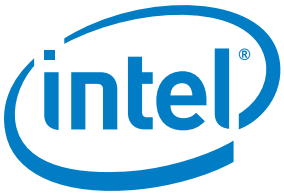

Intel® Xeon Phi™ Processor

Performance Monitoring Reference Manual—Volume 2

Events

Revision 1.0

June 2016

Document Number: 334480-001

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at http://www.intel.com/ or from the OEM or retailer.

No computer system can be absolutely secure.

Intel, the Intel logo, Intel Core, Intel SpeedStep, Intel Xeon Phi, Pentium, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Tables

Revision History

Date	Revision	Description
June 2016	1.0	• Initial Release

§

1 Introduction

The *Intel® Xeon Phi™ processor performance monitoring reference manual,* a two volume set, describes performance monitoring on the Intel® Xeon Phi™ processor (formerly codenamed Knigths Landing).

- This document, the *Intel® Xeon Phi™ processor performance monitoring reference manual—Volume 2: Events* describes a host of performance monitoring mechanisms and performance events on the Intel® Xeon Phi™ processor.
- The *Intel® Xeon Phi™ processor performance monitoring reference manual—Volume 1: Registers* provides a set of model-specific performance monitoring counter MSRs and registers. This manual also provides an overview of the Intel® Xeon Phi™ processor architecture.

The information obtained from these counters can be used for tuning system and compiler performance. Please note that the accuracy of the events and the counters described in this document is subject to change and is based on limited validation.

Refer to the following documents for an in depth understanding of the Intel® Xeon Phi™ processor architecture:

- Sodani, A, “Knights Landing: 2nd Generation Intel® Xeon Phi™ Product”, Hot Chips: A Symposium on High Performance Chips, 2015 located here.
- Sodani, A, et.al. “Knights Landing: 2nd Generation Intel® Xeon Phi™ Product”, Hot Chips Special Issue of IEEE Micro Magazine, March/April 2016.

1.1 Performance Monitoring Events in the Intel® Xeon Phi™ Processor

The Intel® Xeon Phi™ processor (also referred to as "the processor" in this document) provides performance monitoring events. Performance monitoring events in the processor is a unique combination of Intel® Atom™ processor-like core events and Intel® Xeon® processor-based server class uncore events.

1.2 Performance Monitoring Events in the Intel® Xeon Phi™ Processor Tile

An Intel® Xeon Phi™ processor tile is a core-pair based on the Silvermont (SLM) microarchitecture. As such, the processor provides SLM performance monitoring events as well as a few new processor-specific events. Some of the events were changed to meet Intel® Xeon Phi™ processor requirements. For more information on performance monitoring events in SLM microarchitecture, refer to the Section 19.12 in the *Intel 64 and IA-32 Architectures Software Developer’s Manual* (SDM).

For full description of tile performance monitoring events, refer to the Chapter 2.

1.2.1 Event Scopes in the Intel® Xeon Phi™ Processor Tile

In the Intel® Xeon Phi™ processor tile, one L2 cache is shared by two cores, each with up to four logical threads. To support this, most of the SLM microarchitecture core events changed their scope from core-level to thread-level. This means each logical thread sets up and counts events specific to it.

AnyThread support in the processor allows a thread to count across the core. In other words, if an event which is normally thread-scope is set up for AnyThread, its behavior becomes core-scope. AnyThread support is limited to Instructions Retired (on the fixed counter 0), Unhalted Core Cycles (on the fixed counter 1 or general purpose counter with Event Select 0x3C, Umask 0x00), Unhalted Reference Cycles (on the fixed counter 2 or general purpose counter with Event Select 0x3C, Umask 0x01).

The following event is core-scope by definition:

- **CORE_REJECT_L2Q.ALL** – Event Select 0x31, Umask 0x00

  This event counts the number of core requests that were not accepted into the L2Q in the L2 cache.

The following event is tile-scope by definition:

- **OFFCORE_RESP –** Event number 0xB7

  This event counts the matrix events specified by MSR_OFFCORE_RESPx.

1.2.2 New Performance Monitoring Events in the Intel® Xeon Phi™ Processor Tile

The following events are added:

- **UOPS_RETIRED.SCALAR_SIMD** – Event Select 0xC2, Umask 0x20

  This event counts the number of scalar Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512 micro-ops retired. More specifically, it counts scalar Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512 micro-ops except for loads (memory-to-register mov-type micro ops), division, sqrt.

  – This event is defined at the micro-op level and not instruction level. Most instructions are implemented with one micro-op but not all.

- **UOPS_RETIRED.PACKED_SIMD** – Event Select 0xC2, Umask 0x40

  This event counts the number of vector Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512 micro-ops retired. More specifically, it counts packed Intel® SSE, Intel® AVX, Intel® AVX2, Intel® AVX-512 micro-ops (both floating point and integer) except for loads (memory-to-register mov-type micro-ops), packed byte and word multiplies.

- The length of the packed operation (128 bits, 256 bits or 512 bits) is not taken into account when updating the counter; all count the same (+1).
- Mask (k) registers are ignored. For example: a micro-op operating with a mask that only enables one element or even zero elements will still trigger this counter (+1)
- This event is defined at the micro-op level and not instruction level. Most instructions are implemented with one micro-op but not all.

Note that the above new KNL events count the number of retired micro-ops, not FLOPs (floating point operations). As such, they cannot be used to accurately measure FLOPs. Vectorization levels of the code can be inferred by comparing PACKED_SIMD and SCALAR_SIMD event counts.

1.2.3 Updated Performance Monitoring Events in the Intel® Xeon Phi™ Processor Tile

The following events have a new umask value:

- **NO_ALLOC_CYCLES.NOT_DELIVERED** – Event Select 0xCA, Umask 0x90
- **NO_ALLOC_CYCLES.ALL** – Event Select 0xCA, Umask 0x7F

The following event changed its definition and has new name:

- **L2_REQUESTS_REJECT.ALL** – Event Select 0x30, Umask 0x00

  Formerly L2_REJECT_XQ in SLM microarchitecture. This event counts only the L2 cache access requests from L1 cache miss, excluding L2-level SW prefetch requests and L1 evictions, that were rejected. It automatically excludes L2 HW prefetch requests and Uncacheable request rejects.

The following events changed their matrix definition. Refer to Table 1-1 for matrix definition.

- **OFFCORE_RESP.OFFCORE_RESP_0** – Event Select 0xB7, Umask 0x01
- **OFFCORE_RESP.OFFCORE_RESP_1** – Event Select 0xB7, Umask 0x02

1.2.4 Offcore_Response Event Programming in the Intel® Xeon Phi™ Processor Tile

When a programmable counter is set up for an OFFCORE_RESP.OFFCORE_RESP {0, 1} event, the MSR_OFFCORE_RESP {0, 1} register (MSR address 0x1A6, 0x1A7, respectively) is used to specify request and response types.

The following table defines the bit fields of the **MSR_OFFCORE_RESP {0, 1}** registers. The Intel® Xeon Phi™ processor processor offcore_response event definition has been significantly expanded as compared to the SLM microarchitecture.

Table 1-1. Bit fields of the MSR_OFFCORE_RESP {0, 1} Registers

Main	Sub field	Bit	Name	Description
Request Type		0	DEMAND_DATA_RD	Demand cacheable data and L1 prefetch data reads
		1	DEMAND_RFO	Demand cacheable data writes
		2	DEMAND_CODE_RD	Demand code reads and prefetch code reads
		3	Reserved	Reserved
		4	Reserved	Reserved
		5	PF_L2_RFO	L2 data RFO prefetches (includes PREFETCHW instruction)
		6	PF_L2_CODE_RD	L2 code HW prefetches
		7	PARTIAL_READS	Partial reads (UC or WC)
		8	PARTIAL_WRITES	Partial writes (UC or WT or WP). Valid only for OFFCORE_RESP_1 event. Should only be used on PMC1. This bit is reserved for OFFCORE_RESP_0 event.
		9	UC_CODE_READS	UC code reads
		10	BUS_LOCKS	Bus locks and split lock requests
		11	FULL_STREAMING_STORES	Full streaming stores (WC). Valid only for OFFCORE_RESP_1 event. Should only be used on PMC1. This bit is reserved for OFFCORE_RESP_0 event.
		12	SW_PREFETCH	Software prefetches
		13	PF_L1_DATA_RD	L1 data HW prefetches
		14	PARTIAL_STREAMING_STORES	Partial streaming stores (WC). Valid only for OFFCORE_RESP_1 event. Should only be used on PMC1. This bit is reserved for OFFCORE_RESP_0 event.
		15	ANY_REQUEST	Account for any requests
Response Type	Any	16	ANY_RESPONSE	Account for any response
	Sub 1 (Data Supply from Untile)	17	NO_SUPP	No Supplier Details
		18	Reserved	Reserved
		19	L2_HIT_OTHER_TILE_NEAR	Other tile L2 hit E Near
		20	Reserved	Reserved
		21	MCDRAM_NEAR	MCDRAM Local
		22	MCDRAM_FAR_OR_L2_HIT_OTHER_TILE_FAR	MCDRAM Far or Other tile L2 hit far
		23	DRAM_NEAR	DRAM Local
		24	DRAM_FAR	DRAM Far
		25	L2_HITM_THIS_TILE	M-state

Main	Sub field	Bit	Name	Description
	Sub 2 (Data Supply from within same tile)	26	L2_HITE_THIS_TILE	E-state
		27	L2_HITS_THIS_TILE	S-state
		28	L2_HITF_THIS_TILE	F-state
		29	Reserved	Reserved
		30	Reserved	Reserved
	Sub 3 (Snoop Info) Only Valid in case of Data Supply from Untile	31	SNOOP_NONE	None of the cores were snooped
		32	NO_SNOOP_NEEDED	No snoop was needed to satisfy the request
		33	Reserved	Reserved
		34	Reserved	Reserved
		35	HIT_OTHER_TILE_FWD	Snoop request hit in the other tile with data forwarded
		36	HITM_OTHER_TILE	A snoop was needed and it HitM-ed in other core's L1 cache. HitM denotes a cache-line was in modified state before effect as a result of snoop.
		37	NON_DRAM	Target was non-DRAM system address. This includes MMIO transactions
Outstanding requests	Weighted cycles	38	OUTSTANDING (Valid only for MSR_OFFCORE_RESP0. Should only be used on PMC0. This bit is reserved for MSR_OFFCORE_RESP1).	If set, counts total number of weighted cycles of any outstanding offcore requests with data response. Valid only for OFFCORE_RESP_0 event. Should only be used on PMC0. This bit is reserved for OFFCORE_RESP_1 event

At the high level, in the Processor an offcore_response event is defined as combination of request type (e.g., read intention or write intention) and response type (where the data came from, what was the other tile's snoop response, etc.). For example, it can be used to count the number of cacheable data read L2 misses whose data came from the DDR4 memory.

Setting the average latency bit [38] changes counting behavior from event occurrences to weighted cycles. Note that, unlike event occurrence counting, response type is not used in weighted cycles counting.

For request types with valid data response (code reads, data reads and RFOs, prefetch reads and RFOs), select either bit [16] for any response or select bits from bits [17] to [28] for more detailed, selective data response break down.

Responses in the sub group 1 are from the untile, so they come with a snoop response. This means that if any of bits [17] to [24] are set, it requires the sub group 3 (bits [31] to [37]) to be programmed too. Setting all of the bits [31:37] means any snoop response. For example, programming for {request: demand cacheable data read request from L1 cache, data supply: other tile L2 hit, snoop: Hit/HitM forward} gives the number of cacheable data read L2 cache misses whose data were forwarded from other tile. Only Hit and HitM are valid snoop info for other tile data supply types.

Responses in the sub group 2 are from the L2 cache in the same tile, so they do not require specifying snoop response. For example, programming for {request: demand cacheable data read request from L1 cache, data supply: same tile L2 hit E/S/F, snoop: no bit set} gives the number of cacheable data read L2 cache non-modified hits (which do not cause snoop to the other core in the same tile).

Note that for MCDRAM and DDR4 data response types, their near/far response is always as if the processor is operating in the SNC4 (Sub-NUMA Clustering in four quadrants) mode. This means that in the all-to-all mode, both near ([21] or [23]) and far bits ([22] or [24]) should be set to correctly count MCDRAM or DDR4 data responses. For other tile responses, operating mode of the processor is correctly accounted for.

Note that for MCDRAM and DDR4 data response types, only possible snoop responses are snoop none and snoop not needed. More specifically, MCDRAM/DDR4 near ([21] or [23]) can only have snoop none ([31]) while MCDRAM/DDR4 far ([22] or [24]) can only have snoop not needed ([32]).

The following are some of the frequently used cacheable read/write combinations:

- Demand cacheable data read L2 miss satisfied by DDR4: {demand cacheable data read request from L1 cache, data supply: DDR4 near/far, snoop: none/not needed} = bits [0], [23], [24], [31], [32]
- Demand cacheable data write L2 miss satisfied by DDR4: {demand cacheable data RFO request from L1 cache, data supply: DDR4 near/far, snoop: none/not needed} = bits [1], [23], [24], [31], [32]
- Demand cacheable data read L2 miss satisfied by MCDRAM: {demand cacheable data read request from L1 cache, data supply: MCDRAM near/far, snoop: none/not needed} = bits [0], [21], [22], [31], [32]
- Demand cacheable data write L2 miss satisfied by MCDRAM: {demand cacheable data RFO request from L1 cache, data supply: MCDRAM near/far, snoop: none/not needed} = bits [1], [21], [22], [31], [32]
- Demand cacheable data read L2 miss satisfied by other tile: {demand cacheable data read request from L1 cache, data supply: other tile near/far, snoop: Hit/HitM forward} = bits [0], [19], [22], [35], [36]
- Demand cacheable data write L2 miss satisfied by other tile: {demand cacheable data RFO request from L1 cache, data supply: other tile near/far, snoop: Hit/HitM forward} = bits [1], [19], [22], [35], [36]

Some request types do not have a valid data response, for example, full or partial streaming stores. For such types, the response type should be set to any (bit [16]).

Note that the demand cacheable data read request from L1 cache (bit [0]) includes software prefetches (bit [12]) and DCU hardware prefetches (bit [13]). To count only the true demand cacheable reads, software and DCU hardware prefetch counts (counted with bits [12] and [13] set) should be subtracted from the total demand cacheable reads (counted with bit [0] set).

1.3 Performance Monitoring Events in the Intel® Xeon Phi™ Processor Untile

At the high level, the Intel® Xeon Phi™ processor untile performance monitoring facilities are in line with the Intel® Xeon® processor-based server class uncore performance monitoring facilities. For more information on the Intel® Xeon® processor uncore performance monitoring events, refer to the *Intel® Xeon® Processor E5 and E7 v3 Family Uncore Performance Monitoring Reference Manual*.

Notable differences include the following:

- The processor does not have L3 caches so there are no L3 cache-related events.
- The processor is single-socket only so there are no inter-socket events (other than external I/O connection).
- The processor has a completely different set of performance monitoring events in EDC (for on-package MCDRAM memory) and MC (for off-package DDR4 memory)
- Intra-tile interconnect in the processor is a two-dimensional mesh network. Common Mesh Stop (CMS) events are co-located with each untile component's events. For example, CMS events in a given CHA units are counted using the CHA's performance monitoring control and data registers. In other words, the CMS events [Event Select above 0x80] should be considered as part of each untile components. For example, CHA events include CMS events.

For a full description of Intel® Xeon Phi™ processor untile performance monitoring events, refer to Chapters 3 to 8.

1.3.1 Acronyms Frequently Used in the Intel® Xeon Phi™ Processor

Term	Description
The mesh network:	
AD ring	Address ring. Tile read/write requests and memory controller snoops to the CHA.
AK ring	Acknowledge ring. Acknowledges from memory controller to CHA and CHA to tile. Carries snoop responses from core to CHA.
BL ring	Block or data ring. Data transfers (two transfers for one cache line).
IV ring	Invalidate ring. CHA snoop requests of tile caches.

Term	Description
UPI	Ultra Path Interconnect protocol. Used for inter- and intra-socket coherent communication and supersedes the previous standard called QPI (QuickPath Interconnect).
CHA internal queues:	
IPQ	Ingress probe queue on AD ring. Associated with snoops from memory controller.
IRQ	Ingress request queue on AD ring. Associated with requests from tile.
ISMQ	Ingress subsequent message response queue. Associated with message responses to ingress requests (e.g., data responses, Intel UPI complete messages, core snoop response messages and GO reset queue).
RxR	Receive from ring (also known as ingress, IGR). Referring to ingress (requests from tile) queues.
TOR	Table of requests. Tracks pending CHA transactions.
Transgress	Requests 'passed along' or forwarded through a unit toward the destination.
TxR	Transmit to ring (also known as egress, EGR). Referring to egress (requests headed for the ring) queues.
CMS agents:	
AG0	Agent 0. Referring to CHA tile agent in a given tile.
AG1	Agent 1. Referring to core tile agent in a given tile. Not used in EDC, MC, UBOX units.
VN0	VN credit used to avoid protocol deadlocks between for requests to/from DRAM, MMIO, MMCFG, etc.
NCB	Non-coherent bypass message class used primarily for non-coherent messages that require data in their message payload.
NCS	Non-coherent standard message class used primarily for non-coherent messages that do not require data in their message payload.
Untile clocks:	
DCLK	DDR4 clock.
ECLK	MCDRAM clock.
UCLK	Untile clock.

Term	Description
Memory controller:	
CAS	Column Access Strobe. Access/selection of an address in DRAM.
Tile:	
L2Q	Level 2 Queue. Queue of requests going into second level cache.

1.3.2 CHA Filter Registers

Some of the CHA events require programming filter registers properly.

For information on the CHA filter registers, refer to *Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume1: Registers* document, PERF_UNIT_CTL_CHA_n (filter 0 for CHA n) and PERF_UNIT_CTL_1_CHA_n (filter 1 for CHA n) definitions.

Table 1-2. CHA Filter Registers

Register	Bits	Field
PERF_UNIT_CTL_CHA_n (CHA Filter 0 in CHA n)	[31:27]	Reserved
	[26:21]	Reserved
	[20:18]	Snoop filter state 20: SF_M 19: SF_E 18: SF_S
	[17]	Reserved
	[16:13]	Reserved
	[12]	LinkID3
	[11:9]	Reserved
	[8:3]	TileID
	[2:0]	ThreadID
PERF_UNIT_CTL_CHA_n (CHA Filter 1 in CHA n)	[31]	ISOC Op
	[30]	non-coherent Op
	[29]	Reserved
	[28:19]	Opcode1[9:0]
	[18:9]	Opcode0[9:0]
	[8:6]	Reserved
	[5]	Should be set to 1

Register	Bits	Field
	[4]	Reserved
	[3]	All Opcodes
	[2]	Reserved
	[1]	Local Node Request, Should be set to 1
	[0]	Remote Node, Should be set to 1

1.4 Reference Documents

Refer to the following documents for more in depth information.

1. Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual Volume 1: Registers
2. *Intel® Xeon® Processor E5 and E7 v3 Family Uncore Performance Monitoring Reference Manual*
3. *Intel® 64 and IA-32 Architectures Software Developer Manuals*
4. Sodani, A, "Knights Landing: 2nd generation Intel® Xeon Phi™ Product", *Hot Chips: A Symposium on High Performance Chips*, 2015, http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.70-Processors-Epub/HC27.25.710-Knights-Landing-Sodani-Intel.pdf
5. Sodani, A, et.al., "Knights Landing: 2nd generation Intel® Xeon Phi™ Product", *Hot Chips Special Issue of IEEE Micro Magazine*, March/April 2016.

1.5 Terminology

For ease-of-use, the table below contains some of the important terms used in this specification. Please refer to the *Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume 1: Registers* for an in depth look at the Intel® Xeon Phi™ processor architecture and components.

Table 1-3. Terminology

Term	Description
Intel® AVX	Intel® Advanced Vector Extensions (Intel® AVX)
Intel® AVX2	Intel® Advanced Vector Extensions 2 (Intel® AVX2)
Intel® AVX-512	Intel® Advanced Vector Extensions 512-bit wide vector extensions to the 256-bit Advanced Vector Extensions SIMD instructions for x86 instruction set architecture (ISA).
CHA	The functional agent that provides memory coherency between tiles.
DDR4	Fourth generation Double Data Rate synchronous dynamic random access memory (SDRAM) technology.
Integrated Memory Controller (IMC)	A memory controller that is integrated in the processor silicon.
IIO	Integrated I/O Controller in the processor die providing I/O components.
IRP	IIO to ring port. Converts PCIe command to IDI and IDI command to PCIe.
M2PCIe	Mesh-to-PCIe box. Connects IIO to mesh.
MCDRAM	Multi Channel DRAM. A high band width on package DRAM memory.
PMI	Performance monitoring interrupt
Tile	A unit in the processor that contains two cores sharing a second level cache.
UBOX	Untile box that contains global perfmon configuration and control capabilities.
Uncore/Untile	The portion of the processor comprising the IMC, IIO and related components.
Intel® SSE	Intel® Streaming SIMD Extensions

§

2 Tile Performance Monitoring Events

CPU_CLK_UNHALTED

- **Event Select**: N/A
- **Event Description**: Counts the number of unhalted reference clock cycles

Table 2-1. Unit Masks for CPU_CLK_UNHALTED

Extension	UMask	Description
THREAD	fixed ctr1	Fixed Counter: Counts the number of unhalted core clock cycles.
REF_TSC	fixed ctr2	Fixed Counter: Counts the number of unhalted reference clock cycles.

INST_RETIRED

- **Event Select**: N/A
- **Event Description**: Counts the total number of instructions retired

Table 2-2. Unit Masks for INST_RETIRED

Extension	UMask	Description
ANY	fixed ctr0	Fixed Counter: Counts the total number of instructions retired.

RECYCLEQ

- **Event Select**: 3h
- **Event Description**: Counts the number of retired load or store micro-ops that get pushed into the Recycle Queue. Prefetches will not be counted.

Table 2-3. Unit Masks for RECYCLEQ

Extension	UMask	Description
LD_BLOCK_ST_FORWARD	0x01	Counts the number of occurrences a retired load gets blocked because its address partially overlaps with a store.
LD_BLOCK_STD_NOTREADY	0x02	Counts the number of occurrences a retired load gets blocked because its address overlaps with a store whose data is not ready.
ST_SPLITS	0x04	Counts the number of occurrences a retired store that is a cache line split. Each split should be counted only once.
LD_SPLITS	0x08	Counts the number of occurrences a retired load that is a cache line split. Each split should be counted only once.
LOCK	0x10	Counts all the retired locked loads. It does not include stores because we would double count if we count stores.
STA_FULL	0x20	Counts the store micro-ops retired that were pushed in the recycle queue because the store address buffer is full.
ANY_LD	0x40	Counts any retired load that was pushed into the recycle queue for any reason.
ANY_ST	0x80	Counts any retired store that was pushed into the recycle queue for any reason.

MEM_UOPS_RETIRED

- **Event Select**: 4h
- **Event Description**: Counts the number of memory micro-ops retired. Prefetches will not be counted.

Table 2-4. Unit Masks for MEM_UOPS_RETIRED

Extension	UMask	Description
L1_MISS_LOADS	0x01	Counts the number of load micro-ops retired that miss in L1 D cache, HW prefetch misses will not be counted.
L2_HIT_LOADS	0x02	Counts the number of load micro-ops retired that hit in the L2.
L2_MISS_LOADS	0x04	Counts the number of load micro-ops retired that miss in the L2.
DTLB_MISS_LOADS	0x08	Counts the number of load micro-ops retired that cause a DTLB miss.
UTLB_MISS_LOADS	0x10	Counts the number of load micro-ops retired that caused micro TLB miss.
HITM	0x20	Counts the loads retired that get the data from the other core in the same tile in M state.
ALL_LOADS	0x40	Counts all the load micro-ops retired.
ALL_STORES	0x80	Counts all the store micro-ops retired.

PAGE_WALKS

- **Event Select**: 5h
- **Event Description**: Counts the number of core cycles for page walks

Table 2-5. Unit Masks for PAGE_WALKS

Extension	UMask	Description
D_SIDE_WALKS	0x01	Counts the total D-side page walks that are completed or started. The page walks started in the speculative path will also be counted.
D_SIDE_CYCLES	0x01	Counts the total number of core cycles for all the D-side page walks. The cycles for page walks started in speculative path will also be included.
I_SIDE_WALKS	0x02	Counts the total I-side page walks that are completed.
I_SIDE_CYCLES	0x02	Counts the total number of core cycles for all the I-side page walks. The cycles for page walks started in speculative path will also be included.
WALKS	0x03	Counts the total page walks that are completed (I-side and D-side).
CYCLES	0x03	Counts the total number of core cycles for all the page walks. The cycles for page walks started in speculative path will also be included.

L2_REQUESTS

- **Event Select**: 2Eh
- **Event Description**: Counts the number of L2 cache requests

Table 2-6. Unit Masks for L2_REQUESTS

Extension	UMask	Description
REFERENCE	0x4F	Counts the total number of L2 cache references.
MISS	0x41	Counts the number of L2 cache misses.

L2_REQUESTS_REJECT

- **Event Select**: 30h
- **Event Description**: Counts the number of MEC requests from the L2Q that reference a cache line (cacheable requests) exlcuding SW prefetches filling only to L2 cache and L1 evictions (automatically exlcudes L2HWP, UC, WC) that were rejected. Multiple repeated rejects should be counted multiple times.

Table 2-7. Unit Masks for L2_REQUESTS_REJECT

Extension	UMask	Description
ALL	0x00	Counts the number of MEC requests from the L2Q that reference a cache line (cacheable requests) exlcuding SW prefetches filling only to L2 cache and L1 evictions (automatically exlcudes L2HWP, UC, WC) that were rejected - Multiple repeated rejects should be counted multiple times.

CORE_REJECT_L2Q

- **Event Select**: 31h
- **Event Description**: Counts the number of L1 requests that were not accepted into the L2Q because of any L2 queue reject condition

Table 2-8. Unit Masks for CORE_REJECT_L2Q

Extension	UMask	Description
ALL	0x00	Counts the number of L1 requests that were not accepted into the L2Q because of any L2 queue reject condition. There is no concept of at-ret here. It might include requests due to instructions in the speculative path.

CPU_CLK_UNHALTED

- **Event Select**: 3Ch
- **Event Description**: Counts the number of unhalted clock cycles

Table 2-9. Unit Masks for CPU_CLK_UNHALTED

Extension	UMask	Description
THREAD_P	0x00	Counts the number of unhalted core clock cycles.
REF	0x01	Counts the number of unhalted reference clock cycles.

ICACHE

- **Event Select**: 80h
- **Event Description**: None

Table 2-10. Unit Masks for ICACHE

Extension	UMask	Description
ACCESSES	0x03	Counts all instruction fetches, including uncacheable fetches.

Extension	UMask	Description
HIT	0x01	Counts all instruction fetches that hit the instruction cache.
MISSES	0x02	Counts all instruction fetches that miss the Instruction cache or produce memory requests. An instruction fetch miss is counted only once and not once for every cycle it is outstanding.

FETCH_STALL

- **Event Select**: 86h
- **Event Description**: Counts the number of core cycles the instruction fetch pipe was stalls

Table 2-11. Unit Masks for FETCH_STALL

Extension	UMask	Description
ICACHE_FILL_PENDING_CYCLES	0x04	Counts the number of core cycles the fetch stalls because of an icache miss. This is a cumulative count of core cycles the fetch stalled for all icache misses.

OFFCORE_RESP

- **Event Select**: B7h
- **Event Description**: Counts the transactions of request type and response type specified in MSR_OFFCORE_RESPx

Table 2-12. Unit Masks for OFFCORE_RESP

Extension	UMask	Description
OFFCORE_RESP_0	0x01	Counts the matrix events specified by MSR_OFFCORE_RESP0.
OFFCORE_RESP_1	0x02	Counts the matrix events specified by MSR_OFFCORE_RESP1.

INST_RETIRED

- **Event Select**: C0h
- **Event Description**: Counts the number of instructions retired

Table 2-13. Unit Masks for INST_RETIRED

Extension	UMask	Description
ANY_P	0x00	Counts the total number of instructions retired.

UOPS_RETIRED

- **Event Select**: C2h
- **Event Description**: Counts the number of micro-ops retired

Table 2-14. Unit Masks for UOPS_RETIRED

Extension	UMask	Description
MS	0x01	Counts the number of micro-ops retired that are from the complex flows issued by the micro-sequencer (MS).
ALL	0x10	Counts the number of micro-ops retired.
SCALAR_SIMD	0x20	Counts the number of scalar SSE, AVX, AVX2, AVX-512 micro-ops retired. More specifically, it counts scalar SSE, AVX, AVX2, AVX-512 micro-ops except for loads (memory-to-register mov-type micro ops), division, sqrt.
PACKED_SIMD	0x40	Counts the number of vector SSE, AVX, AVX2, AVX-512 micro-ops retired. More specifically, it counts packed SSE, AVX, AVX2, AVX-512 micro-ops (both floating point and integer) except for loads (memory-to-register mov-type micro-ops), packed byte and word multiplies.

MACHINE_CLEARS

- **Event Select**: C3h
- **Event Description**: Counts the number of machine clears at retire

Table 2-15. Unit Masks for MACHINE_CLEARS

Extension	UMask	Description
SMC	0x01	Counts the number of times that the machine clears due to program modifying data within 1K of a recently fetched code page.
MEMORY_ORDERING	0x02	Counts the number of times the machine clears due to memory ordering hazards.
FP_ASSIST	0x04	Counts the number of floating instructions retired that required microcode assists.
ALL	0x08	Counts all nukes.

BR_INST_RETIRED

- **Event Select**: C4h
- **Event Description**: Counts the number of branch instructions retired

Table 2-16. Unit Masks for BR_INST_RETIRED

Extension	UMask	Description
ALL_BRANCHES	0x00	Counts the number of branch instructions retired.
JCC	0x7E	Counts the number of branch instructions retired that were conditional jumps.
TAKEN_JCC	0xFE	Counts the number of branch instructions retired that were conditional jumps and predicted taken.
CALL	0xF9	Counts the number of near CALL branch instructions retired.
REL_CALL	0xFD	Counts the number of near relative CALL branch instructions retired.

Extension	UMask	Description
IND_CALL	0xFB	Counts the number of near indirect CALL branch instructions retired.
RETURN	0xF7	Counts the number of near RET branch instructions retired.
NON_RETURN_IND	0xEB	Counts the number of branch instructions retired that were near indirect CALL or near indirect JMP.
FAR_BRANCH	0xBF	Counts the number of far branch instructions retired.

BR_MISP_RETIRED

- **Event Select**: C5h
- **Event Description**: Counts the number of mispredicted branch instructions retired

Table 2-17. Unit Masks for BR_MISP_RETIRED

Extension	UMask	Description
ALL_BRANCHES	0x00	Counts the number of mispredicted branch instructions retired.
JCC	0x7E	Counts the number of mispredicted branch instructions retired that were conditional jumps.
TAKEN_JCC	0xFE	Counts the number of mispredicted branch instructions retired that were conditional jumps and predicted taken.
CALL	0xF9	Counts the number of mispredicted near CALL branch instructions retired.
REL_CALL	0xFD	Counts the number of mispredicted near relative CALL branch instructions retired.
IND_CALL	0xFB	Counts the number of mispredicted near indirect CALL branch instructions retired.

Extension	UMask	Description
RETURN	0xF7	Counts the number of mispredicted near RET branch instructions retired.
NON_RETURN_IND	0xEB	Counts the number of mispredicted branch instructions retired that were near indirect CALL or near indirect JMP.
FAR_BRANCH	0xBF	Counts the number of mispredicted far branch instructions retired.

NO_ALLOC_CYCLES

- **Event Select**: CAh
- **Event Description**: Counts the number of core cycles when no micro-ops are allocated

Table 2-18. Unit Masks for NO_ALLOC_CYCLES

Extension	UMask	Description
ROB_FULL	0x01	Counts the number of core cycles when no micro-ops are allocated and the ROB is full.
MISPREDICTS	0x04	Counts the number of core cycles when no micro-ops are allocated and the alloc pipe is stalled waiting for a mispredicted branch to retire.
RAT_STALL	0x20	Counts the number of core cycles when no micro-ops are allocated and a RATstall (caused by reservation station full) is asserted.
ALL	0x7F	Counts the total number of core cycles when no micro-ops are allocated for any reason.

RS_FULL_STALL

- **Event Select**: CBh
- **Event Description**: Counts the number of core cycles when the allocate stalls because the required RS is full

Table 2-19. Unit Masks for RS_FULL_STALL

Extension	UMask	Description
MEC	0x01	Counts the number of core cycles when allocation pipeline is stalled and is waiting for a free MEC reservation station entry.
ALL	0x1F	Counts the total number of core cycles the Alloc pipeline is stalled when any one of the reservation stations is full.

CYCLES_DIV_BUSY

- **Event Select**: CDh
- **Event Description**: Cycles the number of core cycles when divider is busy

Table 2-20. Unit Masks for CYCLES_DIV_BUSY

Extension	UMask	Description
ALL	0x01	Cycles the number of core cycles when divider is busy. Does not imply a stall waiting for the divider.

BACLEARS

- **Event Select**: E6h
- **Event Description**: Counts the number of times the Branch Target Buffer (BTB) prediction was corrected by a later branch predictor

Table 2-21. Unit Masks for BACLEARS

Extension	UMask	Description
ALL	0x01	Counts the number of times the front end resteers for any branch as a result of another branch handling mechanism in the front end.
RETURN	0x08	Counts the number of times the front end resteers for RET branches as a result of another branch handling mechanism in the front end.
COND	0x10	Counts the number of times the front end resteers for conditional branches as a result of another branch handling mechanism in the front end.

MS_DECODED

- **Event Select**: E7h
- **Event Description**: Microcode sequencer decode entrypoints

Table 2-22. Unit Masks for MS_DECODED

Extension	UMask	Description
MS_ENTRY	0x01	Counts the number of times the MSROM starts a flow of micro-ops.

§

3 Untile-EDC Performance Monitoring Events

UCLK Ticks

- **Category**: UCLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: UCLK count

EDC Hit / Miss

- **Category:** UCLK Events
- **Event Select**: 0x02
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Number of EDC Hits or Misses

Table 3-1. Unit Masks for EDC Hit / Miss

Extension	UMask	Description
HIT_CLEAN	bxxxxxxx1	Counts the number of read requests and streaming stores that hit in MCDRAM cache and the data in MCDRAM is clean with respect to DDR. This event is only valid in cache and hybrid memory mode.
HIT_DIRTY	bxxxxxx1x	Counts the number of read requests and streaming stores that hit in MCDRAM cache and the data in MCDRAM is dirty with respect to DDR. This event is only valid in cache and hybrid memory mode.

Extension	UMask	Description
MISS_CLEAN	bxxxxx1xx	Counts the number of read requests and streaming stores that miss in MCDRAM cache and the data evicted from the MCDRAM is clean with respect to DDR. This event is only valid in cache and hybrid memory mode.
MISS_DIRTY	bxxxx1xxx	Counts the number of read requests and streaming stores that miss in MCDRAM cache and the data evicted from the MCDRAM is dirty with respect to DDR. This event is only valid in cache and hybrid memory mode.
MISS_INVALID	bxxx1xxxx	Miss I

ECLK Ticks

- **Category:** ECLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: ECLK count

RPQ

- **Category:** ECLK Events
- **Event Select**: 0x01
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: RPQ

Table 3-2. Unit Masks for RPQ

Extension	UMask	Description
Inserts	b00000001	Counts the number of read requests received by the MCDRAM controller. This event is valid in all three memory modes: flat, cache and hybrid. In cache and hybrid memory mode, this event counts all read requests as well as streaming stores that hit or miss in the MCDRAM cache.

WPQ

- **Category:** ECLK Events
- **Event Select**: 0x02
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: WPQ

Table 3-3. Unit Masks for WPQ

Extension	UMask	Description
Inserts	b00000001	Counts the number of write requests received by the MCDRAM controller. This event is valid in all three memory modes: flat, cache and hybrid. In cache and hybrid memory mode, this event counts all streaming stores, writebacks and, read requests that miss in MCDRAM cache.

3.1 Computing MCDRAM Bandwidth in Cache Mode

The MCDRAM cache memory bandwidth can be computed using the following seven untile events:

- ECLK_Events_RPQ_Inserts
- ECLK_Events_WPQ_Inserts
- UCLK_Events_EDC_Hit/Miss_HIT_CLEAN
- UCLK_Events_EDC_Hit/Miss_HIT_DIRTY
- UCLK_Events_EDC_Hit/Miss_MISS_CLEAN

- UCLK_Events_EDC_Hit/Miss_MISS_DIRTY
- DCLK_Events_CAS_Reads

Using MCDRAM in cache memory mode, the UCLK_Events_EDC_Hit/Miss events only count loads and streaming stores from the core but do not count writebacks to the MCDRAM. On the other hand, ECLK_Events_RPQ_Inserts and ECLK_Events_WPQ_Inserts events not only count both, loads and stores including writebacks, from the cores to MCDRAM but also count data transfers between DDR and MCDRAM memory.

Due to the behavior of these two sets of events while using MCDRAM in cache memory mode, the MCDRAM cache mode bandwidth, that is, the rate of data transfer between the core and MCDRAM, is computed differently as opposed to using MCDRAM in the flat memory mode. The MCDRAM cache mode bandwidth can be computed as follows:

MCDRAM Cache read bandwidth
$$= (\text{ECLK\_Events\_RPQ\_Inserts} - \text{UCLK\_Events\_EDC\_Hit/Miss\_MISS\_CLEAN} - \text{UCLK\_Events\_EDC\_Hit/Miss\_MISS\_DIRTY}) * 64 / \text{Time}$$

MCDRAM Cache write bandwidth
$$= (\text{ECLK\_Events\_WPQ\_Inserts} - \text{DCLK\_Events\_CAS\_Reads}) * 64 / \text{Time}$$

MCDRAM Total bytes $= \text{MCDRAM Cache read bandwidth} + \text{MCDRAM Cache write bandwidth}$

Lastly, while using MDRAM in cache memory mode, some streaming store operations may be over counted and may result in higher MCDRAM cache read bandwidth.

§

4 Untile-MC Performance Monitoring Events

UCLK Ticks

- **Category:** UCLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: UCLK count

DCLK Ticks

- **Category:** DCLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: DCLK count

CAS

- **Category:** DCLK Events
- **Event Select**: 0x03
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CAS

Table 4-1. Unit Masks for CAS

Extension	UMask	Description
Reads	bxxxxxx01	Reads
Writes	bxxxxxx10	Writes
All	bxxxxxx11	All

§

5 Untile-CHA Performance Monitoring Events

CLOCKTICKS

- **Category:** UCLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Untile Clocks

RxR_OCCUPANCY

- **Category:** INGRESS Events
- **Event Select**: 0x11
- **Max Inc / cycle**: 8
- **Counter Availability**: 0
- **Event Description**: Ingress Occupancy. Counts number of entries in the specified Ingress queue in each cycle

Table 5-1. Unit Masks for RxR_OCCUPANCY

Extension	Umask	Description
IRQ	b00000001	IRQ
IRQ_REJ	b00000010	IRQ Rejected
IPQ	b00000100	IPQ
PRQ	b00010000	PRQ
PRQ_REJ	b00100000	PRQ Rejected

RxR_INSERTS

- **Category:** INGRESS Events
- **Event Select**: 0x13
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Allocations. Counts number of allocations per cycle into the specified Ingress queue

Table 5-2. Unit Masks for RxR_INSERTS

Extension	Umask	Description
IRQ	bxxxxxxx1	IRQ
IRQ_REJ	bxxxxxx1x	IRQ Rejected
IPQ	bxxxxx1xx	IPQ
PRQ	bxxx1xxxx	PRQ
PRQ_REJ	bxx1xxxxx	PRQ Rejected

RxR_INT_STARVED

- **Category:** INGRESS Events
- **Event Select**: 0x14
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Internal Starvation Cycles. Counts cycles in internal starvation. This occurs when one or more of the entries in the ingress queue are being starved out by other entries in that queue.

Table 5-3. Unit Masks for RxR_INT_STARVED

Extension	Umask	Description
IRQ	bxxxxxxx1	Cycles with the IRQ in Internal Starvation.
IPQ	bxxxxx1xx	Cycles with the IPQ in Internal Starvation.
ISMQ	bxxxx1xxx	Cycles with the ISMQ in Internal Starvation.

Extension	Umask	Description
PRQ	bxxx1xxxx	Cycles with the PRQ in Internal Starvation.

RxR_IRQ0_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x18
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Request Queue Rejects

Table 5-4. Unit Masks for RxR_IRQ0_REJECT

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Request reject due to no AD credit.
BL_NCS_VN0	bxx1xxxxx	Request reject due to no BL credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RxR_IRQ1_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x19
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Request Queue Rejects

Table 5-5. Unit Masks for RxR_IRQ1_REJECT

Extension	Umask	Description
ANY_REJECT_IRQ0	bxxxxxxx1	Any condition listed in the IRQ0 Reject counter was true.

Extension	Umask	Description
SF_VICTIM	bxxxx1xxx	Request did not generate Snoop filter victim.
SF_WAY	bxx1xxxxx	Way conflict with another request that caused the reject.
PA_MATCH	b1xxxxxxx	Address match with an outstanding request that was rejected.

RxR_PRQ0_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x20
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Posted Request Queue Rejects

Table 5-6. Unit Masks for RxR_PRQ0_REJECT

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Request reject due to no AD credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RxR_PRQ1_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x21
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Posted Request Queue Rejects

Table 5-7. Unit Masks for RxR_PRQ1_REJECT

Extension	Umask	Description
ANY_REJECT_IRQ0	bxxxxxxx1	Any condition listed in the IRQ0 Reject counter was true.
SF_VICTIM	bxxxx1xxx	Request did not generate Snoop filter victim.
SF_WAY	bxx1xxxxx	Way conflict with another request that caused the reject.
PA_MATCH	b1xxxxxxx	Address match with an outstanding request that was rejected.

RxR_IPQ0_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x22
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Probe Queue Rejects

Table 5-8. Unit Masks for RxR_IPQ0_REJECT

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Reject due to no VN0 credit for generating a request.
AD_RSP_VN0	bxxxxxx1x	Reject due to no VN0 credit for generating a response.
BL_RSP_VN0	bxxxxx1xx	Reject due to no BL VN0 credit for generating a response.
BL_WB_VN0	bxxxx1xxx	Reject due to no BL VN0 credit for generating a write back.
BL_NCB_VN0	bxxx1xxxx	Reject due to no BL NCB VN0 credit.
BL_NCS_VN0	bxx1xxxxx	Reject due to no BL NCS VN0 credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.

RxR_IPQ1_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x23
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Probe Queue Rejects

Table 5-9. Unit Masks for RxR_IPQ1_REJECT

Extension	Umask	Description
ANY_REJECT_IPQ0	bxxxxxxx1	Any condition listed in the IPQ1 Reject counter was true.
SF_VICTIM	bxxxx1xxx	Request did not generate Snoop filter victim.
SF_WAY	bxx1xxxxx	Way conflict with another request that caused the reject.
ALLOW_SNP	bx1xxxxxx	Reject due to inability to transmit a snoop.
PA_MATCH	b1xxxxxxx	Address match with an outstanding request that was rejected.

RxR_ISMQ0_REJECT

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x24
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: ISMQ Rejects

Table 5-10. Unit Masks for RxR_ISMQ0_REJECT

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Reject due to no VN0 credit for generating a request.
AD_RSP_VN0	bxxxxxx1x	Reject due to no VN0 credit for generating a response.

Extension	Umask	Description
BL_RSP_VN0	bxxxxx1xx	Reject due to no BL VN0 credit for generating a response.
BL_WB_VN0	bxxxx1xxx	Reject due to no BL VN0 credit for generating a write back.
BL_NCB_VN0	bxxx1xxxx	Reject due to no BL NCB VN0 credit.
BL_NCS_VN0	bxx1xxxxx	Reject due to no BL NCS VN0 credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RXR_REQ_Q0_RETRY

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x2A
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: "REQUESTQ" includes: IRQ, PRQ, IPQ, RRQ, WBQ (everything except for ISMQ)

Table 5-11. Unit Masks for RXR_REQ_Q0_RETRY

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Retry due to no VN0 credit for generating a request.
AD_RSP_VN0	bxxxxxx1x	Retry due to no VN0 credit for generating a response.
BL_NCS_VN0	bxx1xxxxx	Retry due to no BL NCS VN0 credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RXR_REQ_Q1_RETRY

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x2B
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: "REQUESTQ" includes: IRQ, PRQ, IPQ, RRQ, WBQ (everything except for ISMQ)

Table 5-12. Unit Masks for RXR_REQ_Q1_RETRY

Extension	Umask	Description
ANY_REJECT_IRQ0	bxxxxxxx1	Any condition listed in the IRQ0 Reject counter was true.
SF_VICTIM	bxxxx1xxx	Request did not generate Snoop filter victim.
SF_WAY	bxx1xxxxx	Way conflict with another request that caused the reject.
ALLOW_SNP	bx1xxxxxx	Retry due to inability to transmit a snoop.
PA_MATCH	b1xxxxxxx	Address match with an outstanding request that was rejected.

RXR_ISMQ0_RETRY

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x2C
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: ISMQ Retries

Table 5-13. Unit Masks for RXR_ISMQ0_RETRY

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Retry due to no VN0 credit for generating a request.

Extension	Umask	Description
AD_RSP_VN0	bxxxxxx1x	Retry due to no VN0 credit for generating a response.
BL_RSP_VN0	bxxxxx1xx	Retry due to no BL VN0 credit for generating a response.
BL_WB_VN0	bxxxx1xxx	Retry due to no BL VN0 credit for generating a write back.
BL_NCB_VN0	bxxx1xxxx	Retry due to no BL NCB VN0 credit.
BL_NCS_VN0	bxx1xxxxx	Retry due to no BL NCS VN0 credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RXR_OTHER0_RETRY

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x2E
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Other Queue Retries

Table 5-14. Unit Masks for RXR_OTHER0_RETRY

Extension	Umask	Description
AD_REQ_VN0	bxxxxxxx1	Retry due to no VN0 credit for generating a request.
AD_RSP_VN0	bxxxxxx1x	Retry due to no VN0 credit for generating a response.
BL_RSP_VN0	bxxxxx1xx	Retry due to no BL VN0 credit for generating a response.
BL_WB_VN0	bxxxx1xxx	Retry due to no BL VN0 credit for generating a write back.
BL_NCB_VN0	bxxx1xxxx	Retry due to no BL NCB VN0 credit.

Extension	Umask	Description
BL_NCS_VN0	bxx1xxxxx	Retry due to no BL NCS VN0 credit.
AK_NON_UPI	bx1xxxxxx	Reject due to inability to inject an AK ring message.
IV_NON_UPI	b1xxxxxxx	Reject due to inability to inject an IV ring message.

RXR_OTHER1_RETRY

- **Category:** INGRESS_RETRY Events
- **Event Select**: 0x2F
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Other Queue Retries

Table 5-15. Unit Masks for RXR_OTHER1_RETRY

Extension	Umask	Description
ANY_REJECT_IRQ0	bxxxxxxx1	Any condition listed in the IRQ0 Reject counter was true.
SF_VICTIM	bxxxx1xxx	Request did not generate Snoop filter victim.
SF_WAY	bxx1xxxxx	Way conflict with another request that caused the reject.
ALLOW_SNP	bx1xxxxxx	Retry due to inability to transmit a snoop.
PA_MATCH	b1xxxxxxx	Address match with an outstanding request that was rejected.

SF_LOOKUP

- **Category:** CACHE Events
- **Event Select**: 0x34
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Snoop Filter Lookups. Counts the number of times the SF was accessed including code, data, prefetches and hints coming from L2. This has numerous filters available. Note the non-standard filtering equation. This event will count requests that lookup the cache multiple times with multiple increments. One must ALWAYS set umask bit 0 and select a state or states to match. Otherwise, the event will count nothing. CBoGlCtrl[22:18] bits correspond to [FMESI] state.

Table 5-16. Unit Masks for SF_LOOKUP

Extension	Umask	Description
DATA_READ	b00000011	Data Read Request
WRITE	b00000101	Write Requests
REMOTE_SNOOP	b00001001	External Snoop Request
ANY	b00010001	Any Request

COUNTER0_OCCUPANCY

- **Category:** OCCUPANCY Events
- **Event Select**: 0x1F
- **Max Inc / cycle**: 32
- **Counter Availability**: 0
- **Event Description**: Counter 0 Occupancy

TOR_INSERTS

- **Category:** TOR Events
- **Event Select**: 0x35
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: TOR Inserts. Counts the number of entries successfully inserted into the TOR that match qualifications specified by the subevent. There are a number of subevent 'filters' but only a subset of the subevent combinations are valid. Subevents that require an opcode or NID match require the Cn_MSR_PMON_BOX_FILTER.{opc, nid} field to be set. If, for example, one wanted to count DRD Local Misses, one should select "MISS_OPC_MATCH" and set Cn_MSR_PMON_BOX_FILTER.opc to DRD (0x182).

Table 5-17. Unit Masks for TOR_INSERTS

Extension	Umask	Description
IRQ	b00110001 Filter Dependency: Filter 1	IRQ or PRQ
IRQ_Hit	b00010001 Filter Dependency: Filter 1	IRQ or PRQ Hit
IRQ_MISS	b00100001 Filter Dependency: Filter 1	IRQ or PRQ Miss
EVICT	b00110010 Filter Dependency: Filter 1	SF Evictions
PRQ	b00110100 Filter Dependency: Filter 1	PRQ
PRQ_HIT	b00010100 Filter Dependency: Filter 1	PRQ Hit
PRQ_MISS	b00100100 Filter Dependency: Filter 1	PRQ Miss
IPQ	b00111000 Filter Dependency: Filter 1	IPQ
IPQ_HIT	b00011000 Filter Dependency: Filter 1	IPQ Hit
IPQ_MISS	b00101000 Filter Dependency: Filter 1	IPQ Miss
HIT	b00011101 Filter Dependency: Filter 1	Hit (Not a Miss)

Extension	Umask	Description
MISS	b00101101 Filter Dependency: Filter 1	Miss

TOR_OCCUPANCY

- **Category:** TOR Events
- **Event Select**: 0x36
- **Max Inc / cycle**: 28
- **Counter Availability**: 0
- **Event Description**: TOR Occupancy. For each cycle, this event accumulates the number of valid entries in the TOR that match qualifications specified by the subevent. There are a number of subevent 'filters' but only a subset of the subevent combinations are valid. Subevents that require an opcode or NID match require the Cn_MSR_PMON_BOX_FILTER.{opc, nid} field to be set. If, for example, one wanted to count DRD Local Misses, one should select "MISS_OPC_MATCH" and set Cn_MSR_PMON_BOX_FILTER.opc to DRD (0x182).

Table 5-18. Unit Masks for TOR_OCCUPANCY

Extension	Umask	Description
IRQ	b00110001 Filter Dependency: Filter 1	IRQ or PRQ
IRQ_HIT	b00010001 Filter Dependency: Filter 1	IRQ or PRQ Hit
IRQ_MISS	b00100001 Filter Dependency: Filter 1	IRQ or PRQ Miss
EVICT	b00110010 Filter Dependency: Filter 1	SF Evictions
PRQ	b00110100 Filter Dependency: Filter 1	PRQ
PRQ_HIT	b00010100 Filter Dependency: Filter 1	PRQ Hit
PRQ_MISS	b00100100 Filter Dependency: Filter 1	PRQ Miss

Extension	Umask	Description
IPQ	b00111000 Filter Dependency: Filter 1	IPQ
IPQ_HIT	b00011000 Filter Dependency: Filter 1	IPQ Hit
IPQ_MISS	b00101000 Filter Dependency: Filter 1	IPQ Miss
HIT	b00011101 Filter Dependency: Filter 1	Hit
MISS	b00101101 Filter Dependency: Filter 1	Miss

MISC

- **Category:** MISC Events
- **Event Select**: 0x39
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Cbo Misc. Miscellaneous events in the Cbo

Table 5-19. Unit Masks for MISC

Extension	UMask	Description
RSPI_WAS_FSE	bxxxxxxx1	Silent Snoop Eviction
WC_ALIASING	bxxxxxx1x	Write Combining Aliasing
RFO_HIT_S	bxxxx1xxx	RFO HitS
CV0_PREF_VIC	bxxx1xxxx	CV0 Prefetch Victim
CV0_PREF_MISS	bxx1xxxxx	CV0 Prefetch Miss

§

6 Untile-CMS Performance Monitoring Events

Common Mesh Stop (CMS) events are co-located with each untile component's events. For example, CMS events in a given CHA units are counted using the CHA's performance monitoring control and data registers. In other words, the CMS events [Event Select above 0x80] should be considered as part of each untile components. For example, CHA events include CMS events.

Please note that the CMS events described in this chapter have been validated only in the CHA.

CLOCKTICKS

- **Category:** UCLK Events
- **Event Select**: 0xC0
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Untile Clocks

AG0_AD_CRD_ACQUIRED

- **Category:** Transgress Credit Events
- **Event Select**: 0x80
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 AD Credits Acquired

Table 6-1. Unit Masks for AG0_AD_CRD_ACQUIRED

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3

Extension	Umask	Description
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_AD_CRD_ACQUIRED_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x81
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 AD Credits Acquired

Table 6-2. Unit Masks for AG0_AD_CRD_ACQUIRED_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG0_AD_CRD_OCCUPANCY

- **Category:** Transgress Credit Events
- **Event Select**: 0x82
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 AD Credits Occupancy

Table 6-3. Unit Masks for AG0_AD_CRD_OCCUPANCY

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4

Extension	Umask	Description
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_AD_CRD_OCCUPANCY_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x83
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 AD Credits Occupancy

Table 6-4. Unit Masks for AG0_AD_CRD_OCCUPANCY_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_AD_CRD_ACQUIRED

- **Category:** Transgress Credit Events
- **Event Select**: 0x84
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 AD Credits Acquired

Table 6-5. Unit Masks for AG1_AD_CRD_ACQUIRED

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5

Extension	Umask	Description
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG1_AD_CRD_ACQUIRED_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x85
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 AD Credits Acquired

Table 6-6. Unit Masks for AG1_AD_CRD_ACQUIRED_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_AD_CRD_OCCUPANCY

- **Category:** Transgress Credit Events
- **Event Select**: 0x86
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 AD Credits Occupancy

Table 6-7. Unit Masks for AG1_AD_CRD_OCCUPANCY

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6

Extension	Umask	Description
TGR7	b1xxxxxxx	For Transgress 7

AG1_AD_CRD_OCCUPANCY_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x87
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 AD Credits Occupancy

Table 6-8. Unit Masks for AG1_AD_CRD_OCCUPANCY_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG0_BL_CRD_ACQUIRED

- **Category:** Transgress Credit Events
- **Event Select**: 0x88
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 BL Credits Acquired

Table 6-9. Unit Masks for AG0_BL_CRD_ACQUIRED

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_BL_CRD_ACQUIRED_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x89
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 BL Credits Acquired

Table 6-10. Unit Masks for AG0_BL_CRD_ACQUIRED_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG0_BL_CRD_OCCUPANCY

- **Category:** Transgress Credit Events
- **Event Select**: 0x8A
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 BL Credits Occupancy

Table 6-11. Unit Masks for AG0_BL_CRD_OCCUPANCY

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_BL_CRD_OCCUPANCY_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x8B
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent0 BL Credits Occupancy

Table 6-12. Unit Masks for AG0_BL_CRD_OCCUPANCY_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_BL_CRD_ACQUIRED

- **Category:** Transgress Credit Events
- **Event Select**: 0x8C
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 BL Credits Acquired

Table 6-13. Unit Masks for AG1_BL_CRD_ACQUIRED

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG1_BL_CRD_ACQUIRED_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x8D
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 BL Credits Acquired

Table 6-14. Unit Masks for AG1_BL_CRD_ACQUIRED_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_BL_CRD_OCCUPANCY

- **Category:** Transgress Credit Events
- **Event Select**: 0x8E
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 BL Credits Occupancy

Table 6-15. Unit Masks for AG1_BL_CRD_OCCUPANCY

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG1_BL_CRD_OCCUPANCY_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0x8F
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Agent1 BL Credits Occupancy

Table 6-16. Unit Masks for AG1_BL_CRD_OCCUPANCY_EXT

Extension	Umask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG0_STALL_NO_CRD_EGRESS_HORZ_AD

- **Category:** Transgress Credit Events
- **Event Select**: 0xD0
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No AD Transgress Credits. Number of cycles the AD Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-17. Unit Masks for AG0_STALL_NO_CRD_EGRESS_HORZ_AD

Extension	Umask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_STALL_NO_CRD_EGRESS_HORZ_AD_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0xD1
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No AD Transgress Credits. Number of cycles the AD Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-18. Unit Masks for AG0_STALL_NO_CRD_EGRESS_HORZ_AD_EXT

Extension	UMask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_STALL_NO_CRD_EGRESS_HORZ_AD

- **Category:** Transgress Credit Events
- **Event Select**: 0xD2
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No AD Transgress Credits. Number of cycles the AD Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-19. Unit Masks for AG1_STALL_NO_CRD_EGRESS_HORZ_AD

Extension	UMask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG1_STALL_NO_CRD_EGRESS_HORZ_AD_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0xD3
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No AD Transgress Credits. Number of cycles the AD Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-20. Unit Masks for AG1_STALL_NO_CRD_EGRESS_HORZ_AD_EXT

Extension	UMask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG0_STALL_NO_CRD_EGRESS_HORZ_BL

- **Category:** Transgress Credit Events
- **Event Select**: 0xD4
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No BL Transgress Credits. Number of cycles the BL Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-21. Unit Masks for AG0_STALL_NO_CRD_EGRESS_HORZ_BL

Extension	UMask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG0_STALL_NO_CRD_EGRESS_HORZ_BL_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0xD5
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No BL Transgress Credits. Number of cycles the BL Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-22. Unit Masks for AG0_STALL_NO_CRD_EGRESS_HORZ_BL_EXT

Extension	UMask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

AG1_STALL_NO_CRD_EGRESS_HORZ_BL

- **Category:** Transgress Credit Events
- **Event Select**: 0xD6
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No BL Transgress Credits. Number of cycles the BL Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-23. Unit Masks for AG1_STALL_NO_CRD_EGRESS_HORZ_BL

Extension	UMask	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5
TGR6	bx1xxxxxx	For Transgress 6
TGR7	b1xxxxxxx	For Transgress 7

AG1_STALL_NO_CRD_EGRESS_HORZ_BL_EXT

- **Category:** Transgress Credit Events
- **Event Select**: 0xD7
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1,2,3
- **Event Description**: Stall on No BL Transgress Credits. Number of cycles the BL Transgress Buffer is stalled waiting for a credit to become available, per transgress

Table 6-24. Unit Masks for AG1_STALL_NO_CRD_EGRESS_HORZ_BL_EXT

Extension	UMask	Description
TGR8	bxxxxxxx1	For Transgress 8
ANY_OF_TGR0_THRU_TGR7	bxxxxxx1x	For Transgress 0-7

EGRESS_VERT_OCCUPANCY

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x90
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vert Egress Occupancy. Occupancy event for the Egress buffers in the Common Mesh Stop. The egress is used to queue up requests destined for the Vertical Ring on the Mesh.

Table 6-25. Unit Masks for EGRESS_VERT_OCCUPANCY

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1

Extension	UMask	Description
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_INSERTS

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x91
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vert Egress Allocations. Number of allocations into the Common Mesh Stop Egress. The Egress is used to queue up requests destined for the Vertical Ring on the Mesh.

Table 6-26. Unit Masks for EGRESS_VERT_INSERTS

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_CYCLES_FULL

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x92
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Cycles CMS Vertical Egress Queue Is Full. Number of cycles the Common Mesh Stop Egress was Not Full. The Egress is used to queue up requests destined for the Vertical Ring on the Mesh.

Table 6-27. Unit Masks for EGRESS_VERT_CYCLES_FULL

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_CYCLES_NE

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x93
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Cycles CMS Vertical Egress Queue Is Not Empty. Number of cycles the Common Mesh Stop Egress was Not Empty. The Egress is used to queue up requests destined for the Vertical Ring on the Mesh.

Table 6-28. Unit Mask for EGRESS_VERT_CYCLES_NE

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_NACK

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x98
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vertical Egress NACKs. Counts cycles the Egress received a ring NACK and could not inject any messages on to the Vertical Ring

Table 6-29. Unit Masks for EGRESS_VERT_NACK

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_STARVED

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x9A
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vertical Egress Injection Starvation. Counts injection starvation. This starvation is triggered when the CMS Egress cannot send a transaction onto the Vertical ring for a long period of time.

Table 6-30. Unit Masks for EGRESS_VERT_STARVED

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_ADS_USED

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x9C
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vertical ADS Used. Number of cycles the Vertical Anti-Deadlock Slot was used, broken down by ring type and CMS Agent

Table 6-31. Unit Masks for EGRESS_VERT_ADS_USED

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1

Extension	UMask	Description
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_VERT_BYPASS

- **Category:** EGRESS Vertical Events
- **Event Select**: 0x9E
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Vertical Egress Bypass

Table 6-32. Unit Masks for EGRESS_VERT_BYPASS

Extension	UMask	Description
AD_AG0	bxxxxxxx1	AD ring Agent 0
AK_AG0	bxxxxxx1x	AK ring Agent 0
BL_AG0	bxxxxx1xx	BL ring Agent 0
IV_AG0	bxxxx1xxx	IV ring Agent 0
AD_AG1	bxxx1xxxx	AD ring Agent 1
AK_AG1	bxx1xxxxx	AK ring Agent 1
BL_AG1	bx1xxxxxx	BL ring Agent 1

EGRESS_HORZ_OCCUPANCY

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x94
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal Egress Occupancy. Occupancy event for the Transgress buffers in the Common Mesh Stop. The egress is used to queue up requests destined for the Horizontal Ring on the Mesh.

Table 6-33. Unit Masks for EGRESS_HORZ_OCCUPANCY

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_INSERTS

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x95
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal Egress Inserts. Number of allocations into the Transgress buffers in the Common Mesh Stop. The egress is used to queue up requests destined for the Horizontal Ring on the Mesh.

Table 6-34. Unit Masks for EGRESS_HORZ_INSERTS

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_CYCLES_FULL

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x96
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Cycles CMS Horizontal Egress Queue is Full. Cycles the Transgress buffers in the Common Mesh Stop are Full. The egress is used to queue up requests destined for the Horizontal Ring on the Mesh.

Table 6-35. Unit Masks for EGRESS_HORZ_CYCLES_FULL

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_CYCLES_NE

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x97
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Cycles CMS Horizontal Egress Queue is Not Empty. Cycles the Transgress buffers in the Common Mesh Stop are Not-Empty. The egress is used to queue up requests destined for the Horizontal Ring on the Mesh.

Table 6-36. Unit Masks for EGRESS_HORZ_CYCLES_NE

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_NACK

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x99
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal Egress NACKs. Counts cycles the Egress received a ring NACK and could not inject any messages onto the Horizontal Ring

Table 6-37. Unit Masks for EGRESS_HORZ_NACK

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_STARVED

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x9B
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal Egress Injection Starvation. Counts injection starvation. This starvation is triggered when the CMS Transgress buffer cannot send a transaction onto the Horizontal ring for a long period of time.

Table 6-38. Unit Masks for EGRESS_HORZ_STARVED

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

EGRESS_HORZ_ADS_USED

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x9D
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal ADS Used. Number of cycles the Horizontal Anti-Deadlock Slot was used, broken down by ring type and CMS Agent

Table 6-39. Unit Masks for EGRESS_HORZ_ADS_USED

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring

EGRESS_HORZ_BYPASS

- **Category:** EGRESS Horizontal Events
- **Event Select**: 0x9F
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: CMS Horizontal Egress Bypass

Table 6-40. Unit Masks for EGRESS_HORZ_BYPASS

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

RING_BOUNCES_VERT

- **Category:** RING Events
- **Event Select**: 0xA0
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: Messages that bounced on the Vertical Ring. Number of incoming messages from the Vertical ring that were bounced, by ring type

Table 6-41. Unit Masks for RING_BOUNCES_VERT

Extension	UMask	Description
AD	bxxxxxxx1	Core AD request to the CHA.
AK	bxxxxxx1x	Acknowledgements to core.

Extension	UMask	Description
BL	bxxxxx1xx	Data Responses to core.
IV	bxxxx1xxx	Snoops of processor's cache.

RING_BOUNCES_HORZ

- **Category:** RING Events
- **Event Select**: 0xA1
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: Messages that bounced on the Horizontal Ring. Number of incoming messages from the Horizontal ring that were bounced, by ring type

Table 6-42. Unit Masks for RING_BOUNCES_HORZ

Extension	UMask	Description
AD	bxxxxxxx1	Core AD request to the CHA.
AK	bxxxxxx1x	Acknowledgements to core.
BL	bxxxxx1xx	Data Responses to core.
IV	bxxxx1xxx	Snoops of processor's cache.

RING_SINK_STARVED_VERT

- **Category:** RING Events
- **Event Select**: 0xA2
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: Vertcial Ring Sink starvation count

Table 6-43. Unit Masks for RING_SINK_STARVED_VERT

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

RING_SINK_STARVED_HORZ

- **Category:** RING Events
- **Event Select**: 0xA3
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1,2,3
- **Event Description**: Horizontal Ring Sink starvation count

Table 6-44. Unit Masks for RING_SINK_STARVED_HORZ

Extension	UMask	Description
AD	bxxxxxxx1	AD ring
AK	bxxxxxx1x	AK ring
BL	bxxxxx1xx	BL ring
IV	bxxxx1xxx	IV ring

RING_SRC_THRTL

- **Category:** RING Events
- **Event Select**: 0xA4
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Counts cycles in throttle mode

FAST_ASSERTED

- **Category:** RING Events
- **Event Select**: 0xA5
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Counts cycles source throttling is asserted

Table 6-45. Unit Masks for FAST_ASSERTED

Extension	UMask	Description
VERT	b00000001	VERT
HORZ	b00000010	HORZ

VERT_RING_AD_IN_USE

- **Category:** RING Events
- **Event Select**: 0xA6
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Vertical AD Ring In Use. Counts the number of cycles that the Vertical AD ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-46. Unit Masks for VERT_RING_AD_IN_USE

Extension	UMask	Description
UP_EVEN	bxxxxxxx1	Up and Even
UP_ODD	bxxxxxx1x	Up and Odd
DN_EVEN	bxxxxx1xx	Down and Even
DN_ODD	bxxxx1xxx	Down and Odd

HORZ_RING_AD_IN_USE

- **Category:** RING Events
- **Event Select**: 0xA7
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Horizontal AD Ring In Use. Counts the number of cycles that the Horizontal AD ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-47. Unit Masks for HORZ_RING_AD_IN_USE

Extension	UMask	Description
LEFT_EVEN	bxxxxxxx1	Left and Even
LEFT_ODD	bxxxxxx1x	Left and Odd
RIGHT_EVEN	bxxxxx1xx	Right and Even
RIGHT_ODD	bxxxx1xxx	Right and Odd

VERT_RING_AK_IN_USE

- **Category:** RING Events
- **Event Select**: 0xA8
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Vertical AK Ring In Use. Counts the number of cycles that the Vertical AK ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-48. Unit Masks for VERT_RING_AK_IN_USE

Extension	UMask	Description
UP_EVEN	bxxxxxxx1	Up and Even
UP_ODD	bxxxxxx1x	Up and Odd
DN_EVEN	bxxxxx1xx	Down and Even
DN_ODD	bxxxx1xxx	Down and Odd

HORZ_RING_AK_IN_USE

- **Category:** RING Events
- **Event Select**: 0xA9
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Horizontal AK Ring In Use. Counts the number of cycles that the Horizontal AK ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-49. Unit Masks for HORZ_RING_AK_IN_USE

Extension	UMask	Description
LEFT_EVEN	bxxxxxxx1	Left and Even
LEFT_ODD	bxxxxxx1x	Left and Odd
RIGHT_EVEN	bxxxxx1xx	Right and Even
RIGHT_ODD	bxxxx1xxx	Right and Odd

VERT_RING_BL_IN_USE

- **Category:** RING Events
- **Event Select**: 0xAA
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Vertical BL Ring in Use. Counts the number of cycles that the Vertical BL ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-50. Unit Masks for VERT_RING_BL_IN_USE

Extension	UMask	Description
UP_EVEN	bxxxxxxx1	Up and Even
UP_ODD	bxxxxxx1x	Up and Odd
DN_EVEN	bxxxxx1xx	Down and Even
DN_ODD	bxxxx1xxx	Down and Odd

HORZ_RING_BL_IN_USE

- **Category:** RING Events
- **Event Select**: 0xAB
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Horizontal BL Ring in Use. Counts the number of cycles that the Horizontal BL ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-51. Unit Masks for HORZ_RING_BL_IN_USE

Extension	UMask	Description
LEFT_EVEN	bxxxxxxx1	Left and Even
LEFT_ODD	bxxxxxx1x	Left and Odd
RIGHT_EVEN	bxxxxx1xx	Right and Even
RIGHT_ODD	bxxxx1xxx	Right and Odd

VERT_RING_IV_IN_USE

- **Category:** RING Events
- **Event Select**: 0xAC
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Vertical IV Ring in Use. Counts the number of cycles that the Vertical IV ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-52. Unit Masks for VERT_RING_IV_IN_USE

Extension	UMask	Description
UP	bxxxxxxx1	Up
DN	bxxxxx1xx	Down

HORZ_RING_IV_IN_USE

- **Category:** RING Events
- **Event Select**: 0xAD
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Horizontal IV Ring in Use. Counts the number of cycles that the Horizontal IV ring is being used at this ring stop. This includes when packets are passing by and when packets are being sunk, but does not include when packets are being sent from the ring stop.

Table 6-53. Unit Masks for HORZ_RING_IV_IN_USE

Extension	UMask	Description
LEFT	bxxxxxxx1	Left
RIGHT	bxxxxx1xx	Right

EGRESS_ORDERING

- **Category:** RING Events
- **Event Select**: 0xAE
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Egress Blocking due to ordering requirements. Counts number of cycles IV was blocked in the TGR Egress due to SNP/GO Ordering requirements.

Table 6-54. Unit Masks for EGRESS_ORDERING

Extension	UMask	Description
IV_SNP_GO_UP	bxxxxxxx1	Up
IV_SNP_GO_DN	bxxxxx1xx	Down

RxR_OCCUPANCY

- **Category:** TG Ingress Events
- **Event Select**: 0xB0
- **Max Inc / cycle**: 24
- **Counter Availability**: 0,1,2,3
- **Event Description**: Transgress Ingress Occupancy. Occupancy event for the Ingress buffers in the CMS. The Ingress is used to queue up requests received from the mesh.

Table 6-55. Unit Masks for RxR_OCCUPANCY

Extension	UMask	Description
AD_BNC	bxxxxxxx1	AD_BNC
AK_BNC	bxxxxxx1x	AK

Extension	UMask	Description
BL_BNC	bxxxxx1xx	BL_BNC
IV_BNC	bxxxx1xxx	IV
AD_CRD	bxxx1xxxx	AD_CRD
BL_CRD	bx1xxxxxx	BL_CRD

RxR_INSERTS

- **Category:** TG Ingress Events
- **Event Select**: 0xB1
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Transgress Ingress Allocations. Number of allocations into the CMS Ingress. The Ingress is used to queue up requests received from the mesh.

Table 6-56. Unit Masks for RxR_INSERTS

Extension	UMask	Description
AD_BNC	bxxxxxxx1	AD_BNC
AK_BNC	bxxxxxx1x	AK
BL_BNC	bxxxxx1xx	BL_BNC
IV_BNC	bxxxx1xxx	IV
AD_CRD	bxxx1xxxx	AD_CRD
BL_CRD	bx1xxxxxx	BL_CRD

RxR_BYPASS

- **Category:** TG Ingress Events
- **Event Select**: 0xB2
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Transgress Ingress Bypass. Number of packets bypassing the CMS Ingress

Table 6-57. Unit Masks for RxR_BYPASS

Extension	UMask	Description
AD_BNC	bxxxxxxx1	AD_BNC
AK_BNC	bxxxxxx1x	AK
BL_BNC	bxxxxx1xx	BL_BNC
IV_BNC	bxxxx1xxx	IV
AD_CRD	bxxx1xxxx	AD_CRD
BL_CRD	bx1xxxxxx	BL_CRD

RxR_CRD_STARVED

- **Category:** TG Ingress Events
- **Event Select**: 0xB3
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Transgress Injection Starvation. Counts cycles under injection starvation mode. This starvation is triggered when the CMS Ingress cannot send a transaction onto the mesh for a long period of time. In this case, the Ingress is unable to forward to the Egress due to a lack of credit.

Table 6-58. Unit Masks for RxR_CRD_STARVED

Extension	UMask	Description
AD_BNC	bxxxxxxx1	AD_BNC
AK_BNC	bxxxxxx1x	AK
BL_BNC	bxxxxx1xx	BL_BNC
IV_BNC	bxxxx1xxx	IV
AD_CRD	bxxx1xxxx	AD_CRD
BL_CRD	bx1xxxxxx	BL_CRD
IFV	b1xxxxxxx	IFV

RxR_BUSY_STARVED

- **Category:** TG Ingress Events
- **Event Select**: 0xB4
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Transgress Injection Starvation. Counts cycles under injection starvation mode. This starvation is triggered when the CMS Ingress cannot send a transaction onto the mesh for a long period of time.; in this case, because a message from the other queue has higher priority.

Table 6-59. Unit Masks for RxR_BUSY_STARVED

Extension	UMask	Description
AD_BNC	bxxxxxxx1	AD_BNC
BL_BNC	bxxxxx1xx	BL_BNC
AD_CRD	bxxx1xxxx	AD_CRD
BL_CRD	bx1xxxxxx	BL_CRD

§

7 Untile-M2PCIe Performance Monitoring Events

RxR_CYCLES_NE

- **Category:** INGRESS Events
- **Event Select**: 0x10
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Ingress Queue Cycles Not Empty. Counts the number of cycles when the M2PCIe Ingress is not empty

Table 7-1. Unit Masks for RxR_CYCLES_NE

Extension	UMask	Description
CBO_IDI	bxxxxxxx1	CBO_IDI
CBO_NCB	bxxxxxx1x	CBO_NCB
CBO_NCS	bxxxxx1xx	CBO_NCS
ALL	b1xxxxxxx	ALL

TxC_CYCLES_NE

- **Category:** EGRESS Events
- **Event Select**: 0x23
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: Egress (to CMS) Cycles Not Empty. Counts the number of cycles when the M2PCIe Egress is not empty. This tracks messages for one of the two CMS ports that are used by the M2PCIe agent. This can be used in conjunction with the M2PCIe Ingress Occupancy Accumulator event in order to calculate average queue occupancy. Multiple egress buffers can be tracked at a given time using multiple counters.

Table 7-2. Unit Masks for TxC_CYCLES_NE

Extension	UMask	Description
AD_0	bxxxxxxx1	AD_0
AK_0	bxxxxxx1x	AK_0
BL_0	bxxxxx1xx	BL_0
AD_1	bxxxx1xxx	AD_1
AK_1	bxxx1xxxx	AK_1
BL_1	bxx1xxxxx	BL_1

TxC_INSERTS

- **Category:** EGRESS Events
- **Event Select**: 0x24
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Egress (to CMS) Ingress. Counts the number of number of messages inserted into the M2PCIe Egress queue. This tracks messages for one of the two CMS ports that are used by the M2PCIe agent. This can be used in conjunction with the M2PCIe Ingress Occupancy Accumulator event in order to calculate average queue occupancy.

Table 7-3. Unit Mask for TxC_INSERTS

Extension	UMask	Description
AD_0	bxxxxxxx1	AD_0
AK_0	bxxxxxx1x	AK_0
BL_0	bxxxxx1xx	BL_0
AK_CRD_0	bxxxx1xxx	AK_CRD_0
AD_1	bxxx1xxxx	AD_1
AK_1	bxx1xxxxx	AK_1
BL_1	bx1xxxxxx	BL_1
AK_CRD_1	b1xxxxxxx	AK_CRD_1

TxC_CYCLES_FULL

- **Category:** EGRESS Events
- **Event Select**: 0x25
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1,2,3
- **Event Description**: Egress (to CMS) Cycles Full. Counts the number of cycles when the M2PCIe Egress is full. This tracks messages for one of the two CMS ports that are used by the M2PCIe agent.

Table 7-4. Unit Masks for TxC_CYCLES_FULL

Extension	UMask	Description
AD_0	bxxxxxxx1	AD_0
AK_0	bxxxxxx1x	AK_0
BL_0	bxxxxx1xx	BL_0
AD_1	bxxxx1xxx	AD_1
AK_1	bxxx1xxxx	AK_1
BL_1	bxx1xxxxx	BL_1

§

8 Untile-IRP Performance Monitoring Events

IO_CLKS_COUNT_ESEL

- **Category:** UCLK Events
- **Event Select**: 0x00
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: IRP Clocks

BL_INGRESS_ALLOCATIONS_DRS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x01
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL DRS Ingress Inserts

BL_INGRESS_ALLOCATIONS_NCB_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x02
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL NCB Ingress Inserts

BL_INGRESS_ALLOCATIONS_NCS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x03
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL NCS Ingress Inserts

BL_INGRESS_FULL_DRS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x04
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL DRS Ingress Cycles Full

BL_INGRESS_FULL_NCB_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x05
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL NCB Ingress Cycles Full

BL_INGRESS_FULL_NCS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x06
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: BL NCS Ingress Cycles Full

BL_INGRESS_OCCUPANCY_DRS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x07
- **Max Inc / cycle**: 16
- **Counter Availability**: 0,1
- **Event Description**: BL DRS Ingress Occupancy

BL_INGRESS_OCCUPANCY_NCB_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x08
- **Max Inc / cycle**: 10
- **Counter Availability**: 0,1
- **Event Description**: BL NCB Ingress Occupancy

BL_INGRESS_OCCUPANCY_NCS_ESEL

- **Category:** BL Ingress Events
- **Event Select**: 0x09
- **Max Inc / cycle**: 10
- **Counter Availability**: 0,1
- **Event Description**: BL NCS Ingress Occupancy

AK_INGRESS_ALLOCATIONS_ESEL

- **Category:** AK Ingress Events
- **Event Select**: 0x0a
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1
- **Event Description**: AK Ingress Allocations

OUTBOUND_REQUESTS_REQUEST_Q_OCCUPANCY_ESEL

- **Category:** Outbound Request Events
- **Event Select**: 0x0d
- **Max Inc / cycle**: 8
- **Counter Availability**: 0,1
- **Event Description**: Outbound Request Queue Occupancy. Accumulates the number of outstanding outbound requests from the IRP to the switch (towards the devices). This can be used in conjuection with the allocations event in order to calculate average latency of outbound requests.

OUTBOUND_REQUESTS_REQUEST_Q_ALLOCATIONS_NCB_ESEL

- **Category:** Outbound Request Events
- **Event Select**: 0x0e
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: Outbound Read Requests. Counts the number of requests issued to the switch (towards the devices)

OUTBOUND_REQUESTS_REQUEST_Q_ALLOCATIONS_NCS_ESEL

- **Category:** Outbound Request Events
- **Event Select**: 0x0f
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: Outbound Read Requests. Counts the number of requests issued to the switch (towards the devices)

WRITE_CACHE_TOTAL_ESEL

- **Category:** Cache Events
- **Event Select**: 0x12
- **Max Inc / cycle**: 128
- **Counter Availability**: 0,1
- **Event Description**: Total Write Cache Occupancy. Accumulates the number of reads and writes that are outstanding in the untile in each cycle. This is effectively the sum of the READ_OCCUPANCY and WRITE_OCCUPANCY events.

Table 8-1. Unit Masks for WRITE_CACHE_TOTAL_ESEL

Extension	UMask	Description
ANY	bxxxxxxx1	Any Source
IV_Q	bxxxxxx1x	Counts snoops

COHERENT_OP_ESEL

- **Category:** Cache Events
- **Event Select**: 0x13
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1
- **Event Description**: Coherent Ops. Counts the number of coherency related operations serviced by the IRP

Table 8-2. Unit Masks for COHERENT_OP_ESEL

Extension	UMask	Description
RDCUR	bxxxxxxx1	PCIRdCur
RFO	bxxxx1xxx	RFO
I2M	bxxx1xxxx	I2M
WBMTOI	bx1xxxxxx	WbMtoI
CLFLUSH	b1xxxxxxx	CLFlush

TRANSACTIONS_COUNT_ESEL

- **Category:** Misc Events
- **Event Select**: 0x16
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: Inbound Transaction Count. Counts the number of "Inbound" transactions from the IRP to the Untile. This can be filtered based on request type in addition to the source queue. Note the special filtering equation. We do OR-reduction on the request type. If the SOURCE bit is set, then we also do AND qualification based on the source portID.

Table 8-3. Unit Masks for TRANSACTIONS_COUNT_ESEL

Extension	UMask	Description
READ_FETCH	bxxxxxxx1	Reads
WRITE_FETCH	bxxxxxx1x	Writes
WRITE_PREFETCH	bxxxx1xxx	Write Prefetches
ATOMIC	bxxx1xxxx	Atomic
OTHER	bxx1xxxxx	Other
SOURCE	bx1xxxxxx	Select Source

SNOOP_RESPONSES_ESEL

- **Category:** Misc Events
- **Event Select**: 0x17
- **Max Inc / cycle**: 2
- **Counter Availability**: 0,1
- **Event Description**: Snoop Responses

Table 8-4. Unit Masks for SNOOP_RESPONSES_ESEL

Extension	UMask	Description
IRPMISS	bxxxxxxx1	Miss
IRPHITI	bxxxxxx1x	Hit I
IRPHITES	bxxxxx1xx	Hit E or S
IRPHITM	bxxxx1xxx	Hit M

Extension	UMask	Description
SNPCODE	bxxx1xxxx	SnpCode
SNPDATA	bxx1xxxxx	SnpData
SNPINV	bx1xxxxxx	SnpInv

STALL_CYCLES_AD_EGRESS_CREDITS_ESEL

- **Category:** Stall Events
- **Event Select**: 0x18
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: No AD Egress Credit Stalls. Counts the number times when it is not possible to issue a request to the R2PCIe because there are no AD Egress Credits available

STALL_CYCLES_BL_EGRESS_CREDITS_ESEL

- **Category:** Stall Events
- **Event Select**: 0x19
- **Max Inc / cycle**: 1
- **Counter Availability**: 0,1
- **Event Description**: No BL Egress Credit Stalls. Counts the number times when it is not possible to issue data to the R2PCIe because there are no BL Egress Credits available

§