. However, in this case, no explicit PRET packets need to be generated if the last valid FRP value has been embedded in at least one packet.

The link master must always embed the current value of the retry pointer collection register into the RRP field of every non-NULL packet transmitted, including:

- Normal transactional packets from the link output buffer
- IRTRY packets transmitted during the StartRetry stream
- IRTRY packets transmitted during the LinkRetry sequence
- Pending packets transmitted during the LinkRetry_FLIT state of the LinkRetry sequence. In this case, the old pointer values of the RRP field (previously transmitted) are overwritten by the latest pointer. The CRC must be recalculated for each packet due to the RRP change.

11.2.4 Return Retry Pointer Reception

The link master not only receives the FRPs for packets traveling in the opposite direction from the local link slave, as described in the previous section, but also receives RRPs from the local link slave. Receipt of an RRP completes the loop that the retry pointer travels and acts as an acknowledgement of the successful transmission of the associated packet.

11.2.5 Link Master Sequences

The link master provides one of two different sequences, depending on where the link error occurs, that interrupts the normal packet flow being transmitted from the link master. When only one link error occurs (the most likely case), the local link master adjacent to the link slave that detected the error will execute the StartRetry sequence. The link master transmitting in the direction where the link error occurred will execute the LinkRetry sequence.

11.2.5.1 StartRetry Sequence

When a link slave detects an error, it sets “Error Abort Mode”, which is monitored by the local link master. When this signal activates, it triggers the local link master to initiate its StartRetry Sequence, which interrupts the normal transaction packet flow and transmits a programmable number of single-FLIT IRTRY packets. A packet currently being transmitted must be allowed to complete before starting the IRTRY stream. Within each of

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

these IRTRY packets, the StartRetry flag is set with FRP[0]=1. This stream of IRTRY packets must be contiguous, meaning each IRTRY FLIT must be followed by the next IRTRY FLIT without the insertion of any other types of FLITs/packets, including NULL FLITs, for the duration of the stream. The transmission of the IRTRY stream follows these rules:

- The link master should always embed the latest FRP it received into the RRP field of each IRTRY packet as it does for any outgoing packet.
- None of the IRTRY packets should be loaded into the Retry Buffer.

The execution of the StartRetry sequence also starts a Retry Timer. This is used to monitor the success of the link retry. If the link retry is unsuccessful and Error Abort Mode does not clear, the retry timer will time-out which will initiate the execution of another StartRetry sequence. This will continue until the number of retry attempts equals the retry attempt limit specified in the link retry control register.

11.2.5.1.1 StartRetry Sequence States

11.2.5.1.1.1StartRetry_Begin State

This state enables the completion of an in-flight packet transmission, after which the normal packet flow stops. The in-flight packet must also be saved for retransmission.

StartRetry_Begin State Control

- Enter: When the link master detects the rising edge of the error abort mode signal from the local link slave, OR when the retry timer expires and Error Abort Mode is set.

Actions:

1. Allow any packet currently being transmitted to complete.
2. Stop normal packet flow and enter StartRetry_Init state. This will force the packet stream to be queued-up in the link output FIFO.

- Exit: When the in-flight packet transmission is complete.

11.2.5.1.1.2StartRetry_Init State

In this state, the link master transmits a programmable number of single-FLIT IRTRY packets across the link. These packets are not saved for retransmission, thus their forward retry pointer and sequence number aren't used. Within each IRTRY packet, Bit 0 of the FRP field is redefined to hold the StartRetry flag, which signals the other end of the link to initiate its LinkRetry sequence. The link master also embeds the last FRP value it received in the RRP field of every IRTRY packet. This stream of IRTRY packets accomplishes the following:

- It signals to the link slave on the far end of the link that this end detected a link error.
- It provides the latest value of the retry pointer, embedded in the RRP field, to the other end of the link.

StartRetry_Init State Control

- Set: When StartRetry_Begin state clears
- Action: Start transmission of contiguous stream of IRTRY packets that include the StartRetry flag (FRP[0]=1). The number of packets is determined by the IRTRY packet transmit number field in the Link Retry Control Register. NULL FLITs must not be inserted between IRTRY packets.
- Clear: After the programmable number of IRTRY packets has been transmitted.

When the link master exits the StartRetry_Init state, it reverts back to its normal packet stream that has been accumulating in its link output buffer.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

11.2.5.1.2 Retry Timer

The Retry Timer monitors the success of the current retry attempt by timing the assertion of the Error Abort mode signal from the local link slave. The time required to complete a retry attempt which clears error abort mode is described in Section 10.5, Retry Latency.

The vast majority of link retries will be successful on the first retry attempt, in which case the Error Abort mode will clear out within the normal retry loop time, and the Retry Timer will not affect the delay of the overall retry operation. Given that the retry loop time is less than or equal to the retry buffer full period, the retry timer should be set to at least 3 times the retry buffer full period, as shown in the right most column of Table 42 on page 71, Retry Pointer Loop Time. This is to eliminate the possibility of issuing a second StartRetry before the first retry has a chance of completing during a multiple-error case where both sides of the link may be executing a link retry sequence, before the first Init_Retry/StartRetry stream is allowed to be transmitted.

11.2.5.1.2.1Retry Timer Algorithm

If Error_Abort_Mode = 0

- Retry_Timer = 0
- retry_attempts = 0
- Retry_Timer_expired = 0

ElseIf Retry_Timer = Retry Timeout Period AND retry_attempts < retry_attempt_limit

- Retry_Timer_expired = 1
- Retry_attempts = Retry_attempts + 1
- Retry_Timer = 0

Elseif retry_attempts = retry_attempt_limit

- Set link_retry_failed

Else

- Retry_Timer = Retry_Timer + 1
- Retry_Timer_expired = 0

EndIF

11.2.5.2 LinkRetry Sequence

If the link master receives a StartRetry pulse from the local link slave, it is an indication that the link slave on the other end of the link detected an error, and the link master should initiate its LinkRetry sequence. This sequence will suspend the flow of packets from the link output buffer and send the packets currently being saved for re-transmission.

11.2.5.2.1 LinkRetry Sequence States

The LinkRetry sequence includes three states: LinkRetry_Begin, LinkRetry_Init, and LinkRetry_FLIT. The LinkRetry_FLIT state may or may not be entered depending on if there are any packets to be retransmitted.

11.2.5.2.1.1 LinkRetry_Begin State

This state enables the completion of an in-flight packet transmission, after which the normal packet flow stops. The in-flight packet must also be saved for retransmission.

LinkRetry_Begin State Control

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

- Enter: When the link master receives a StartRetry pulse from the local link slave.
- Actions:
  1. Enable the packet currently in transmission to complete.
  2. Stop normal packet flow and enter next retry state.
- Exit: When the in-flight packet transmission is complete.

11.2.5.2.1.2 LinkRetry_Init State

In this state, the link master transmits a programmable number of single-FLIT IRTRY packets across the link. Within each of these IRTRY packets, the ClearErrorAbort flag will be set (FRP[1]=1). These packets are not saved for retransmission, thus their Forward Retry Pointer and SEQ number are not used. The link master also embeds the last valid FRP it received from its local link slave in the RRP field of every IRTRY packet. The stream of IRTRY packets enables the following:

- The ClearErrorAbort flag enables the link slave section on the far end of the link to detect the link master is in the retry sequence.
- It provides a period of time for the link slave to detect a number of good IRTRY packets that establish confidence in the link.

11.2.5.2.1.3 LinkRetry_FLIT State

In this state, the link master transmits the packets currently being saved for retransmission. These packets have already been transmitted but have been aborted at the receive end of the link. If an error occurs in a packet that this link master had previously transmitted, this state will be entered. If an error occurs only during the transmission of a stream of continuous NULL FLITs and there is not any FLITs currently saved for retransmission, this state will not be entered.

If the link master is retransmitting packets in the LinkRetry_FLIT state and it detects the assertion of the Error Abort mode signal, it can choose to either:

1. Finish the LinkRetry_FLIT state and allow all packets to be retransmitted before sending a stream of IRTRY packets with the StartRetry flag set.
2. Pause the LinkRetry_FLIT state and send a stream of IRTRY packets with the StartRetry flag set, and then revert back to the completion of the LinkRetry_FLIT state.

The HMC implementation completes the LinkRetry_FLIT state to empty the retry buffer, and then sends the stream of IRTRY packets.

LinkRetry_FLIT State Control

- Enter: When LinkRetry_Init state clears AND there are packets currently saved for retransmission.
- Actions:
  Retransmit all saved packets across the link. All packets retransmitted must have the latest valid FRP value received from the local link slave embedded into their RRP field.
- Exit: When all packets saved for retransmission have been sent to the link.

11.2.5.2.2 LinkRetry Sequence Exit

The link master exits the retry sequence from either the LinkRetry_Init state or the LinkRetry_FLIT state depending on whether there were packets saved for retransmission when the LinkRetry_Init state was exited. The normal packet stream may now resume.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

11.3 Link Slave Retry Functions

The link slave section parses incoming packets, performs the CRC and sequence check for each packet, and if no error forwards the packets to the internal destination. The link slave also extracts the FRP and RRP fields from each good packet and forwards those to the local (adjacent) link master section. When a link error occurs, the link slave goes into error abort mode, which stops the forwarding of the incoming packets. It also stops the forwarding of FRP and RRP pointers to the local link master. The assertion of error abort mode causes the local link master to transmit a stream of IRTRY packets to signal to the other end to activate its link retry sequence. When a link retry is in progress, the link slave detects and counts two different types of IRTRY packets, including those with the StartRetry flag set (FRP[0]=1) and those with the ClearErrorAbort flag set (FRP[1]=1).

11.3.1 Packet CRC/Sequence Check

As incoming packets flow through the link slave CRC is calculated from the header to the tail (inserting 0s into the CRC field) of every packet. As the tail passes through, the calculated CRC is compared to the incoming CRC included in the tail of the packet. A mismatch indicates that bits in the packet have changed since the CRC was originally generated at the link master. If the calculated CRC compares correctly, an additional packet check is provided by extracting the packet sequence number out of the tail and comparing it with the saved value of the sequence number from the last successful packet. The sequence number should always be incremented by + 1 for a correct transmission of packets across the link.

11.3.1.1 FRP and RRP Extraction

If the incoming packet passes the CRC and sequence checks, and the link slave is not in error abort mode, the link slave extracts the values of the FRP and RRP fields and forwards them to the local link master section.

11.3.2 Error Abort Mode

If an LNG, CRC, or sequence error occurs on an incoming packet, the link slave executes the following actions:

- The link slave sets the Error Abort mode.
- The link slave drops or poisons the packet that had the CRC or sequence error. Poisoning involves inverting the calculated CRC for the packet, so it will be dropped by a downstream receiver of the packet. This action is required because in the case of packets that may span multiple core cycles, the beginning of the packet may have been forwarded before the CRC check was performed at the tail, to minimize latency.
- During error abort mode, the link slave will drop (not forward) all subsequent packets following the packet that caused the error.
- During error abort mode the link slave normally does not extract values from the RRP, FRP, or RTC fields. There is one exception to this, which is described in 11.3.3 “IRTRY Packet Operation” on page 69.
- Error Abort mode is cleared by monitoring the IRTRY packets as described in 11.3.3 “IRTRY Packet Operation” on page 69.

11.3.3 IRTRY Packet Operation

When the link slave receives an IRTRY packet, it performs a LNG check and CRC check similar to checking any other incoming packet. The SEQ check is inhibited for IRTRY packets because they are not retried if they have a failure. A stream of multiple IRTRY packets up to a programmable threshold must be received before an action is performed

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

so that a data payload that looks like an IRTRY packet arriving after a link error occurred doesn't falsely trigger a retry action. Setting the IRTRY Packet Receiver Number register to expect 16 IRTRY packets is recommended as it allows the IRTRY packet stream to be longer than the longest normal transaction packet (9 FLITs), but short enough to minimize the latency of the retry sequence.

Each successfully received IRTRY packet will cause the link slave to increment one of two counters as described below. This occurs even if Error Abort mode is set. No IRTRY packets are forwarded beyond the link slave. If an IRTRY packet has a LNG error or CRC error, the link slave will set error abort mode if it is not already set.

The link slave decodes two types of IRTRY packets:

1. StartRetry: IRTRY with FRP[0] = 1, indicating the StartRetry flag. In this case the link slave will increment the StartRetry counter for each StartRetry flag that it receives in a continuous stream of IRTRY packets. Once the counter reaches the threshold specified in the Init_Retry Packet Receiver Number field of the link retry control register, the link slave will generate a StartRetry pulse that is sent to the local link master. Additional StartRetry flags may arrive after the threshold is met if the transmitting link master has its Init_Retry Packet Transmit Number programmed to be greater than the threshold. These excess StartRetry flags should not affect the operation and should not cause a second StartRetry pulse. If a FLIT or packet other than an IRTRY is received, the link slave will clear the StartRetry counter, which should arm the counter to begin counting again if additional IRTRY packets with the StartRetry flag set arrive later. If the link slave detects a link error and sets Error Abort mode, it will also clear the StartRetry counter. When receiving IRTRY packets with the StartRetry flag, there are two cases of RRP extraction:
   1a. If error abort mode is not set, the link slave extracts an RRP out of every good packet that it receives, including all IRTRY packets with the StartRetry flag set. These RRPs are sent to the local link master.
   1b. When error abort mode is set, normally the link slave does not extract any retry pointers or tokens from incoming packets. There is one exception: If Error Abort mode is set and the link slave is receiving IRTRY packets with the StartRetry flag, this is a special case where a link error occurred on both sides of the link. The link slave will count the StartRetry flags, and when reaching the threshold it will extract the RRP from the IRTRY packet that caused the threshold to be reached. The link slave will send this RRP along with the StartRetry pulse to the local link master.
2. ClearErrorAbort : IRTRY with FRP[1] = 1, indicating the ClearErrorAbort flag. In this case the link slave will increment the ClearErrorAbort counter for each ClearErrorAbort flag that it receives in a continuous stream of IRTRY packets. Once the counter reaches the threshold specified in the Init_Retry Packet Receiver Number field of the link retry control register, the link slave will clear error abort mode if it is currently set. It will also extract the RRP out of the IRTRY packet that reached the receiver number threshold and send the RRP to the local link master. Additional IRTRY packets with the ClearErrorAbort flag set may arrive after the threshold is met if the transmitting link master has its Init_Retry Packet Transmit Number programmed to be greater than the threshold. Specific logical implementation may require more IRTRY packets with the ClearErrorAbort flag set to be transmitted than the link slave is expecting. These excess ClearErrorAbort flags will not affect the operation as long as Error Abort mode clears before a retried packet is received. The link slave should follow normal procedure and extract all RRPs after error abort mode clears and send them to the local link master. If a FLIT or packet other than an IRTRY is received, the link slave will clear the ClearErrorAbort counter, which should arm the counter to begin counting

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

again if additional IRTRY packets with the ClearErrorAbort flag set arrive later. If the link slave detects a link error on any IRTRY packet with ClearErrorAbort flag set when error abort mode is clear, it should set error abort mode and also clear the ClearErrorAbort counter.

11.3.4 Resumption of Normal Packet Stream after the Retry Sequence

At some point after the retry sequence, a normal packet (either a retried one or a new one) will be received by the link slave. The link slave operates normally on that packet, verifying the LNG, CRC, and sequence number fields and forwarding it on to its internal destination. Because the link slave captured a valid sequence number from the last good packet received before the retry sequence (no sequence numbers were extracted from the IRTRY packets), it will compare that sequence number with the first one received after the retry sequence. The sequence number should be incremented by 1 if the retry sequence was successful and the normal packet stream resumed correctly.

11.4 Retry Pointer Loop Time

Table 42: Retry Pointer Loop Time Implementation Example

Link		256-FLIT Retry Buffer Full Period (ns)	HMC Delay (ns)	Host Allowable Delay (ns)	Recommended Retry Timer Value (ns)	Recommended Timeout Period Encode Value
Bit Rate	ps/FLIT					
10 Gbps	800	204.80	26.47	178.33	>614.4	5
12.5 Gbps	640	163.84	25.87	137.97	>491.52	4
15 Gbps	533	136.45	22.27	114.18	>409.35	4

The HMC protocol includes retry pointer space to support a retry buffer that can hold up to 256 FLITs. In order to maintain link performance, the maximum allowable retry pointer loop time must be less than the “Retry Buffer Full Period” shown in the table. Note that the maximum retry pointer loop time could be allowed to be greater than the time it takes to fill the retry buffer, but the retry buffer full condition will then throttle the packet stream.

The result is that NULL FLITs will be sent between transaction packets, degrading the performance of the link. To avoid link throttling, the host’s portion of the retry pointer loop delay must be less than the “Host Allowable Delay” shown in the table.

The following is an implementation example that highlights the steps involved to identify the total retry pointer loop time:

- The time it takes for a packet (and its FRP) to travel from a link master to a link slave, including:
  - Serialization time at the transmitting end of the link
  - Deserialization time at the receiving end of the link
- The time for the longest packet’s tail to be received in the link slave, to the point of CRC check and FRP extraction
- The transfer of the extracted FRP from the link slave to the local link master
- Time to wait in the link master for just missing embedding the retry pointer into the RRP field of the previous packet and waiting for the next tail of the longest packet
- The time it takes for a packet with the embedded retry pointer to travel back to the source end of the link, including:
  - Serialization time at the transmitting end of the link
  - Deserialization time at the receiving end of the link

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

- Time for the longest packet's tail to be received in the link slave, to the point of CRC check and RRP extraction
- The transfer of the RRP from the link slave to the local link master (the original transmitter in the first step)

11.5 Link Flow Control During Retry

When the normal packet stream is transmitted by the link master, it has already determined that the link slave has enough buffer space in its buffers to accept all FLITs of the transmitted packets, as controlled by the return token count protocol. If a link error occurs and the packet stream is aborted during Error Abort mode in the link slave, the reserved space in the receive buffers still exists, and therefore, when those packets are retransmitted across the link, there is no need for any additional flow control, except for the following clarification. At the receiver, the link slave may poison the packet that has errored, but it has already propagated it forward into the receive buffer. Because this takes up some FLIT locations, the return token count control logic must not return the tokens for a poisoned packet. That packet will be retransmitted, and when it is finally loaded into the receive buffer with good CRC and propagated further, the tokens will be returned. All subsequent packets (after the errored one) dropped in the link slave will not be forwarded into the receive buffer, as a result, the buffer space will be available for those packets as they are retransmitted.

A consequence of forwarding the poisoned packet and not returning tokens until the packet is retransmitted successfully is as follows: The poisoned packet could have been the last packet that caused the full condition at the receive buffer, and then the receive buffer could be temporarily stalled due to downstream flow control. If this were the case, the failed packet would be the only packet to be transmitted. When the packet was retransmitted, there would be no room for that packet in the receive buffer. An easy solution to this is to adjust the maximum tokens available for that receive buffer to 9 FLITs less than the actual buffer size. This accommodates an extra copy (the poisoned one) of the largest packet that could be retransmitted during a link retry.

11.6 Example Implementation Link Error and Retry Sequence

For the following example, refer to Figure 17 on page 62.

The green link master, upper left, transmits packets across the link to the green link slave in the upper right. The blue link master, lower right, transmits packets across the link to the blue link slave in the lower left.

1. Successful green packets flowing from the green link master to the green link slave across the top of the diagram. Green packets have green FRPs and blue RRPs embedded in them. Successful blue packets flowing from the blue link master to the blue link slave across the bottom of the diagram. Blue packets have blue FRPs and green RRPs embedded in them.
2. An error occurs in the green packet stream across the top. The green link slave detects CRC error and sets error abort mode. The errored packet is poisoned and all following packets are dropped.
3. Flow of FRPs and RRPs stops from the green link slave to the blue link master.
4. The blue link master receives the error abort mode signal from the green link slave, suspends its normal packet stream from its output FIFO, and inserts a stream of IRTRY packets each with the StartRetry flag set (FRP[0] = 1). The blue link master embeds the latest FRP it received from the green link slave (via its retry pointer collection register) into the RRP field of each IRTRY packet. The value embedded into the

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

RRP field is the same as the FRP for the last good packet that was received at the green link slave before the link error occurred. After sending the stream of IRTRY packets, the blue link master resumes it normal packet stream from its output FIFO.

5. The blue link slave detects the IRTRY stream and counts the StartRetry flags. During this time it extracts the RRP and sends it to the green link master, which updates its retry buffer read pointer to point to the packet that had errored. When the IRTRY receive threshold is reached, the blue link slave sends a StartRetry pulse to the green link master.
6. The blue link slave continues to receive normal packets, forwarding them to its input FIFO, and extracts all RRPs, FRPs, and RTCs.
7. The green link master receives the StartRetry pulse which starts its retry sequence and enters into its LinkRetry_Begin state, finishes sending the current packet being transmitted from its output FIFO, then stops. That packet is the last packet that gets loaded into the retry buffer.
8. The green link master enters its LinkRetry_Init state and transmits a stream of IRTRY packets, each with the ClearErrorAbort flag set (FRP[1]=1). During this time, the green link master is receiving valid FRPs from the blue link slave (via its retry pointer collection register), and it embeds those into the RRP field of every IRTRY packet.
9. The green link slave, currently in Error Abort mode, detects the stream of IRTRY packets and counts the ClearErrorAbort flags. When it reaches the IRTRY receive threshold, it clears Error Abort mode. At that time it will begin extracting the RRPs from the IRTRY packet(s) and sends them to the blue link master.
10. The green link master enters its LinkRetry_FLIT state and transmits the pending packets from its retry buffer until its read pointer equals its write pointer. It continues to embed the latest value of its retry pointer collection register into the RRP field of each of the pending packets being retransmitted. As the pending packet stream completes from the retry buffer, the green link master switches to new packets waiting in its output FIFO.
11. The green link slave receives the retried packet stream and resumes the sequence check (along with CRC check, etc). The first retried packet received should have a sequence number +1 greater than the last successfully received packet before the link error occurred. All successfully received retried packets, followed by new packets, will be forwarded to the link input FIFO.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

12 Warm Reset

Warm reset provides a mechanism to re-initialize the HMC state machines and tokens to allow on-line activity again after a system or HMC fatal error has occurred. The occurrence of an HMC fatal error is communicated to the host through an Error Response packet (if possible) and by the assertion of the FERR_N signal.

Figure 19: Warm Reset Flow Chart

13 Retraining

13.1 Retraining a Link with High Error Rate

The host can monitor the frequency of link retry occurrences in the ERRSTAT field of response packets and initiate a retraining procedure when it determines the error rate on any downstream link is above an acceptable threshold. The steps for this procedure are as follows:

1. Host detects an unacceptable rate of link retries on any downstream link.
2. Host quiesces traffic (ensures closure on all transactions in flight).
3. Host blocks packets from being received into its link slave.
4. Host retrains links through sleep mode entry and exit. All specifications described in “Power State Management” are applicable.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

13.2 Host Recovery after Link Retry Fails

Figure 20: Host Recovery after Link Retry Fails

Notes: 1. Host detection of link retry failure on link not directly connected to host will occur via an Error Response packet from a downstream HMC.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

14 Functional Characteristics

14.1 Packet Ordering and Data Consistency

There can be multiple packet-reordering points within the HMC, including the following implementation examples:

1. The output of each link slave input buffer could reorder packets arriving from one link that are destined for different vaults. This would prevent a busy vault from blocking the packet flow out of the receive buffer and enable the flow of packets across the link to continue.
2. Each vault controller command queue could reorder packet execution to banks that are not busy, potentially optimizing the memory bus usage.

All reordering points within an HMC must maintain the order of a stream of packets from a specific link to a specific bank within a vault. This ensures that if a host sends a write request packet to address x followed by a read request packet to the same address x, the READ will always return the newly written data.

Packet ordering performed by the HMC supports messaging or data-passing programming between two or more hosts with access to the same HMC. This common method of host-to-host messaging is known as producer-consumer programming, and proceeds as follows:

1. Host 0 wants to pass a message/data to host 1. Host 0 writes memory location A (this could be multiple locations) with the message/data.
2. Upon receiving all expected successful write responses for the write(s) to location A, host 0 sets a flag in memory location B indicating that the message/data is available in location A.
3. Host 1 executes a loop, polling (that is, issuing read requests to) location B to observe the set flag.
4. Upon detecting the set flag in the read response for location B, host 1 issues a read request to location A.
5. Host 1 receives the read response for location A containing the newly-written data from host 0.

This scenario proceeds as defined if host 0 waits for the successful write response for the write to location A before issuing the write request packet to location B.

The write response is generated at the vault controller after the write request is fully executed and is identified with the same tag as the write request. This write response could be an explicit write response packet, or it could optionally be embedded within a response packet being sent back on the same link. Locations A and B can be any memory locations. Note that this scenario is not supported with the use of posted write requests.

14.2 Data Access Performance Considerations

Due to the internal 32-byte granularity of the DRAM data bus within each vault in the HMC, inefficient utilization of this bus occurs when starting or ending on a 16-byte boundary rather than a 32-byte boundary. An extreme example of this would be the host issuing a series of 16-byte read requests. Each read request would fetch 32 bytes from the DRAM and return half of the data in the response packet, throwing away the other 16 bytes of data. For bandwidth optimization it is advantageous to issue requests with 32-byte granularity.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

If the host issues a series of commands that access portions of the same data block (as defined by the maximum block size) rather than issuing one request accessing the entire data block, two performance disadvantages exist:

1. The probability of bank conflicts will increase in the accessed vault.
2. The multiple request and response packets have additional header/tail overhead on the link.

Because of this, it is desirable to perform commands with larger data sizes. In an example implementation, if a host has defined the maximum block size as 128 bytes and instead of issuing a 128-byte request to a specific location, it issues four consecutive 32-byte requests to the same block location. The cube will send these four independent requests to the specific bank whereby it must open and close the row in the same bank four times, reducing bandwidth utilization and increasing latency as compared to the case in which a single 128-byte request is issued.

To maximize bandwidth utilization and reduce system latency the following should be taken into consideration:

1. Request data block sizes that are as large as possible, up to the configured maximum block size, that meet the requirements of the system.
2. Configure the maximum block size based upon the most frequently requested data size. If the most frequently requested data size does not equal any of the possible maximum block sizes (32B, 64B, or 128B), select the smallest block size that encompasses the most frequently requested data size. For example, if a system most often requires 48B data transfers, 64B should be chosen as the maximum block size.

14.3 Vault ECC and Reference Error Detection

Memory READ and WRITE operations have a parity bit added to the command and address signals that are sent from each vault controller to the memory partitions and banks associated with each controller. If the bank receiving a request determines that there is a parity error, the reference is retried.

Data is protected in memory using ECC. The ECC should provide correction of a single-bit error as well as all data bits that are transmitted over a single, failed TSV between the memory and the logic base.

Memory locations are initialized with correct ECC data patterns before the host begins using the locations. This eliminates a write-before-read requirement to avoid SBEs and MUEs.

14.4 Refresh

DRAM refresh is handled internally within the HMC by the vault controllers. As an example implementation, every vault controller could have a unique refresh rate for each of its bank groups (where a bank group is a set of banks within one memory die). An implementation may also choose to vary refresh rates dynamically, driven by conditions such as temperature and/or detected memory error rates.

14.5 Scrubbing

Scrubbing is used inside the cube to mitigate soft failures in memory. Two categories of scrubbing are provided in each vault controller:

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

- **Demand scrubbing:** If a correctable error is detected in the read data of a read request, the data is corrected and returned to the host in the response packet. A scrubbing cycle is invoked that writes the corrected data back into the memory array.
- **Patrol scrubbing:** This form of scrubbing is performed by the cube refresh logic. If a correctable error is detected, a scrubbing cycle is invoked that writes the correct data back into the memory array.

The scrubbing cycle writes the corrected data immediately after the correctable error is discovered on the read. This read-update-write operation is atomic, meaning that no other reference to the location will be possible between the read and the associated write. This is handled as a bank conflict within the vault controller command queue.

All scrubbing cycles can be programmed to execute a reread of the scrubbed memory location to determine whether the scrubbing operation was successful. If it was successful, a “soft error” is logged. If it was unsuccessful and another correctable error occurs during the reread, a “hard error” is assumed and it is logged. Correctable errors that are designated as hard errors enable the system to continue to run, but they also increase the probability that the occurrence of another soft or hard error within the 16-byte ECC word will turn the correctable situation into an uncorrectable one. To cover this case, the HMC provides In-Field repair of the hard failures, the mechanism of which is vendor specific.

14.6 Response Open Loop Mode

During normal transaction layer packet flow, return token counts (RTCs) are returned in both directions on each link. This controls the flow of request packets in one direction and response packets in the other direction. Response open loop mode can be configured on the upstream links that are connected to the host. This mode eliminates the requirement for the host to send tokens back to the cube, and results in two simplifications:

- The host link slave does not have to generate tokens when response packets are consumed.
- The host link master does not have to embed any RTCs into request packets being transmitted to the cube.

The response open loop mode will force the link master to send response packets to the host without the token requirements. (Other packet flow controls such as retry buffer full conditions and retry in progress will still suspend packet flow if necessary.)

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

15 JTAG Interface

HMC incorporates a standard test access port (TAP) controller that operates in accordance with IEEE Standard 1149.1-2001. The input and output signals of the test access port use V_{DD} as a supply.

The JTAG test access port uses the TAP controller on the HMC, from which the instruction, bypass, ID, boundary scan, and CADATA registers can be selected. Each of these functions of the TAP controller is described in detail below.

The HMC includes boundary scan support that complies with the IEEE 1149.1 and 1149.6 standards. Boundary scan chain order is provided within BSDL files for each HMC configuration.

15.1 Disabling the JTAG Feature

It is possible to operate HMC without using the JTAG feature. To disable the TAP controller, tie the Test Reset Signal, TRST_N, LOW. TCK, TDI, and TMS can all be considered "Don't Care" when TRST_N is LOW. TDO should be left unconnected. Upon power up, the device comes up in a reset state; this will not interfere with the operation of the HMC device.

15.2 Test Access Port (TAP)

15.2.1 Test Clock (TCK)

The test clock is used only with the TAP controller. All inputs are captured on the rising edge of TCK. All outputs are driven from the falling edge of TCK.

15.2.2 Test Mode Select (TMS)

The TMS input is used to issue commands to the TAP controller and is sampled on the rising edge of TCK.

All of the states in Figure 21: "TAP Controller State Diagram" on page 81 are entered through the serial input of the TMS pin. A 0 in the diagram represents a LOW on the TMS pin during the rising edge of TCK; a 1 represents a HIGH on TMS.

15.2.3 Test Reset (TRST_N)

The TRST_N input provides asynchronous initialization of the TAP controller. TRST_N must be LOW during power-up and track the transition timing of the system reset signal, P_RST_N. To ensure deterministic operation of the test logic, TMS should be held HIGH while the signal applied at TRST_N changes from LOW to HIGH.

15.2.4 Test Data-In (TDI)

The TDI pin is used to input test instructions and data into the registers serially and can be connected to the input of any registers. The register between TDI and TDO is selected by the instruction that is loaded into the TAP instruction register. For information on loading the instruction register, see Figure 25: "JTAG Operation – Loading Instruction Code and Shifting Out Data" on page 86. TDI is connected to the most significant bit (MSB) of any register (see Figure 22: "TAP Controller Block Diagram" on page 81).

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

15.2.5 Test Data-Out (TDO)

The TDO output pin is used to clock test instructions and data serially out of the registers. The TDO output driver is only active during the Shift-IR and Shift-DR TAP controller states. In all other states, the TDO pin is in a High-Z state. The output changes on the falling edge of TCK. TDO is connected to the least significant bit (LSB) of any register (see Figure 22: "TAP Controller Block Diagram" on page 81).

15.3 TAP Controller

The TAP controller is a finite state machine that uses the state of the TMS pin at the rising edge of TCK to navigate through its various operating modes. The TAP controller state diagram is provided in Figure 21 on page 81. Each state is described in detail below.

15.3.1 Test-Logic-Reset

The test-logic-reset controller state is entered when TMS is held HIGH for at least five consecutive rising edges of TCK. As long as TMS remains HIGH, the TAP controller remains in the test-logic-reset state. The test logic is inactive during this state.

15.3.2 Run-Test/Idle

The run-test/idle is a controller state in between scan operations. This state can be maintained by holding TMS LOW. From here, either the data register scan, or subsequently, the instruction register scan can be selected.

15.3.3 Select-DR-Scan

Select-DR-scan is a temporary controller state. All test data registers retain their previous state while here.

15.3.4 Capture-DR

The capture-DR state is where the data is parallel-loaded into the test data registers.

15.3.5 Shift-DR

Data is shifted serially through the data register while in this state. As new data is input through the TDI pin, data is shifted out of the TDO pin.

15.3.6 Exit1-DR, Pause-DR, and Exit2-DR

The purpose of exit1-DR is to provide a path for return to the run-test/idle state (through the update-DR state). The pause-DR state is entered when there is a need to suspend data shifting through the test registers. When shifting is to enable reconvene, the controller enters the exit2-DR state and can then re-enter the shift-DR state.

15.3.7 Update-DR

Data is parallel loaded from the shift register to the parallel output register on the falling edge of TCK in the update-DR controller state.

15.3.8 Instruction Register States

The instruction register states of the TAP controller are similar to the data register states. The desired instruction is shifted serially into the instruction register during the shift-IR state and is loaded during the update-IR state.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 21: TAP Controller State Diagram

Figure 22: TAP Controller Block Diagram

Notes: 1. x is dependent upon number of links in the HMC device.

15.4 Performing a TAP RESET

The TAP A RESET is performed by forcing TMS HIGH (V_{DD}) for five rising edges of TCK. This RESET does not affect the operation of the HMC and can be performed during HMC operation.

The TAP controller is also reset when TRST_N is driven LOW which ensures that TDO comes up in a High-Z state.

15.5 TAP Registers

Registers are connected between the TDI and TDO pins and enable data scanning into and out of the HMC test circuitry. Only one register can be selected at a time through the instruction register. Data is serially loaded into the TDI pin on the rising edge of TCK. Data is output on the TDO pin on the falling edge of TCK.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

15.5.1 Instruction Register

Eight-bit instructions can be loaded serially into the instruction register. This register is loaded during the update-IR state of the TAP controller. Upon power up, the instruction register is loaded with the IDCODE instruction. It is also loaded with the IDCODE instruction if the controller is placed in a reset state as described in the previous section.

When the TAP controller is in the capture-IR state, the two LSBs are loaded with a binary 01 pattern to accommodate fault isolation of the board-level serial test data path.

15.5.2 Bypass Register

To save time when shifting data serially through registers, it is sometimes advantageous to skip certain chips. The bypass register is a single-bit register that can be placed between the TDI and TDO pins. This enables data shifting through the HMC with minimal delay. The bypass register is set LOW (V_{SS}) when the BYPASS instruction is executed.

15.5.3 Identification (ID) Register

The ID register is loaded with a vendor-specific, 32-bit code during the capture-DR state when the IDCODE command is loaded in the instruction register. The IDCODE is hard-wired into the HMC and can be shifted out when the TAP controller is in the shift-DR state. The ID register has a vendor code and other device-specific information described in Table 45 on page 87.

15.5.4 Boundary Scan Register

The boundary-scan register is loaded with the contents of the I/O ring when the TAP controller is in the capture-DR state and is then placed between the TDI and TDO balls when the controller is moved to the shift-DR state.

The behavior of the boundary scan register is completely specified by the IEEE 1149.1 standard.

15.5.5 CADATA Register

The CADATA is a 64-bit register used for configuration and status register access. This register is not defined by the IEEE 1149.1 standard.

Table 43: CADATA Register

Register Name	Sub-block	Description
CADATA	ADRS[63:32]	Stores the address for configuration and status register access.
	DATA[31:0]	1) Stores data to be written during configuration write operations. 2) Stores results of configuration read operations.

15.6 TAP Instruction Set

15.6.1 Overview

Many different instructions are possible with the 8-bit instruction register. All combinations used are listed in Table 47 on page 88. These instructions are described in detail below. The remaining instructions are reserved and must not be used.

The TAP controller used in this HMC is fully compliant with the 1149.1 convention.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Instructions are loaded into the TAP controller during the shift-IR state when the instruction register is placed between TDI and TDO. During this state, instructions are shifted through the instruction register through via the TDI and TDO pins. To execute the instruction after it is shifted into the controller, the TAP controller must be moved into the update-IR state.

15.6.2 EXTEST

The EXTEST instruction enables circuitry external to the component package to be tested. Boundary-scan register cells at output balls are used to apply a test vector, while those at input balls capture test results.

15.6.3 EXTEST_PULSE & EXTEST_TRAIN

The EXTEST_PULSE and EXTREST_TRAIN instructions provide an edge-detecting test mechanism for AC-coupled signals. The EXTEST_PULSE instruction provides a pulse of data onto an AC signal whereby the pulse width is controlled by the time spent in the Run-Test/Idle TAP Controller state. This produces a single “wide” pulse with a variable and controllable period. The EXTEST_TRAIN instruction provides a pulse train, the edges of which are generated by each falling edge of TCK while in the Run-Test/Idle TAP Controller state. DC pins behave according to the IEEE Std 1149.1 EXTEST instruction when either the EXTEST_PULSE or EXTEST_TRAIN instructions are selected.

15.6.4 HIGH-Z

The High-Z instruction causes the bypass register to be connected between the TDI and TDO. This places all HMC outputs into a High-Z state.

15.6.5 CLAMP

When the CLAMP instruction is loaded into the instruction register, the data driven by the output balls are determined from the values held in the boundary-scan register.

15.6.6 SAMPLE/PRELOAD

When the SAMPLE/PRELOAD instruction is loaded into the instruction register and the TAP controller is in the capture-DR state, a snapshot can be taken of the states of the component's input and output signals without interfering with the normal operation of the assembled board. The snapshot is taken on the rising edge of TCK and is captured in the boundary-scan register. The data can then be viewed by shifting through the components TDO output.

15.6.7 IDCODE

The IDCODE instruction causes loading of a vendor-specific, 32-bit code to be loaded into the instruction register. It also places the instruction register between the TDI and TDO pins and enables shifting the IDCODE out of the device when the TAP controller enters the shift-DR state. The IDCODE instruction is loaded into the instruction register upon power-up or whenever the TAP controller is given a test logic reset state.

15.6.8 BYPASS

When the BYPASS instruction is loaded in the instruction register and the TAP is placed in a shift-DR state, the bypass register is placed between TDI and TDO.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

15.6.9 CFG_RDA

Selects the CADATA register so configuration and status register address and read data can be serially shifted between TDI and TDO pins. This instruction is not defined by the IEEE 1149.1 standard.

15.6.10 CFG_WRA

Selects the CADATA register so configuration and status register address and write data can be serially shifted between TDI and TDO pins. This instruction is not defined by the IEEE 1149.1 standard.

15.6.11 Reserved for Future Use

The remaining instructions are not implemented but are reserved for future use. Do not use these instructions.

15.7 JTAG Configuration Register Write and Status Register Read

Figure 23: JTAG Configuration Register Write Flow Chart

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 24: JTAG Status Register Read Flow Chart

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 25: JTAG Operation – Loading Instruction Code and Shifting Out Data

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 44: JTAG Voltage and Timing Parameters

Parameter Description	Symbol	Min	Max	Units	Notes
JTAG Voltage Parameters					
Input LOW voltage - DC	V_{IL_DC}	V_{SS} - 0.4V	V_{SS} + 0.5V	V	1
Input HIGH voltage - DC	V_{IH_DC}	V_{DDK} - 0.5V	V_{DDK} + 0.2V	V	2
Output LOW voltage	V_{OL}	–	V_{SS} + 0.1V	V	3
Output HIGH voltage	V_{OH}	V_{DDK} - 0.1V	–	V	4
JTAG Timing Parameters					
Clock cycle time	^{t}THTH	10	–	ns	
Clock frequency	^{t}TF	–	100	MHz	
Clock LOW time	^{t}TLTH	5	–	ns	
Clock HIGH time	^{t}THTL	5	–	ns	
TCK LOW to TDO unknown	^{t}TLOX	0	–	ns	
TCK LOW to TDO valid	^{t}TLOV	5	–	ns	
TDI valid to TCK HIGH	^{t}DVTH	2.5	–	ns	
TCK HIGH to TDI invalid	^{t}THDX	2.5	–	ns	
TMS, capture setup	^{t}MVTH	2.5	–	ns	
TMS, capture hold	^{t}THMX	2.5	–	ns	

Notes:
1. V_{SS} measured at HMC package ball.
2. V_{DDK} measured at HMC package ball.
3. I_{OL} = 1mA
4. I_{OH} = 1mA

Figure 26: TAP Timing

Table 45: Identification Register Definitions

Instruction Field	All Devices	Description
Revision number (31:28)	abcd	0
Device ID (27:12)	0x0008	Identification of HMC
JEDEC ID code (11:1)	Vendor specific	Enables unique identification of HMC vendors

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 45: Identification Register Definitions (Continued)

Instruction Field	All Devices	Description
ID register presence indicator (0)	1	Indicates the presence of an ID register

Table 46: Scan Register Sizes

Register Name	Bit Size
Instruction	8
Bypass	1
Device ID	32
Boundary scan	Dependent on number of links
CADATA	64

Table 47: Instruction Codes

Instruction	Code	Active Register	Description
SAMPLE/PRELOAD	0x01	Boundary Scan	1149.1 SAMPLE/PRELOAD instruction
IDCODE	0x02	Device ID	1149.1 IDCODE instruction
CLAMP	0x04	Boundary Scan	1149.1 CLAMP instruction
HIGHZ	0x08	Boundary Scan	1149.1 HIGHZ instruction
EXTEST	0x09	Boundary Scan	1149.1 EXTEST instruction
EXTEST_PULSE	0x0A	Boundary Scan	1149.6 EXTEST_PULSE instruction
EXTEST_TRAIN	0x0B	Boundary Scan	1149.6 EXTEST_TRAIN instruction
CFG_RDA	0x2C	CADATA	Allows Configuration & Status Register to be read
CFG_WRA	0x2D	CADATA	Allows Configuration & Status Register to be written
BYPASS	0xFF	BYPASS	1149.1 BYPASS instruction

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

16 I^{2}C Interface

The I^{2}C bus is provided as a sideband interface to program, monitor and access various modes within the HMC. The I^{2}C bus complies with the UM-10204 I2C bus specification and includes the following attributes:

1. Operates with V_{DDK} supply (1.5V nominal) with either an open-drain or open-collector output stage
2. 7-bit slave addressing
3. Operates at 100Kb/s (standard-mode) and 400Kb/s (fast-mode)
4. Clock stretching
5. Device ID
6. Software reset
7. FTDI I^{2}C compatibility
8. Complies with the standard-mode and fast-mode electrical and timing parameters found in the UM-10204 I^{2}C bus specification. The V_{DD} referred to in these tables is equivalent to the HMC 1.5V V_{DDK} supply.

Figure 27: I^{2}C Block Diagram

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 28: I²C State Diagram

1. Idle
2. Start
3. Slave Address
4. Slave Address Ack
5. CFG Address
6. CFG Address Ack
7. WDATA
8. RDATA
9. Data Ack
10. Stop

Figure 29: I²C Register Accessibility

Notes:
1. SCL = ≤400 kHz I²C clock
   Core clock = Internal 625 MHz clock
   SDA = I²C data
2. CFG registers will be available to the user via I²C, JTAG, and SerDes I/O interface.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 30: I²C Packet Protocol Key

Number of bits in field:	1	7	1	1	8	1	1
	S	Slave address	Wr	A	Data	A	P
				x		x	

Symbol	Description
S	Start condition
Sr	Repeated start condition
Rd	Read (bit value of 1)
Wr	Write (bit value of 0)
x	Value under field indicating required value
A	Acknowledge (Ack is 0, Nack is 1)
P	Stop condition
PEC	Packet error code
☐	Master-to-slave
▒	Slave-to-master
...	Continuation of protocol

Figure 31: Configuration Register WRITE and READ Commands

CFG Write

1	7	1	1	8	1	8	1	8	1	8	1	
S	Slave address	Wr	A	CFG address 1	A	CFG address 2	A	CFG address 3	A	CFG address 4	A	...
	0010 CUB[2:0][1]	0	0	[31:24]	0	[23:16]	0	[15:8]	0	[7:0]	0	

8	1	8	1	8	1	8	1	1
Data 1	A	Data 2	A	Data 3	A	Data 4	A	P
[31:24]	0	[23:16]	0	[15:8]	0	[7:0]	0	

CFG Read

1	7	1	1	8	1	8	1	8	1	8	1	
S	Slave address	Wr	A	CFG address 1	A	CFG address 2	A	CFG address 3	A	CFG address 4	A	...
	0010 CUB[2:0][1]	0	0	[31:24]	0	[23:16]	0	[15:8]	0	[7:0]	0	

1	7	1	1	8	1	8	1	8	1	8	1	1
Sr	Slave address	Rd	A	Data 1	A	Data 2	A	Data 3	A	Data 4	A	P
	0010 CUB[2:0][1]	1	0	[31:24]	0	[23:16]	0	[15:8]	0	[7:0]	0	

Notes: 1. Value of the CUB[2:0] bits in the slave address match the values that the CUB[2:0] pins are tied to.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

17 HMC-15G-SR Electrical Specifications

17.1 Absolute Maximum Ratings

Stresses greater than those listed may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions outside those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may adversely affect reliability.

Table 48: Absolute Maximum Ratings

Description	Parameter	Minimum	Maximum	Units	Comments
Logic core source	V_{DD}	–0.4	1.15	V	
PLL sources	$V_{DDPLLA[3:0]}$, $V_{DDPLLB[3:0]}$, V_{DDPLLR}	–0.5	1.4	V	
Link termination sources	V_{TT}, V_{TR}	–0.5	1.5	V	
DRAM source	V_{DDM}	–0.4	1.7	V	
DRAM wordline boost source	V_{CCP}	–0.4	2.75	V	
JTAG, NVM, I^2C source	V_{DDK}	–0.4	1.95	V	
Link input signal pin voltage (LxRXP, LxRXN)	Link input signal pin voltage (LxRXP, LxRXN)	More positive of –0.1V or V_{TR} - 1.1V	V_{TR} + 0.3V	V	
Link output signal pin voltage (LxTXP, LxTXN)	Link output signal pin voltage (LxTXP, LxTXN)	More positive of –0.1V or V_{TT} - 1.1V	V_{TT}	V	

Table 49: Maximum Current Conditions
Notes 1 and 2 apply to entire table

Condition	State Symbol	Voltage Supply									Units	Notes
		V_{DD}	V_{DDPLLA}[4]	V_{DDPLLB}[4]	V_{DDPLLR}	V_{TT}	V_{TR}	V_{DDM}	V_{CCP}	V_{DDK}		
Reset P_RST_N LOW	H0	–	–	–	–	–	–	–	–	–	mA	
Active standby at 10Gb/s NULL commands on all links	H1	–	–	–	–	–	–	–	–	–	mA	
Active standby at 15Gb/s NULL commands on all links	H2	–	–	–	–	–	–	–	–	–	mA	
All links operating 15Gb/s	H3	–	–	–	–	–	–	–	–	–	mA	3
All links operating 10 Gb/s	H4	–	–	–	–	–	–	–	–	–	mA	3
One link operating 15 Gb/s – All other links in sleep mode	H5a	–	–	–	–	–	–	–	–	–	mA	3

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 49: Maximum Current Conditions (Continued)
Notes 1 and 2 apply to entire table

Condition	State Symbol	Voltage Supply									Units	Notes
		V_{DD}	V_{DDPLLA}[4]	V_{DDPLLB}[4]	V_{DDPLLR}	V_{TT}	V_{TR}	V_{DDM}	V_{CCP}	V_{DDK}		
One link operating 15 Gb/s – All other Links in down mode	H5b	–	–	–	–	–	–	–	–	–		3
One link operating 10 Gb/s – All other links in sleep mode	H6a	–	–	–	–	–	–	–	–	–	mA	3
One link operating 10 Gb/s – All other links in down mode	H6b	–	–	–	–	–	–	–	–	–		3
Self refresh All links in down mode	H7	–	–	–	–	–	–	–	–	–	mA	
Field programming of NVM and JTAG port active at 25 MHz	H8	–	–	–	–	–	–	–	–	–	mA	5

Notes:
1. Unless otherwise noted, the following conditions apply: 95°C T_{J_dram}, 105°C T_{J_logic}, MAX voltage for each supply.
2. Contact memory vendor for specific maximum current values.
3. 50% read/write mix on active links, with 64B accesses cycling through all vaults and banks with low-interleave sequential addressing.
4. Values for V_{DDPLLA} and V_{DDPLLB} represent the collective current of all link PLL supplies.
5. The current from H7 will be added to whichever state the HMC is operating in while field programming or JTAG activity is in progress.

17.2 DC Operating Conditions

Table 50: DC Electrical Characteristics

Description	Parameter	Minimum (V)	Nominal (V)	Maximum (V)	Notes
Ground	V_{SS}	0	0	0	
Logic core source	V_{DD}	0.873	0.9	0.927	1
Link TX termination source	V_{TT}	1.14	1.2	1.26	
Link RX termination source	V_{TR}	1.14	1.2	1.26	2
Link PLLA source	$V_{DDPLLA[3:0]}$	1.14	1.2	1.26	2, 3
Link PLLB source	$V_{DDPLLB[3:0]}$	1.14	1.2	1.26	2, 3
Intermediate frequency PLL source	V_{DDPLLR}	1.14	1.2	1.26	2, 3
DRAM source	V_{DDM}	1.14	1.2	1.26	
DRAM wordline boost source	V_{CCP}	2.375	2.5	2.625	
JTAG, NVM, I^2C source	V_{DDK}	1.45	1.5	1.7	

Notes:
1. AC noise tolerance is vendor specific.
2. The same external voltage supply can be shared between all 1.2V supplies.
3. $V_{DDPLLA[3:0]}$, $V_{DDPLLB[3:0]}$, and V_{DDPLLR} each require a ferrite bead filter between the supply and HMC pins as is shown in the figure below. Minimum isolation is 35dB for frequencies below 8.4MHz. Supply bypass requirements for all other supplies are vendor specific.
4. Specifications within this table represent limits that must be met at the HMC package balls.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 32: V_{DDPLLX} Filtering Requirement

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

18 HMC-15G-SR Physical Link Specifications

18.1 Physical Link Electrical Interface

The HMC transmitter and receiver circuits are designed to allow for flexible integration with the host. Various supported configurations are shown in "Termination Configurations" below. Both the HMC Rx and Tx circuits are meant to be connected to their respective host lanes through the use of 100Ω differential line impedance. The SR PHY supports channels consistent with existing standards (OIF CEI-11-SR, INCITS FC-PI-5, IEEE nAUI).

18.1.1 Termination Configurations

The impedance values shown are nominal. Host transmitter and receiver termination and signaling parameters may differ from those of the HMC as long as the HMC receiver parameters are still met as per Table 52, "TX Signaling Parameters," on page 99 and Table 53, "RX Signaling Parameters," on page 101.

The HMC has been designed to minimize the probability of long DC run lengths with its scrambling and descrambling circuitry. The host transmitter must meet the requirements found in 4.3 "Lane Run Length Limitation" on page 17. Additional run length limit circuitry to avoid run lengths of longer than 85UI at the HMC transmitter output is also available (See 4.3 "Lane Run Length Limitation" on page 17). Specific host AC coupled receiver encoding requirements beyond this, such as 64b/66b encoding or 8b/10b encoding, are not supported.

Figure 33: HMC Tx Driving Host Rx - Example #1

x = +1 or –1 depending on data value

V_{TT} = Transmit termination source (measured at HMC pin)

V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

V_{HRX} is an on-die voltage supply that must equal one of the following:

1) $V_{TT}/2$ (±50mV tolerance)
2) V_{TT} - 200mV < V_{HRX} < V_{TT} - 100mV

Notes: 1. Implementation of 2) is subject to junction temperature restrictions of the HMC.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 34: HMC Tx Driving Host Rx - Example #2

x = +1 or -1 depending on data value

V_{TT} = Transmit termination source (measured at HMC pin)

V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

Capacitor (C_T) recommended to control Rx common-mode return loss

Figure 35: HMC Tx Driving Host Rx with External AC Coupling Present

x = +1 or -1 depending on data value

V_{TT} = Transmit termination source (measured at HMC pin)

V_{DIFF_TX} = Differential peak-to-peak transmit voltage (measured at HMC pin)

Tx common mode voltage (V_{CMTX}) = $V_{TT}/2$

Value of V_{HRX} depends on the host receiver design
Rx common mode voltage (V_{CMRX}) = V_{HRX}

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 36: Host Tx Driving HMC Rx Without External AC Coupling

x = +1 or –1 depending on data value
V_{CMTX} = Transmit common mode voltage
V_{SW} = Voltage swing

Receive common mode voltage = V_{TR} /2

Shared common mode voltage
$V_{CM} = V_{CMTX}$

Notes: 1. Tx output impedance is 50Ω nominal (not shown).

Figure 37: Host Tx Driving HMC Rx With External AC Coupling

x = +1 or –1 depending on data value
V_{CMTX} = Transmit common mode voltage
V_{SW} = Voltage swing

Receive common mode voltage = V_{TR} /2

Notes: 1. Tx output impedance is 50Ω nominal (not shown).

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

18.2 Equalization Schemes

The HMC uses equalization schemes on both its transmitter and receiver to mitigate signal integrity issues that arise from media losses, crosstalk, and intersymbol interference.

The HMC transmitter implements feed forward equalization (FFE) using a programmable 3-tap, baud-spaced finite impulse response (FIR) driver with the following equation:

$$H(Z) = K(C_0z^{+1} + C_1z^0 + C_2z^{-1})$$

The driver amplitude (K) is adjusted in the range of 500 to 1000 mV in 33 power settings. The relative weights of C_0 to C_2 are user-configurable to create a wide variety of transmitter FIR pulse shaping filters.

The receiver provides a powerful combination of an automatic gain control (AGC) amplifier with dynamic peaking control (DPC) plus a baud-spaced decision feedback equalizer (DFE) circuit that complements the transmitter equalization capability. The DFE adaptation circuit examines the incoming serial stream and dynamically adjusts the H1 tap coefficient to maximize the internal eye opening.

Using the equalization capabilities of the high-speed SerDes cores, channel losses in excess of 12dB at the fundamental frequency are readily overcome and a BER <1E-17 can be achieved provided a fully compatible SerDes is used by the host.

18.3 Link Bit Rate

The HMC link interface is synchronous on both the upstream and downstream lanes. It is not plesiochronous. The choices for link bit rate are shown in Table 51. Any fixed skew is allowable between the reference clock at the host and HMC. Note that any variation below approximately 20MHz is indistinguishable from sinusoidal jitter, and will be subtracted from the amount allowed in Figure 38 on page 102.

Table 51: Synchronous Link Bit Rate Specifications

Parameter	Symbol	Test Conditions/Comments	Min	Typical	Max	Unit	Notes
Bit rate	BR_{10}	f_{REFCLK} = 125.00 MHz	10.00	–	10.00	Gbps	1
Bit rate	$BR_{12.5}$	f_{REFCLK} = 125.00 MHz	12.50	–	12.50	Gbps	1
Bit rate	BR_{15}	f_{REFCLK} = 125.00 MHz	15.00	–	15.00	Gbps	1
Bit rate range	BRR		0	–	0	ppm	

Notes: 1. The link interface is synchronous. In this table, the bit rate is exactly 80, 100, or 120 times the reference clock frequency. See Table 56, "Reference Clock Parameters," on page 103 for alternative reference clock frequencies.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

18.4 High Speed Signaling Parameters

All parameters within Table 52 and Table 53 represent those of the HMC. Host transmitter and receiver termination and signaling parameters may differ from those of the HMC as long as the parameters within the tables are still met for the HMC.

Table 52: TX Signaling Parameters

Parameter	Symbol	Test Conditions / Comments	Min	Typical	Max	Unit	Notes
Link supply Tx termination voltage	V_{TT}	1.2 V ±0.06V	1.14	1.2	1.26	V	
Differential peak-to-peak output voltage	V_{DIFF_TX} SMALL_SWING	$V_{DIFF_TX} = 2 \times \lvert V_{TX_P} - V_{TX_N} \rvert$ Measured with slow square wave (64 zeros followed by 64 ones) with ideal 100Ω floating differential termination	430mV		–	mV_{PP}	1
	V_{DIFF_TX} LARGE_SWING		860mV		$1.1 \times V_{TT}$	mV_{PP}	2
Single-ended voltage (with respect to V_{SS}) on V_{TX_P}, V_{TX_N}	V_{TX_SE}	Active mode	$0.2 \times V_{TT}$		$0.8 \times V_{TT}$	mV	
Down or sleep mode output voltage	V_{TX_PD}			High-Z		mV	3
DC common mode output voltage	V_{TX_CM}	V_{TX_CM} = DC(avg) of $\lvert V_{TX_P} + V_{TX_N} \rvert$		See Note 4		mV	4
AC common mode noise	$V_{TX_CM_NZ}$ SQUARE WAVE	1000mV differential peak-to-peak slow square wave output amplitude			50	mV_{PP}	
	$V_{TX_CM_NZ}$ PRBS	PRBS data pattern			15	mV_{TMS}	
Short circuit current	I_{SHORT_TX}	Output pins shorted to GND or each other	–50		50	mA	
Differential Tx output rise/fall time	t_{TX_RISE}, t_{TX_FALL}	Measured from 20% to 80% into ideal 100Ω load	18			ps	
Differential transmitter resistance	R_{OD}		80	100	120	Ω	5
Single-ended transmitter resistance	R_{OSE}		40	50	60	Ω	5
Single-ended transmitter termination mismatch	$R_{TX_DELTA_AC}$	AC-coupled or DC-coupled ($V_{HRX} = V_{TT} \pm 50mV$); Measured within a differential pair			5	%	
	$R_{TX_DELTA_DC}$	DC-coupled (V_{TT} - 200mV < V_{HRX} < V_{TT} - 100mV); Measured within a differential pair			25	%	6, 7

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 52: TX Signaling Parameters (Continued)

Parameter	Symbol	Test Conditions / Comments	Min	Typical	Max	Unit	Notes
Output Return Loss - Relative to 100Ω differential system	$RL_{TX-DIFF}$	50 MHz–2.8 GHz			–12	dB	
		2.8 GHz–4.25 GHz			–8.15 + 13.3 × log (f/5.5 GHz)	dB	
		4.25 GHz–7.5 GHz			-10	dB	
		7.5 GHz–15 GHz			–10 + 0.67 × (f - 7.5 GHz)	dB	
		15 GHz–16.5 GHz			–5 + 1.21 × (f - 15 GHz)	dB	
Common mode return loss - relative to ideal 25Ω impedance	RL_{TX-CM}	50 MHz to 16.5 GHz			–6	dB	
Jitter generation, 15.0 Gbps	DJ_{TX}	Measured at output ball			0.15	UI	
	DCD_{TX}	Measured at output ball; Considered part of and included within DJ_{TX}			5	%UI	
	RJ_{TX}	1E-12 BER (peak-to-peak jitter)			0.12	UI	
	TJ_{TX}	1E-12 BER			0.27	UI	
Output differential skew	$^{t}TX-SKEW_diff$		–		15	ps	
Lane-to-lane output skew at TX	$L_{TX-SKEW}$		–		200	ps	8
Tx drift tolerance	t_{TX_DRIFT}		–	–	±6	UI	

Notes:
1. Transmit amplitude register set for small swing output voltage.
2. Transmit amplitude register set for large swing output voltage.
3. The Tx goes into a High-Z state and the output will float HIGH or LOW, depending upon the termination. The Tx output will nominally pull to $V_{TT}/2$ if external AC coupling is used (see Figure 35 on page 96). The Tx output voltage in all other cases will depend on the host receiver termination.
4. Tx output common-mode voltage in a DC-coupled link is dependent on the type of termination used at the receiver. See 18.1.1 “Termination Configurations” on page 95 for supported termination topologies.
5. Refers to the resistive portion of the termination.
6. Tx performance may degrade in DC-coupled mode. In the presence of poor Rx common-mode return loss the AC common-mode noise can increase by 2x. The actual Rx termination voltage will impact the single ended output resistance matching.
7. This behavior is not compensated for with impedance calibration.
8. Measured at the crossing point of each differential pair.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 53: RX Signaling Parameters

Parameter	Symbol	Test Conditions/ Comments	Min	Typical	Max	Unit	Notes
Link supply Rx termination voltage	V_{TR}	1.2 V ±0.06 V	1.14	1.2	1.26	V	
Differential peak-to-peak input voltage	V_{DIFF_RX}	$V_{DIFF_RX} = 2 \times \lvert V_{RX_P} - V_{RX_N} \rvert$ measured with reference load	100		1400	mV	1
Single-ended voltage (with respect to V_{SS}) on D+, D-	V_{RX_SE}	V_{TR} = 1.2V	0		V_{TR} + 300	mV	
Common mode of the input voltage	V_{RX_CM}	V_{RX_CM} = DC(avg) of $\lvert V_{RX_P} + V_{RX_N} \rvert / 2$	300		V_{TR}	mV	
Short circuit current, Rx off	$I_{SHORT\ RX_OFF}$		-100		100	mA	2
Short circuit current, Rx on	$I_{SHORT\ RX_ON}$		-100		100	mA	2
Differential input resistance	R_{ID}		80	100	120	Ω	
	R_{ISE}		40	50	60	Ω	
Single-ended input resistance matching	R_{RX_DELTA}		–		5	%	
Differential input return loss – relative to 100Ω differential system	RL_{RX_DIFF}	F_B/1667 to (3/4)F_B	-8		–	dB	3
		(3/4)F_B to F_B	-8 +16.6 × log (f/(0.75 × F_B))			dB	3
Common mode return loss – relative to ideal 25Ω	RL_{RX_CM}	Measured over F_B/1667 to 3/4F_B	-6		–	dB	3
Jitter tolerance	DJ_{RX}	Comprised of ISI, DCD, reflections, and crosstalk (does not include SJ)	–		0.42	UI	4
	SJ_{RX}	Sinusoidal, low frequency	0.1		8	UI	4,5, 7
	TJ_{RX}				0.65	UI	4
Input differential skew	RX_{SKEW_DIFF}				±25	ps	
Lane-to-lane skew at RX	L_{RX_SKEW}	Lane-to-lane skew at a receiver that must be tolerated	–		16	UI	6, 7
RX drift tolerance	t_{RX_DRIFT}		–		±7	UI	

Notes:
1. The $V_{DIFF_RX_min}$ is an estimate which will be revised after the benefit of the Rx DFE has been fully assessed. $V_{DIFF_RX_max}$ value is based on a low frequency data pattern (16 ones followed by 16 zeros).
2. The receiver input absolute maximum voltage ratings in Table 48 on page 92 cannot be exceeded for extended periods of time without affecting device reliability.
3. Half-baud frequency, F_B, specified in GHz.
4. The Rx decision feedback equalization (DFE) can effectively open a "closed" eye signal at its inputs, to achieve BER of 1E-12 or better. These jitter tolerance specifications correspond to the minimum capability of the Rx when DFE is disabled for debug purposes. Assessment of DFE capabilities and BER for a given host Tx and channel model can be performed using IBIS-AMI receiver models.
5. See Table 38, "Receiver Sinusoidal Jitter Tolerance," on page 102.
6. Lane-to-lane skew contributes directly to memory latency, so minimizing skew between Rx lanes is recommended.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

7. Any reduction in SJrx below the maximum allowed can be added directly to the lane-to-lane skew budget. In other words, Lrx_skew + SJrx = 24UI, yet SJrx must not exceed 8UI.

Figure 38: Receiver Sinusoidal Jitter Tolerance

18.5 Non-High Speed Link Parameters

Table 54: Initialization Timing Parameters

Description	Symbol	Min	Max	Unit
Power supply ramp time	^{t}DD	–	200	ms
Power supply slew rate	V_{DVDT}	–	10	V/ms
Assertion time for P_RST_N	^{t}RST	20	–	ns
P_RST_N slew rate	$V_{RSTDVDT}$	0.1	–	V/ns
PLL and register configuration time	^{t}INIT	–	20	ms
Link receiver-phase acquisition (no DFE)	^{t}RESP1	–	1	µs
Link receiver-phase acquisition (with DFE)	^{t}RESP1_DFE	–	1.2	ms
FLIT synchronization time	^{t}RESP2	–	1	µs

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 55: Link Power Management Parameters

Parameter Description	Symbol	Min	Max	Units	Comments
Voltage Parameters					
LxRXPS input LOW voltage	V_{IL}	–	V_{SS} + 0.5V	V	V_{SS} measured at HMC package ball
LxRXPS input HIGH voltage	V_{IH}	V_{DDK} - 0.5V	–	V	V_{DDK} measured at HMC package ball
LxTXPS output LOW voltage	V_{OL}	–	V_{SS} + 0.1V	V	I_{OL} = 1mA
LxTXPS output HIGH voltage	V_{OH}	V_{DDK} - 0.1V	–	V	I_{OH} = 1mA
Timing Parameters					
Power state transition timing	^{t}PST	–	80	ns	Delay from the transition of a link's LxRXPS signal to the transition of its LxTXPS signal
LxRXPS setup timing to enter down mode	^{t}IS	10	–	ns	Setup time of LxRXPS to P_RST_N transition from LOW to HIGH
Simultaneous power transition stagger	^{t}SS		500	ns	Entry/exit time from a state during simultaneous link power state transitions
Sleep mode entry	^{t}SME	–	600	ns	Time from transition LxTXPS LOW to sleep mode entry
Time to transition links from sleep mode to down mode	^{t}SD		150	us	Measured from last link to take LxTXPS LOW to down mode Entry of all links
Time required to stay in self refresh	^{t}SREF	1.0	–	ms	Measured from last LxRXPS LOW to first LxRXPS HIGH
Time required to stay within active mode	^{t}OP	1.0	–	ms	Measured by LxTXPS HIGH time
HSS PLL self calibration timing	^{t}PSC	–	Vendor specific		Required after returning to active mode from down mode
Low power exit timing: transition of LxRXPS to RX PRBS	^{t}RXD	–	Vendor specific		Delay from transition of responder's LxRXPS to receiving PRBS at its RX
Low power exit timing: transition of LxRXPS to RX NULL	^{t}RXD_NULL	–	Vendor specific		Delay from transition of responder's LxRXPS to receiving NULL FLITs at its RX (only applicable to HMC links connected to host)
Low power exit timing: transition of LxTXPS to TX PRBS	^{t}TXD	–	Vendor specific		Delay from transition of responder's LxTXPS to its TX sending PRBS
Low power exit timing: TX PRBS to Rx NULL	^{t}SigDet	–	Vendor specific		Delay from responder's TX sending PRBS to receiving NULL FLITs at its RX (only applicable for cube-to-cube link)

Table 56: Reference Clock Parameters

Parameter	Symbol	Min	Nom	Max	Units	Notes
Reference clock input frequency	f_{REFCLK}	–	125, 156.25, 166.67	–	MHz	1
AC Coupled Inputs (LVPECL, LVDS Compatible Signaling)						
Input swing voltage LVPECL	V_{ID_LVPECL}	310	–	–	mV_{PP}	2
Input swing voltage LVDS	V_{ID_LVDS}	200	–	–	mV_{PP}	2
DC Coupled Inputs (HSTL Compatible Signaling)						
Input swing voltage HSTL	V_{ID_HSTL}	400	–	–	mV	
Crossing voltage	V_{IX}	680	750	900	mV	
Reference Clock Phase Noise Requirements						

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 56: Reference Clock Parameters (Continued)

Parameter	Symbol	Min	Nom	Max	Units	Notes
Offset from nominal input frequency	100Hz	–	–	–85	dBc/Hz	
	1 kHz	–	–	–97	dBc/Hz	
	10 kHz	–	–	–97	dBc/Hz	
	100 kHz	–	–	–114	dBc/Hz	
	1 MHz–1 GHz	–	–	–126	dBc/Hz	

Notes: 1. Recommend using 0.1µF AC coupling capacitor.

18.6 Impedance Calibration

The HMC includes internal calibration circuitry to maintain tight termination impedance tolerance. This circuitry auto-calibrates and requires no action from the user other than the placement of a 200Ω high-precision resistor (0.1% tolerance recommended) between EXTRESTP and EXTRESTN as well as one between EXTRESBP and EXTRESBN.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

19 Packages for HMC-15G-SR Devices

Figure 39: HMC Package Drawing (4-Link HMC-15G-SR Device)

Notes: 1. All dimensions are in millimeters.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 40: HMC Ballout for 4-link HMC-15G-SR

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
A		VSS	L2TXP[4]	L2TXN[4]	VSS	VSS	L2RXP[8]	L2RXN[8]	VSS	VSS	L2TXP[13]	L2TXN[13]	VSS	VSS	L0RXN[2]	L0RXP[2]	VSS	VSS	L0TXN[6]	L0TXP[6]	VSS	VSS	L0RXN[3]	L0RXP[3]	VSS	VSS	L0TXN[15]	L0TXP[15]	VSS		A
B	VSS	VSS	VSS	L2TXP[14]	L2TXN[14]	VSS	VSS	L2RXP[12]	L2RXN[12]	VSS	VSS	L2TXP[12]	L2TXN[12]	VSS	VSS	VSS	VSS	L0TXN[7]	L0TXP[7]	VSS	VSS	L0RXN[7]	L0RXP[7]	VSS	VSS	L0TXN[5]	L0TXP[5]	VSS	VSS	VSS	B
C	L2RXP[1]	L2RXN[1]	VSS	VSS	L2TXP[0]	L2TXN[0]	VSS	VSS	L2RXP[13]	L2RXN[13]	VSS	DNU	L2TXP[8]	L2TXN[8]	VSS	VSS	L0TXN[3]	L0TXP[3]	VSS	VSS	L0RXN[6]	L0RXP[6]	VSS	DNU	L0TXN[11]	L0TXP[11]	VSS	VSS	L0RXN[10]	L0RXP[10]	C
D	VSS	L2RXP[0]	L2RXN[0]	VSS	VSS	L2TXP[15]	L2TXN[15]	VSS	VSS	L2RXP[14]	L2RXN[14]	VSS	VSS	L2TXP[9]	L2TXN[9]	L0TXN[2]	L0TXP[2]	VSS	VSS	L0RXN[5]	L0RXP[5]	VSS	VSS	L0TXN[4]	L0TXP[4]	VSS	VSS	L0RXN[11]	L0RXP[11]	VSS	D
E	DNU	VSS	L2RXP[4]	L2RXN[4]	VSS	VSS	L2TXP[11]	L2TXN[11]	VSS	VSS	L2RXP[15]	L2RXN[15]	VSS	VSS	VSS	VSS	VSS	VSS	L0RXN[4]	L0RXP[4]	VSS	VSS	L0TXN[0]	L0TXP[0]	VSS	VSS	L0RXN[15]	L0RXP[15]	VSS	VSS	E
F	VSS	VSS	VSS	L2RXP[5]	L2RXN[5]	VSS	VSS	L2TXP[10]	L2TXN[10]	VSS	VSS	L2RXP[10]	L2RXN[10]	VSS	L2RXP[9]	L2RXN[9]	VSS	L0RXN[1]	L0RXP[1]	VSS	VSS	L0TXN[1]	L0TXP[1]	VSS	VSS	L0RXN[14]	L0RXP[14]	VSS	VSS	VSS	F
G	L2TXP[5]	L2TXN[5]	VSS	VSS	L2RXP[6]	L2RXN[6]	VSS	VSS	L2TXP[1]	L2TXN[1]	VSS	VSS	L2RXP[11]	L2RXN[11]	VSS	VSS	L0RXN[0]	L0RXP[0]	VSS	VSS	L0TXN[10]	L0TXP[10]	VSS	VSS	L0RXN[13]	L0RXP[13]	VSS	VSS	L0TXN[14]	L0TXP[14]	G
H	VSS	L2TXP[6]	L2TXN[6]	VSS	VSS	L2RXP[7]	L2RXN[7]	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	L0RXN[12]	L0RXP[12]	VSS	VSS	L0TXN[13]	L0TXP[13]	VSS	H
J	EXTREST P	VSS	L2TXP[7]	L2TXN[7]	VSS	VSS	VSS	VTR	VSS	VTT	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VSS	VTR	VSS	VTR	VSS	VSS	VSS	VSS	L0TXN[12]	L0TXP[12]	VSS	DNU	J
K	VSS	EXTREST N	VSS	L2TXP[3]	L2TXN[3]	VSS	VSS	VSS	VTT	VSS	VTT	VDDPLLB2	VDDPLLA2	VDDPLLB2	VDDPLLA2	VDDPLLB0	VDDPLLA0	VDDPLLB0	VDDPLLA0	VSS	VTT	VSS	VTT	VSS	VSS	L0TXN[8]	L0TXP[8]	VSS	VSS	VSS	K
L	L2RXP[3]	L2RXN[3]	VSS	VSS	L2TXP[2]	L2TXN[2]	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTR	VSS	VSS	L0TXN[9]	L0TXP[9]	VSS	VSS	L0RXN[8]	L0RXP[8]	L
M	VSS	VSS	L2RXP[2]	L2RXN[2]	VSS	VSS	DNU	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS[1]	VSS	VSS	L0RXN[9]	L0RXP[9]	VSS	VSS	M
N	REFCLK_ BOOT[0]	VSS	VSS	VSS	VSS	L2RXPS	DNU	DNU	VDD	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	L0TXPS	DNU	VSS	VSS	VSS	VSS	VSS	N
P	DNU	L3RXPS	L3TXPS	DNU	DNU	VSS[1]	TDI	VSS	DNU	DNU	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	CUB[1]	DNU	DNU	DNU	DNU	L1RXPS	VDDPLL R	P
R	DNU	VSS	DNU	VDDK	TMS	VDDK	TRST_N	DNU	VSS	VDDK[1]	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VSS[1]	VSS	VSS[1]	VSS	VSS[1]	VSS	VSS[1]	VDDK	SCL	VSS	REFCLK N	R
T	TDO	VSS	VSS[1]	VSS	TCK	VSS	DNU	VSS[1]	VDD	VDD	VSS	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	SDA	VDDK	VSS[1]	VSS	DNU	VSS	REFCLK P	T
U	DNU	L2TXPS	DNU	DNU	FERR_N	VSS[1]	DNU	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VCCP	VSS	DNU	DNU	L0RXPS	REFCLK_ BOOT[1]	P_RST_N	U
V	VSS[1]	VSS	VSS	VSS	VSS	REFCLK SEL	DNU	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	VDD	VSS	DNU	L1TXPS	CUB[0]	VSS	VSS	VSS	VSS	VSS[1]	V
W	VSS	VSS	L3RXP[2]	L3RXN[2]	VSS	VSS	CUB[2]	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VDDM	VSS	VSS	VSS	VSS	L1RXN[9]	L1RXP[9]	VSS	VSS	W
Y	L3RXP[3]	L3RXN[3]	VSS	VSS	L3TXP[2]	L3TXN[2]	VSS	VSS	VTR	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VSS	L1TXN[9]	L1TXP[9]	VSS	VSS	L1RXN[8]	L1RXP[8]	Y
AA	VSS	EXTRES BN	VSS	L3TXP[3]	L3TXN[3]	VSS	VSS	VTT	VSS	VTT	VSS	VDDPLLB3	VDDPLLA3	VDDPLLB3	VDDPLLA3	VDDPLLB1	VDDPLLA1	VDDPLLB1	VDDPLLA1	VTT	VSS	VTT	VSS	VSS	VSS	L1TXP[8]	L1TXP[8]	VSS	VSS	VSS	AA
AB	EXTRESB P	VSS	L3TXP[7]	L3TXN[7]	VSS	VSS	VSS	VSS	VTR	VSS	VTR	VSS	VSS	VTT	VSS	VTR	VSS	VTT	VSS	VTR	VTT	VSS	VTR	VSS	VSS	VSS	L1TXN[12]	L1TXP[12]	VSS	DNU	AB
AC	VSS	L3TXP[6]	L3TXN[6]	VSS	VSS	L3RXP[7]	L3RXN[7]	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	L1RXN[12]	L1RXP[12]	VSS	VSS	L1TXN[13]	L1TXP[13]	VSS	AC
AD	L3TXP[5]	L3TXN[5]	VSS	VSS	L3RXP[6]	L3RXN[6]	VSS	VSS	L3TXP[1]	L3TXN[1]	VSS	VSS	L3RXP[11]	L3RXN[11]	VSS	VSS	L1RXN[0]	L1RXP[0]	VSS	VSS	L1TXN[10]	L1TXP[10]	VSS	VSS	L1RXN[13]	L1RXP[13]	VSS	VSS	L1TXN[14]	L1TXP[14]	AD
AE	VSS	VSS	VSS	L3RXP[5]	L3RXN[5]	VSS	VSS	L3TXP[10]	L3TXN[10]	VSS	VSS	L3RXP[10]	L3RXN[10]	VSS	L3RXP[9]	L3RXN[9]	VSS	L1RXN[1]	L1RXP[1]	VSS	VSS	L1TXN[1]	L1TXP[1]	VSS	VSS	L1RXN[14]	L1RXP[14]	VSS	VSS	VSS	AE
AF	DNU	VSS	L3RXP[4]	L3RXN[4]	VSS	VSS	L3TXP[11]	L3TXN[11]	VSS	VSS	L3RXP[15]	L3RXN[15]	VSS	VSS	VSS	VSS	VSS	VSS	L1RXN[4]	L1RXP[4]	VSS	VSS	L1TXN[0]	L1TXP[0]	VSS	VSS	L1RXN[15]	L1RXP[15]	VSS	VSS	AF
AG	VSS	L3RXP[0]	L3RXN[0]	VSS	VSS	L3TXP[15]	L3TXN[15]	VSS	VSS	L3RXP[14]	L3RXN[14]	VSS	VSS	L3TXP[9]	L3TXN[9]	L1TXN[2]	L1TXP[2]	VSS	VSS	L1RXN[5]	L1RXP[5]	VSS	VSS	L1TXN[4]	L1TXP[4]	VSS	VSS	L1RXN[11]	L1RXP[11]	VSS	AG
AH	L3RXP[1]	L3RXN[1]	VSS	VSS	L3TXP[0]	L3TXN[0]	VSS	VSS	L3RXP[13]	L3RXN[13]	VSS	DNU	L3TXP[8]	L3TXN[8]	VSS	VSS	L1TXN[3]	L1TXP[3]	VSS	VSS	L1RXN[6]	L1RXP[6]	VSS	DNU	L1TXN[11]	L1TXP[11]	VSS	VSS	L1RXN[10]	L1RXP[10]	AH
AJ	VSS	VSS	VSS	L3TXP[14]	L3TXN[14]	VSS	VSS	L3RXP[12]	L3RXN[12]	VSS	VSS	L3TXP[12]	L3TXN[12]	VSS	VSS	VSS	VSS	L1TXN[7]	L1TXP[7]	VSS	VSS	L1RXN[7]	L1RXP[7]	VSS	VSS	L1TXN[5]	L1TXP[5]	VSS	VSS	VSS	AJ
AK		VSS	L3TXP[4]	L3TXN[4]	VSS	VSS	L3RXP[8]	L3RXN[8]	VSS	VSS	L3TXP[13]	L3TXN[13]	VSS	VSS	L1RXN[2]	L1RXP[2]	VSS	VSS	L1TXN[6]	L1TXP[6]	VSS	VSS	L1RXN[3]	L1RXP[3]	VSS	VSS	L1TXN[15]	L1TXP[15]	VSS		AK
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	

Notes:
1. Pin is used for internal test purposes and is not part of power delivery network. It must be connected as indicated in ball-out.
2. Ballout represents an X-ray view, looking through package down to the balls.
3. Vendors may choose to use any power rail pin or V_{SS} pin for internal test purposes instead of including it within the power delivery network.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

20 HMC-10G-USR Electrical Specifications

20.1 Absolute Maximum Ratings

Stresses greater than those listed may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions outside those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may adversely affect reliability.

Table 57: Absolute Maximum Ratings

Description	Parameter	Minimum	Maximum	Units	Comments
Logic core source	V_{DD}	–0.4	1.15	V	
PLL sources	$V_{DDPLL[7:0]}$	–0.5	1.7	V	
Link termination sources	V_{TT}, V_{TR}	–0.4	1.3	V	
DRAM source	V_{DDM}	–0.4	1.5	V	
DRAM wordline boost source	V_{CCP}	–0.4	2.75	V	
JTAG, NVM, I^2C source	V_{DDK}	–0.4	1.95	V	
Link input signal pin voltage (LxRXP, LxRXN)	Link input signal pin voltage (LxRXP, LxRXN)	More positive of -0.1V or V_{TR}-1.1V	V_{TR} + 0.3V	V	
Link output signal pin voltage (LxTXP, LxTXN)	Link output signal pin voltage (LxTXP, LxTXN)	More positive of -0.1V or V_{TT} -1.1V	V_{TT}	V	

Table 58: Maximum Current Conditions
Note 1 applies to entire table

Condition	State Symbol	Voltage Supply							Units	Notes
		V_{DD}	V_{DDPLL}[3]	V_{TT}	V_{TR}	V_{DDM}	V_{CCP}	V_{DDK}		
Reset P_RST_N LOW	H0	–	–	–	–	–	–	–	mA	2
Active standby at 10Gb/s NULL commands on all links	H1	–	–	–	–	–	–	–	mA	2
All links operating 10 Gb/s	H2	–	–	–	–	–	–	–	mA	2
One link operating 10 Gb/s – All other links in sleep mode	H3a	–	–	–	–	–	–	–	mA	2
One link operating 10 Gb/s – All other Links in down mode	H3b	–	–	–	–	–	–	–	mA	2
Self refresh All links in down mode	H4	–	–	–	–	–	–	–	mA	

Notes:
1. Unless otherwise noted, the following conditions apply: 95°C T_{J_dram}, 105°C T_{J_logic}, MAX voltage for each supply.
2. 50% read/write mix on active links, with 64B accesses cycling through all vaults and banks with low-interleave sequential addressing.
3. Values for V_{DDPLL} represent the collective current of all link PLL supplies.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

4. The current from H4 will be added to whichever state the HMC is operating in while field programming or JTAG activity is in progress.
5. Contact memory vendor for specific maximum current values.

20.2 DC Operating Conditions

Table 59: DC Electrical Characteristics

Description	Parameter	Minimum (V)	Nominal (V)	Maximum (V)	Notes
Ground	V_{SS}	0	0	0	
Logic core source	V_{DD}	0.87	0.9	0.93	1
Link TX termination source	V_{TT}	0.95	1	1.05	2
Link RX termination source	V_{TR}	0.95	1	1.05	2
Link TX PLLA source	$V_{DDPLL[7:0]}$	1.45	1.5	1.7	3, 4
DRAM source	V_{DDM}	1.14	1.2	1.26	
DRAM wordline boost source	V_{CCP}	2.38	2.5	2.63	
JTAG, NVM, I^2C source	V_{DDK}	1.45	1.5	1.7	3

Notes:
1. AC noise tolerance will be addressed in subsequent draft specification version.
2. The same external voltage supply can be shared between all 1V supplies.
3. The same external voltage supply can be shared between all 1.5V supplies.
4. A ferrite bead filter is required between the 1.2V PLL supplies and HMC pins. More details of the filter requirements will be released in future revision. Supply bypass requirements for all other supplies is TBD.
5. Specifications within this table represent limits that must be met at the HMC package balls.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

21 HMC-10G-USR Physical Link Specifications

21.1 Physical Link Electrical Interface

The HMC transmitter and receiver circuits for the 10Gb/s USR physical link (PHY) are designed to be DC coupled to the host device. Both the HMC Rx and Tx circuits are meant to be connected to their respective host lanes through the use of 100Ω differential line impedance. Channel characteristics are expected to be "well-behaved" (to be more rigorously defined in the following sections) in order to support low-power (on the order of 1pJ/bit or 1mW/Gb/s) USR PHY circuitry. This places restrictions on the tolerable channel loss as well as lane-to-lane skew within a link.

Although the HMC-15G-SR physical link can operate at 10Gb/s, there is no intent for interoperability between the HMC-10G-USR and HMC-15G-SR physical links.

21.1.1 Termination Configuration

The impedance values shown are nominal. Host transmitter and receiver termination and signaling parameters may differ from those of the HMC as long as the HMC receiver parameters are still met per Table 61, "HMC-USR TX Signaling Parameters," on page 112" and Table 63, "HMC-USR RX Signaling Parameters," on page 116. The HMC has been designed to minimize the probability of long DC run lengths with its scrambling and descrambling circuitry. Seeding values vary per lane, as described in more detail in "Scrambling and Descrambling" on page 15.

AC coupling of the HMC to the host device is not supported. In all lane connections, the transmitter must supply an appropriate common-mode level for DC compatibility with the receiver at the opposite end of the connection. The common-mode requirements are described in Table 61 and Table 63.

Figure 41 shows the HMC-USR transmitter driving a receiver in the host device. The HMC-USR transmitter has adjustable transmit swing controlled through bias current I_{DRIV}. This current is 3b controllable with 50mVppd launch amplitude steps, for a maximum launch amplitude of 400mVppd, as specified in Table 61. During startup calibration, the launch amplitude is adjusted such that the lane can run at a target BER (1E-12, 1E-14, or 1E-16) programmed by the user, while providing sufficient voltage margin to guard against power supply fluctuations. Alternatively, the value of I_{DRIV} can be programmed through the JTAG interface. Common-mode resistor R_{CM} sets the transmitter common-mode level roughly equal to $V_{TT}/4$. Capacitor C_{TT} ensures proper common-mode return loss as specified in Table 61. Capacitance values for C_{TT} will vary among HMC vendors, but it is recommended that this value be in the range of 25–35pF.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 41: HMC-USR Tx Driving Host Rx

V_{TT} = Transmit termination source (measured at HMC pin)

Capacitor (C_T) recommended to control Rx common-mode return loss

I_{DRIV} = Programmable current used to set differential peak-to-peak transmit voltage

Figure 42 depicts a host device driving an HMC-USR receiver. As the HMC-10G-USR specifies a DC-coupled connection, the common-mode output level from the host driver sets the common-mode input level to the HMC-USR receiver. As specified in Table 63 on page 116, this common-mode level is roughly VTR/4, or 250mV. Capacitor CRT ensures proper common-mode return loss as specified in Table 63. Capacitance values for CTT will vary among HMC vendors, but it is recommended that this value be in the range of 25–35pF.

Figure 42: Host Tx Driving HMC-USR Rx

x = +1 or -1 depending on data value
V_{CMTX} = Transmit common mode voltage
V_{SW} = Voltage swing

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

21.2 Link Bit Rate

The HMC-USR link is a source-synchronous interface. Each lane operates at a bit rate of 10Gb/s. A half-rate 5GHz differential clock is forwarded from the transmitter to the receiver.

Table 60: Source Synchronous Link Bit Rate Specification

Parameter	Symbol	Test Conditions/Comments	Min	Max	Unit	Notes
Bit rate	BR	^{f}REFCLK = 100 MHz	10	10	Gb/s	1
Forwarded clock rate	^{f}FC	^{f}REFCLK = 100 MHz	5	5	GHz	1
Bit rate range	BRR		–300	300	ppm	

Notes: 1. The link interface is synchronous. In this table, the bit rate is exactly 60, 64, 80 or 100 times the reference clock frequency. See Table 66, "Reference Clock Parameters," on page 119 for alternative reference clock frequencies.

21.3 High Speed Signaling Parameters

All parameters within the "HMC-USR TX Signaling Parameters" and "HMC-USR RX Signaling Parameters" tables represent those of the HMC. Host transmitter and receiver termination and signaling parameters may differ from those of the HMC as long as the parameters within the tables are still met for the HMC.

21.3.1 HMC-USR TX Signaling Specifications

21.3.1.1 HMC-USR TX Return Loss

To bound the amount of energy reflected off of the transmitter in a highly reflective and high crosstalk environment the transmitters differential and common-mode return loss is specified by means of a frequency dependent mask from 50 MHz to the fundamental of the maximum frequency. The low frequency end of this mask is within the frequency range of most 2 port VNA's and is sufficiently close to DC to provide an accurate bound on the voltage swing of the transmitter into the reference plane load of 50Ω (100Ω differential).

21.3.1.2 HMC-USR TX Jitter

In the source-synchronous HMC-10G-USR, data is transmitted at the same time as the clock edge that is subsequently used to sample the data at the receiver. The transmitter random jitter (RJ) should be low, limited by the cycle-to-cycle jitter of the PLL. The RJ should be less than 0.25 ps(rms).

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 61: HMC-USR TX Signaling Parameters

Symbol	Parameter	Min	Nom	Max	Units	Comments
UI	Unit interval		100		ps	
$V_{TX-DIFF}$	Differential peak-to-peak output swing	50	200	400	mV	3b control of the launch amplitude; should be set to the minimum to allow for operation at a target BER[1] with sufficient voltage margin. 0.5dB of through loss to the package can be tolerated in addition to the voltage swing specified.
$V_{TX-CM-DC}$	DC common-mode output voltage	100		250	mV	$V_{TX-CM-DC}$ = (TXP + TXN)/2 averaged over 10^6 UI using the scramble pattern.
$I_{SHORT-TX}$	Short circuit current	–30		30	mA	Output data or clock pins shorted to supply, GND, or each other. Can be only tolerated for a short (<1s) duration.
T_{TX-DJ}	Maximum transmit deterministic jitter			0.2	UI	
T_{TX-DCD}	Maximum data and clock duty cycle distortion			0.05	UI	Maximum pulse width deviation from 1UI measured over 10^6 UI using a 1010 pattern.
T_{TX-PW}	Minimum data output pulse width	0.9			UI	Measured at 50% level of edges voltage transition measured over 10^6 UI using the scramble pattern.
T_{TX-TRF}	Differential output rise and fall times			20	ps	20-80% rise fall times, measured at the ball[2]
$Z_{TX-DIFF}$	Differential transmitter output impedance	80	100	120	Ω	
$RL_{TX-DIFF-HF}$	Differential return loss at high frequencies			–8	dB	Measured from 2.5 GHz to 5 GHz at the ball[2]
$RL_{TX-DIFF-LF}$	Differential return loss at low frequencies			–12	dB	Measured from 50 MHz to 2.5GHz at the ball[2]
RL_{TX-CM}	Common-mode return loss			–6	dB	Measured from 50 MHz to 5GHz at the ball[2]
T_{TX-RJ}	Random Jitter			0.25	ps	1 sigma jitter. (phase noise integration range of 200MHz to 5GHz) (source synchronous link)
T_{TX-TJ}	Total jitter			0.3	UI	Total jitter measured at the ball[2]
$T_{TX-DSKEW}$	Maximum skew allowed between any two data lanes in clock forwarded group			5	ps	
$T_{TX-DCLKSKEW}$	Maximum skew allowed between CLK and any data in clock forwarded group			5	ps	
$T_{TX-PNDSKEW}$	In-pair data differential skew			3	ps	Measured at the ball[2]
$T_{TX-PNCKSKEW}$	In-pair clock differential skew			3	ps	Measured at the ball[2]

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Notes:
1. The target operating BER can be programmed by the user to values of 1E-12, 1E-14, or 1E-16. This BER is not measured, but instead is extrapolated from BER measurements near the edges of the bathtub curve. Details of the extrapolation method will be described in a future revision.
2. The ball refers to the C4 bump for a chip-scale HMC, or the package ball for a packaged HMC.

21.3.2 HMC-USR Supported Channels

The HMC-10G-USR ultra-short-reach PHY targets communication at 10 Gb/s per lane over channels with 5–6dB insertion loss at 5 GHz. Figure 43 below shows example implementation channels and their characteristics. These are shown for the purpose of illustrating channel characteristics, and are not intended to limit the packaging or channels to the materials shown in the example. Deviation from the channel guidelines and specifications are allowed but not recommended for interoperability.

Figure 43: Example HMC-USR Implementation

The on-module link consists of a multi-chip-module (MCM) with the host and HMC devices mounted to an LTCC substrate and communicating over differential chip-to-chip wiring channels on this substrate. The differential channels are anticipated to be no more than 30mm in length and the channel insertion loss at a frequency equal to half the baud rate is roughly 2dB for 10Gb/s signaling. The off-module link enables communication between a host device and an HMC residing in two different packages. In this scenario, the channel consists of wiring within both LTCC substrates, through the LGA sockets, and over up to 5” of PCB trace on a Megtron4 material. In this case, the loss is slightly higher than the on-module link. At 5 GHz, roughly 6dB of loss is expected. Figure 44 below plots the insertion loss for an on-module 30mm link, and off-module links with 3” and 5” PCB traces on a Megtron4 material. The insertion losses for the respective channels at 5 GHz are 1dB, 5dB, and 5.5dB.

Achieving ultra-low power in the HMC-USR PHY places strict requirements on channel routing. The receiver can compensate up to 25ps of lane-to-lane skew, and can tolerate up to 10ps of in-pair differential skew. It is recommended that 3ps of the in-pair skew be allocated to the transmitter, and that the channel not consume more than 7ps of this differential skew. The latter can be achieved through spread glass or angular compensation techniques.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Figure 44: Example Insertion Loss Plots for Recommended HMC-10G-USR Applications

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 62: HMC-USR Channel Parameters

Symbol	Parameter	Min	Nom	Max	Units	Comments
IL_{DC}	DC insertion loss			1	dB	Measured from TX ball[1] at 50 MHz
IL_{HB}	Insertion loss at a frequency equal to half the baud rate			6	dB	Measured from TX ball to RX ball[1] at 5 GHz
ILD_{HB}	Insertion loss deviation from DC to a frequency equal to half the baud rate	–1		1	dB	Measured from TX ball to RX ball[1], from 50 MHz to 5 GHz
$FEXT_{HB}$	Far end crosstalk at a frequency equal to half the baud rate			–22	dB	Measured at 5 GHz relative to 0dB reference, with 2 adjacent aggressors enabled
$NEXT_{HB}$	Near end crosstalk at a frequency equal to half the baud rate			–15	dB	Measured at 5 GHz relative to 0dB reference, with 2 adjacent aggressors enabled
$Z_{CH\text{-}DIFF}$	Channel differential impedance	90	100	110	Ω	

Notes: 1. The ball refers to the C4 bump for a chip-scale HMC, or the package ball for a packaged HMC.

The insertion loss deviation is a metric of the how far the insertion loss data deviates from a linear approximation. An illustration of the procedure for calculating ILD_{HB} in Table 62 is shown in Figure 5. In the plot in the top half of the figure, the insertion loss of the channel is depicted, along with a linear fit of the data from 50MHz to the half baud frequency of 5GHz. The insertion loss deviation is then found as the difference between the channel insertion loss and the linear fit, and is plotted in the lower half of Figure 5.

Figure 45: Illustration of Determining Insertion Loss Deviation (ILD_{HB})

21.3.3 HMC-USR RX Signaling Specifications

The receiver port accepts the DDR source synchronous clock from the dedicated clock lane. This lane is not interchangeable with a data bit. The architecture includes 17 data lanes (including one spare) and the single forwarded clock. Lane-to-lane skew within a USR link is bounded to 1/4UI (25 ps) end to end.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

To maximize the power transfer to the receiver from the transmission line and to bound the amount of energy reflected back to the source, the differential and common-mode return loss is specified by means of a frequency dependent mask from 50MHz to the fundamental of the maximum bit rate. The return loss is measured with the device powered up and biased over the range of $V_{TR}/4$ to $3V_{TR}/4$. Two single port S_{11} measurements are made for each half of the differential input. These results are then combined into the mixed mode differential S-parameter matrix to extract S_{DD11}.

21.3.3.1 Jitter Model

Interoperability is ensured by specifying the total jitter (TJ) of the transmitter and the total jitter of the receiver. TJ can be calculated by the Dual Dirac approximation of the random and deterministic components of jitter, RJ and DJ. The total jitter of the Transmitter $T_{TX\text{-}EYE}$ is 30 % of the UI (30ps). The total jitter of the receiver $T_{RX\text{-}EYE}$ is 40% of the UI (40ps). Jitter measurements for this source synchronous link are made with the clock as the trigger.

Table 63: HMC-USR RX Signaling Parameters

Symbol	Parameter	Min	Nom	Max	Units	Comments
UI	Unit interval		100		ps	± is set by the reference clock oscillator
$V_{RX\text{-}DIFF\text{-}MAX}$	Maximum differential peak-to-peak input swing			400	mV	Maximum outer eye, measured over 10^6 UI with compliant TX and channel. TX sends the scramble pattern.
$V_{RX\text{-}DIFF\text{-}MIN}$	Minimum differential peak-to-peak input swing	50			mV	Minimum inner eye, measured over 10^6 UI with compliant TX and channel. TX sends the scramble pattern.
$V_{RX\text{-}CM\text{-}DC}$	DC common-mode output voltage	100		250	mV	$V_{TX\text{-}CM\text{-}DC}$ = (RXP + RXN)/2 averaged over 10^6 UI with the transmitter sending the scramble pattern.
$I_{SHORT\text{-}RX}$	Short circuit current	–15		15	mA	Input data or clock pins shorted to supply, GND, or each other. Can only be tolerated for short (<1s) duration.
$T_{RX\text{-}DJ}$	Maximum deterministic jitter receiver must tolerate			0.5	UI	
$T_{RX\text{-}CLK\text{-}TJ\text{-}HF}$	High frequency total jitter on clock			0.05	UI	Cycle-to-cycle jitter applied to receiver's CLK during jitter tolerance testing.
$T_{RX\text{-}PW}$	Minimum data input pulse width	0.5			UI	Measured at 50% level of edge's voltage transition.
$T_{RX\text{-}CLK\text{-}PW}$	Minimum clock input pulse width	95	100	105	ps	Measured at 50% level of edge's voltage transition.
$T_{RX\text{-}DSKEW}$	Maximum skew allowed between any two data lanes in clock forwarded group			25	ps	The receiver is designed to compensate up to 25ps lane-to-lane static skew, measured at the input ball[1].
$T_{RX\text{-}DCLKSKEW}$	Maximum skew allowed between CLK and any data in clock forwarded group			25	ps	The clock can fall anywhere in the 25ps total data skew, measured at the input ball[1].

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 63: HMC-USR RX Signaling Parameters

Symbol	Parameter	Min	Nom	Max	Units	Comments
$T_{RX-P/NSKEW}$	In pair skew at the RX pins			10	ps	End-to-end skew in a differential pair at the input ball[1].
T_{RX-TRF}	Differential output rise and fall times			50	ps	20-80% rise fall times, measured at the RX ball[1].
$Z_{RX-DIFF}$	Differential receiver input impedance	80	100	120	Ω	
$RL_{RX-DIFF-HF}$	Differential return loss at high frequencies			–8	dB	Measured from 2.5 GHz to 5 GHz at the input ball[1].
$RL_{RX-DIFF-LF}$	Differential return loss at low frequencies			–12	dB	Measured at 50 MHz to 2.5 GHz at the input ball[1].
RL_{RX-CM}	Common-mode return loss at high frequencies			–6	dB	Measured from 50 MHz to 5 GHz at the input ball[1].
T_{RX-RJ}				0.25	ps	1 sigma jitter. Phase noise integration range of 200MHz to 5GHz. Source synchronous link
T_{RX-EYE}	Cumulative receive total eye width			0.4	UI	Receiver jitter tolerance. Eye width is measured relative to the clock at the user target BER.

Notes: 1. The ball refers to the C4 bump for a chip-scale HMC, or the package ball for a packaged HMC.

21.4 Non-High Speed Link Parameters

Table 64: Initialization Timing Parameters

Description	Symbol	Min	Max	Units
Power supply ramp time	^{t}DD	–	200	ms
Power supply slew rate	V_{DVDT}	–	10	V/ms
Assertion time for P_RST_N	^{t}RST	20	–	ns
P_RST_N slew rate	$V_{RSTDVDT}$	0.1	–	V/ns
PLL and register configuration time	$^{t}INIT$	–	20	ms
Link receiver-phase acquisition	$^{t}RESP1$	–	1	µs
FLIT synchronization time	$^{t}RESP2$	–	1	µs

Table 65: Link Power Management Parameters

Description	Symbol	Min	Max	Units	Comments
Voltage Parameters					
LxRXPS input LOW voltage	V_{IL}	–	Vss + 0.5V	V	
LxRXPS input HIGH voltage	V_{IH}	VDDK - 0.5V	–	V	
LxTXPS output LOW voltage	V_{OL}	–	Vss + 0.1V	V	
LxTXPS output HIGH voltage	V_{OH}	VDDK - 0.1V	–	V	
Timing Parameters					
Power state transition timing	^{t}PST	–	80	ns	Delay from the transition of a link's LxRXPS signal to the transition of its LxTXPS signal

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 65: Link Power Management Parameters (Continued)

Description	Symbol	Min	Max	Units	Comments
LxRXPS setup timing to enter down mode	^{t}IS	10	–	ns	Setup time of LxRXPS to P_RST_N transition from LOW to HIGH
Simultaneous power transition stagger	^{t}SS	–	500	ns	Entry/exit time from a state during simultaneous link power state transitions
Sleep mode entry	^{t}SME	–	600	ns	Time from transition LxTXPS LOW to sleep mode entry
Time to transition links from sleep mode to down mode	^{t}SD	–	150	us	Measured from last link to take LxTXPS LOW to down mode entry of all links
Time required to stay in self refresh	^{t}SREF	1	–	ms	Measured from last LxRXPS LOW to first LxRXPS HIGH
Time required to stay in active mode	^{t}OP	1	–	ms	Measured by LxTXPS HIGH time
HSS PLL self calibration timing	^{t}PSC		Vendor specific		Required after returning to active mode from down mode
Low power exit timing: transition of LxRXPS to RX PRBS	^{t}RXD	–	Vendor specific		Delay from transition of responder's LxRXPS to receiving PRBS at its RX
Low power exit timing: transition of LxRXPS to RX NULL	^{t}RXD_NULL	–	Vendor specific		Delay from transition of responder's LxRXPS to receiving NULL FLITs at its RX (only applicable to HMC links connected to host)
Low power exit timing: transition of LxTXPS to TX PRBS	^{t}TXD	–	Vendor specific		Delay from transition of responder's LxTXPS to its TX sending PRBS
Low power exit timing: TX PRBS to Rx NULL	^{t}SigDet	–	Vendor specific		Delay from responder's TX sending PRBS to receiving NULL FLITs at its RX (only applicable for cube-to-cube link)

21.4.1 PLL Reference Clock

The HMC-10G-USR uses a 100 MHz reference clock with 300 ppm tolerance. The reference clock sources may be spread spectrum modulated with up to a 0.5% frequency downspread at a modulation rate of 30–33 kHz. Due to the nature of source-synchronous architectures, the impact of spread-spectrum frequency modulation on link performance is negligible.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

Table 66: Reference Clock Parameters

Description	Symbol	Min	Nom	Max	Units	Notes
Reference clock input frequency	^{f}REFCLK	–	100, 125, 156.25, 166.67	–	MHz	1
AC Coupled Inputs (LVPECL, LVDS Compatible Signaling)						
Input swing voltage LVPECL	V_{ID_LVPECL}	310	–	–	mVpp	2
Input swing voltage LVDS	V_{ID_LVDS}	200	–	–	mVpp	2
DC Coupled Inputs (HSTL Compatible Signaling)						
Input swing voltage HSTL	V_{ID_HSTL}	400	–	–	mV	
Crossing voltage	V_{IX}	680	750	900	mV	
Reference Clock Phase Noise Requirements						
Offset from nominal input frequency	100Hz	–	–	–85	dBc/Hz	3
	1kHz	–	–	–97	dBc/Hz	3
	10kHz	–	–	–97	dBc/Hz	3
	100kHz	–	–	–114	dBc/Hz	3
	1MHz–1GHz	–	–	–126	dBc/Hz	3

Notes:
1. Nominal values other than the primary frequency can be enabled through the JTAG or I^2C bus.
2. Recommend using 0.1µF AC coupling capacitor.
3. Listed reference clock phase noise requirements are the same as those for the HMC-15G-SR. As HMC-10G-USR is a source synchronous link, sufficient link performance may be achieved without meeting these phase noise specifications. Common reference clock phase noise specifications as listed for PCIe Gen 3 are also acceptable.

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

22 Appendix A: Glossary of Terms

Table 67: Glossary of Terms

Term	Definition
ADR or ADRS	Address
Downstream	Away from the host
FIFO	First in, first out
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
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. The Gen 2 HMC has 4 links.
LSB	Least significant bit
MSB	Most significant bit
MUE	Multiple uncorrectable errors
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.
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.
RMW	Read-modify-write sequence
RRP	Return retry pointer.
SBE	Single bit error
TSV	Through-silicon via: Electrical connection through a die to enable die stacking without wires or spacers, providing excellent electrical properties.
Upstream	Toward the host
Vault	Independent memory controller and local memory stacked directly above, in a cube

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.

23 Revision History

Rev. 1.0 . . . . . 1/2013

- HMCC Specification

The Hybrid Memory Cube Consortium reserves the right to change specifications without notice.