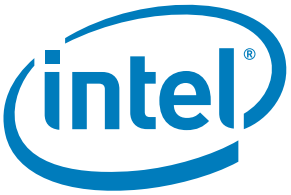
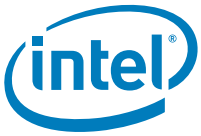


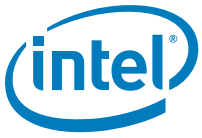
Intel® Xeon® Processor Scalable Memory Family Uncore Performance Monitoring

Reference Manual

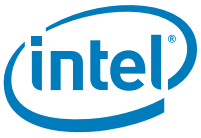
July 2017

Reference Number: 336274-001US



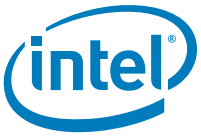


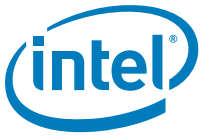
Contents



Figures

Tables





Revision History

Document Number	Revision Number	Description	Date
336274	001	• Initial Release	July 2017

§

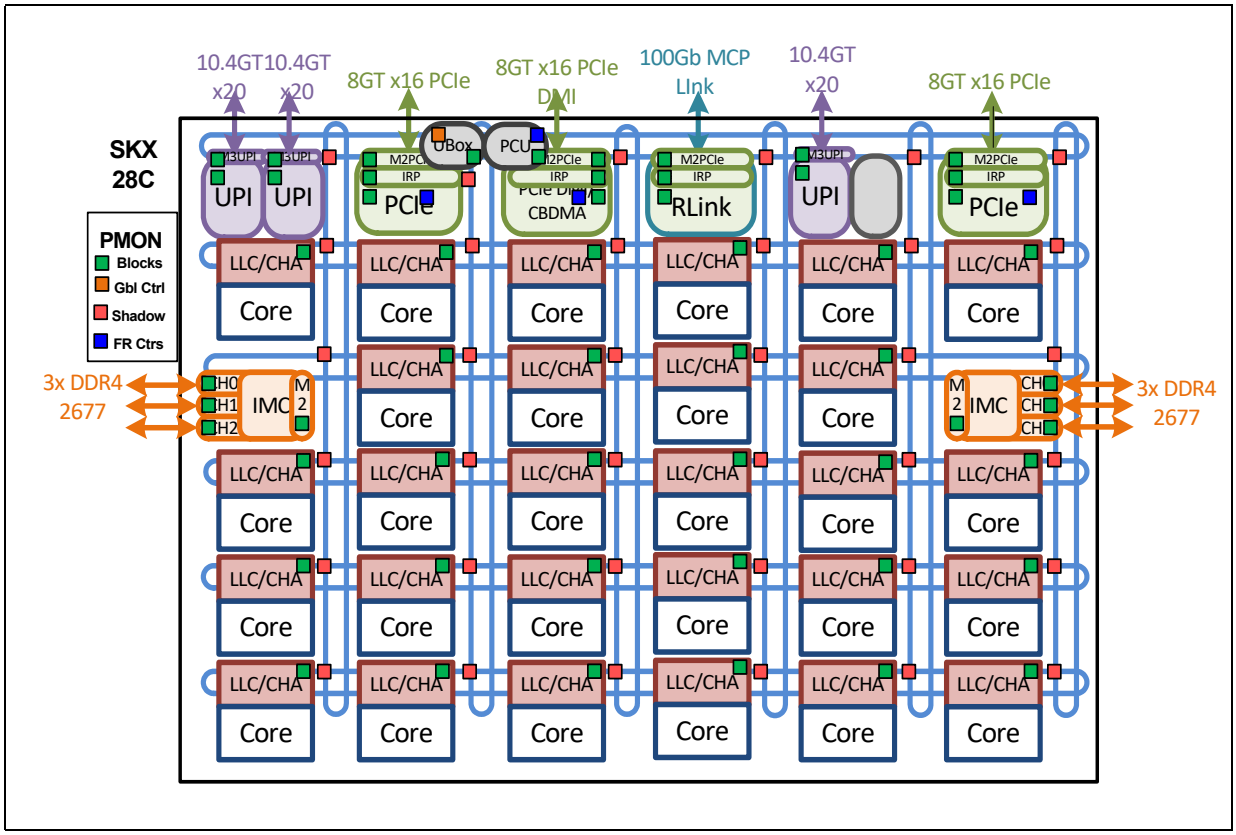
1 Introduction

1.1 Introduction

'Uncore' roughly equates to logic outside the CPU cores but residing on the same die. Traffic (e.g. Data Reads) generated by threads executing on CPU cores or IO devices may be operated on by logic in the Uncore. Logic responsible for managing coherency, managing access to the DIMMs, managing power distribution and sleep states, and so forth.

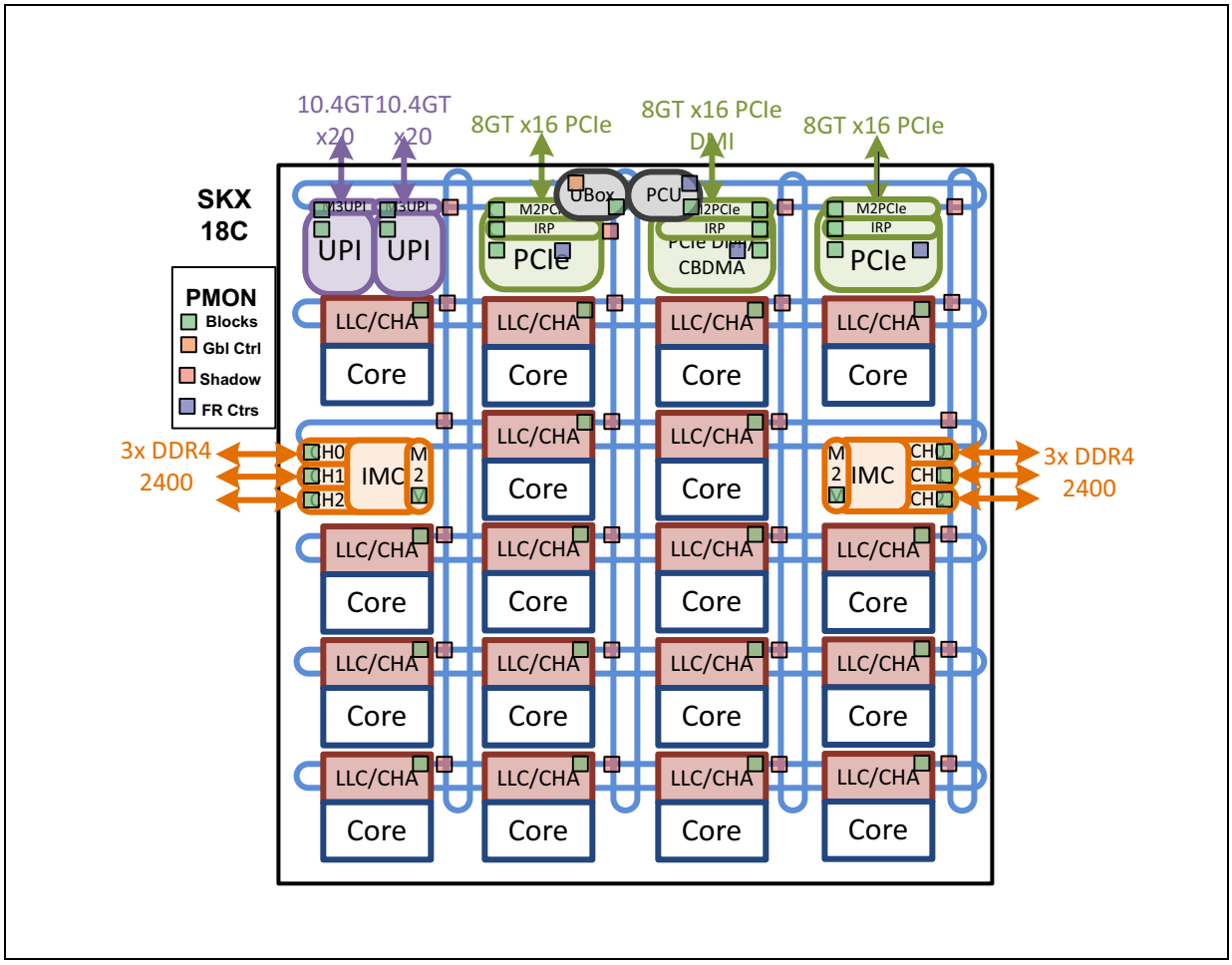
The uncore sub-system of the next generation Intel® Xeon® Processor Scalable Memory Family is shown in the following figure. The uncore sub-system consists of a variety of components, many assigned to the aforementioned responsibilities, ranging from the CHA cache/home agent to the power controller unit (PCU) and integrated memory controller (IMC), to name a few. Most of these components provide similar performance monitoring capabilities.

Figure 1-1. Intel® Xeon® Processor Scalable Memory Family - Block diagram for a 28C part



Note: This diagram represents one possible configuration. The number of cores and Intel® Ultra Path Interconnect (Intel® UPI) links supported will vary by SKU. Not all features supported on all SKUs.

Figure 1-2. Intel® Xeon® Processor Scalable Memory Family - Example 18C part

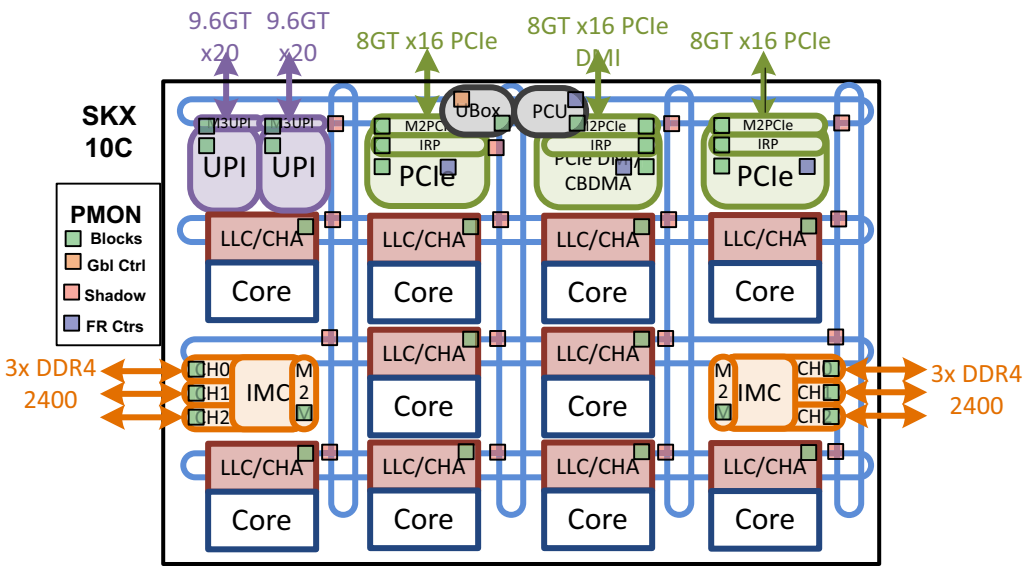


Note: This diagram represents one possible configuration. The number of cores and Intel UPI links supported will vary by SKU. Not all features are supported on all SKUs.

Before going in to the details of Intel® Xeon® Processor Scalable Memory Family's Uncore PMON, the following sections will provide

- A general overview of Uncore PMON operation and the state provided SW to manage its operation.
- Functionality common to individual units with the common logic to support the functionality.
- A summary of Intel® Xeon® Processor Scalable Memory Family uncore performance monitoring capabilities.
- Addressing all Intel® Xeon® Processor Scalable Memory Family uncore performance monitoring state.
- Some guidance to SW including how to manage a monitoring session and find the bus number for state in PCICFG space.

Figure 1-3. Intel® Xeon® Processor Scalable Memory Family - Example 10C part



Note: This diagram represents one possible configuration. The number of cores and Intel UPI links supported will vary by SKU. Not all features are supported on all SKUs

1.2 Section References

The following sections provide a breakdown of the performance monitoring capabilities for each box.

- Section 2.1, "UBox Performance Monitoring"
- Section 2.2, "Mesh Performance Monitoring"
- Section 2.2.4, "Caching/Home Agent (CHA) Performance Monitoring"
- Section 2.3, "Memory Controller (IMC) Performance Monitoring"
- Section 2.4, "IIO Performance Monitoring"
- Section 2.5, "IRP Performance Monitoring"
- Section 2.6, "Intel® UPI Link Layer Performance Monitoring"
- Section 2.7, "M2M Performance Monitoring"
- Section 2.8, "M3UPI Performance Monitoring"
- Section 2.9, "Power Control (PCU) Performance Monitoring"
- Section 3.1, "Packet Matching Reference(s)"

1.3 Uncore PMON Overview

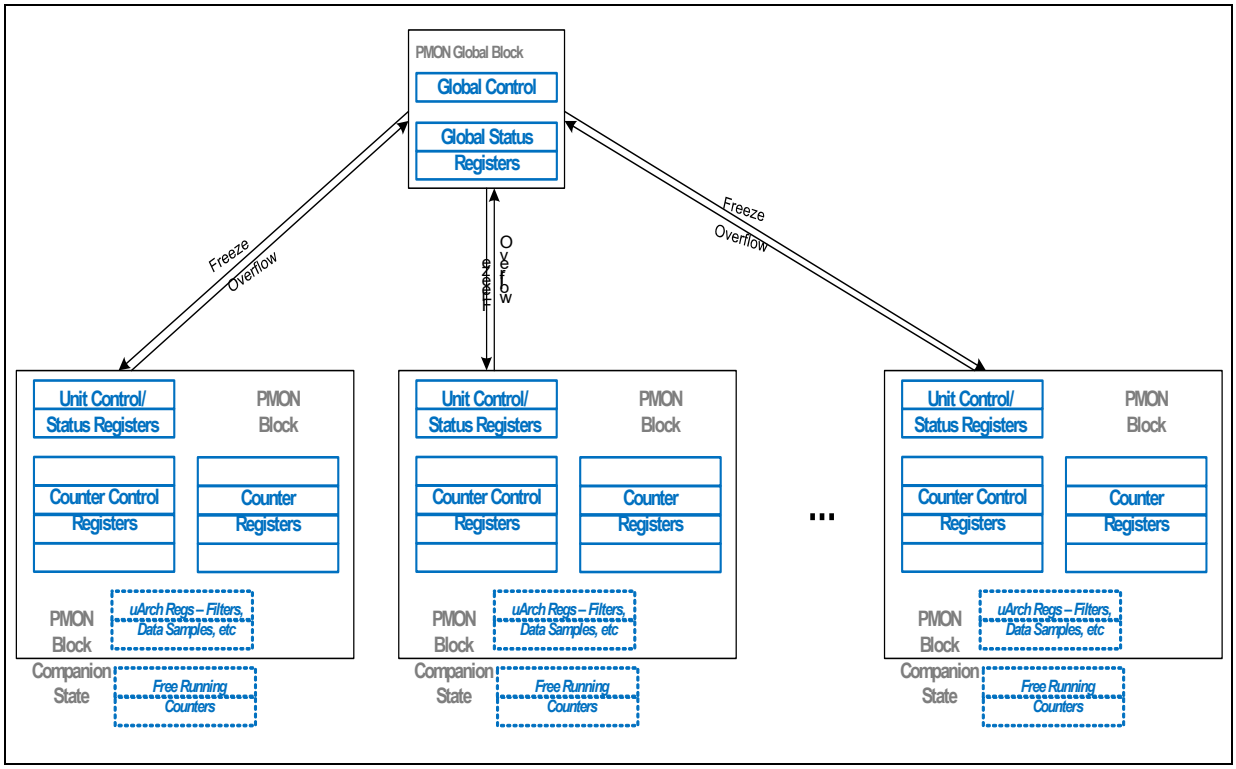
1.3.1 A Simple Hierarchy

Uncore performance monitoring is managed through a very simple hierarchy. There are some number of performance monitoring (or '**PMON**') units governed by a global control.

Each PMON block contains a set of counters with paired control registers. Each unit provides a set of events for SW to select from. SW can ask HW to collect an event by specifying what to count in a counter's control register. SW can then periodically read the collected value from the paired counter.

Some units offer an expanded event set that require additional counter control bits. (for example, IIO). Some units offer the ability to further refine, or 'filter', the monitored events (for example, CHA; see Section 2.2.6.2, "CHA Filter Registers (Cn_MSR_PMON_BOX_FILTER{0,1})").

Figure 1-4. Uncore PMON Components and Hierarchy



Note: Uncore performance monitors represent a per-socket resource not meant to be affected by context switches and thread migration performed by the OS. It is recommended that the monitoring software agent establish a fixed affinity binding to prevent event count cross-talk across different uncore PMU.

To manage the large number of counter registers distributed across so many units and collect event data efficiently, each block has a modest amount of control/status governed by a similar global control/status.

SW can directly synchronize actions across counters (for example, to start/stop/reset counting) within each PMON block or across all PMON blocks through this control state.

SW can indirectly synchronize actions across counters (for example, stop counting) in all the PMON blocks by telling HW what to do when a counter overflows. SW can set a counter to overflow, after a set number of events have been captured, by pre-seeding the counter. For each counter, SW can then choose whether to notify the global PMON control that a counter has overflowed.

Upon receipt of an overflow, the global control will assert the global freeze signal. Once the global freeze has been detected, each box will disable (or 'freeze') all of its counters. In the process of generating a global freeze, SW can configure the global control to send a PMI signal to the core executing the monitoring software.

The following sections will detail the basic control state provided to SW to control performance monitoring in the uncore.

1.3.2 Global PMON State

Table 1-1. Global Performance Monitoring Control MSRs

MSR Name	MSR Address	Size (bits)	Description
U_MSR_PMON_GLOBAL_STATUS	0x0701	32	UBox PMON Global Status
U_MSR_PMON_GLOBAL_CTL	0x0700	32	UBox PMON Global Control

1.3.2.1 Global PMON Global Control/Status Registers

The following registers represent state governing all PMUs in the uncore, both to exert global control and collect unit-level information.

U_MSR_PMON_GLOBAL_CTL contains bits that can stop (*.frz_all*) / restart (*.unfrz_all*) all the uncore counters.

Figure 1-5. PMON Global Control Register for Intel® Xeon® Processor Scalable Memory Family

Table 1-2. U_MSR_PMON_GLOBAL_CTL Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Attr	HW Reset Val	Description
frz_all	63	WO	0	Freeze all uncore performance monitors.
wk_on_pmi	62	RW	0	If PMI event requested to send to core... 0 - Send event to cores already awakened 1 - Wake any sleeping core and send PMI to all cores.
unfrz_all	61	WO	0	Unfreeze all uncore performance monitors.
rsv	60:28	RV	0	Reserved

Table 1-2. U_MSR_PMON_GLOBAL_CTL Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Attr	HW Reset Val	Description
pmi_core_sel	27:0	RW	0	PMI Core Select Ex: If counter overflow is sent to UBox... 00000000000000000000000000000 - No PMI sent 00000000000000000000000000001 - Send PMI to core 0 00000000000000000000001000000 - Send PMI to core 6 00000000000000000000001100010 - Send PMI to core 2, 5 & 6 etc. NOTE: If wk_on_pmi is set to 1, a wake will be sent to any sleeping core in the mask prior to sending the PMI.

If an overflow is detected in any of the uncore's PMON registers, it will be summarized in U_MSR_PMON_GLOBAL_STATUS. This register accumulates overflows sent to it from uncore boxes with PMON blocks. To reset these overflow bits, a user must set the corresponding bits in U_MSR_PMON_GLOBAL_STATUS to 1, which will act to clear them.

Figure 1-6. PMON Global Status Register for Intel® Xeon® Processor Scalable Memory Family

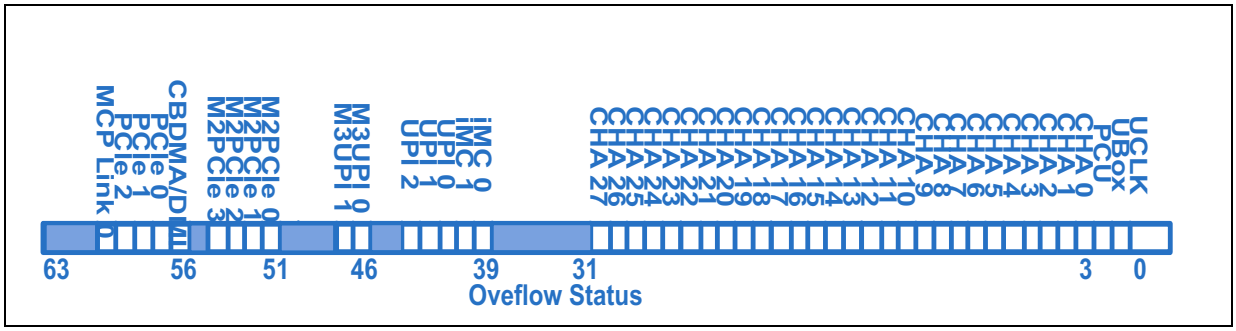


Table 1-3. U_MSR_PMON_GLOBAL_STATUS Register – Field Definitions (Sheet 1 of 2)

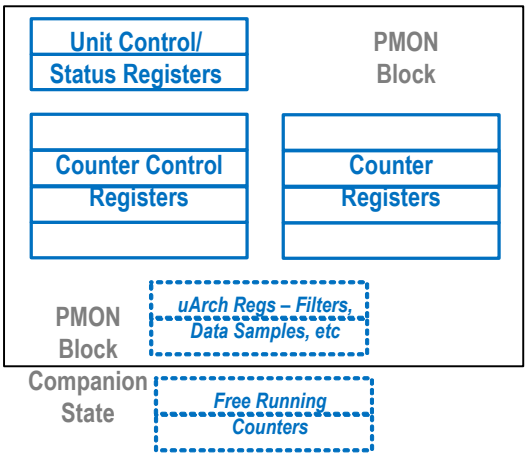
Field	Bits	Attr	HW Reset Val	Description
ov_m2m1	63	RW1C	0	Overflow detected in an M2M 1 PMON register.
ov_m2m0	62	RW1C	0	Overflow detected in an M2M 0 PMON register.
rsv	61	RV	0	Reserved
ov_iom0	60	RW1C	0	Overflow detected in a PMON register in MCP Link 0 (either in IIO or IRP block).
ov_iop2	59	RW1C	0	Overflow detected in a PMON register in PStack 2 (either in IIO or IRP block).
ov_iop1	58	RW1C	0	Overflow detected in a PMON register in PStack 1 (either in IIO or IRP block).
ov_iop0	57	RW1C	0	Overflow detected in a PMON register in PStack 0 (either in IIO or IRP block).
ov_ioc0	56	RW1C	0	Overflow detected in a PMON register in CStack 0 (either in IIO or IRP block).

Table 1-3. U_MSR_PMON_GLOBAL_STATUS Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Attr	HW Reset Val	Description
rsv	55:48	RV	0	Reserved
ov_m3u1	47	RW1C	0	Overflow detected in a M3UPI1 PMON register.
ov_m3u0	46	RW1C	0	Overflow detected in a M3UPI0 PMON register.
rsv	45:44	RV	0	Reserved
ov_upi2	43	RW1C	0	Overflow detected in a UPI1 - Agent 0 PMON register.
ov_upi1	42	RW1C	0	Overflow detected in a UPI0 - Agent 1 PMON register.
ov_upi0	41	RW1C	0	Overflow detected in a UPI0 - Agent 0 PMON register.
ov_mc1	40	RW1C	0	Overflow detected in an iMC1 PMON register.
ov_mc0	39	RW1C	0	Overflow detected in an iMC0 PMON register.
rsv	38:31	RV	0	Reserved
ov_c[27-0]	30:3	RW1C	0	Overflow detected in a CHA PMON register, 1 bit for each CHA where bit 5 corresponds CHA 0, etc.
ov_p	2	RW1C	0	Overflow detected in a PCU PMON register.
ov_u	1	RW1C	0	Overflow detected in a UBox PMON register.
ov_u_fixed	0	RW1C	0	Overflow detected in UBox fixed PMON register.

1.4 Unit Level PMON State

Each PMON block in the uncore is composed of the following state:



- A Unit Control register to aid software sample collection.
- Status registers to record when a counter within the Block overflows.
- A set of data registers
- A set of control registers, each paired to a data register, to allow SW to specify what event should be captured.
- Additional micro-architectural specific state designed to enhance performance monitoring collection within a block. e.g. event or traffic Filters.
- Some free running counters, although not subject to the PMON hierarchy, may be included in this document with the Unit they are associated with.

Every PMON block in the system is governed by a modest amount of Unit level Control. Each bit intended to assist SW in more efficiently managing the PMON state within the block. Reset bits help reduce the time SW needs to setup a new sample.

Note: If the PMON registers within the unit are shared among different users, either those users should leave this register untouched or they should agree on the user allowed to affect the Unit level control state.

Figure 1-7. PMON Unit Control Register for Intel® Xeon® Processor Scalable Memory Family - Common to all PMON Blocks

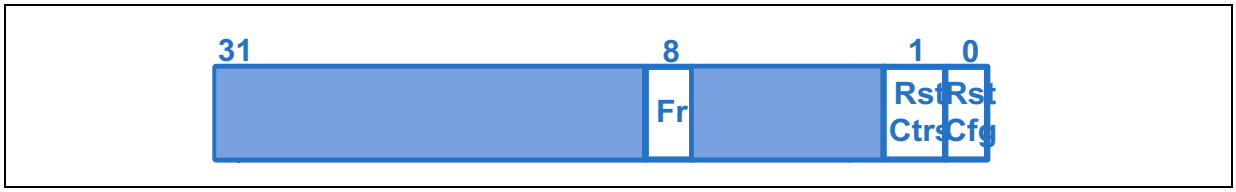
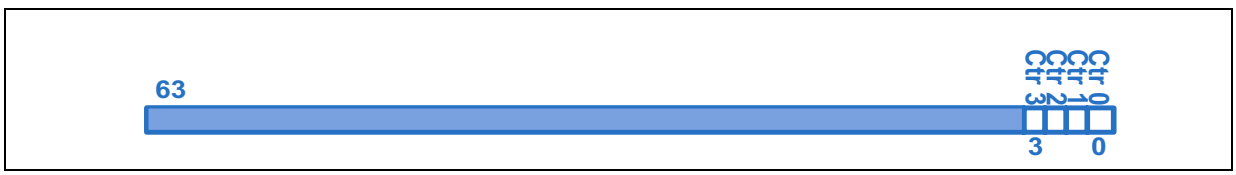


Table 1-4. PMON_UNIT_CTL Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	31:18	RV	0	Reserved
rsv	17:16	RV	0	Reserved; SW must write to 1 else behavior is undefined.
rsv	15:9	RV	0	Reserved
frz	8	WO	0	Freeze. If set to 1 the counters in this box will be frozen.
rsv	7:2	RV	0	Reserved
rst_ctrs	1	WO	0	Reset Counters. When set to 1, the Counter Registers will be reset to 0.
rst_ctrl	0	WO	0	Reset Control. When set to 1, the Counter Control Registers will be reset to 0.

Figure 1-8. PMON Unit Status Register for Intel® Xeon® Processor Scalable Memory Family - Format common to all PMON Blocks (except the iMC)



Note: If an overflow is detected from one of the Unit's PMON registers, the corresponding bit in the PMON_UNIT_STATUS.ov field will be set. To reset these overflow bits, a user must write a value of '1' to them (which will clear the bits). There are typically four counters per PMON block. But that number may vary. Check Table 1-8, "Per-Box Performance Monitoring Capabilities" or the section detailing each unit's functionality for the number counters it supports.

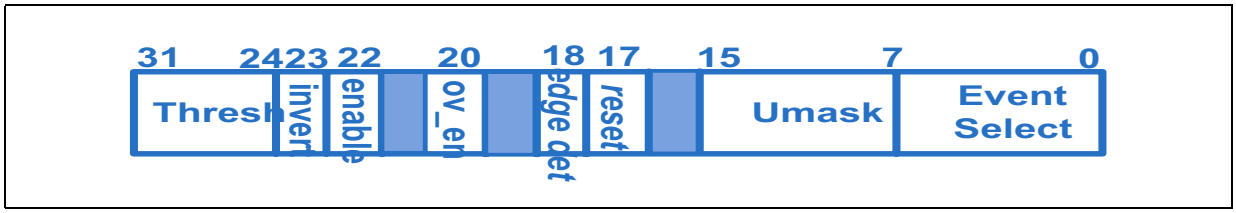
Table 1-5. PMON_UNIT_STATUS Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	31:4	RV	0	Reserved
ov	3:0	RW1C	0	If an overflow is detected from the corresponding PMON_CTR register, it's overflow bit will be set. NOTE: Write of '1' will clear the bit.

1.4.1 Unit PMON state - Counter/Control Pairs

The following table defines the layout for the standard performance monitor control registers. Their main task is to select the event to be monitored by their respective data counter (*.ev_sel*, *.umask*). Additional control bits are provided to shape the incoming events (e.g. *.invert*, *.edge_det*, *.thresh*) as well as provide additional functionality for monitoring software (*.rst,.ov_en*).

Figure 1-9. PMON Counter Control Register for Intel® Xeon® Processor Scalable Memory Family - Fields common to all PMON Blocks



Note: Per Unit considerations - please refer to each unit section for more detail on:

- Certain units may make use of additional bits in these counter control registers.
- The width of the Thresh field is dependent on a unit's 'widest' event (i.e. the event that can increment the most per cycle, typically measuring per-cycle occupancy of a large queue).
- As of Intel® Xeon® Processor Scalable Memory Family, several unit counter control registers are still 32b, some 64b.

An overview of the Counter Control logic is in the next section.

Table 1-6. Baseline *_PMON_CTLx Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	63:32	RV	0	Reserved - Only relevant to unit's that use 64b control registers
thresh	31:24	RW	0	Threshold is used, along with the invert bit, to compare against the counter's incoming increment value. i.e. the value that will be added to the counter. For events that increment by more than 1 per cycle, if the threshold is set to a value greater than 1, the data register will accumulate instances in which the event increment is >= threshold. e.g. say you have an event to accumulate the occupancy of a 64-entry queue every cycle. By setting the threshold value to 60, the data register would count the number of cycles the queue's occupancy was >= 60.
invert	23	RW	0	Invert comparison against Threshold. 0 - comparison will be 'is event increment >= threshold?'. 1 - comparison is inverted - 'is event increment < threshold?' e.g. for a 64-entry queue, if SW wanted to know how many cycles the queue had fewer than 4 entries, SW should set the threshold to 4 and set the invert bit to 1. Intel® Xeon® Processor Scalable Memory Family NOTE: .invert is in series following .thresh, Due to this, the .thresh field must be set to a non-0 value. For events that increment by no more than 1 per cycle, set .thresh to 0x1. Also, if .edge_det is set to 1, the counter will increment when a 1 to 0 transition (i.e. falling edge) is detected.
en	22	RW	0	Local Counter Enable
rsv	21	RV	0	Reserved. SW must write to 0 else behavior is undefined.
ov_en	20	RW	0	When this bit is set to 1 and the corresponding counter overflows, an overflow message is sent to the UBox's global logic. The message identifies the unit that sent it. Once received, the global status register will record the overflow in the corresponding U_MSR_PMON_GLOBAL_STATUS bit.
rsv	19	RV	0	Reserved
edge_det	18	RW	0	When set to 1, rather than measuring the event in each cycle it is active, the corresponding counter will increment when a 0 to 1 transition (i.e. rising edge) is detected. When 0, the counter will increment in each cycle that the event is asserted. NOTE: .edge_det is in series following .thresh, Due to this, the .thresh field must be set to a non-0 value. For events that increment by no more than 1 per cycle, set .thresh to 0x1.
rst	17	WO	0	When set to 1, the corresponding counter will be cleared to 0.
rsv	16	RV	0	Reserved. SW must write to 0 else behavior is undefined.
umask	15:8	RW	0	Select subevents to be counted within the selected event.
ev_sel	7:0	RW	0	Select event to be counted.

The default width for performance monitor data registers are 48b wide. A counter overflow occurs when a carry out from bit 47 is detected. Software can force all uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N and setting the control register to send an overflow message to the UBox (refer to Section 1.3.2, "Global PMON State"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

To ensure accuracy, SW should stop the counter and check the overflow status before reading its value. But, if accessible, software can continuously read the data registers without disabling event collection.

Figure 1-10. PMON Counter Register for Intel® Xeon® Processor Scalable Memory Family - Common to all PMON Blocks

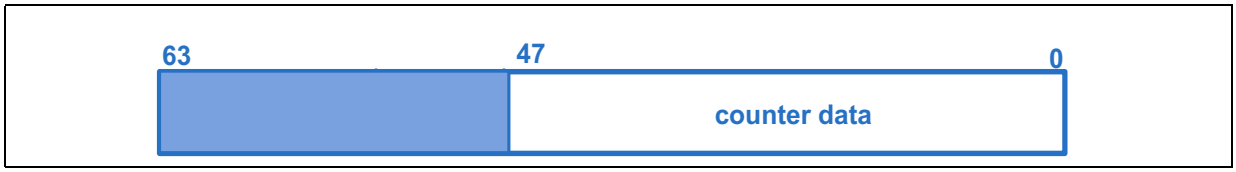


Table 1-7. Baseline *_PMON_CTRx Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	63:48	RV	0	Reserved
event_count	47:0	RW-V	0	48-bit performance event counter

1.4.1.1 Unit PMON Registers - On Overflow and the Consequences (PMI/ Freeze)

If an overflow is detected from a Unit's performance counter, the overflow bit is set at the unit level (*_PMON_UNIT_STATUS.ov).

If the counter is enabled to communicate the overflow (*_PMON_CTL.ov_en is set to 1), an overflow message is sent to the UBox. When the UBox receives the overflow signal, the *_PMON_GLOBAL_STATUS.ov_x bit is set, a global freeze signal is sent and a PMI can be generated. 'x' represents the box generating the overflow (see Table 1-3, "U_MSR_PMON_GLOBAL_STATUS Register – Field Definitions").

Once a freeze has occurred, in order to see a new freeze, the overflow responsible for the freeze must be cleared by setting the corresponding bit in *_PMON_UNIT_STATUS.ov and U_MSR_PMON_GLOBAL_STATUs.ov_x to 1 (which acts to clear the bits).

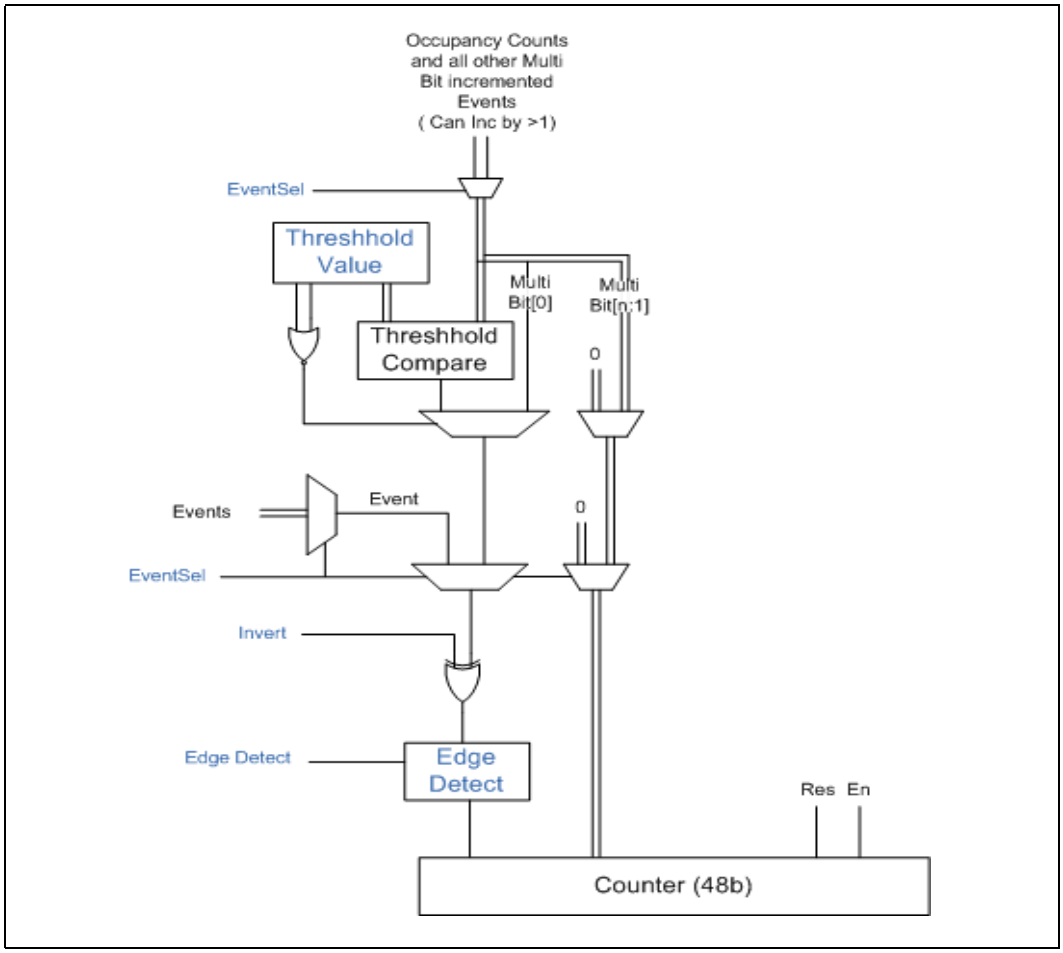
Assuming all counters have been locally enabled (*.en* bit is set to 1 in every control register meant to monitor events) and the overflow bits have been cleared, the Unit is prepared for a new sample interval. Once the global controls have been re-enabled (Section 1.9.4, "Enabling a New Sample Interval from Frozen Counters"), counting will resume.

1.5 Uncore PMON - Typical Counter Control Logic

Following is a logic diagram for the standard perfmon counter control. It illustrates how event information is routed, selected, filtered (by other bits in the control register) and sent to the paired data register for storage.

Note: The PCU uses an adaptation of this block (refer to Section 2.9, "Power Control (PCU) Performance Monitoring" more information). Also note that only a subset of the available control bits are presented in the diagram.

Figure 1-11. Perfmon Counter Control Block Diagram



Selecting What To Monitor: The main task of a configuration register is to select the event to be monitored by its respective data counter. Setting the *.ev_sel* and *.umask* fields performs the event selection.

Note: Only the *.ev_sel* is pictured in the previous figure. The *.umask* field is generally used to select subevents of the event. Once the proper subevent combination has been selected, it is passed on to the per Counter EventSel Mux.

Additional control bits used to filter and create information related to the selected Event:

Applying a Threshold to Incoming Events: *.thresh* - since most counters can increment by a value greater than 1, a threshold can be applied to generate an event based on the outcome of the comparison. If *.thresh* is set to a non-zero value, that value is compared against the incoming count for that event in each cycle. If the incoming count is >= the threshold value, then the event count captured in the data register will be incremented by 1.

Using the threshold field to generate additional events can be particularly useful when applied to a queue occupancy count. For example, if a queue is known to contain eight entries, it may be useful to know how often it contains 6 or more entries (i.e. Almost Full) or when it contains 1 or more entries (i.e. Not Empty).

Note: For Intel® Xeon® Processor Scalable Memory Family the *.invert* and *.edge_det* bits follow the threshold comparison in sequence. If a user wishes to apply these bits to events that only increment by 1 per cycle, *thresh* must be set to 0x1.

Inverting the Threshold Comparison: *.invert* - Changes *.thresh* test condition to '<'.

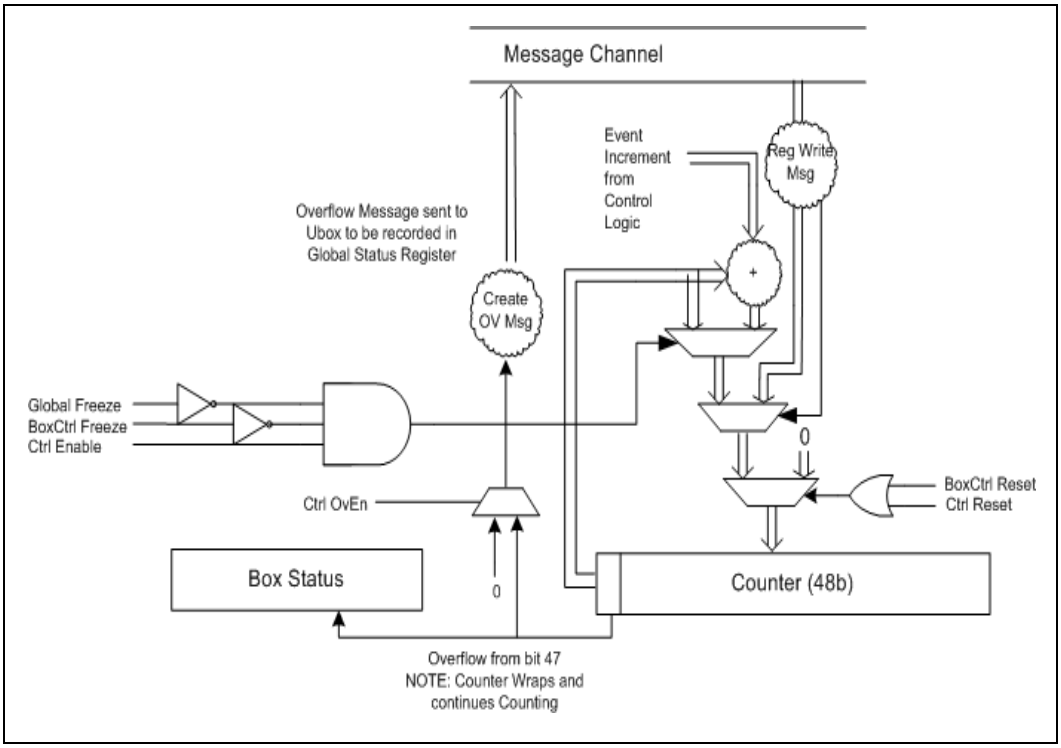
Counting State Transitions Instead of per-Cycle Events: *.edge_det* - Rather than accumulating the raw count each cycle (for events that can increment by 1 per cycle), the register can capture transitions from no event to an event incoming (i.e. the 'Rising Edge').

1.6 Uncore PMON - Typical Counter Logic

Following is a diagram of a standard perfmon counter. It illustrates the control involved in managing the Data Register. Functionality includes how to start/stop the register, reset it, indicate an overflow and capture the information sent from the Counter Control block. The diagram contains bits from all levels in the Counter Control Hierarchy - Global, Unit level as well as from the paired Counter Control register.

Details on how to perform counter management, including how to set up a Monitoring Session and periodically sample the counters can be found in Section 1.9.2, "Setting up a Monitoring Session".

Figure 1-12. Perfmon Counter Block Diagram



Telling HW that the Control Register Is Set: *.en* bit must be set to 1 to enable counting. Once counting has been enabled at all levels of the Performance Monitoring Hierarchy (refer to Section 1.9.2, "Setting up a Monitoring Session" for more information), the paired data register will begin to collect events.

Notification after X events: *.ov_en* - Instead of manually stopping the counters at intervals (often wall clock time) pre-determined by software, hardware can be set to notify monitoring software when a set number of events has occurred. The Overflow Enable bit is provided for just that purpose. See Section 1.4.1.1, "Unit PMON Registers - On Overflow and the Consequences (PMI/Freeze)" for more information on how to use this mechanism.

1.7 Intel® Xeon® Processor Scalable Memory Family's Uncore PMON

The general performance monitoring capabilities of each box are outlined in the following table.

Table 1-8. Per-Box Performance Monitoring Capabilities

Box	# Boxes	# Counters/ Box	# Queue Enabled	Packet Match/ Mask Filters?	Bit Width
CHA	up to 28	4	1	Y	48
IIO	up to 6 between C, P and M flavors	4 (+1) per stack (+4 per port)	0	N	48
IRP	up to 6	2	4	N	48
IMC	up to 2 (each with up to 3 channels)	4 (+1) (per channel)	4	N	48
Intel® UPI	up to 2 (2 or 3 links)	4 (per link)	4	Y	48
M3UPI	up to 2 (2 or 3 links)	3 (per link)	1	N	48
M2M	up to 2	4	1	Y	48
PCU	1	4 (+2)	4	N	48
UBox	1	2 (+1)	0	N	48

The programming interface of the counter registers and control registers fall into two address spaces:

- Accessed by MSR are PMON registers within the CHA units, IIO, IRP, PCU, and U-Box, see Table 1-10.
- Access by PCI device configuration space are PMON registers within the IMC, Intel UPI and M3UPI units, see Table 1-11.

Irrespective of the address-space difference and with only minor exceptions, the bit-granular layout of the control registers to program event code, unit mask, start/stop, and signal filtering via threshold/edge detect are the same.

1.7.1 Querying number of CHAs

Note: The number of CHAs varies with the number of Cores in a system. To determine the number of CHAs, SW should read bits 27:0 in the CAPID6 register located at Device 30, Function 3, Offset 0x9C. These 28 bits form a bit vector of available LLC slices and the CHAs that manage those slices. For example: If bits 27:0 read 0x000F0F, the PMON blocks corresponding to CHAs 0-3 and 8-11 are available and CHAs 4-7 and 12-27 are not available.

1.7.2 Querying number of Intel UPI Links

The number of Intel UPI Links varies according to the specific version of the product. To determine the number of UPI Links, SW should read bits 7:6 in the CAPID4 register located at Device 30, Function 3, Offset 0x94.

00 = 2 Intel UPI Links

01 = 2 Intel UPI Links

10 = 3 Intel UPI Links

1.8 Addressing Uncore PMON State

Following is a list of registers provided in the Intel® Xeon® Processor Scalable Memory Family Server Uncore for Performance Monitoring. The registers are split between MSR space and PCICFG space.

1.8.1 Uncore Performance Monitoring State in MSR space

First off, the Global Control / Status Registers

Table 1-9. Global Performance Monitoring Registers (MSR)

MSR Addresses	Description
0x700	Global Control
0x701	Global Status

As mentioned previously, PMON blocks in the Uncore have some number of paired counter/control (typically 4) registers, a unit status and unit control register (with the exception of the UBox). Many Units may offer extra PMON state such as event Filters or Fixed counters.

The addresses for all basic PMON state addressed through MSR space are laid out in the table below.

Table 1-10. Uncore Performance Monitoring Registers (MSR) (Sheet 1 of 3)

Unit	Unit Status	Unit Ctrl	Ctr3	Ctr2	Ctr1	Ctr0	Ctrl3	Ctrl2	Ctrl1	Ctrl0	Extra	
CHA											Filter1	Filter0
CHA 0	0x0E07	0x0E00	0x0E0B	0x0E0A	0x0E09	0x0E08	0x0E04	0x0E03	0x0E02	0x0E01	0x0E06	0x0E05
CHA 1	0x0E17	0x0E10	0x0E1B	0x0E1A	0x0E19	0x0E18	0x0E14	0x0E13	0x0E12	0x0E11	0x0E16	0x0E15

Table 1-10. Uncore Performance Monitoring Registers (MSR) (Sheet 2 of 3)

Unit	Unit Status	Unit Ctrl	Ctr3	Ctr2	Ctr1	Ctr0	Ctrl3	Ctrl2	Ctrl1	Ctrl0	Extra	
CHA 2	0x0E27	0x0E20	0x0E2B	0x0E2A	0x0E29	0x0E28	0x0E24	0x0E23	0x0E22	0x0E21	0x0E26	0x0E25
CHA 3	0x0E37	0x0E30	0x0E3B	0x0E3A	0x0E39	0x0E38	0x0E34	0x0E33	0x0E32	0x0E31	0x0E36	0x0E35
CHA 4	0x0E47	0x0E40	0x0E4B	0x0E4A	0x0E49	0x0E48	0x0E44	0x0E43	0x0E42	0x0E41	0x0E46	0x0E45
CHA 5	0x0E57	0x0E50	0x0E5B	0x0E5A	0x0E59	0x0E58	0x0E54	0x0E53	0x0E52	0x0E51	0x0E56	0x0E55
CHA 6	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64	0x0E64
CHA 7	0x0E77	0x0E70	0x0E7B	0x0E7A	0x0E79	0x0E78	0x0E74	0x0E73	0x0E72	0x0E71	0x0E76	0x0E75
CHA 8	0x0E87	0x0E80	0x0E8B	0x0E8A	0x0E89	0x0E88	0x0E84	0x0E83	0x0E82	0x0E81	0x0E86	0x0E85
CHA 9	0x0E97	0x0E90	0x0E9B	0x0E9A	0x0E99	0x0E98	0x0E94	0x0E93	0x0E92	0x0E91	0x0E96	0x0E95
CHA 10	0x0EA7	0x0EA0	0x0EAB	0x0EAA	0x0EA9	0x0EA8	0x0EA4	0x0EA3	0x0EA2	0x0EA1	0x0EA6	0x0EA5
CHA 11	0x0EB7	0x0EB0	0x0EBB	0x0EBA	0x0EB9	0x0EB8	0x0EB4	0x0EB3	0x0EB2	0x0EB1	0x0EB6	0x0EB5
CHA 12	0x0EC7	0x0EC0	0x0ECB	0x0ECA	0x0EC9	0x0EC8	0x0EC4	0x0EC3	0x0EC2	0x0EC1	0x0EC6	0x0EC5
CHA 13	0x0ED7	0x0ED0	0x0EDB	0x0EDA	0x0ED9	0x0ED8	0x0ED4	0x0ED3	0x0ED2	0x0ED1	0x0ED6	0x0ED5
CHA 14	0x0EE7	0x0EE0	0x0EEB	0x0EEA	0x0EE9	0x0EE8	0x0EE4	0x0EE3	0x0EE2	0x0EE1	0x0EE6	0x0EE5
CHA 15	0x0EF7	0x0EF0	0x0EFB	0x0EFA	0x0EF9	0x0EF8	0x0EF4	0x0EF3	0x0EF2	0x0EF1	0x0EF6	0x0EF5
CHA 16	0x0F07	0x0F00	0x0F0B	0x0F0A	0x0F09	0x0F08	0x0F04	0x0F03	0x0F02	0x0F01	0x0F06	0x0F05
CHA 17	0x0F17	0x0F10	0x0F1B	0x0F1A	0x0F19	0x0F18	0x0F14	0x0F13	0x0F12	0x0F11	0x0F16	0x0F15
CHA 18	0x0F27	0x0F20	0x0F2B	0x0F2A	0x0F29	0x0F28	0x0F24	0x0F23	0x0F22	0x0F21	0x0F26	0x0F25
CHA 19	0x0F37	0x0F30	0x0F3B	0x0F3A	0x0F39	0x0F38	0x0F34	0x0F33	0x0F32	0x0F31	0x0F36	0x0F35
CHA 20	0x0F47	0x0F40	0x0F4B	0x0F4A	0x0F49	0x0F48	0x0F44	0x0F43	0x0F42	0x0F41	0x0F46	0x0F45
CHA 21	0x0F57	0x0F50	0x0F5B	0x0F5A	0x0F59	0x0F58	0x0F54	0x0F53	0x0F52	0x0F51	0x0F56	0x0F55
CHA 22	0x0F67	0x0F60	0x0F6B	0x0F6A	0x0F69	0x0F68	0x0F64	0x0F63	0x0F62	0x0F61	0x0F66	0x0F65
CHA 23	0x0F77	0x0F70	0x0F7B	0x0F7A	0x0F79	0x0F78	0x0F74	0x0F73	0x0F72	0x0F71	0x0F76	0x0F75
CHA 24	0x0F87	0x0F80	0x0F8B	0x0F8A	0x0F89	0x0F88	0x0F84	0x0F83	0x0F82	0x0F81	0x0F86	0x0F85
CHA 25	0x0F97	0x0F90	0x0F9B	0x0F9A	0x0F99	0x0F98	0x0F94	0x0F93	0x0F92	0x0F91	0x0F96	0x0F95
CHA 26	0x0FA7	0x0FA0	0x0FAB	0x0FAA	0x0FA9	0x0FA8	0x0FA4	0x0FA3	0x0FA2	0x0FA1	0x0FA6	0x0FA5
CHA 27	0x0FB7	0x0FB0	0x0FBB	0x0FBA	0x0FB9	0x0FB8	0x0FB4	0x0FB3	0x0FB2	0x0FB1	0x0FB6	0x0FB5
IIO											Clock	
CBDMA	0x0A47	0x0A40	0x0A44	0x0A43	0x0A42	0x0A41	0x0A4B	0x0A4A	0x0A49	0x0A48	0x0A45	
PCIe 0	0x0A67	0x0A60	0x0A64	0x0A63	0x0A62	0x0A61	0x0A6B	0x0A6A	0x0A69	0x0A68	0x0A65	
PCIe 1	0x0A87	0x0A80	0x0A84	0x0A83	0x0A82	0x0A81	0x0A8B	0x0A8A	0x0A89	0x0A88	0x0A85	
PCIe 2	0x0AA7	0x0AA0	0x0AA4	0x0AA3	0x0AA2	0x0AA1	0x0AAB	0x0AAA	0x0AA9	0x0AA8	0x0AA5	
MCP 0	0x0AC7	0x0AC0	0x0AC4	0x0AC3	0x0AC2	0x0AC1	0x0ACB	0x0ACA	0x0AC9	0x0AC8	0x0AC5	
MCP 1	0x0AE7	0x0AE0	0x0AE4	0x0AE3	0x0AE2	0x0AE1	0x0AEB	0x0AEA	0x0AE9	0x0AE8	0x0AE5	
IRP												
CBDMA	0x0A5F	0x0A58			0x0A5A	0x0A59			0x0A5C	0x0A5B		
PCIe 0	0x0A7F	0x0A78			0x0A7A	0x0A79			0x0A7C	0x0A7B		
PCIe 1	0x0A9F	0x0A98			0x0A9A	0x0A99			0x0A9C	0x0A9B		
PCIe 2	0x0ABF	0x0AB8			0x0ABA	0x0AB9			0x0ABC	0x0ABB		
MCP 0	0x0ADF	0x0AD8			0x0ADA	0x0AD9			0x0ADC	0x0ADB		

Table 1-10. Uncore Performance Monitoring Registers (MSR) (Sheet 3 of 3)

Unit	Unit Status	Unit Ctrl	Ctr3	Ctr2	Ctr1	Ctr0	Ctrl3	Ctrl2	Ctrl1	Ctrl0	Extra	
MCP 1	0x0AFF	0x0AF8			0x0AFA	0x0AF9			0x0AFC	0x0AFB		
PCU											Filter	
PCU	0x0716	0x0710	0x071A	0x0719	0x0718	0x0717	0x0714	0x0713	0x0712	0x0711	0x0715	
UBox											UCLK Ctr	UCLK Ctrl
UBox	0x0708				0x070A	0x0709			0x0706	0x0705	0x0704	0x0703

As of Intel® Xeon® Processor Scalable Memory Family, there are a number of free-running counters in each IIO Stack that collect counts for Input/Output x BW/Utilization for each Port. The MSR addresses used to access that state are detailed in the following two tables.

Table 1-11. Free-Running IIO Bandwidth Counters in MSR space

	Port 3 BW In	Port 2 BW In	Port 1 BW In	Port 0 BW In	Port 3 BW Out	Port 2 BW Out	Port 1 BW Out	Port 0 BW Out
CBDMA	0x0B03	0x0B02	0x0B01	0x0B00	0x0B07	0x0B06	0x0B05	0x0B04
PCIe 0	0x0B13	0x0B12	0x0B11	0x0B10	0x0B17	0x0B16	0x0B15	0x0B14
PCIe 1	0x0B23	0x0B22	0x0B21	0x0B20	0x0B27	0x0B26	0x0B25	0x0B24
PCIe 2	0x0B33	0x0B32	0x0B31	0x0B30	0x0B37	0x0B36	0x0B35	0x0B34
MCP 0	0x0B43	0x0B42	0x0B41	0x0B40	0x0B47	0x0B46	0x0B45	0x0B44
MCP 1	0x0B53	0x0B52	0x0B51	0x0B50	0x0B57	0x0B56	0x0B55	0x0B54

Table 1-12. Free-running IIO Utilization Counters in MSR space

	Port 3 Util In	Port 2 Util In	Port 1 Util In	Port 0 Util In	Port 3 Util Out	Port 2 Util Out	Port 1 Util Out	Port 0 Util Out
CBDMA	0x0B0E	0x0B0C	0x0B0A	0x0B08	0x0B0F	0x0B0D	0x0B0B	0x0B09
PCIe 0	0x0B1E	0x0B1C	0x0B1A	0x0B18	0x0B1F	0x0B1D	0x0B1B	0x0B19
PCIe 1	0x0B2E	0x0B2C	0x0B2A	0x0B28	0x0B2F	0x0B2D	0x0B2B	0x0B29
PCIe 2	0x0B3E	0x0B3C	0x0B3A	0x0B38	0x0B3F	0x0B3D	0x0B3B	0x0B39
MCP 0	0x0B4E	0x0B4C	0x0B4A	0x0B48	0x0B4F	0x0B4D	0x0B4B	0x0B49
MCP 1	0x0B5E	0x0B5C	0x0B5A	0x0B58	0x0B5F	0x0B5D	0x0B5B	0x0B59

Note: Please refer to each Unit's performance monitoring section for any related state not covered here.

1.8.2 Uncore Performance Monitoring State in PCICFG space

The addresses for all basic PMON state addressed through PCICFG space are laid out in the table below. Each such block will have a PCICFG B:D:F and DeviceID. The registers are presented as offsets to the PMON block's base address.

Note: Although a bus number is provided as part of the B:D:F address each PMON block is located, the Bus is configurable by BIOS. It will be necessary to walk the OS provided PCI device list to find the appropriate header and the assigned bus. Some pseudo code is provided to illustrate the process

Table 1-13. Uncore Performance Monitoring Registers (PCICFG)

Unit	Unit Status	Unit Ctrl	Ctr3	Ctr2	Ctr1	Ctr0	Ctrl3	Ctrl2	Ctrl1	Ctrl0	Extra	
Unit - PMON Block			PCICFG B:D:F Base Address					Device ID				
IMC0 - Channel 0					B2:D10:F2			0x2042				
IMC0 - Channel 1					B2:D10:F6			0x2046				
IMC0 - Channel 2					B2:D11:F2			0x204A				
IMC1 - Channel 0					B2:D12:F2			0x2042				
IMC1 - Channel 1					B2:D12:F6			0x2046				
IMC1 - Channel 2					B2:D13:F2			0x204A			DCLK Ctrl	DCLK Ctr
iMC	0xF8	0xF4	0xB8	0xB0	0xA8	0xA0	0xE4	0xE0	0xDC	0xD8	0xF0	0xD0
M2M - for iMC0					B2:D8:F0			0x2066				
M2M - for iMC1					B2:D9:F0			0x2066			See M2M Section	
M2M	0x260	0x258	0x218	0x210	0x208	0x200	0x240	0x238	0x230	0x228		
M3UPI - Link 0					B3:D18:F0			0x204C				
M3UPI - Link 1					B3:D18:F1			0x204D				
M3UPI - Link 2					B3:D18:F4			0x204C				
M3UPI	0xF8	0xF4		0xB0	0xA8	0xA0		0xE0	0xDC	0xD8		
UPI LL - Link 0					B3:D14:F0			0x2058				
UPI LL - Link 1					B3:D15:F0			0x2058				
UPI LL - Link 2					B3:D16:F0			0x2058				
UPI LL	0x37C	0x378	0x330	0x328	0x320	0x318	0x368	0x360	0x358	0x350		

1.9 Some Guidance for SW

1.9.1 On Finding the Package's Bus number for Uncore PMON registers in PCICFG Space

PCI-based uncore units in Intel® Xeon® Processor Scalable Memory Family can be found using bus, device and functions numbers. However, the **busno** has to be found dynamically in each package. The code is embedded below.

First, for each package, it is necessary to read the node ID offset in the Ubox. That needs to match the GID offset of the Ubox in a specific pattern to get the busno for the package. This busno can then be used with the given D:F (device:function) listed with each box's counters that are accessed through PCICfg space (Table 1-8, "Per-Box Performance Monitoring Capabilities").

Note: ed: The one undefined piece in the following code is PCI_Read_Ulong, a function that simply reads the value from the PCI address. This function, or ones like it, can be found in a more general PCI library the composition of which is OS dependent.

Unfortunately, a link to a suitable version of the library was not readily available. Below are links to a comparable open source version of the library. Included for reference:

https://github.com/opcm/pcm/blob/master/ pci.h and pci.cpp

```
#define DRV_IS_PCI_VENDOR_ID_INTEL              0x8086
#define VENDOR_ID_MSASK                         0x0000FFFF
#define DEVICE_ID_MASK                          0xFFFF00000
#define DEVICE_ID_BITSHIFT                      16

#define PCI_ENABLE                              0x80000000
#define FORM_PCI_ADDR(bus,dev,fun,off)          (((PCI_ENABLE))      | \
                                                 ((bus & 0xFF) << 16)| \
                                                 ((dev & 0x1F) << 11)| \
                                                 ((fun & 0x07) << 8) | \
                                                 ((off & 0xFF) << 0))


#define SKYLAKE_SERVER_SOCKETID_UBOX_DID    0x2014

//the below LNID and GID applies to Skylake Server
#define UNC_SOCKETID_UBOX_LNID_OFFSET    0xC0
#define UNC_SOCKETID_UBOX_GID_OFFSET     0xD4

for (bus_no = 0; bus_no < 256; bus_no++) {
  for (device_no = 0; device_no < 32; device_no++) {
   for (function_no = 0; function_no < 8; function_no++) {

     // find bus, device, and function number for socket ID UBOX device
     pci_address = FORM_PCI_ADDR(bus_no, device_no, function_no, 0);
     value = PCI_Read_Ulong(pci_address);

     vendor_id = value & VENDOR_ID_MASK;
     device_id = (value & DEVICE_ID_MASK) >> DEVICE_ID_BITSHIFT;

     if (vendor_id != DRV_IS_PCI_VENDOR_ID_INTEL) {
       continue;
        }
     if (device_id == SKYLAKE_SERVER_SOCKETID_UBOX_DID) {
```

```
        // first get node id for the local socket
        pci_address = FORM_PCI_ADDR(bus_no, device_no, function_no,
                      UNC_SOCKETID_UBOX_LNID_OFFSET);
        nodeid = PCI_Read_Ulong(pci_address) & 0x00000007;

        // Every 3b of the Node ID mapping register maps to a specific node
        // Read the Node ID Mapping Register and find the node that matches
        // the gid read from the Node ID configuration register (above).
        // e.g. Bits 2:0 map to node 0, bits 5:3 maps to package 1, etc.

        pci_address = FORM_PCI_ADDR(bus_no, device_no, function_no,
                      UNC_SOCKETID_UBOX_GID_OFFSET);
        mapping = PCI_Read_Ulong(pci_address);

         for (i = 0; i < 8; i++){
          if (nodeid == ((mapping >> (3 * i)) & 0x7)) {
                gid = i;
                break;
          }
         }

        UNC_UBOX_package_to_bus_map[gid] = bus_no;
       }
     }
   }

}
```

1.9.2 Setting up a Monitoring Session

On HW reset, all the counters are disabled. Enabling is hierarchical. So the following steps, which include programming the event control registers and enabling the counters to begin collecting events, must be taken to set up a monitoring session. Section 1.9.3 covers the steps to stop/re-start counter registers during a monitoring session.

Global Settings in the UBox: (NOTE: Necessary for U-Box monitoring).

a) Freeze all the uncore counters by setting U_MSR_PMON_GLOBAL_CTL.frz_all to 1

OR (if box level freeze control preferred)

a) Freeze the box's counters while setting up the monitoring session.

for example, set Cn_MSR_PMON_BOX_CTL.frz to 1

For each event to be measured within each box:

b) Enable counting for each monitor

for example, Set C0_MSR_PMON_CTL2.en to 1

Note: Recommended: set the *.en* bit for all counters in each box a user intends to monitor, and leave alone for the duration of the monitoring session.

Note: For cases where there is no sharing of these counters among software agents that are independently sampling the counters, software could set the enable bits for all counters it intends to use during the setup phase. For cases where sharing is expected, each agent could use the individual enable bits in order to perform sampling rather than using the box-level freeze from steps (a) and (d).

c) Select event to monitor if the event control register hasn't been programmed:

Program the *.ev_sel* and *.umask* bits in the control register with the encoding necessary to capture the requested event along with any signal conditioning bits (*.thresh*/*.edge_det*/*.invert*) used to qualify the event.

e.g. Set C0_MSR_PMON_CTL2.{ev_sel, umask} to {0x37, 0x1} in order to capture LLC_VICTIMS.M_STATE in CHA 0's C0_MSR_PMON_CTR2.

Note: It is also important to program any additional filter registers used to further qualify the events (e.g. setting the opcode match field in Cn_MSR_BOX_FILTER1 to qualify TOR_INSERTS by a specific opcode).

Back to the box level:

d) Reset counters in each box to ensure no stale values have been acquired from previous sessions. Resetting the control registers, particularly those that won't be used is also recommended if for no other reason than to prevent errant overflows. To reset both the counters and control registers write the following registers:

- For each CHAx, set Cn_MSR_PMON_UNIT_CTL[1:0] to 0x3.
- For each DRAM Channel, set MCn_CHy_PCI_PMON_UNIT_CTL[1:0] to 0x3.
- Set PCU_MSR_PMON_UNIT_CTL[1:0] to 0x3.
- For each Intel® UPI Link, set M3_Ly_PCI_PMON_UNIT_CTL[1:0] to 0x3.
- For each Intel® UPI Link, set UPI_Ly_PCI_PMON_UNIT_CTL[1:0] to 0x3.
- For each IIO stack, set M2n_PCI_PMON_UNIT_CTL[1:0] to 0x3.
- For each IIO stack, set IIOn_MSR_PMON_UNIT_CTL[1:0] to 0x3
- For each IIO stack, set IRPn_MSR_PMON_UNIT_CTL[1:0] to 0x3

Note: The UBox does not have a Unit Control register. The counters will need to be manually reset by writing a 0 in each data register.

e) Select how to gather data. *If polling, skip to f*. If sampling:

To set up a **sample interval**, software can pre-program the data register with a value of [2^(register bit width - up to 48) - sample interval length]. Doing so allows software, through use of the pmi mechanism, to be **notified** when the number of events in the sample have been captured. Capturing a performance monitoring sample every 'X cycles' (the fixed counter in the UBox counts uncore clock cycles) is a common use of this mechanism.

i.e. To stop counting and receive notification when the 1,000,000th data flit is transmitted from Intel UPI on Link 0

— set UPI_L0_PCI_PMON_CTR1 to (2^48- 1000)

— set UPI_L0_PCI_PMON_CTL1.ev_sel to 0x2

— set UPI_L0_PCI_PMON_CTL1.umask to 0xF

— set U_MSR_PMON_GLOBAL_CTL.pmi_core_sel to which core the monitoring thread is executing on.

f) Enable counting at the global level by setting the U_MSR_PMON_GLOBAL_CTL.unfrz_all bit to 1.

OR

f) Enable counting at the box level by unfreezing the counters in each box

e.g. set Cn_MSR_PMON_BOX_CTL.frz to 0

And with that, counting will begin.

Note: The UBox does not have a Unit Control register, so there's no box-level freeze to help isolate the UBox from agents counting in other boxes. Once enabled and programmed with a valid event, the UBox counters will collect events. For somewhat better synchronization, a user can keep the U_MSR_PMON_CTL.ev_sel at 0x0 while enabled and write it with a valid value just prior to unfreezing the registers in other boxes.

1.9.3 Reading the Sample Interval

Software can **poll** the counters whenever it chooses, or wait to be **notified** that a counter has overflowed (by receiving a PMI).

a) **Polling** - before reading, it is recommended that software freeze the counters at either the Global level (U_MSR_PMON_GLOBAL_CTL.frz_all) or in each box with active counters (by setting *_PMON_UNIT_CTL.frz to 1). After reading the event counts from the counter registers, the monitoring agent can choose to reset the event counts to avoid event-count wrap-around; or resume the counter register without resetting their values. The latter choice will require the monitoring agent to check and adjust for potential wrap-around situations.

b) **Frozen** counters - If software set the counters to freeze on overflow and send notification when it happens, the next question is: Who caused the freeze?

Overflow bits are stored hierarchically within the Intel® Xeon® Processor Scalable Memory Family uncore. First, software should read the U_MSR_PMON_GLOBAL_STATUS.ov_* bits to determine which box(es) sent an overflow. Then read that box's *_PMON_GLOBAL_STATUS.ov field to find the overflowing counter.

Note: More than one counter may overflow at any given time.

Note: Certain boxes may have more than one PMON block (e.g. IMC has a PMON block in each Channel). It may be necessary to read all STATUS registers in the box to determine which counter overflowed.

1.9.4 Enabling a New Sample Interval from Frozen Counters

a) **Clear all uncore counters**: For each box in which counting occurred, set *_PMON_BOX_CTL.rst_ctrs to 1.

b) **Clear all overflow bits**. This includes clearing U_MSR_PMON_GLOBAL_STATUS.ov_* as well as any *_BOX_STATUS registers that have their overflow bits set.

e.g. If counter 3 in Intel UPI Link 1 overflowed, software should set UPI_L1_PCI_PMON_BOX_STATUS.ov[3] to 1 to clear the overflow.

c) **Create the next sample**: Reinitialize the sample by setting the monitoring data register to (2^48 - sample_interval). Or set up a new sample interval as outlined in Section 1.9.2, "Setting up a Monitoring Session".

d) **Re-enable counting**: Set U_MSR_PMON_GLOBAL_CTL.unfrz_all to 1.

1.10 On Parsing and Using Derived Events

For many of the sections covering each box's Performance Monitoring capabilities, a set of commonly measured metrics (or 'Derived Events') has been included. For the most part, these derived events are simple mathematical combinations of events found within the box. However, there are some extensions to the notation used by the metrics.

The following is a breakdown of a CHA Derived Event to illustrate a couple of the notations used.

To calculate "Average Number of Data Read Entries that Miss the LLC when the TOR is not empty".

(TOR_OCCUPANCY.MISS_OPCODE / COUNTER0_OCCUPANCY{edge_det,thresh=0x1})) with:Cn_MSR_PMON_BOX_FILTER1.opc=0x182.

First term is a normal Event/Subevent.

Second Term requires setting extra control bits in the register the event has been programmed in:

- event_name[.subevent_name]{ctrl_bit[=value],}
- for example, `COUNTER0_OCCUPANCY{edge_det,thresh=0x1}`

NOTE: If there is no [=value] specified it is assumed that the bit must be set to 1.

Third Term requires programming an extra control register (often for filtering):

- For a single field: with:`Register_Name.field=value1`
- For multiple fields:
  with:`Register_Name.{field1,field2,...}={value1,value2,...}`
- for example,
  `with:Cn_MSR_PMON_BOX_FILTER1.{opc,nid}={0x182,my_node}`

Following is a breakdown of an IMC Derived Event to illustrate a couple more of the notations used.

To calculate "Percent Cycles DRAM Rank x in CKE".

POWER_CKE_CYCLES.RANKx / MC_Chy_PCI_PMON_CTR_FIXED

First Term requires more input to software to determine the specific event/subevent

- In some cases, there may be multiple events/subevents that cover the same information across multiple like hardware units. Rather than manufacturing a derived event for each combination, the derived event will use a lower case variable in the event name.
- for example, `POWER_CKE_CYCLES.RANKx` where 'x' is a variable to cover events `POWER_CKE_CYCLES.RANK0` through `POWER_CKE_CYCLES.RANK7`

Second Term requires reading a fixed data register

- For the case where the metric requires the information contained in a fixed data register, the pnemonic for the register will be included in the equation. Software will

be responsible for configuring the data register and setting it to start counting with the other events used by the metric.

- for example, `MC_Chy_PCI_PMON_CTR_FIXED`

In addition to these formats, some equations require gathering of extra information outside the box (often for common terms):

- See following section for a breakdown of common terms found in Derived Events.

1.10.1 On Common Terms found in Derived Events

To convert a Latency term from a count of clocks to a count of nanoseconds:

- (**Latency Metric**) - {Box}_CLOCKTICKS * (1000 / UNCORE_FREQUENCY)

To convert a Bandwidth term from a count of raw bytes at the operating clock to GB/sec:

- ((**Traffic Metric in Bytes**) / (TOTAL_INTERVAL / (TSC_SPEED * 1000000))) / GB_CONVERSION
- e.g. For READ_MEM_BW, an event derived from IMC:CAS_COUNT.RD * 64, which is the amount of memory bandwidth consumed by read requests, put 'READ_MEM_BW' into the bandwidth term to convert the measurement from raw bytes to GB/sec.

Following are some other terms that may be found within Metrics and how they should be interpreted.

- GB_CONVERSION: 1024^3
- TSC_SPEED: Time Stamp Counter frequency in MHz
- TOTAL_INTERVAL: Overall sample interval (TSC) for the instructions retired event. Typically used to compute a per send metric. Dividing the TOTAL_INTERVAL by CPU_SPEED * 1,000,000 is the number of seconds in the sample interval.
- TOTAL_PROC_CYC: Total number of CPU cycles for a processor event value. Used with processor event data to determine time or work per time as in MB/sec. UPI_LINKS: 2-3 for Intel® Xeon® Processor Scalable Memory Family
- IMC_CHANNELS: Up to 6 for Intel® Xeon® Processor Scalable Memory Family

§

2 Intel® Xeon® Processor Scalable Memory Family Uncore Performance Monitoring

2.1 UBox Performance Monitoring

The UBox serves as the system configuration controller for Intel® Xeon® Processor Scalable Memory Family

In this capacity, the UBox acts as the central unit for a variety of functions:

- The master for reading and writing physically distributed registers across using the Message Channel.
- The UBox is the intermediary for interrupt traffic, receiving interrupts from the system and dispatching interrupts to the appropriate core.
- The UBox serves as the system lock master used when quiescing the platform (e.g., Intel® UPI bus lock).

2.1.1 UBox Performance Monitoring Overview

The UBox supports event monitoring through two programmable 48-bit wide counters (U_MSR_PMON_CTR{1:0}), and a 48-bit fixed counter which increments each U-clock. Each of these counters can be programmed (U_MSR_PMON_CTL{1:0}) to monitor any UBox event.

2.1.2 Additional UBox Performance Monitoring

Note: The UBox is the only PMON block without a separate Unit Control register. The only way to freeze the UBox counters is to use the global freeze.

Figure 2-1. PMON Control Register for UCLK

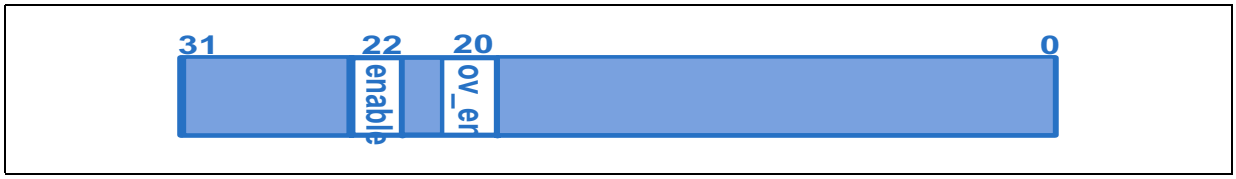


Table 2-1. U_MSR_PMON_FIXED_CTL Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Attr	HW Reset Val	Description
rsv	31:23	RV	0	Reserved
en	22	RW-V	0	Enable counter when global enable is set.

Table 2-1. U_MSR_PMON_FIXED_CTL Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Attr	HW Reset Val	Description
rsv	21	RV	0	Reserved. SW must write to 0 else behavior is undefined.
ov_en	20	RW-V	0	When this bit is set to 1 and the corresponding counter overflows, a UBox overflow message is sent to the UBox's global logic. Once received, the global status register will record the overflow in U_MSR_PMON_GLOBAL_STATUS.ov_u_fixed.
rsv	19:0	RV	0	Reserved

Table 2-2. U_MSR_PMON_FIXED_CTR Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	63:48	RV	0	Reserved
event_count	47:0	RW-V	0	48-bit performance event counter

2.1.3 UBox Performance Monitoring Events

The set of events that can be monitored in the UBox are as follows:

2.1.4 UBox Events Ordered By Code

The following table summarizes the directly measured UBox events.

Symbol Name	Event Code	Ctrs	Extra Select Bit	Max Inc/ Cyc	Description
EVENT_MSG	0x42	0-1	0	0	Message Received
LOCK_CYCLES	0x44	0-1	0	0	IDI Lock/SplitLock Cycles
PHOLD_CYCLES	0x45	0-1	0	0	Cycles PHOLD Assert to Ack
RACU_REQUESTS	0x46	0-1	0	0	RACU Request
RACU_DRNG	0x4c		0	0	

2.1.5 UBox Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the UBox.

EVENT_MSG

- **Title:** Message Received
- **Category:** EVENT_MSG Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Virtual Logical Wire (legacy) message were received from Uncore.

Table 2-3. Unit Masks for EVENT_MSG

Extension	umask [15:8]	Description
VLW_RCVD	bxxxxxxx1	VLW
MSI_RCVD	bxxxxxx1x	MSI Message Signaled Interrupts - interrupts sent by devices (including PCIe via IOxAPIC) (Socket Mode only)
IPI_RCVD	bxxxxx1xx	IPI Inter Processor Interrupts
DOORBELL_RCVD	bxxxx1xxx	
INT_PRIO	bxxx1xxxx	

LOCK_CYCLES

- **Title:** IDI Lock/SplitLock Cycles
- **Category:** LOCK Events
- **Event Code:** 0x44
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Number of times an IDI Lock/SplitLock sequence was started

PHOLD_CYCLES

- **Title:** Cycles PHOLD Assert to Ack
- **Category:** PHOLD Events
- **Event Code:** 0x45
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** PHOLD cycles.

Table 2-4. Unit Masks for PHOLD_CYCLES

Extension	umask [15:8]	Description
ASSERT_TO_ACK	bxxxxxxx1	Assert to ACK

RACU_DRNG

- **Title:**
- **Category:** RACU Events
- **Event Code:** 0x4c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-5. Unit Masks for RACU_DRNG

Extension	umask [15:8]	Description
RDRAND	bxxxxxxx1	
RDSEED	bxxxxxx1x	
PFTCH_BUF_EMPTY	bxxxxx1xx	

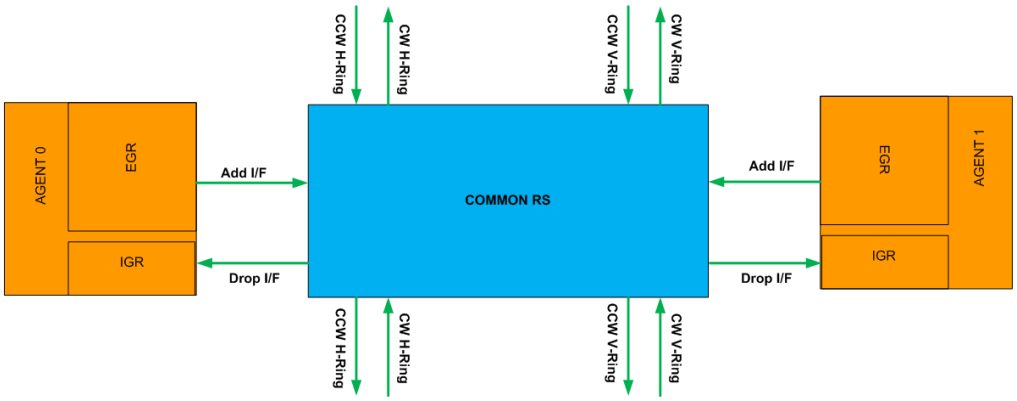
RACU_REQUESTS

- **Title:** RACU Request
- **Category:** RACU Events
- **Event Code:** 0x46
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Number outstanding register requests within message channel tracker

2.2 Mesh Performance Monitoring

For all Boxes that must communicate with the Mesh, there are a common set of events to capture various kinds of information about traffic flowing through their connection to the Mesh. The same encodings are used to request the mesh events in each box.

This common mesh stop event list is available in the CHA, M2M and M3UPI



2.2.1 Mesh Performance Monitoring Events

There are events to track information related to all traffic passing through each box's connection to the Mesh.

- Credit Tracking and Stalls due to lack of credits

  Mesh

2.2.2 CMS Events Ordered By Code

The following table summarizes the directly measured CMS events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
AG0_AD_CRD_ACQUIRED	0x80		0	CMS Agent0 AD Credits Acquired
AG0_AD_CRD_OCCUPANCY	0x82		0	CMS Agent0 AD Credits Occupancy
AG1_AD_CRD_ACQUIRED	0x84		0	CMS Agent1 AD Credits Acquired
AG1_AD_CRD_OCCUPANCY	0x86		0	CMS Agent1 AD Credits Occupancy
AG0_BL_CRD_ACQUIRED	0x88		0	CMS Agent0 BL Credits Acquired
AG0_BL_CRD_OCCUPANCY	0x8a		0	CMS Agent0 BL Credits Occupancy
AG1_BL_CREDITS_ACQUIRED	0x8c		0	CMS Agent1 BL Credits Acquired
AG1_BL_CRD_OCCUPANCY	0x8e		0	CMS Agent1 BL Credits Occupancy
TxR_VERT_OCCUPANCY	0x90		0	CMS Vert Egress Occupancy
TxR_VERT_INSERTS	0x91		0	CMS Vert Egress Allocations
TxR_VERT_CYCLES_FULL	0x92		0	Cycles CMS Vertical Egress Queue Is Full

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
TxR_VERT_CYCLES_NE	0x93		0	Cycles CMS Vertical Egress Queue Is Not Empty
TxR_HORZ_OCCUPANCY	0x94		0	CMS Horizontal Egress Occupancy
TxR_HORZ_INSERTS	0x95		0	CMS Horizontal Egress Inserts
TxR_HORZ_CYCLES_FULL	0x96		0	Cycles CMS Horizontal Egress Queue is Full
TxR_HORZ_CYCLES_NE	0x97		0	Cycles CMS Horizontal Egress Queue is Not Empty
TxR_VERT_NACK	0x98		0	CMS Vertical Egress NACKs
TxR_HORZ_NACK	0x99		0	CMS Horizontal Egress NACKs
TxR_VERT_STARVED	0x9a		0	CMS Vertical Egress Injection Starvation
TxR_HORZ_STARVED	0x9b		0	CMS Horizontal Egress Injection Starvation
TxR_VERT_ADS_USED	0x9c		0	CMS Vertical ADS Used
TxR_HORZ_ADS_USED	0x9d		0	CMS Horizontal ADS Used
TxR_VERT_BYPASS	0x9e		0	CMS Vertical ADS Used
TxR_HORZ_BYPASS	0x9f		0	CMS Horizontal Bypass Used
RING_BOUNCES_VERT	0xa0		0	Messages that bounced on the Vertical Ring.
RING_BOUNCES_HORZ	0xa1		0	Messages that bounced on the Horizontal Ring.
RING_SINK_STARVED_VERT	0xa2		0	Sink Starvation on Vertical Ring
RING_SINK_STARVED_HORZ	0xa3		0	Sink Starvation on Horizontal Ring
RING_SRC_THRTL	0xa4		0	Source Throttle
FAST_ASSERTED	0xa5		0	FaST wire asserted
VERT_RING_AD_IN_USE	0xa6		0	Vertical AD Ring In Use
HORZ_RING_AD_IN_USE	0xa7		0	Horizontal AD Ring In Use
VERT_RING_AK_IN_USE	0xa8		0	Vertical AK Ring In Use
HORZ_RING_AK_IN_USE	0xa9		0	Horizontal AK Ring In Use
VERT_RING_BL_IN_USE	0xaa		0	Vertical BL Ring in Use
HORZ_RING_BL_IN_USE	0xab		0	Horizontal BL Ring in Use
VERT_RING_IV_IN_USE	0xac		0	Vertical IV Ring in Use
HORZ_RING_IV_IN_USE	0xad		0	Horizontal IV Ring in Use
EGRESS_ORDERING	0xae		0	Egress Blocking due to Ordering requirements
RxR_OCCUPANCY	0xb0		0	Transgress Ingress Occupancy
RxR_INSERTS	0xb1		0	Transgress Ingress Allocations
RxR_BYPASS	0xb2		0	Transgress Ingress Bypass
RxR_CRD_STARVED	0xb3		0	Transgress Injection Starvation
RxR_BUSY_STARVED	0xb4		0	Transgress Injection Starvation
CMS_CLOCKTICKS	0xc0		0	CMS Clockticks
STALL_NO_TxR_HORZ_CRD_AD_AG0	0xd0		0	Stall on No AD Agent0 Transgress Credits
STALL_NO_TxR_HORZ_CRD_AD_AG1	0xd2		0	Stall on No AD Agent1 Transgress Credits
STALL_NO_TxR_HORZ_CRD_BL_AG0	0xd4		0	Stall on No BL Agent0 Transgress Credits
STALL_NO_TxR_HORZ_CRD_BL_AG1	0xd6		0	Stall on No BL Agent1 Transgress Credits

2.2.3 CMS Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the CMS.

AG0_AD_CRD_ACQUIRED

- **Title:** CMS Agent0 AD Credits Acquired
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x80
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 0 AD credits acquired in a given cycle, per transgress.
- **NOTE:** If multiple masks are selected, will count the OR of all selected

Table 2-6. Unit Masks for AG0_AD_CRD_ACQUIRED

Extension	umask [15:8]	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

AG0_AD_CRD_OCCUPANCY

- **Title:** CMS Agent0 AD Credits Occupancy
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x82
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 0 AD credits in use in a given cycle, per transgress
- **NOTE:** If multiple masks are selected, will count the SUM of all selected

Table 2-7. Unit Masks for AG0_AD_CRD_OCCUPANCY

Extension	umask [15:8]	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

AG0_BL_CRD_ACQUIRED

- **Title:** CMS Agent0 BL Credits Acquired
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x88
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:** Number of CMS Agent 0 BL credits acquired in a given cycle, per transgress.

Table 2-8. Unit Masks for AG0_BL_CRD_ACQUIRED

Extension	umask [15:8]	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

AG0_BL_CRD_OCCUPANCY

- **Title:** CMS Agent0 BL Credits Occupancy
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x8a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 0 BL credits in use in a given cycle, per transgress

Table 2-9. Unit Masks for AG0_BL_CRD_OCCUPANCY

Extension	umask [15:8]	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

AG1_AD_CRD_ACQUIRED

- **Title:** CMS Agent1 AD Credits Acquired
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x84
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 1 AD credits acquired in a given cycle, per transgress.
- **NOTE:** If multiple masks are selected, will count the OR of all selected

Table 2-10. Unit Masks for AG1_AD_CRD_ACQUIRED (Sheet 1 of 2)

Extension	umask [15:8]	Description
TGR0	bxxxxxxx1	For Transgress 0
TGR1	bxxxxxx1x	For Transgress 1
TGR2	bxxxxx1xx	For Transgress 2
TGR3	bxxxx1xxx	For Transgress 3

Table 2-10. Unit Masks for AG1_AD_CRD_ACQUIRED (Sheet 2 of 2)

Extension	umask [15:8]	Description
TGR4	bxxx1xxxx	For Transgress 4
TGR5	bxx1xxxxx	For Transgress 5

AG1_AD_CRD_OCCUPANCY

- **Title:** CMS Agent1 AD Credits Occupancy
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x86
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 1 AD credits in use in a given cycle, per transgress
- **NOTE:** If multiple masks are selected, will count the SUM of all selected

Table 2-11. Unit Masks for AG1_AD_CRD_OCCUPANCY

Extension	umask [15:8]	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

AG1_BL_CRD_OCCUPANCY

- **Title:** CMS Agent1 BL Credits Occupancy
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x8e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of CMS Agent 1 BL credits in use in a given cycle, per transgress

Table 2-12. Unit Masks for AG1_BL_CRD_OCCUPANCY

Extension	umask [15:8]	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

AG1_BL_CREDITS_ACQUIRED

- **Title:** CMS Agent1 BL Credits Acquired
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0x8c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:** Number of CMS Agent 1 BL credits acquired in a given cycle, per transgress.

Table 2-13. Unit Masks for AG1_BL_CREDITS_ACQUIRED

Extension	umask [15:8]	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

CMS_CLOCKTICKS

- **Title:** CMS Clockticks
- **Category:** Misc Events
- **Event Code:** 0xc0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

EGRESS_ORDERING

- **Title:** Egress Blocking due to Ordering requirements
- **Category:** Horizontal In Use RING Events
- **Event Code:** 0xae
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts number of cycles IV was blocked in the TGR Egress due to SNP/GO Ordering requirements

Table 2-14. Unit Masks for EGRESS_ORDERING

Extension	umask [15:8]	Description
IV_SNOOPGO_UP	bxxxxxxx1	Up
IV_SNOOPGO_DN	bxxxxx1xx	Down

FAST_ASSERTED

- **Title:** FaST wire asserted
- **Category:** Horizontal RING Events
- **Event Code:** 0xa5
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of cycles either the local or incoming distress signals are asserted. Incoming distress includes up, dn and across.

Table 2-15. Unit Masks for FAST_ASSERTED

Extension	umask [15:8]	Description
VERT	b00000001	Vertical
HORZ	b00000010	Horizontal

HORZ_RING_AD_IN_USE

- **Title:** Horizontal AD Ring In Use
- **Category:** Horizontal In Use RING Events
- **Event Code:** 0xa7
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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. We really have two rings -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-16. Unit Masks for HORZ_RING_AD_IN_USE

Extension	umask [15:8]	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

HORZ_RING_AK_IN_USE

- **Title:** Horizontal AK Ring In Use
- **Category:** Horizontal In Use RING Events
- **Event Code:** 0xa9
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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.We really have two rings -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-17. Unit Masks for HORZ_RING_AK_IN_USE

Extension	umask [15:8]	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

HORZ_RING_BL_IN_USE

- **Title:** Horizontal BL Ring in Use
- **Category:** Horizontal In Use RING Events
- **Event Code:** 0xab
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:** 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.We really have two rings -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-18. Unit Masks for HORZ_RING_BL_IN_USE

Extension	umask [15:8]	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

HORZ_RING_IV_IN_USE

- **Title:** Horizontal IV Ring in Use
- **Category:** Horizontal In Use RING Events
- **Event Code:** 0xad
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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. There is only 1 IV ring. Therefore, if one wants to monitor the "Even" ring, they should select both UP_EVEN and DN_EVEN. To monitor the "Odd" ring, they should select both UP_ODD and DN_ODD.

Table 2-19. Unit Masks for HORZ_RING_IV_IN_USE

Extension	umask [15:8]	Description
LEFT	bxxxxxxx1	Left
RIGHT	bxxxxx1xx	Right

RING_BOUNCES_HORZ

- **Title:** Messages that bounced on the Horizontal Ring.
- **Category:** Horizontal RING Events
- **Event Code:** 0xa1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles incoming messages from the Horizontal ring that were bounced, by ring type.

Table 2-20. Unit Masks for RING_BOUNCES_HORZ

Extension	umask [15:8]	Description
AD	bxxxxxxx1	AD
AK	bxxxxxx1x	AK
BL	bxxxxx1xx	BL
IV	bxxxx1xxx	IV

RING_BOUNCES_VERT

- **Title:** Messages that bounced on the Vertical Ring.
- **Category:** Vertical RING Events
- **Event Code:** 0xa0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles incoming messages from the Vertical ring that were bounced, by ring type.

Table 2-21. Unit Masks for RING_BOUNCES_VERT

Extension	umask [15:8]	Description
AD	bxxxxxxx1	AD
AK	bxxxxxx1x	Acknowledgments to core
BL	bxxxxx1xx	Data Responses to core
IV	bxxxx1xxx	Snoops of processor's cache.

RING_SINK_STARVED_HORZ

- **Title:** Sink Starvation on Horizontal Ring
- **Category:** Horizontal RING Events
- **Event Code:** 0xa3
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-22. Unit Masks for RING_SINK_STARVED_HORZ

Extension	umask [15:8]	Description
AD	bxxxxxxx1	AD
AK	bxxxxxx1x	AK
BL	bxxxxx1xx	BL
IV	bxxxx1xxx	IV
AK_AG1	bxx1xxxxx	Acknowledgments to Agent 1

RING_SINK_STARVED_VERT

- **Title:** Sink Starvation on Vertical Ring
- **Category:** Vertical RING Events
- **Event Code:** 0xa2
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-23. Unit Masks for RING_SINK_STARVED_VERT

Extension	umask [15:8]	Description
AD	bxxxxxxx1	AD
AK	bxxxxxx1x	Acknowledgments to core
BL	bxxxxx1xx	Data Responses to core
IV	bxxxx1xxx	Snoops of processor's cache.

RING_SRC_THRTL

- **Title:** Source Throttle
- **Category:** Horizontal RING Events
- **Event Code:** 0xa4
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxR_BUSY_STARVED

- **Title:** Transgress Injection Starvation
- **Category:** CMS Transgress INGRESS Events
- **Event Code:** 0xb4
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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
- **NOTE:** If both masks are selected for one ring type (for example: AD CRD + BNC), will count the OR of the two. Selecting multiple ring types NOT supported

Table 2-24. Unit Masks for RxR_BUSY_STARVED

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

RxR_BYPASS

- **Title:** Transgress Ingress Bypass
- **Category:** CMS Transgress INGRESS Events
- **Event Code:** 0xb2
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of packets bypassing the CMS Ingress
- **NOTE:** If both masks are selected for one ring type (ex: AD CRD + BNC), will count the SUM of the two. Selecting multiple ring types NOT supported

Table 2-25. Unit Masks for RxR_BYPASS

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

RxR_CRD_STARVED

- **Title:** Transgress Injection Starvation
- **Category:** CMS Transgress INGRESS Events
- **Event Code:** 0xb3
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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.
- **NOTE:** If both masks are selected for one ring type (ex: AD CRD + BNC), will count the OR of the two. For this purpose IFV is considered an AK ring type. Selecting multiple ring types NOT supported

Table 2-26. Unit Masks for RxR_CRD_STARVED

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit
IFV	b1xxxxxxx	IFV - Credit

RxR_INSERTS

- **Title:** Transgress Ingress Allocations
- **Category:** CMS Transgress INGRESS Events
- **Event Code:** 0xb1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of allocations into the CMS Ingress The Ingress is used to queue up requests received from the mesh
- **NOTE:** If both masks are selected for one ring type (ex: AD CRD + BNC), will count the SUM of the two. Selecting multiple ring types NOT supported

Table 2-27. Unit Masks for RxR_INSERTS

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

RxR_OCCUPANCY

- **Title:** Transgress Ingress Occupancy
- **Category:** CMS Transgress INGRESS Events
- **Event Code:** 0xb0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Occupancy event for the Ingress buffers in the CMS The Ingress is used to queue up requests received from the mesh
- **NOTE:** If both masks are selected for one ring type (for example: AD CRD + BNC), will count the SUM of the two. Selecting multiple ring types NOT supported

Table 2-28. Unit Masks for RxR_OCCUPANCY

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

STALL_NO_TxR_HORZ_CRD_AD_AG0

- **Title:** Stall on No AD Agent0 Transgress Credits
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0xd0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles the AD Agent 0 Egress Buffer is stalled waiting for a TGR credit to become available, per transgress.

Table 2-29. Unit Masks for STALL_NO_TxR_HORZ_CRD_AD_AG0

Extension	umask [15:8]	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

STALL_NO_TxR_HORZ_CRD_AD_AG1

- **Title:** Stall on No AD Agent1 Transgress Credits
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0xd2
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles the AD Agent 1 Egress Buffer is stalled waiting for a TGR credit to become available, per transgress.

Table 2-30. Unit Masks for STALL_NO_TxR_HORZ_CRD_AD_AG1

Extension	umask [15:8]	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

STALL_NO_TxR_HORZ_CRD_BL_AG0

- **Title:** Stall on No BL Agent0 Transgress Credits
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0xd4
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles the BL Agent 0 Egress Buffer is stalled waiting for a TGR credit to become available, per transgress.

Table 2-31. Unit Masks for STALL_NO_TxR_HORZ_CRD_BL_AG0

Extension	umask [15:8]	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

STALL_NO_TxR_HORZ_CRD_BL_AG1

- **Title:** Stall on No BL Agent1 Transgress Credits
- **Category:** CMS Transgress Credit Events
- **Event Code:** 0xd6
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of cycles the BL Agent 1 Egress Buffer is stalled waiting for a TGR credit to become available, per transgress.

Table 2-32. Unit Masks for STALL_NO_TxR_HORZ_CRD_BL_AG1

Extension	umask [15:8]	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

TxR_HORZ_ADS_USED

- **Title:** CMS Horizontal ADS Used
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x9d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of packets using the Horizontal Anti-Deadlock Slot, broken down by ring type and CMS Agent.

Table 2-33. Unit Masks for TxR_HORZ_ADS_USED

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_BYPASS

- **Title:** CMS Horizontal Bypass Used
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x9f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of packets bypassing the Horizontal Egress, broken down by ring type and CMS Agent.

Table 2-34. Unit Masks for TxR_HORZ_BYPASS

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_CYCLES_FULL

- **Title:** Cycles CMS Horizontal Egress Queue is Full
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x96
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-35. Unit Masks for TxR_HORZ_CYCLES_FULL

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_CYCLES_NE

- **Title:** Cycles CMS Horizontal Egress Queue is Not Empty
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x97
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-36. Unit Masks for TxR_HORZ_CYCLES_NE

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_INSERTS

- **Title:** CMS Horizontal Egress Inserts
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x95
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-37. Unit Masks for TxR_HORZ_INSERTS

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce

Table 2-37. Unit Masks for TxR_HORZ_INSERTS

Extension	umask [15:8]	Description
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_NACK

- **Title:** CMS Horizontal Egress NACKs
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x99
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts number of Egress packets NACK'ed on to the Horizontal Ring

Table 2-38. Unit Masks for TxR_HORZ_NACK

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxx1xxxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_OCCUPANCY

- **Title:** CMS Horizontal Egress Occupancy
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x94
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-39. Unit Masks for TxR_HORZ_OCCUPANCY

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce
AD_CRD	bxxx1xxxx	AD - Credit
BL_CRD	bx1xxxxxx	BL - Credit

TxR_HORZ_STARVED

- **Title:** CMS Horizontal Egress Injection Starvation
- **Category:** CMS Horizontal EGRESS Events
- **Event Code:** 0x9b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:** 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 2-40. Unit Masks for TxR_HORZ_STARVED

Extension	umask [15:8]	Description
AD_BNC	bxxxxxxx1	AD - Bounce
AK_BNC	bxxxxxx1x	AK - Bounce
BL_BNC	bxxxxx1xx	BL - Bounce
IV_BNC	bxxxx1xxx	IV - Bounce

TxR_VERT_ADS_USED

- **Title:** CMS Vertical ADS Used
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x9c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of packets using the Vertical Anti-Deadlock Slot, broken down by ring type and CMS Agent.

Table 2-41. Unit Masks for TxR_VERT_ADS_USED

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0
AK_AG0	bxxxxxx1x	AK - Agent 0
BL_AG0	bxxxxx1xx	BL - Agent 0
AD_AG1	bxxx1xxxx	AD - Agent 1
AK_AG1	bxx1xxxxx	AK - Agent 1
BL_AG1	bx1xxxxxx	BL - Agent 1

TxR_VERT_BYPASS

- **Title:** CMS Vertical ADS Used
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x9e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of packets bypassing the Vertical Egress, broken down by ring type and CMS Agent.

Table 2-42. Unit Masks for TxR_VERT_BYPASS

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0
AK_AG0	bxxxxxx1x	AK - Agent 0
BL_AG0	bxxxxx1xx	BL - Agent 0
IV	bxxxx1xxx	IV
AD_AG1	bxxx1xxxx	AD - Agent 1

Table 2-42. Unit Masks for TxR_VERT_BYPASS

Extension	umask [15:8]	Description
AK_AG1	bxx1xxxxx	AK - Agent 1
BL_AG1	bx1xxxxxx	BL - Agent 1

TxR_VERT_CYCLES_FULL

- **Title:** Cycles CMS Vertical Egress Queue Is Full
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x92
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-43. Unit Masks for TxR_VERT_CYCLES_FULL

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0 Ring transactions from Agent 0 destined for the AD ring. Some example include outbound requests, snoop requests, and snoop responses.
AK_AG0	bxxxxxx1x	AK - Agent 0 Ring transactions from Agent 0 destined for the AK ring. This is commonly used for credit returns and GO responses.
BL_AG0	bxxxxx1xx	BL - Agent 0 Ring transactions from Agent 0 destined for the BL ring. This is commonly used to send data from the cache to various destinations.
IV	bxxxx1xxx	IV Ring transactions from Agent 0 destined for the IV ring. This is commonly used for snoops to the cores.
AD_AG1	bxxx1xxxx	AD - Agent 1 Ring transactions from Agent 1 destined for the AD ring. This is commonly used for outbound requests.
AK_AG1	bxx1xxxxx	AK - Agent 1 Ring transactions from Agent 1 destined for the AK ring.
BL_AG1	bx1xxxxxx	BL - Agent 1 Ring transactions from Agent 1 destined for the BL ring. This is commonly used for transfering writeback data to the cache.

TxR_VERT_CYCLES_NE

- **Title:** Cycles CMS Vertical Egress Queue Is Not Empty
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x93
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-44. Unit Masks for TxR_VERT_CYCLES_NE

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0 Ring transactions from Agent 0 destined for the AD ring. Some example include outbound requests, snoop requests, and snoop responses.
AK_AG0	bxxxxxx1x	AK - Agent 0 Ring transactions from Agent 0 destined for the AK ring. This is commonly used for credit returns and GO responses.
BL_AG0	bxxxxx1xx	BL - Agent 0 Ring transactions from Agent 0 destined for the BL ring. This is commonly used to send data from the cache to various destinations.
IV	bxxxx1xxx	IV Ring transactions from Agent 0 destined for the IV ring. This is commonly used for snoops to the cores.
AD_AG1	bxxx1xxxx	AD - Agent 1 Ring transactions from Agent 1 destined for the AD ring. This is commonly used for outbound requests.
AK_AG1	bxx1xxxxx	AK - Agent 1 Ring transactions from Agent 1 destined for the AK ring.
BL_AG1	bx1xxxxxx	BL - Agent 1 Ring transactions from Agent 1 destined for the BL ring. This is commonly used for transfering writeback data to the cache.

TxR_VERT_INSERTS

- **Title:** CMS Vert Egress Allocations
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x91
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-45. Unit Masks for TxR_VERT_INSERTS

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0 Ring transactions from Agent 0 destined for the AD ring. Some example include outbound requests, snoop requests, and snoop responses.
AK_AG0	bxxxxxx1x	AK - Agent 0 Ring transactions from Agent 0 destined for the AK ring. This is commonly used for credit returns and GO responses.
BL_AG0	bxxxxx1xx	BL - Agent 0 Ring transactions from Agent 0 destined for the BL ring. This is commonly used to send data from the cache to various destinations.
IV	bxxxx1xxx	IV Ring transactions from Agent 0 destined for the IV ring. This is commonly used for snoops to the cores.
AD_AG1	bxxx1xxxx	AD - Agent 1 Ring transactions from Agent 1 destined for the AD ring. This is commonly used for outbound requests.

Table 2-45. Unit Masks for TxR_VERT_INSERTS

Extension	umask [15:8]	Description
AK_AG1	bxx1xxxxx	AK - Agent 1 Ring transactions from Agent 1 destined for the AK ring.
BL_AG1	bx1xxxxxx	BL - Agent 1 Ring transactions from Agent 1 destined for the BL ring. This is commonly used for transferring writeback data to the cache.

TxR_VERT_NACK

- **Title:** CMS Vertical Egress NACKs
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x98
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts number of Egress packets NACK'ed on to the Vertical Ring

Table 2-46. Unit Masks for TxR_VERT_NACK

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0
AK_AG0	bxxxxxx1x	AK - Agent 0
BL_AG0	bxxxxx1xx	BL - Agent 0
IV	bxxxx1xxx	IV
AD_AG1	bxxx1xxxx	AD - Agent 1
AK_AG1	bxx1xxxxx	AK - Agent 1
BL_AG1	bx1xxxxxx	BL - Agent 1

TxR_VERT_OCCUPANCY

- **Title:** CMS Vert Egress Occupancy
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x90
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-47. Unit Masks for TxR_VERT_OCCUPANCY

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0 Ring transactions from Agent 0 destined for the AD ring. Some example include outbound requests, snoop requests, and snoop responses.
AK_AG0	bxxxxxx1x	AK - Agent 0 Ring transactions from Agent 0 destined for the AK ring. This is commonly used for credit returns and GO responses.
BL_AG0	bxxxxx1xx	BL - Agent 0 Ring transactions from Agent 0 destined for the BL ring. This is commonly used to send data from the cache to various destinations.
IV	bxxxx1xxx	IV Ring transactions from Agent 0 destined for the IV ring. This is commonly used for snoops to the cores.

Table 2-47. Unit Masks for TxR_VERT_OCCUPANCY

Extension	umask [15:8]	Description
AD_AG1	bxxx1xxxx	AD - Agent 1 Ring transactions from Agent 1 destined for the AD ring. This is commonly used for outbound requests.
AK_AG1	bxx1xxxxx	AK - Agent 1 Ring transactions from Agent 1 destined for the AK ring.
BL_AG1	bx1xxxxxx	BL - Agent 1 Ring transactions from Agent 1 destined for the BL ring. This is commonly used for transferring writeback data to the cache.

TxR_VERT_STARVED

- **Title:** CMS Vertical Egress Injection Starvation
- **Category:** CMS Vertical EGRESS Events
- **Event Code:** 0x9a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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 2-48. Unit Masks for TxR_VERT_STARVED

Extension	umask [15:8]	Description
AD_AG0	bxxxxxxx1	AD - Agent 0
AK_AG0	bxxxxxx1x	AK - Agent 0
BL_AG0	bxxxxx1xx	BL - Agent 0
IV	bxxxx1xxx	IV
AD_AG1	bxxx1xxxx	AD - Agent 1
AK_AG1	bxx1xxxxx	AK - Agent 1
BL_AG1	bx1xxxxxx	BL - Agent 1

VERT_RING_AD_IN_USE

- **Title:** Vertical AD Ring In Use
- **Category:** Vertical In Use RING Events
- **Event Code:** 0xa6
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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. We really have two rings -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-49. Unit Masks for VERT_RING_AD_IN_USE (Sheet 1 of 2)

Extension	umask [15:8]	Description
UP_EVEN	bxxxxxxx1	Up and Even
UP_ODD	bxxxxxx1x	Up and Odd

Table 2-49. Unit Masks for VERT_RING_AD_IN_USE (Sheet 2 of 2)

Extension	umask [15:8]	Description
DN_EVEN	bxxxxx1xx	Down and Even
DN_ODD	bxxxx1xxx	Down and Odd

VERT_RING_AK_IN_USE

- **Title:** Vertical AK Ring In Use
- **Category:** Vertical In Use RING Events
- **Event Code:** 0xa8
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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.We really have two rings in -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-50. Unit Masks for VERT_RING_AK_IN_USE

Extension	umask [15:8]	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

VERT_RING_BL_IN_USE

- **Title:** Vertical BL Ring in Use
- **Category:** Vertical In Use RING Events
- **Event Code:** 0xaa
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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.We really have two rings -- a clockwise ring and a counter-clockwise ring. On the left side of the ring, the "UP" direction is on the clockwise ring and "DN" is on the counter-clockwise ring. On the right side of the ring, this is reversed. The first half of the CBos are on the left side of the ring, and the 2nd half are on the right side of the ring. In other words (for example), in a 4c part, Cbo 0 UP AD is NOT the same ring as CBo 2 UP AD because they are on opposite sides of the ring.

Table 2-51. Unit Masks for VERT_RING_BL_IN_USE

Extension	umask [15:8]	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

VERT_RING_IV_IN_USE

- **Title:** Vertical IV Ring in Use
- **Category:** Vertical In Use RING Events
- **Event Code:** 0xac
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** 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. There is only 1 IV ring. Therefore, if one wants to monitor the "Even" ring, they should select both UP_EVEN and DN_EVEN. To monitor the "Odd" ring, they should select both UP_ODD and DN_ODD.

Table 2-52. Unit Masks for VERT_RING_IV_IN_USE

Extension	umask [15:8]	Description
UP	bxxxxxxx1	Up
DN	bxxxxx1xx	Down

2.2.4 Caching/Home Agent (CHA) Performance Monitoring

The LLC coherence engine and Home agent (CHA) merges the caching agent and home agent (HA) responsibilities of the chip into a single block. In its capacity as a caching agent the CHA manages the interface between the core the IIO devices and the last level cache (LLC). In its capacity as a home agent the CHA manages the interface between the LLC and the rest of the UPI coherent fabric as well as the on die memory controller.

All core and IIO DMA transactions that access the LLC are directed from their source to a CHA via the mesh interconnect. The CHA is responsible for managing data delivery from the LLC to the requestor and maintaining coherence between the all the cores and IIO devices within the socket that share the LLC. It is also responsible for generating snoops and collecting snoop responses from the local cores when the MESIF protocol requires it.

Similarly, all incoming traffic from remote sockets that maps to the socket's local memory are directed from the UPI link(s) to a CHA via the mesh interconnect. The CHA is responsible for managing the coherence across all sockets in the system for the socket's memory following the protocols defined in the Intel UPI Specification. It manages directory state for the local memory, conflicts, and memory ordering rules for such requests.

In the process of maintaining cache coherency within the socket, and across the system in a multi-socket system, the CHA is the gate keeper for all Intel® UPI Interconnect messages that have addresses mapping to the socket's memory as well as the originator of all Intel® UPI Interconnect messages the originate from cores within its socket when attempts are made to access memory in another socket. It is responsible for ensuring that all Intel® UTI messages that pass through the socket remain coherent.

The CHA can manage a large number of simultaneous requests in parallel, but in order to maintain proper memory ordering it does ensure that whenever multiple incoming requests to the same address are pending (whether they originated from a core or IIO

device within the socket or came in from another socket through one of the UPI links) only one of those requests is being processed at a time.

The Intel® Xeon® Processor Scalable Memory Family uncore contains up to 28 instances of the CHA, each assigned to manage a distinct 1.125MB slice of the processor's total LLC capacity. Since this LLC cache is not inclusive of the IA cores' internal caches, the total cache capacity of the socket is much larger than the LLC capacity (with an additional 1MB of caching capacity for each IA core in the socket), and each CHA is responsible for monitor a portion of that available IA core cache capacity for the purpose of maintaining coherence between the IA core caches and the rest of the UPI coherent fabric.

Every physical memory address in the system is uniquely associated with a single CHA instance via a proprietary hashing algorithm that is designed to keep the distribution of traffic across the CHA instances relatively uniform for a wide range of possible address patterns. This enables the individual CHA instances to operate independently, each managing its slice of the physical address space without any CHA in a given socket ever needing to communicate with the other CHAs in that same socket.

2.2.5 CHA Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each of the CHAs in the Intel® Xeon® Processor Scalable Memory Family uncore supports event monitoring through four 48-bit wide counters (Cn_MSR_PMON_CTR{3:0}). With but a small number of exceptions, each of these counters can be programmed (Cn_MSR_PMON_CTL{3:0}) for any available event.

NOTE: Occupancy Events can only be measured in Counter 0.

CHA counter 0 can increment by a maximum of 20 per cycle; counters 1-3 can increment by 1 per cycle.

Some uncore performance events that monitor transaction activities require additional details that must be programmed in a filter register. Each CHA provides two filter registers and allows only one such event to be programmed at a given time, see Section 2.2.6.2.

2.2.5.1 Special Note on CHA Occupancy Events

Although only counter 0 supports occupancy events, it is possible to program counters 1-3 to monitor the same occupancy event by selecting the "OCCUPANCY_COUNTER0" event code on counters 1-3.

This allows:

Thresholding

2.2.6 Additional CHA Performance Monitoring

2.2.6.1 CHA PMON Counter Control - Difference from Baseline

CHA performance monitoring control registers provide a small amount of additional functionality. The following table defines those cases.

Figure 2-2. CHA Counter Control Register for Intel® Xeon® Processor Scalable Memory Family

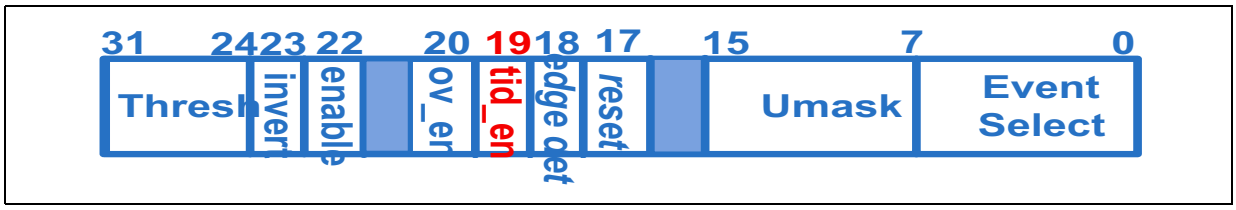


Table 2-53. Cn_MSR_PMON_CTL{3-0} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	21:20	RV	0	Reserved; SW must write to 0 else behavior is undefined.
tid_en	19	RW-V	0	TID Filter Enable
rsv	16	RV	0	Reserved. SW must write to 0 else behavior is undefined.

2.2.6.2 CHA Filter Registers (Cn_MSR_PMON_BOX_FILTER{0,1})

In addition to generic event counting, each CHA provides a pair of FILTER registers that allow a user to filter various traffic as it applies to specific events (see Event Section for more information). LLC_LOOKUP may be filtered by the cacheline state, while TOR_INSERTS and TOR_OCCUPANCY may be filtered by the opcode of the queued request, whether it was serviced locally or remotely, etc.

Any of the CHA events may be filtered by Thread/Core-ID. To do so, the control register's *.tid_en* bit must be set to 1 and the tid field in the FILTER register filled out.

Note: Only one of these filtering criteria may be applied at a time.

Figure 2-3. CHA PMON Filter 0 Register

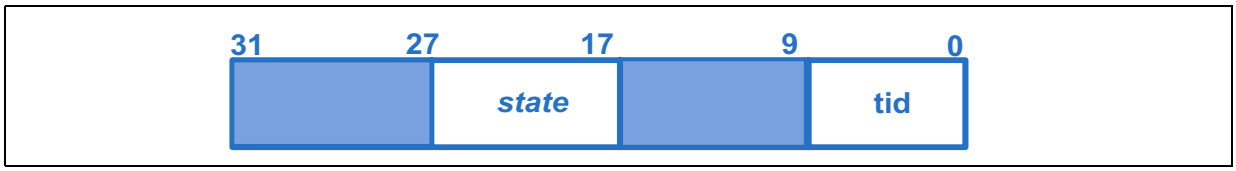


Table 2-54. Cn_MSR_PMON_BOX_FILTER0 Register – Field Definitions

Field	Bits	Atrtr	HW Reset Val	Description
rsv	31:27	RV	0	Reserved SW must set to 0 else behavior is undefined
state	26:17	RW	0	Select state to monitor for **LLC_LOOKUP** event. Setting multiple bits in this field will allow a user to track multiple states. bxx1xxxxxxx - LLC - F state. bxxx1xxxxxx - LLC - M state. bxxxx1xxxxx - LLC - E state. bxxxxx1xxxx - LLC - S state. bxxxxxx1xxx - SF - H state bxxxxxxx1xx - SF - E state. bxxxxxxxx1x - SF - S state. bxxxxxxxxx1 - LLC - I state.
rsv	16:9	RV	0	Reserved SW must set to 0 else behavior is undefined
tid	8:0	0	0	[8:3] Core-ID [2:0] Thread 3-0 When .tid_en is 0; the specified counter will count ALL events. On Intel® Xeon® Processor Scalable Memory Family, 'ALL events' includes traffic from the CPU (e.g. Data Reqeusts) and IIO.

Note: The **default value for FILTER1**, which essentially turns the filter OFF, should be **0x0000003B**. The default value should be 0xXXXXXX33 if SW wants to use the opcode matcher.

Figure 2-4. CHA PMON Filter 1 Register

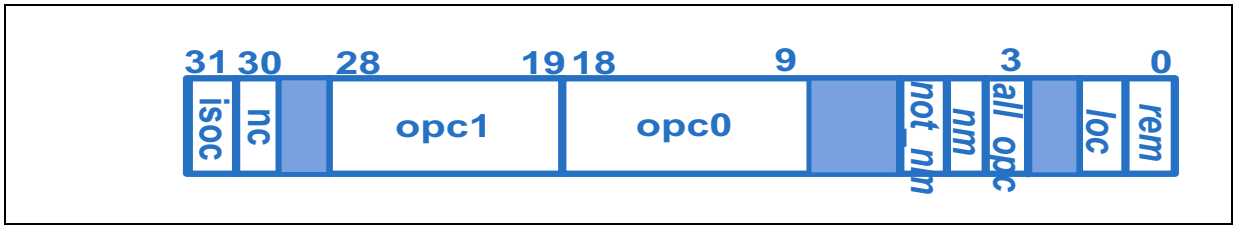


Table 2-55. Cn_MSR_PMON_BOX_FILTER1 Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Atrtr	HW Reset Val	Description
isoc	31	RW	0	Match on ISOC Requests
nc	30	RW	0	Match on Non-Coherent Requests
opc1 (10b IDI Opcode w/top 2b 0x3)	28:19	RW	0	Match on Opcode (see Section 3.1.1, "Reference for CHA Packet Matching") Can be used to track transaction by opocde relevant to each key queue in the CHA pipeline: IPQ, IRQ, RRQ and WBQ
opc0 (10b IDI Opcode w/top 2b 0x3)	18:9	RW	0	Match on Opcode (see Section 3.1.1, "Reference for CHA Packet Matching") Can be used to track transaction by opocde relevant to each key queue in the CHA pipeline: IPQ, IRQ, RRQ and WBQ

Table 2-55. Cn_MSR_PMON_BOX_FILTER1 Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Atrtr	HW Reset Val	Description
rsv	8:6	RV	0	Reserved
not_nm	5	RW	1	Non Near Memory Cacheable
nm	4	RW	1	Near Memory Cacheable
all_opc	3	RW	1	All Opcodes When set to 1 - opcode filtering is disabled When set to 0 - opcode filtering is enabled
rsv	2	RV	0	Reserved
loc	1	RW	1	Match on Local Node Target
rem	0	RW	1	Match on Remote Node Target

2.2.7 CHA Performance Monitoring Events

The performance monitoring events within the CHA include all events internal to the LLC and HA as well as events which track mesh related activity at the CHA/Core mesh stops (see Section 2.2.1, "Mesh Performance Monitoring Events" for the available Mesh Stop events).

CHA performance monitoring events can be used to track LLC access rates, LLC hit/miss rates, LLC eviction and fill rates, HA access rates, HA conflicts, and to detect evidence of back pressure on the internal CHA pipelines. In addition, the CHA has performance monitoring events for tracking MESIF state transitions that occur as a result of data sharing across sockets in a multi-socket system.

Every event in the CHA is from the point of view of the CHA and is not associated with any specific core since all cores in the socket send their LLC transactions to all CHAs in the socket. However, the Intel® Xeon® Processor Scalable Memory Family CHA provides a thread-id field in the Cn_MSR_PMON_BOX_FILTER0 register which can be applied to the CHA events to obtain the interactions between specific cores and threads.

There are separate sets of counters for each CHA instance. For any event, to get an aggregate count of that event for the entire LLC, the counts across the CHA instances must be added together. The counts can be averaged across the CHA instances to get a view of the typical count of an event from the perspective of the individual CHAs. Individual per-CHA deviations from the average can be used to identify hot-spotting across the CHAs or other evidences of non-uniformity in LLC behavior across the CHAs. Such hot-spotting should be rare, though a repetitive polling on a fixed physical address is one obvious example of a case where an analysis of the deviations across the CHAs would indicate hot-spotting.

2.2.7.1 Acronyms frequently used in CHA Events

The Rings:

AD (Address) Ring - Core Read/Write Requests and Intel UPI Snoops. Carries Intel UPI requests and snoop responses from C to Intel UPI.

BL (Block or Data) Ring - Data == 2 transfers for 1 cache line

AK (Acknowledge) Ring - Acknowledges Intel UPI to CHA and CHA to Core. Carries snoop responses from Core to CHA.

IV (Invalidate) Ring - CHA Snoop requests of core caches

2.2.7.2 Key Queues

IRQ - Requests from iA Cores

IPQ - Ingress Probe Queue on AD Ring. Remote socket snoops sent from Intel UPI LL.

ISMQ - Ingress Subsequent Messages (response queue). Associated with message responses to ingress requests (e.g. data responses, Intel UPI completion messages, core snoop response messages and the GO reset queue).

PRQ - Requests from IIO

RRQ - Remote Request Queue. Remote socket read requests, from UPI to the local home agent.

WBQ - Writeback Queue. Remote socket write requests, from UP to the local home agent.

TOR - Table Of Requests. Tracks pending CHA transactions.

RxC (aka IGR) /TxC (aka EGR) - Ingress, requests from Cores (by way of the CMS), and Egress, requests headed for the Mesh (by way of the CMS), queues.

2.2.8 CHA Box Events Ordered By Code

The following table summarizes the directly measured CHA Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CLOCKTICKS	0x00	0-3	0	Uncore Clocks
RxC_OCCUPANCY	0x11	0	0	Ingress (from CMS) Occupancy
RxC_INSERTS	0x13	0-3	0	Ingress (from CMS) Allocations
CORE_PMA	0x17	0-3	0	Core PMA Events
RxC_IRQ0_REJECT	0x18		0	Ingress (from CMS) Request Queue Rejects
RxC_IRQ1_REJECT	0x19		0	Ingress (from CMS) Request Queue Rejects
COUNTER0_OCCUPANCY	0x1f	0-3	0	Counter 0 Occupancy
RxC_PRQ0_REJECT	0x20		0	Ingress (from CMS) Request Queue Rejects
RxC_PRQ1_REJECT	0x21		0	Ingress (from CMS) Request Queue Rejects
RxC_IPQ0_REJECT	0x22		0	Ingress Probe Queue Rejects
RxC_IPQ1_REJECT	0x23		0	Ingress Probe Queue Rejects
RxC_ISMQ0_REJECT	0x24		0	ISMQ Rejects
RxC_ISMQ1_REJECT	0x25		0	ISMQ Rejects
RxC_RRQ0_REJECT	0x26		0	RRQ Rejects
RxC_RRQ1_REJECT	0x27		0	RRQ Rejects
RxC_WBQ0_REJECT	0x28		0	WBQ Rejects
RxC_WBQ1_REJECT	0x29		0	WBQ Rejects
RxC_REQ_Q0_RETRY	0x2a		0	Request Queue Retries

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
RxC_REQ_Q1_RETRY	0x2b		0	Request Queue Retries
RxC_ISMQ0_RETRY	0x2c		0	ISMQ Retries
RxC_ISMQ1_RETRY	0x2d		0	ISMQ Retries
RxC_OTHER0_RETRY	0x2e		0	Other Retries
RxC_OTHER1_RETRY	0x2f		0	Other Retries
XSNP_RESP	0x32	0-3	0	Core Cross Snoop Responses
CORE_SNP	0x33		0	Core Cross Snoops Issued
LLC_LOOKUP	0x34	0-3	0	Cache and Snoop Filter Lookups
TOR_INSERTS	0x35	0-3	0	TOR Inserts
TOR_OCCUPANCY	0x36	0	0	TOR Occupancy
LLC_VICTIMS	0x37	0-3	0	Lines Victimized
MISC	0x39	0-3	0	Cbo Misc
SF_EVICTION	0x3d		0	Snoop Filter Eviction
REQUESTS	0x50	0-3	0	Read and Write Requests
SNOOPS_SENT	0x51		0	Snoops Sent
DIR_LOOKUP	0x53		0	Directory Lookups
DIR_UPDATE	0x54		0	Directory Updates
OSB	0x55		0	OSB Snoop Broadcast
WB_PUSH_MTOI	0x56		0	WbPushMtoI
BYPASS_CHA_IMC	0x57		0	CHA to iMC Bypass
READ_NO_CREDITS	0x58		0	CHA iMC CHNx READ Credits Empty
IMC_READS_COUNT	0x59		0	HA to iMC Reads Issued
WRITE_NO_CREDITS	0x5a		0	CHA iMC CHNx WRITE Credits Empty
IMC_WRITES_COUNT	0x5b		0	CHA to iMC Full Line Writes Issued
SNOOP_RESP	0x5c		0	Snoop Responses Received
SNOOP_RESP_LOCAL	0x5d		0	Snoop Responses Received Local
HITME_LOOKUP	0x5e		0	Counts Number of times HitMe Cache is accessed
HITME_HIT	0x5f		0	Counts Number of Hits in HitMe Cache
HITME_MISS	0x60		0	Counts Number of Misses in HitMe Cache
HITME_UPDATE	0x61		0	Counts the number of Allocate/Update to HitMe Cache
IODC_ALLOC	0x62		0	Counts Number of times IODC entry allocation is attempted
IODC_DEALLOC	0x63		0	Counts number of IODC deallocations

2.2.9 CHA Box Common Metrics (Derived Events)

The following table summarizes metrics commonly calculated from CHA Box events.

Symbol Name: Definition	Equation
AVG_CRD_MISS_LATENCY: Average Latency of Code Reads from an iA Core that miss the LLC	(TOR_OCCUPANCY.IA_MISS / TOR_INSERTS.IA_MISS) with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x259,0x201,1,1,0,1,1}

Symbol Name: Definition	Equation
AVG_DEMAND_RD_HIT_LATENCY: Average Latency of Data Reads that hit the LLC	(TOR_OCCUPANCY.ALL_HIT / (TOR_INSERTS.ALL_HIT with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x202,1,1,0,1,1}
AVG_DEMAND_RD_MISS_LOCAL_LATENCY: Average Latency of Data Reads from an IA Core that miss the LLC and were satisfied by Local Memory	(TOR_OCCUPANCY.IA_MISS / TOR_INSERTS.IA_MISS) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x202,1,1,0,1,0}
AVG_DRD_MISS_LATENCY: Average Latency of Data Reads or Data Read Prefetches from an IA Core that miss the LLC	(TOR_OCCUPANCY.IA_MISS / TOR_INSERTS.IA_MISS) with:Cn_MSR_PMON_BOX_FILTER1.{opc1, opc0,not_nm,nm,all_opc,loc,rem}={0x25A,0x202,1,1,0,1,1}
AVG_IA_CRD_LLC_HIT_LATENCY: Average Latency of Code Reads from an iA Core that miss the LLC	(TOR_OCCUPANCY.IA_HIT / TOR_INSERTS.IA_HIT) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x201,1,1,0,1,1}
AVG_INGRESS_DEPTH: Average Depth of the Ingress Queue through the sample interval	RxC_OCCUPANCY.IRQ / SAMPLE_INTERVAL
AVG_INGRESS_LATENCY: Average Latency of Requests through the Ingress Queue in Uncore Clocks	RxC_OCCUPANCY.IRQ / RxC_INSERTS.IRQ
AVG_INGRESS_LATENCY_WHEN_NE: Average Latency of Requests through the Ingress Queue in Uncore Clocks when Ingress Queue has at least one entry	RxC_OCCUPANCY.IRQ / COUNTER0_OCCUPANCY{edge_det,thresh=0x1}
AVG_RFO_MISS_LATENCY: Average Latency of RFOs from an iA Core that miss the LLC	(TOR_OCCUPANCY.IA_MISS / TOR_INSERTS.IA_MISS) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x200,1,1,0,1,1}
AVG_TOR_DRDS_MISS_WHEN_NE: Average Number of Data Read Entries that Miss the LLC when the TOR is not empty.	(TOR_OCCUPANCY.ALL_MISS / COUNTER0_OCCUPANCY{edge_det,thresh=0x1})) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x202,1,1,0,1,1}
AVG_TOR_DRDS_WHEN_NE: Average Number of Data Read Entries when the TOR is not empty.	(TOR_OCCUPANCY.ALL / COUNTER0_OCCUPANCY{edge_det,thresh=0x1}) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x202,1,1,0,1,1}
CYC_INGRESS_BLOCKED: Cycles the Ingress Request Queue arbiter was Blocked	RxC_EXT_STARVED.IRQ / SAMPLE_INTERVAL
FAST_STR_LLC_HIT: Number of ItoM (fast string) operations that reference the LLC	TOR_INSERTS.IA_HIT with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x248,1,1,0,1,1}
FAST_STR_LLC_MISS: Number of ItoM (fast string) operations that miss the LLC	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x248,1,1,0,1,1}
INGRESS_REJ_V_INS: Ratio of Ingress Request Entries that were rejected vs. inserted	RxC_INSERTS.IRQ_REJECTED / RxC_INSERTS.IRQ
LLC_CRD_MISS_TO_LOC_MEM: LLC Code Read and Code Prefetch misses satisfied by local memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x259,0x201,1,1,0,1,0})
LLC_CRD_MISS_TO_REM_MEM: LLC Code Read and Code Read Prefetch misses satisfied by a remote cache or remote memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x259,0x201,1,1,0,0,1})
LLC_DRD_MISS_PCT:	LLC_LOOKUP.DATA_READ (Cn_MSR_PMON_BOX_FILTER0.state=0x1) / LLC_LOOKUP.DATA_READ (Cn_MSR_PMON_BOX_FILTER0.state=0xF1)
LLC_DRD_MISS_TO_LOC_MEM: LLC Data Read and Data Prefetch misses satisfied by local memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x25A,0x202,1,1,0,1,0})

Symbol Name: Definition	Equation
LLC_DRD_MISS_TO_REM_MEM: LLC Data Read and Data Prefetch misses satisfied by a remote cache or remote memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x25A,0x202,1,1,0,0,1})
LLC_DRD_PREFETCH_HITS:	TOR_INSERTS.IA_HIT with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x25A,1,1,0,1,1})
LLC_DRD_PREFETCH_MISSES:	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x25A,1,1,0,1,1})
LLC_IA_CRD_HITS: LLC Code Read and Code Prefetch misses satisfied by local memory.	TOR_INSERTS.IA_HIT with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x201,1,1,0,1,0})
LLC_MPI: LLC Misses Per Instruction (code, read, RFO and prefetches)	LLC_LOOKUP.ANY (Cn_MSR_PMON_BOX_FILTER0.state=0x1) / INST_RETIRED.ALL (on Core)
LLC_PCIE_DATA_BYTES: LLC write miss (disk/network reads) bandwidth in MB	TOR_INSERTS.IO with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x248,1,1,0,1,1} * 64
LLC_RFO_MISS_PCT: LLC RFO Miss Ratio	(TOR_INSERTS.ALL_MISS / TOR_INSERTS.ALL) with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x200,1,1,0,1,1}
LLC_RFO_MISS_TO_LOC_MEM: LLC RFO and RFO Prefetch misses satisfied by local memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x258,0x200,1,1,0,1,0})
LLC_RFO_MISS_TO_REM_MEM: LLC RFO and RFO Prefetch misses satisfied by a remote cache or remote memory.	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc1,opc0,not_nm,nm,all_opc,loc,rem}={0x258,0x200,1,1,0,0,1})
LLC_RFO_PREFETCH_HITS:	TOR_INSERTS.IA_HIT with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x258,1,1,0,1,1})
LLC_RFO_PREFETCH_MISSES:	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x258,1,1,0,1,1})
MMIO_READ_BW: IO Read Bandwidth in MB - Disk or Network Reads	(TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{nc,opc0,not_nm,nm,all_opc,loc,rem}={1,0x20E,1,1,0,1,1} * 64 / 1000000
MMIO_WRITE_BW: IO Write Bandwidth in MB - Disk or Network Writes	(TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{nc,opc0,not_nm,nm,all_opc,loc,rem}={1,0x20F,1,1,0,1,1} * 64 / 1000000
PCIE_FULL_WRITES: Number of full PCI writes	TOR_INSERTS.IO with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x248,1,1,0,1,1}
PCI_PARTIAL_WRITES: Number of partial PCI writes	TOR_INSERTS.IO with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x200,1,1,0,1,1}
PCI_READS: Number of PCI reads	TOR_INSERTS.IO with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x21E,1,1,0,1,1}
PCT_RD_REQUESTS: Percentage of HA traffic that is from Read Requests	REQUESTS.READS / (REQUESTS.READS + REQUESTS.WRITES)
PCT_WR_REQUESTS: Percentage of HA traffic that is from Write Requests	REQUESTS.WRITES / (REQUESTS.READS + REQUESTS.WRITES)
STREAMED_FULL_STORES: Number of Streamed Store (of Full Cache Line) Transactions	TOR_INSERTS.IA with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x20C,1,1,0,1,1}

Symbol Name: Definition	Equation
STREAMED_PART_STORES: Number of Streamed Store (of Partial Cache Line) Transactions	TOR_INSERTS.IA with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x20D,1,1,0,1,1}
UC_READS: Uncachable Read Transactions	TOR_INSERTS.IA_MISS with:Cn_MSR_PMON_BOX_FILTER1.{opc0,not_nm,nm,all_opc,loc,rem}={0x207,1,1,0,1,1}

2.2.10 CHA Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the CHA Box.

BYPASS_CHA_IMC

- **Title:** CHA to iMC Bypass
- **Category:** HA BYPASS Events
- **Event Code:** 0x57
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of times when the CHA was able to bypass HA pipe on the way to iMC. This is a latency optimization for situations when there is light loadings on the memory subsystem. This can be filtered by when the bypass was taken and when it was not.

Table 2-56. Unit Masks for BYPASS_CHA_IMC

Extension	umask [15:8]	Description
TAKEN	bxxxxxxx1	Taken Filter for transactions that succeeded in taking the full bypass.
INTERMEDIATE	bxxxxxx1x	Intermediate bypass Taken Filter for transactions that succeeded in taking the intermediate bypass.
NOT_TAKEN	bxxxxx1xx	Not Taken Filter for transactions that could not take the bypass, and issues a read to memory. Note that transactions that did not take the bypass but did not issue read to memory will not be counted.

CLOCKTICKS

- **Title:** Uncore Clocks
- **Category:** UCLK Events
- **Event Code:** 0x00
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

CORE_PMA

- **Title:** Core PMA Events
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x17
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-57. Unit Masks for CORE_PMA

Extension	umask [15:8]	Description
C1_STATE	bxxxxxxx1	C1 State
C1_TRANSITION	bxxxxxx1x	C1 Transition
C6_STATE	bxxxxx1xx	C6 State
C6_TRANSITION	bxxxx1xxx	C6 Transition
GV	bxxx1xxxx	GV

CORE_SNP

- **Title:** Core Cross Snoops Issued
- **Category:** ISMQ Events
- **Event Code:** 0x33
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of transactions that trigger a configurable number of cross snoops. Cores are snooped if the transaction looks up the cache and determines that it is necessary based on the operation type and what CoreValid bits are set. For example, if 2 CV bits are set on a data read, the cores must have the data in S state so it is not necessary to snoop them. However, if only 1 CV bit is set the core my have modified the data. If the transaction was an RFO, it would need to invalidate the lines. This event can be filtered based on who triggered the initial snoop(s).

Table 2-58. Unit Masks for CORE_SNP

Extension	umask [15:8]	Description
EXT_ONE	b00100001	Single External Snoops
EXT_GTONE	b00100010	Multiple External Snoops
EXT_REMOTE	b00100100	External Snoop to Remote Node
CORE_ONE	b01000001	Single Core Requests
CORE_GTONE	b01000010	Multiple Core Requests
CORE_REMOTE	b01000100	Core Request to Remote Node
EVICT_ONE	b10000001	Single Eviction
EVICT_GTONE	b10000010	Multiple Eviction
EVICT_REMOTE	b10000100	Eviction to Remote Node
ANY_ONE	b11100001	Any Single Snoop
ANY_GTONE	b11100010	Any Cycle with Multiple Snoops
ANY_REMOTE	b11100100	Any Snoop to Remote Node

COUNTER0_OCCUPANCY

- **Title:** Counter 0 Occupancy
- **Category:** OCCUPANCY Events
- **Event Code:** 0x1f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Since occupancy counts can only be captured in the Cbo's 0 counter, this event allows a user to capture occupancy related information by filtering the Cb0 occupancy count captured in Counter 0. The filtering available is found in the control register - threshold, invert and edge detect. E.g. setting threshold to 1 can effectively monitor how many cycles the monitored queue has an entry.

DIR_LOOKUP

- **Title:** Directory Lookups
- **Category:** HA DIRECTORY Events
- **Event Code:** 0x53
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of transactions that looked up the Home Agent directory. Can be filtered by requests that had to snoop and those that did not have to.
- **NOTE:** Only valid for parts that implement the Directory

Table 2-59. Unit Masks for DIR_LOOKUP

Extension	umask [15:8]	Description
SNP	bxxxxxxx1	Snoop Needed Filters for transactions that had to send one or more snoops because the directory was not clean.
NO_SNP	bxxxxxx1x	Snoop Not Needed Filters for transactions that did not have to send any snoops because the directory was clean.

DIR_UPDATE

- **Title:** Directory Updates
- **Category:** HA DIRECTORY Events
- **Event Code:** 0x54
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of directory updates that were required. These result in writes to the memory controller.
- **NOTE:** Only valid for parts that implement the Directory. Note that any directory update that's a part of an explicit (eviction or core/IO) or an implicit (snoop response) writeback is not counted.

Table 2-60. Unit Masks for DIR_UPDATE

Extension	umask [15:8]	Description
HA	bxxxxxxx1	from HA pipe
TOR	bxxxxxx1x	from TOR pipe

HITME_HIT

- **Title:** Counts Number of Hits in HitMe Cache
- **Category:** HA HitME Events
- **Event Code:** 0x5f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-61. Unit Masks for HITME_HIT

Extension	umask [15:8]	Description
EX_RDS	bxxxxxxx1	Exclusive hit and op is RdCode, RdData, RdDataMigratory, RdCur, RdInv*, Inv*
SHARED_OWNREQ	bxxxxx1xx	Shared hit and op is RdInvOwn, RdInv, Inv*

Table 2-61. Unit Masks for HITME_HIT

Extension	umask [15:8]	Description
WBMTOE	bxxxx1xxx	op is WbMtoE
WBMTOI_OR_S	bxxx1xxxx	op is WbMtoI, WbPushMtoI, WbFlush, or WbMtoS

HITME_LOOKUP

- **Title:** Counts Number of times HitMe Cache is accessed
- **Category:** HA HitME Events
- **Event Code:** 0x5e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-62. Unit Masks for HITME_LOOKUP

Extension	umask [15:8]	Description
READ	bxxxxxxx1	op is RdCode, RdData, RdDataMigratory, RdCur, RdInvOwn, RdInv, Inv*
WRITE	bxxxxxx1x	op is WbMtoE, WbMtoI, WbPushMtoI, WbFlush, or WbMtoS

HITME_MISS

- **Title:** Counts Number of Misses in HitMe Cache
- **Category:** HA HitME Events
- **Event Code:** 0x60
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-63. Unit Masks for HITME_MISS

Extension	umask [15:8]	Description
SHARED_RDINVOWN	bxx1xxxxx	SF/LLC HitS/F and op is RdInvOwn
NOTSHARED_RDINVOWN	bx1xxxxxx	No SF/LLC HitS/F and op is RdInvOwn
READ_OR_INV	b1xxxxxxx	op is RdCode, RdData, RdDataMigratory, RdCur, RdInv, Inv*

HITME_UPDATE

- **Title:** Counts the number of Allocate/Update to HitMe Cache
- **Category:** HA HitME Pipe Events
- **Event Code:** 0x61
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-64. Unit Masks for HITME_UPDATE

Extension	umask [15:8]	Description
DEALLOCATE_RSPFWDI_LOC	bxxxxxxx1	op is RspIFwd or RspIFwdWb for a local request Received RspFwdI* for a local request, but converted HitME SF entry
RSPFWDI_REM	bxxxxxx1x	op is RspIFwd or RspIFwdWb for a remote request Updated HitME RspFwdI* or local HitM/E received for a remote request
SHARED	bxxxxx1xx	Update HitMe Cache to SHARed
RDINVOWN	bxxxx1xxx	Update HitMe Cache on RdInvOwn even if not RspFwdI*
DEALLOCATE	bxxx1xxxx	Deallocate HtiME Reads without RspFwdI*

IMC_READS_COUNT

- **Title:** HA to iMC Reads Issued
- **Category:** HA READ WRITE Events
- **Event Code:** 0x59
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Count of the number of reads issued to any of the memory controller channels. This can be filtered by the priority of the reads.
- **NOTE:** To match the number of reads seen at the IMC, it's necessary to account for any bypasses. IMC_READS_COUNT.* + BYPASS_CHA_IMC.TAKEN == CAS_COUNT.RD

Table 2-65. Unit Masks for IMC_READS_COUNT

Extension	umask [15:8]	Description
NORMAL	bxxxxxxx1	Normal
PRIORITY	bxxxxxx1x	ISOCH

IMC_WRITES_COUNT

- **Title:** CHA to iMC Full Line Writes Issued
- **Category:** HA READ WRITE Events
- **Event Code:** 0x5b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the total number of full line writes issued from the HA into the memory controller. This counts for all four channels. It can be filtered by full/partial and ISOCH/non-ISOCH.
- **NOTE:** Directory bits are stored in memory. Remote socket RFOs will result in a directory update which, in turn, will cause a write command.

Table 2-66. Unit Masks for IMC_WRITES_COUNT

Extension	umask [15:8]	Description
FULL	bxxxxxxx1	Full Line Non-ISOCH
PARTIAL	bxxxxxx1x	Partial Non-ISOCH
FULL_PRIORITY	bxxxxx1xx	ISOCH Full Line
PARTIAL_PRIORITY	bxxxx1xxx	ISOCH Partial

Table 2-66. Unit Masks for IMC_WRITES_COUNT

Extension	umask [15:8]	Description
FULL_MIG	bxxx1xxxx	Full Line MIG
PARTIAL_MIG	bxx1xxxxx	Partial MIG Filter for memory controller 5 only.

IODC_ALLOC

- **Title:** Counts Number of times IODC entry allocation is attempted
- **Category:** HA IODC Events
- **Event Code:** 0x62
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-67. Unit Masks for IODC_ALLOC

Extension	umask [15:8]	Description
INVITOM	bxxxxxxx1	Number of IODC allocations
IODCFULL	bxxxxxx1x	Number of IODC allocations dropped due to IODC Full
OSBGATED	bxxxxx1xx	Number of IDOC allocation dropped due to OSB gate

IODC_DEALLOC

- **Title:** Counts number of IODC deallocations
- **Category:** HA IODC Events
- **Event Code:** 0x63
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-68. Unit Masks for IODC_DEALLOC

Extension	umask [15:8]	Description
WBMTOE	bxxxxxxx1	IODC deallocated due to WbMtoE
WBMTOI	bxxxxxx1x	IODC deallocated due to WbMtoI
WBPUSHMTOI	bxxxxx1xx	IODC deallocated due to WbPushMtoI Moved to Cbo section
SNPOUT	bxxxx1xxx	IODC deallocated due to conflicting transaction
ALL	bxxx1xxxx	IODC deallocated due to any reason

LLC_LOOKUP

- **Title:** Cache and Snoop Filter Lookups
- **Category:** CACHE Events
- **Event Code:** 0x34
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times the LLC was accessed - this includes 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. CHA-Filter0[24:21,17] bits correspond to [FMESI] state.

- **NOTE:** Bit 0 of the umask must always be set for this event. This allows us to match against a given state (or states) as programmed in the Cn_MSR_PMON_BOX_FILTER0.state field bitmask. 0 = I (miss), 4 = S, 5 = E, 6 = M, 7 = F. For example, if you wanted to monitor F and S hits, you could set 00001001b in the 8-bit state field. To monitor any lookup, set the field to 0x1F. Extra note - it may be a little confusing for customers of earlier products. With the CBo and HA functionality combined, it's possible to also measure Snoop Filter lookups with bits 1-3 of the FILTER0.state field

Table 2-69. Unit Masks for LLC_LOOKUP

Extension	umask [15:8]	Filter Dep	Description
DATA_READ	b00000011	CHAFilter0[26:17]	Data Read Request Read transactions
WRITE	b00000101	CHAFilter0[26:17]	Write Requests Writeback transactions from L2 to the LLC This includes all write transactions -- both Cachable and UC.
REMOTE_SNOOP	b00001001	CHAFilter0[26:17]	External Snoop Request Filters for only snoop requests coming from the remote socket(s) through the IPQ.
ANY	b00010001	CHAFilter0[26:17]	Any Request Filters for any transaction originating from the IPQ or IRQ. This does not include lookups originating from the ISMQ.
LOCAL	b00110001	CHAFilter0[26:17]	Local
REMOTE	b10010001	CHAFilter0[26:17]	Remote

LLC_VICTIMS

- **Title:** Lines Victimized
- **Category:** CACHE Events
- **Event Code:** 0x37
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of lines that were victimized on a fill. This can be filtered by the state that the line was in.
- **NOTE:** Does not include evict cleans

Table 2-70. Unit Masks for LLC_VICTIMS

Extension	umask [15:8]	Description
LOCAL_M	b00100001	Local - Lines in M State
LOCAL_E	b00100010	Local - Lines in E State
LOCAL_S	b00100100	Local - Lines in S State
LOCAL_F	b00101000	Local - Lines in F State
LOCAL_ALL	b00101111	Local - All Lines
REMOTE_M	b10000001	Remote - Lines in M State
REMOTE_E	b10000010	Remote - Lines in E State
REMOTE_S	b10000100	Remote - Lines in S State
REMOTE_F	b10001000	Remote - Lines in F State
REMOTE_ALL	b10001111	Remote - All Lines

Table 2-70. Unit Masks for LLC_VICTIMS

Extension	umask [15:8]	Description
TOTAL_M	b10100001	Lines in M State
TOTAL_E	b10100010	Lines in E State
TOTAL_S	b10100100	Lines in S State
TOTAL_F	b10101000	Lines in F State

MISC

- **Title:** Cbo Misc
- **Category:** MISC Events
- **Event Code:** 0x39
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Miscellaneous events in the Cbo.

Table 2-71. Unit Masks for MISC

Extension	umask [15:8]	Description
RSPI_WAS_FSE	bxxxxxxx1	Silent Snoop Eviction Counts the number of times when a Snoop hit in FSE states and triggered a silent eviction. This is useful because this information is lost in the PRE encodings.
WC_ALIASING	bxxxxxx1x	Write Combining Aliasing Counts the number of times that a USWC write (WCIL(F)) transaction hit in the LLC in M state, triggering a WBMtoI followed by the USWC write. This occurs when there is WC aliasing.
RFO_HIT_S	bxxxx1xxx	RFO HitS Number of times that an RFO hit in S state. This is useful for determining if it might be good for a workload to use RspIWB instead of RspSWB.
CV0_PREF_VIC	bxxx1xxxx	CV0 Prefetch Victim
CV0_PREF_MISS	bxx1xxxxx	CV0 Prefetch Miss

OSB

- **Title:** OSB Snoop Broadcast
- **Category:** HA OSB Events
- **Event Code:** 0x55
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Count of OSB snoop broadcasts. Counts by 1 per request causing OSB snoops to be broadcast. Does not count all the snoops generated by OSB.

READ_NO_CREDITS

- **Title:** CHA iMC CHNx READ Credits Empty
- **Category:** HA READ WRITE Events
- **Event Code:** 0x58
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of times when there are no credits available for sending reads from the CHA into the iMC. In order to send reads into the memory controller, the HA must first acquire a credit for the iMC's AD Ingress queue.

Table 2-72. Unit Masks for READ_NO_CREDITS

Extension	umask [15:8]	Description
MC0_SMI0	bxxxxxxx1	MC0_SMI0 Filter for memory controller 0 only.
MC1_SMI1	bxxxxxx1x	MC1_SMI1 Filter for memory controller 1 only.
EDC0_SMI2	bxxxxx1xx	EDC0_SMI2 Filter for memory controller 2 only.
EDC1_SMI3	bxxxx1xxx	EDC1_SMI3 Filter for memory controller 3 only.
EDC2_SMI4	bxxx1xxxx	EDC2_SMI4 Filter for memory controller 4 only.
EDC3_SMI5	bxx1xxxxx	EDC3_SMI5 Filter for memory controller 5 only.

REQUESTS

- **Title:** Read and Write Requests
- **Category:** HA REQUEST Events
- **Event Code:** 0x50
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the total number of read requests made into the Home Agent. Reads include all read opcodes (including RFO). Writes include all writes (streaming, evictions, HitM, etc).

Table 2-73. Unit Masks for REQUESTS

Extension	umask [15:8]	Description
READS_LOCAL	bxxxxxxx1	Reads Local
READS_REMOTE	bxxxxxx1x	Reads Remote
READS	b00000011	Reads
WRITES_LOCAL	bxxxxx1xx	Writes Local
WRITES_REMOTE	bxxxx1xxx	Writes Remote
WRITES	b00001100	Writes
INVITOE_LOCAL	bxxx1xxxx	InvalItoE Local
INVITOE_REMOTE	bxx1xxxxx	InvalItoE Remote

RxC_INSERTS

- **Title:** Ingress (from CMS) Allocations
- **Category:** INGRESS Events
- **Event Code:** 0x13
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts number of allocations per cycle into the specified Ingress queue.
- **NOTE:** IRQ_REJECTED should not be Ored with the other umasks.

Table 2-74. Unit Masks for RxC_INSERTS

Extension	umask [15:8]	Description
IRQ	bxxxxxxx1	IRQ
IRQ_REJ	bxxxxxx1x	IRQ Rejected
IPQ	bxxxxx1xx	IPQ
PRQ	bxxx1xxxx	PRQ
PRQ_REJ	bxx1xxxxx	PRQ
RRQ	bx1xxxxxx	RRQ
WBQ	b1xxxxxxx	WBQ

RxC_IPQ0_REJECT

- **Title:** Ingress Probe Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x22
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-75. Unit Masks for RxC_IPQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_IPQ1_REJECT

- **Title:** Ingress Probe Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x23
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-76. Unit Masks for RxC_IPQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim

Table 2-76. Unit Masks for RxC_IPQ1_REJECT

Extension	umask [15:8]	Description
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_IRQ0_REJECT

- **Title:** Ingress (from CMS) Request Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x18
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-77. Unit Masks for RxC_IRQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_IRQ1_REJECT

- **Title:** Ingress (from CMS) Request Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x19
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-78. Unit Masks for RxC_IRQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_ISMQ0_REJECT

- **Title:** ISMQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x24
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the ISMQ had to retry. Transaction pass through the ISMQ as responses for requests that already exist in the Cbo. Some examples include: when data is returned or when snoop responses come back from the cores.

Table 2-79. Unit Masks for RxC_ISMQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_ISMQ0_RETRY

- **Title:** ISMQ Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the ISMQ had to retry. Transaction pass through the ISMQ as responses for requests that already exist in the Cbo. Some examples include: when data is returned or when snoop responses come back from the cores.

Table 2-80. Unit Masks for RxC_ISMQ0_RETRY

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_ISMQ1_REJECT

- **Title:** ISMQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x25
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the ISMQ had to retry. Transaction pass through the ISMQ as responses for requests that already exist in the Cbo. Some examples include: when data is returned or when snoop responses come back from the cores.

Table 2-81. Unit Masks for RxC_ISMQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA

RxC_ISMQ1_RETRY

- **Title:** ISMQ Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the ISMQ had to retry. Transaction pass through the ISMQ as responses for requests that already exist in the Cbo. Some examples include: when data is returned or when snoop responses come back from the cores.

Table 2-82. Unit Masks for RxC_ISMQ1_RETRY

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA

RxC_OCCUPANCY

- **Title:** Ingress (from CMS) Occupancy
- **Category:** INGRESS Events
- **Event Code:** 0x11
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:** Counts number of entries in the specified Ingress queue in each cycle.
- **NOTE:** IRQ_REJECTED should not be Ored with the other umasks.

Table 2-83. Unit Masks for RxC_OCCUPANCY

Extension	umask [15:8]	Description
IRQ	b00000001	IRQ
IPQ	b00000100	IPQ
RRQ	b01000000	RRQ
WBQ	b10000000	WBQ

RxC_OTHER0_RETRY

- **Title:** Other Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Retry Queue Inserts of Transactions that were already in another Retry Q (sub-events encode the reason for the next reject)

Table 2-84. Unit Masks for RxC_OTHER0_RETRY

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_OTHER1_RETRY

- **Title:** Other Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Retry Queue Inserts of Transactions that were already in another Retry Q (sub-events encode the reason for the next reject)

Table 2-85. Unit Masks for RxC_OTHER1_RETRY

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_PRQ0_REJECT

- **Title:** Ingress (from CMS) Request Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x20
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-86. Unit Masks for RxC_PRQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_PRQ1_REJECT

- **Title:** Ingress (from CMS) Request Queue Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-87. Unit Masks for RxC_PRQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	LLC OR SF Way
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_REQ_Q0_RETRY

- **Title:** Request Queue Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** "REQUESTQ" includes: IRQ, PRQ, IPQ, RRQ, WBQ (everything except for ISMQ)

Table 2-88. Unit Masks for RxC_REQ_Q0_RETRY

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0

Table 2-88. Unit Masks for RxC_REQ_Q0_RETRY

Extension	umask [15:8]	Description
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_REQ_Q1_RETRY

- **Title:** Request Queue Retries
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x2b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** "REQUESTQ" includes: IRQ, PRQ, IPQ, RRQ, WBQ (everything except for ISMQ)

Table 2-89. Unit Masks for RxC_REQ_Q1_RETRY

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_RRQ0_REJECT

- **Title:** RRQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x26
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the RRQ (Remote Response Queue) had to retry.

Table 2-90. Unit Masks for RxC_RRQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0

Table 2-90. Unit Masks for RxC_RRQ0_REJECT

Extension	umask [15:8]	Description
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_RRQ1_REJECT

- **Title:** RRQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x27
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the RRQ (Remote Response Queue) had to retry.

Table 2-91. Unit Masks for RxC_RRQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

RxC_WBQ0_REJECT

- **Title:** WBQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x28
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the WBQ (Writeback Queue) had to retry.

Table 2-92. Unit Masks for RxC_WBQ0_REJECT

Extension	umask [15:8]	Description
AD_REQ_VN0	bxxxxxxx1	AD REQ on VN0
AD_RSP_VN0	bxxxxxx1x	AD RSP on VN0
BL_RSP_VN0	bxxxxx1xx	BL RSP on VN0
BL_WB_VN0	bxxxx1xxx	BL WB on VN0
BL_NCB_VN0	bxxx1xxxx	BL NCB on VN0
BL_NCS_VN0	bxx1xxxxx	BL NCS on VN0
AK_NON_UPI	bx1xxxxxx	Non UPI AK Request
IV_NON_UPI	b1xxxxxxx	Non UPI IV Request

RxC_WBQ1_REJECT

- **Title:** WBQ Rejects
- **Category:** INGRESS_RETRY Events
- **Event Code:** 0x29
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a transaction flowing through the WBQ (Writeback Queue) had to retry.

Table 2-93. Unit Masks for RxC_WBQ1_REJECT

Extension	umask [15:8]	Description
ANY0	bxxxxxxx1	ANY0
HA	bxxxxxx1x	HA
LLC_VICTIM	bxxxxx1xx	LLC Victim
SF_VICTIM	bxxxx1xxx	SF Victim
VICTIM	bxxx1xxxx	Victim
LLC_OR_SF_WAY	bxx1xxxxx	Merging these two together to make room for ANY_REJECT_*0
ALLOW_SNP	bx1xxxxxx	Allow Snoop
PA_MATCH	b1xxxxxxx	PhyAddr Match

SF_EVICTION

- **Title:** Snoop Filter Eviction
- **Category:** CACHE Events
- **Event Code:** 0x3d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-94. Unit Masks for SF_EVICTION

Extension	umask [15:8]	Description
M_STATE	bxxxxxxx1	M state
E_STATE	bxxxxxx1x	E state
S_STATE	bxxxxx1xx	S state

SNOOPS_SENT

- **Title:** Snoops Sent
- **Category:** HA REQUEST Events
- **Event Code:** 0x51
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of snoops issued by the HA.

Table 2-95. Unit Masks for SNOOPS_SENT

Extension	umask [15:8]	Description
ALL	bxxxxxxx1	All
LOCAL	bxxxxx1xx	Broadcast or directed Snoops sent for Local Requests Counts the number of broadcast or directed snoops issued by the HA per request. This filter includes only requests coming from the local socket.
REMOTE	bxxxx1xxx	Broadcast or directed Snoops sent for Remote Requests Counts the number of broadcast or directed snoops issued by the HA per request. This filter includes only requests coming from the remote socket.
BCST_LOCAL	bxxx1xxxx	Broadcast snoop for Local Requests Counts the number of broadcast snoops issued by the HA. This filter includes only requests coming from local sockets.
BCST_REMOTE	bxx1xxxxx	Broadcast snoops for Remote Requests Counts the number of broadcast snoops issued by the HA.This filter includes only requests coming from remote sockets.
DIRECT_LOCAL	bx1xxxxxx	Directed snoops for Local Requests Counts the number of directed snoops issued by the HA. This filter includes only requests coming from local sockets.
DIRECT_REMOTE	b1xxxxxxx	Directed snoops for Remote Requests Counts the number of directed snoops issued by the HA. This filter includes only requests coming from remote sockets.

SNOOP_RESP

- **Title:** Snoop Responses Received
- **Category:** HA SNOOP RESPONSE Events
- **Event Code:** 0x5c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the total number of RspI snoop responses received. Whenever a snoops are issued, one or more snoop responses will be returned depending on the topology of the system. In systems larger than 2s, when multiple snoops are returned this will count all the snoops that are received. For example, if 3 snoops were issued and returned RspI, RspS, and RspSFwd; then each of these sub-events would increment by 1.

Table 2-96. Unit Masks for SNOOP_RESP

Extension	umask [15:8]	Description
RSPI	bxxxxxxx1	RspI Filters for snoops responses of RspI. RspI is returned when the remote cache does not have the data, or when the remote cache silently evicts data (such as when an RFO hits non-modified data).
RSPS	bxxxxxx1x	RspS Filters for snoop responses of RspS. RspS is returned when a remote cache has data but is not forwarding it. It is a way to let the requesting socket know that it cannot allocate the data in E state. No data is sent with S RspS.
RSPIFWD	bxxxxx1xx	RspIFwd Filters for snoop responses of RspIFwd. This is returned when a remote caching agent forwards data and the requesting agent is able to acquire the data in E or M states. This is commonly returned with RFO transactions. It can be either a HitM or a HitFE.

Table 2-96. Unit Masks for SNOOP_RESP

Extension	umask [15:8]	Description
RSPSFWD	bxxxx1xxx	RspSFwd Filters for a snoop response of RspSFwd. This is returned when a remote caching agent forwards data but holds on to its currentl copy. This is common for data and code reads that hit in a remote socket in E or F state.
RSP_WBWB	bxxx1xxxx	Rsp*WB Filters for a snoop response of RspIWB or RspSWB. This is returned when a non-RFO request hits in M state. Data and Code Reads can return either RspIWB or RspSWB depending on how the system has been configured. InvItoE transactions will also return RspIWB because they must acquire ownership.
RSP_FWD_WB	bxx1xxxxx	Rsp*Fwd*WB Filters for a snoop response of Rsp*Fwd*WB. This snoop response is only used in 4s systems. It is used when a snoop HITM's in a remote caching agent and it directly forwards data to a requestor, and simultaneously returns data to the home to be written back to memory.
RSPCNFLCTS	bx1xxxxxx	RSPCNFLCT* Filters for snoops responses of RspConflict. This is returned when a snoop finds an existing outstanding transaction in a remote caching agent when it CAMs that caching agent. This triggers conflict resolution hardware. This covers both RspCnflct and RspCnflctWbI.
RSPFWD	b1xxxxxxx	RspFwd Filters for a snoop response of RspFwd to a CA request. This snoop response is only possible for RdCur when a snoop HITM/E in a remote caching agent and it directly forwards data to a requestor without changing the requestor's cache line state.

SNOOP_RESP_LOCAL

- **Title:** Snoop Responses Received Local
- **Category:** HA SNOOP RESPONSE Events
- **Event Code:** 0x5d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of snoop responses received for a Local request

Table 2-97. Unit Masks for SNOOP_RESP_LOCAL

Extension	umask [15:8]	Description
RSPI	bxxxxxxx1	RspI Filters for snoops responses of RspI to local CA requests. RspI is returned when the remote cache does not have the data, or when the remote cache silently evicts data (such as when an RFO hits non-modified data).
RSPS	bxxxxxx1x	RspS Filters for snoop responses of RspS to local CA requests. RspS is returned when a remote cache has data but is not forwarding it. It is a way to let the requesting socket know that it cannot allocate the data in E state. No data is sent with S RspS.
RSPIFWD	bxxxxx1xx	RspIFwd Filters for snoop responses of RspIFwd to local CA requests. This is returned when a remote caching agent forwards data and the requesting agent is able to acquire the data in E or M states. This is commonly returned with RFO transactions. It can be either a HitM or a HitFE.
RSPSFWD	bxxxx1xxx	RspSFwd Filters for a snoop response of RspSFwd to local CA requests. This is returned when a remote caching agent forwards data but holds on to its currentl copy. This is common for data and code reads that hit in a remote socket in E or F state.

Table 2-97. Unit Masks for SNOOP_RESP_LOCAL

Extension	umask [15:8]	Description
RSP_WB	bxxx1xxxx	Rsp*WB Filters for a snoop response of RspIWB or RspSWB to local CA requests. This is returned when a non-RFO request hits in M state. Data and Code Reads can return either RspIWB or RspSWB depending on how the system has been configured. InvItoE transactions will also return RspIWB because they must acquire ownership.
RSP_FWD_WB	bxx1xxxxx	Rsp*FWD*WB Filters for a snoop response of Rsp*Fwd*WB to local CA requests. This snoop response is only used in 4s systems. It is used when a snoop HITM's in a remote caching agent and it directly forwards data to a requestor, and simultaneously returns data to the home to be written back to memory.
RSPFWD	b1xxxxxxx	RspFwd Filters for a snoop response of RspFwd to local CA requests. This snoop response is only possible for RdCur when a snoop HITM/E in a remote caching agent and it directly forwards data to a requestor without changing the requestor's cache line state.

TOR_INSERTS

- **Title:** TOR Inserts
- **Category:** TOR Events
- **Event Code:** 0x35
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of entries successfuly inserted into the TOR that match qualifications specified by the subevent.
- **NOTE:** HW does not strictly OR each subevent. The equation is roughly (IRQ|EVICT|PRQ|IPQ|RRQ|WBQ) & (HIT|MISS). Meaing it is necessary to set one of the queue bits before one can measure .HIT or .MISS. Also note this event is subect to CHA Filter1 which allows a user to opcode match against TOR entries, distinguish those requests satsified locally vs. those that came from a remote node, etc.

Table 2-98. Unit Masks for TOR_INSERTS

Extension	umask [15:8]	Filter Dep	Description
IRQ	bxxxxxxx1	CHAFilter1[31:0]	IRQ
EVICT	bxxxxxx1x	CHAFilter1[31:0]	SF/LLC Evictions TOR allocation occurred as a result of SF/LLC evictions (came from the ISMQ)
PRQ	bxxxxx1xx	CHAFilter1[31:0]	PRQ
IPQ	bxxxx1xxx	CHAFilter1[31:0]	IPQ
HIT	bxxx1xxxx	CHAFilter1[31:0]	Hit (Not a Miss) HITs (hit is defined to be "not a miss" [see below], as a result for any request allocated into the TOR, one of either HIT or MISS must be true)
IA_HIT	b00010001	CHAFilter1[31:0]	Hits from Local iA
IO_HIT	b00010100	CHAFilter1[31:0]	Hits from Local IO
ALL_HIT	b00010101	CHAFilter1[31:0]	Hits from Local

Table 2-98. Unit Masks for TOR_INSERTS

Extension	umask [15:8]	Filter Dep	Description
MISS	bxx1xxxxx	CHAFilter1[31:0]	Miss Misses. (a miss is defined to be any transaction from the IRQ, PRQ, RRQ, IPQ or (in the victim case) the ISMQ, that required the CHA to spawn a new UPI/SMI3 request on the UPI fabric (including UPI snoops and/or any RD/WR to a local memory controller, in the event that the CHA is the home node)). Basically, if the LLC/SF/MLC complex were not able to service the request without involving another agent...it is a miss. If only IDI snoops were required, it is not a miss (that means the SF/MLC complex took care of it).
IA_MISS	b00100001	CHAFilter1[31:0]	Misses from Local iA
IO_MISS	b00100100	CHAFilter1[31:0]	Misses from Local IO
ALL_MISS	b00100101	CHAFilter1[31:0]	Misses from Local
IA	b00110001	CHAFilter1[31:0]	All from Local iA All locally initiated requests from iA Cores
IO	b00110100	CHAFilter1[31:0]	All from Local IO All locally generated IO traffic
ALL_IO_IA	b00110101	CHAFilter1[31:0]	All from Local iA and IO All locally initiated requests

TOR_OCCUPANCY

- **Title:** TOR Occupancy
- **Category:** TOR Events
- **Event Code:** 0x36
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:** For each cycle, this event accumulates the number of valid entries in the TOR that match qualifications specified by the subevent. T
- **NOTE:** HW does not strictly OR each subevent. The equation is roughly (IRQ|EVICT|PRQ|IPQ|RRQ|WBQ) & (HIT|MISS). Meaning it is necessary to set one of the queue bits before one can measure .HIT or .MISS. Also note this event is subject to CHA Filter1 which allows a user to opcode match against TOR entries, distinguish those requests satisfied locally vs. those that came from a remote node, etc.

Table 2-99. Unit Masks for TOR_OCCUPANCY

Extension	umask [15:8]	Filter Dep	Description
IRQ	bxxxxxxx1	CHAFilter1[31:0]	IRQ
EVICT	bxxxxxx1x	CHAFilter1[31:0]	SF/LLC Evictions TOR allocation occurred as a result of SF/LLC evictions (came from the ISMQ)
PRQ	bxxxxx1xx	CHAFilter1[31:0]	PRQ
IPQ	bxxxx1xxx	CHAFilter1[31:0]	IPQ
HIT	bxxx1xxxx	CHAFilter1[31:0]	Hit (Not a Miss) HITs (hit is defined to be "not a miss" [see below], as a result for any request allocated into the TOR, one of either HIT or MISS must be true)

Table 2-99. Unit Masks for TOR_OCCUPANCY

Extension	umask [15:8]	Filter Dep	Description
IA_HIT	b00010001	CHAFilter1[31:0]	Hits from Local iA
IO_HIT	b00010100	CHAFilter1[31:0]	Hits from Local IO
ALL_HIT	b00010111	CHAFilter1[31:0]	Hits from Local
MISS	bxx1xxxxx	CHAFilter1[31:0]	Miss Misses. (a miss is defined to be any transaction from the IRQ, PRQ, RRQ, IPQ or (in the victim case) the ISMQ, that required the CHA to spawn a new UPI/SMI3 request on the UPI fabric (including UPI snoops and/or any RD/WR to a local memory controller, in the event that the CHA is the home node)). Basically, if the LLC/SF/MLC complex were not able to service the request without involving another agent...it is a miss. If only IDI snoops were required, it is not a miss (that means the SF/MLC complex took care of it).
IA_MISS	b00100001	CHAFilter1[31:0]	Misses from Local iA
IO_MISS	b00100100	CHAFilter1[31:0]	Misses from Local IO
ALL_MISS	b00100111	CHAFilter1[31:0]	Misses from Local
IA	b00110001	CHAFilter1[31:0]	All from Local iA All locally initiated requests from iA Cores
IO	b00110100	CHAFilter1[31:0]	All from Local IO All locally generated IO traffic

WB_PUSH_MTOI

- **Title:** WbPushMtoI
- **Category:** HA WBPUSHMTOI Events
- **Event Code:** 0x56
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of times when the CHA was received WbPushMtoI

Table 2-100. Unit Masks for WB_PUSH_MTOI

Extension	umask [15:8]	Description
LLC	bxxxxxxx1	Pushed to LLC Counts the number of times when the CHA was able to push WbPushMToI to LLC
MEM	bxxxxxx1x	Pushed to Memory Counts the number of times when the CHA was unable to push WbPushMToI to LLC (hence pushed it to MEM)

WRITE_NO_CREDITS

- **Title:** CHA iMC CHNx WRITE Credits Empty
- **Category:** HA READ WRITE Events
- **Event Code:** 0x5a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:** Counts the number of times when there are no credits available for sending WRITEs from the CHA into the iMC. In order to send WRITEs into the memory controller, the HA must first acquire a credit for the iMC's BL Ingress queue.

Table 2-101. Unit Masks for WRITE_NO_CREDITS

Extension	umask [15:8]	Description
MC0_SMI0	bxxxxxxx1	MC0_SMI0 Filter for memory controller 0 only.
MC1_SMI1	bxxxxxx1x	MC1_SMI1 Filter for memory controller 1 only.
EDC0_SMI2	bxxxxx1xx	EDC0_SMI2 Filter for memory controller 2 only.
EDC1_SMI3	bxxxx1xxx	EDC1_SMI3 Filter for memory controller 3 only.
EDC2_SMI4	bxxx1xxxx	EDC2_SMI4 Filter for memory controller 4 only.
EDC3_SMI5	bxx1xxxxx	EDC3_SMI5 Filter for memory controller 5 only.

XSNP_RESP

- **Title:** Core Cross Snoop Responses
- **Category:** ISMQ Events
- **Event Code:** 0x32
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of core cross snoops. Cores are snooped if the transaction looks up the cache and determines that it is necessary based on the operation type. This event can be filtered based on who triggered the initial snoop(s): from Evictions, Core or External (i.e. from a remote node) Requests. And the event can be filtered based on the responses: RspX_Fwd/HitY where Y is the state prior to the snoop response and X is the state following.

Table 2-102. Unit Masks for XSNP_RESP

Extension	umask [15:8]	Description
EXT_RSP_HITFSE	b00100001	External RspHitFSE External Request - Response any to Hit F/S/E
EXT_RSPS_FWDFE	b00100010	External RspSFwdFE External Request - Response S to Fwd F/E
EXT_RSPI_FWDFE	b00100100	External RspIFwdFE External Request - Response I to Fwd F/E
EXT_RSPS_FWDM	b00101000	External RspSFwdM External Request - Response S to Fwd M
EXT_RSPI_FWDM	b00110000	External RspIFwdM External Request - Response I to Fwd M
CORE_RSP_HITFSE	b01000001	Core RspHitFSE Core Request - Response any to Hit F/S/E
CORE_RSPS_FWDFE	b01000010	Core RspSFwdFE Core Request - Response S to Fwd F/E
CORE_RSPI_FWDFE	b01000100	Core RspIFwdFE Core Request - Response I to Fwd F/E
CORE_RSPS_FWDM	b01001000	Core RspSFwdM Core Request - Response S to Fwd M

Table 2-102. Unit Masks for XSNP_RESP

Extension	umask [15:8]	Description
CORE_RSPI_FWDM	b01010000	Core RspIFwdM Core Request - Response I to Fwd M
EVICT_RSP_HITFSE	b10000001	Evict RspHitFSE Eviction Request - Response any to Hit F/S/E
EVICT_RSPS_FWDFE	b10000010	Evict RspSFwdFE Eviction Request - Response S to Fwd F/E
EVICT_RSPI_FWDFE	b10000100	Evict RspIFwdFE Eviction Request - Response I to Fwd F/E
EVICT_RSPS_FWDM	b10001000	Evict RspSFwdM Eviction Request - Response S to Fwd M
EVICT_RSPI_FWDM	b10010000	Evict RspIFwdM Eviction Request - Response I to Fwd M
ANY_RSP_HITFSE	b11100001	Any RspHitFSE Any Request - Response any to Hit F/S/E
ANY_RSPS_FWDFE	b11100010	Any RspSFwdFE Any Request - Response S to Fwd F/E
ANY_RSPI_FWDFE	b11100100	Any RspIFwdFE Any Request - Response I to Fwd F/E
ANY_RSPS_FWDM	b11101000	Any RspSFwdM Any Request - Response S to Fwd M

2.3 Memory Controller (IMC) Performance Monitoring

The Intel® Xeon® Processor Scalable Memory Family integrated Memory Controller provides the interface to DRAM and communicates to the rest of the Uncore through the Mesh2Mem block.

The memory controller also provides a variety of RAS features, such as ECC, lockstep, memory access retry, memory scrubbing, thermal throttling, mirroring, and rank sparing.

2.3.1 Functional Overview

The memory controller communicates to DRAM, translating read and write commands into specific memory commands and schedules them with respect to memory timing. The other main function of the memory controller is advanced ECC support.

The Intel® Xeon® Processor Scalable Memory Family supports up to 6 channels of DDR4 with 3 channels per memory controller. The Intel® Xeon® Processor Scalable Memory Family supports up to 2 DIMMs per channel.

Each memory controller supports chanel speeds up to 2667 MT/s. The Intel® Xeon® Processor Scalable Memory Family does not support DDR3 DIMMs.

A selection of IMC functionality that performance monitoring provides some insight into:

- Supports up to 16 ranks per channel with 8 independent banks per rank.
- ECC support (correct any error within a x4 device)
- Open or closed page policy - (closed only on EX)

- ISOCH (not supported on EX)
- Demand and Patrol Scrubbing support (EP and EX parts)
- Support for LR-DIMMs (load reduced) for a buffered memory solution demanding higher capacity memory subsytems.

2.3.2 IMC Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

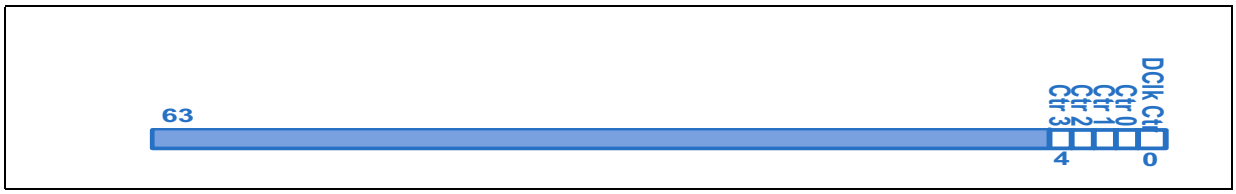
The IMC supports event monitoring through four 48-bit wide counters (MC_CHy_PCI_PMON_CTR{3:0}) and one fixed counter (MC_CHy_PCI_PMON_FIXED_CTR) for each DRAM channel (of which there are 3 in Intel® Xeon® Processor Scalable Memory Family) the MC is attached to. Each of these counters can be programmed (MC_CHy_PCI_PMON_CTL{3:0}) to capture any MC event. The MC counters will increment by a maximum of 8b per cycle.

2.3.3 Additional IMC Performance Monitoring

2.3.3.1 iMC Unit Status - Difference from Baseline

The iMC Unit Status register differs from the common Unit Status layout in it also includes the fixed counter's overflow status.

Figure 2-5. PMON Unit Status Register for Intel® Xeon® Processor Scalable Memory Family - Format common to all PMON Blocks (except the iMC)



Note: If an overflow is detected from one of the Unit's PMON registers + the iMC's fixed DCLK register, the corresponding bit in the PMON_UNIT_STATUS.ov field will be set (DCLK at bit 0. PMON counter 0 at bit 1, etc). To reset these overflow bits, a user must write a value of '1' to them (which will clear the bits).

Table 2-103. PMON_UNIT_STATUS Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	31:4	RV	0	Reserved
ov	4:1	RW1C	0	If an overflow is detected from the corresponding PMON_CTR register, it's overflow bit will be set. (i.e. Bit 1 for Ctr0, Bit 2 for Ctr1, etc) NOTE: Write of '1' will clear the bit.
ov_dclk	0	RW1C	0	If an overflow is detected from the fixed DCLK register, this bit will be set. NOTE: Write of '1' will clear the bit.

Following is a counter that always tracks the number of DRAM clocks (dclks - half of DDR speed) in the IMC. The dclk never changes frequency (on a given system), and therefore is a good measure of wall clock (unlike the Uncore clock which can change frequency based on system load). This clock is generally a bit slower than the uclk (~800MHz to ~1.066GHz) and therefore has less fidelity.

Figure 2-6. PMON Control Register for DCLK

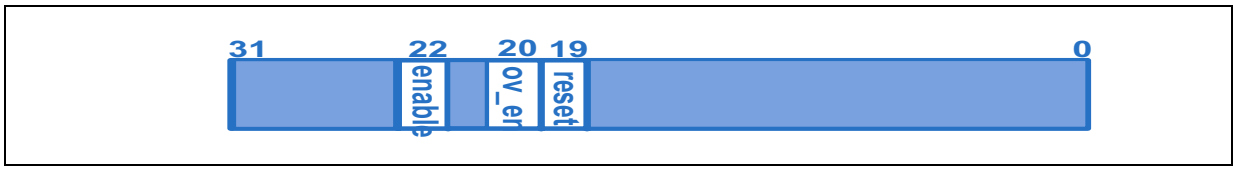


Table 2-104. MC_CHy_PCI_PMON_FIXED_CTL Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	31:23	RV	0	Ignored
en	22	RW-V	0	Local Counter Enable.
ig	21	RV	0	Ignored
ov_en	20	RW-V	0	When this bit is asserted and the corresponding counter overflows, a PMI exception is sent to the UBox.
rst	19	WO	0	When set to 1, the corresponding counter will be cleared to 0.
ig	18:0	RV	0	Ignored

Table 2-105. MC_CHy_PCI_PMON_CTR{FIXED,3-0} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:48	RV	0	Ignored
event_count	47:0	RW-V	0	48-bit performance event counter

2.3.4 IMC Performance Monitoring Events

A sampling of events available for monitoring in the IMC:

- **Translated commands**: Various Read and Write CAS commands
- **Memory commands**: CAS, Precharge, Refresh, Preemptions, etc,
- Page hits and page misses.
- **Page Closing** Events
- **Control of power consumption**: Thermal Throttling by Rank, Time spent in CKE ON mode, etc.

and many more.

Internal IMC Queues:

RPQ - Read Pending Queue.

WPQ - Write Pending Queue.

2.3.5 iMC Box Events Ordered By Code

The following table summarizes the directly measured iMC Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CLOCKTICKS	0x00	0-3	0	DRAM Clockticks
ACT_COUNT	0x01	0-3	0	DRAM Activate Count
PRE_COUNT	0x02	0-3	0	DRAM Precharge commands.
CAS_COUNT	0x04	0-3	0	DRAM CAS (Column Address Strobe) Commands.
DRAM_REFRESH	0x05	0-3	0	Number of DRAM Refreshes Issued
DRAM_PRE_ALL	0x06	0-3	0	DRAM Precharge All Commands
MAJOR_MODES	0x07	0-3	0	Cycles in a Major Mode
PREEMPTION	0x08	0-3	0	Read Preemption Count
ECC_CORRECTABLE_ERRORS	0x09	0-3	0	ECC Correctable Errors
RPQ_INSERTS	0x10	0-3	0	Read Pending Queue Allocations
RPQ_CYCLES_NE	0x11	0-3	0	Read Pending Queue Not Empty
RPQ_CYCLES_FULL	0x12	0-3	0	Read Pending Queue Full Cycles
WPQ_INSERTS	0x20	0-3	0	Write Pending Queue Allocations
WPQ_CYCLES_NE	0x21	0-3	0	Write Pending Queue Not Empty
WPQ_CYCLES_FULL	0x22	0-3	0	Write Pending Queue Full Cycles
WPQ_READ_HIT	0x23	0-3	0	Write Pending Queue CAM Match
WPQ_WRITE_HIT	0x24	0-3	0	Write Pending Queue CAM Match
POWER_THROTTLE_CYCLES	0x41	0-3	0	Throttle Cycles for Rank 0
POWER_PCU_THROTTLING	0x42	0-3	0	
POWER_SELF_REFRESH	0x43	0-3	0	Clock-Enabled Self-Refresh
RPQ_OCCUPANCY	0x80	0-3	0	Read Pending Queue Occupancy
POWER_CKE_CYCLES	0x83	0-3	0	CKE_ON_CYCLES by Rank
POWER_CHANNEL_DLLOFF	0x84	0-3	0	Channel DLLOFF Cycles
POWER_CHANNEL_PPD	0x85	0-3	0	Channel PPD Cycles
POWER_CRITICAL_THROTTLE_CYCLES	0x86	0-3	0	Critical Throttle Cycles
RD_CAS_PRIO	0xa0	0-3	0	
BYP_CMDS	0xa1	0-3	0	
RD_CAS_RANK0	0xb0	0-3	0	RD_CAS Access to Rank 0
RD_CAS_RANK1	0xb1	0-3	0	RD_CAS Access to Rank 1
RD_CAS_RANK2	0xb2	0-3	0	RD_CAS Access to Rank 2
RD_CAS_RANK3	0xb3	0-3	0	RD_CAS Access to Rank 3
RD_CAS_RANK4	0xb4	0-3	0	RD_CAS Access to Rank 4
RD_CAS_RANK5	0xb5	0-3	0	RD_CAS Access to Rank 5
RD_CAS_RANK6	0xb6	0-3	0	RD_CAS Access to Rank 6
RD_CAS_RANK7	0xb7	0-3	0	RD_CAS Access to Rank 7
WR_CAS_RANK0	0xb8	0-3	0	WR_CAS Access to Rank 0

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
WR_CAS_RANK1	0xb9	0-3	0	WR_CAS Access to Rank 1
WR_CAS_RANK2	0xba	0-3	0	WR_CAS Access to Rank 2
WR_CAS_RANK3	0xbb	0-3	0	WR_CAS Access to Rank 3
WR_CAS_RANK4	0xbc	0-3	0	WR_CAS Access to Rank 4
WR_CAS_RANK5	0xbd	0-3	0	WR_CAS Access to Rank 5
WR_CAS_RANK6	0xbe	0-3	0	WR_CAS Access to Rank 6
WR_CAS_RANK7	0xbf	0-3	0	WR_CAS Access to Rank 7
WMM_TO_RMM	0xc0	0-3	0	Transition from WMM to RMM because of low threshold
WRONG_MM	0xc1	0-3	0	Not getting the requested Major Mode

2.3.6 iMC Box Common Metrics (Derived Events)

The following table summarizes metrics commonly calculated from iMC Box events.

Symbol Name: Definition	Equation
MEM_BW_READS: Memory bandwidth consumed by reads. Expressed in bytes.	(CAS_COUNT.RD * 64)
MEM_BW_TOTAL: Total memory bandwidth. Expressed in bytes.	MEM_BW_READS + MEM_BW_WRITES
MEM_BW_WRITES: Memory bandwidth consumed by writes Expressed in bytes.	(CAS_COUNT.WR * 64)
PCT_CYCLES_CRITICAL_THROTTLE: The percentage of cycles all DRAM ranks in critical thermal throttling	POWER_CRITICAL_THROTTLE_CYCLES / MC_Chy_PCI_PMON_CTR_FIXED
PCT_CYCLES_DLLOFF: The percentage of cycles all DRAM ranks in CKE slow (DLOFF) mode	POWER_CHANNEL_DLLOFF / MC_Chy_PCI_PMON_CTR_FIXED
PCT_CYCLES_DRAM_RANKx_IN_CKE: The percentage of cycles DRAM rank (x) spent in CKE ON mode.	POWER_CKE_CYCLES.RANKx / MC_Chy_PCI_PMON_CTR_FIXED
PCT_CYCLES_DRAM_RANKx_IN_THR: The percentage of cycles DRAM rank (x) spent in thermal throttling.	POWER_THROTTLE_CYCLES.RANKx / MC_Chy_PCI_PMON_CTR_FIXED
PCT_CYCLES_PPD: The percentage of cycles all DRAM ranks in PPD mode	POWER_CHANNEL_PPD / MC_Chy_PCI_PMON_CTR_FIXED
PCT_CYCLES_SELF_REFRESH: The percentage of cycles Memory is in self refresh power mode	POWER_SELF_REFRESH / MC_Chy_PCI_PMON_CTR_FIXED
PCT_RD_REQUESTS: Percentage of read requests from total requests.	RPQ_INSERTS / (RPQ_INSERTS + WPQ_INSERTS)
PCT_REQUESTS_PAGE_EMPTY: Percentage of memory requests that resulted in Page Empty	(ACT_COUNT - PRE_COUNT.PAGE_MISS)/ (CAS_COUNT.RD + CAS_COUNT.WR)
PCT_REQUESTS_PAGE_HIT: Percentage of memory requests that resulted in Page Hits	1 - (PCT_REQUESTS_PAGE_EMPTY + PCT_REQUESTS_PAGE_MISS)

Symbol Name: Definition	Equation
PCT_REQUESTS_PAGE_MISS: Percentage of memory requests that resulted in Page Misses	PRE_COUNT.PAGE_MISS / (CAS_COUNT.RD + CAS_COUNT.WR)
PCT_WR_REQUESTS: Percentage of write requests from total requests.	WPQ_INSERTS / (RPQ_INSERTS + WPQ_INSERTS)

2.3.7 iMC Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the iMC Box.

ACT_COUNT

- **Title:** DRAM Activate Count
- **Category:** ACT Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of DRAM Activate commands sent on this channel. Activate commands are issued to open up a page on the DRAM devices so that it can be read or written to with a CAS. One can calculate the number of Page Misses by subtracting the number of Page Miss precharges from the number of Activates.

Table 2-106. Unit Masks for ACT_COUNT

Extension	umask [15:8]	Description
RD	bxxxxxxx1	Activate due to Read
WR	bxxxxxx1x	Activate due to Write
BYP	bxxxx1xxx	Activate due to Bypass

BYP_CMDS

- **Title:**
- **Category:** BYPASS Command Events
- **Event Code:** 0xa1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-107. Unit Masks for BYP_CMDS

Extension	umask [15:8]	Description
ACT	bxxxxxxx1	ACT command issued by 2 cycle bypass
CAS	bxxxxxx1x	CAS command issued by 2 cycle bypass
PRE	bxxxxx1xx	PRE command issued by 2 cycle bypass

CAS_COUNT

- **Title:** DRAM CAS (Column Address Strobe) Commands.
- **Category:** PRE Events
- **Event Code:** 0x04
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3

- **Definition:**

Table 2-108. Unit Masks for CAS_COUNT

Extension	umask [15:8]	Description
RD_REG	bxxxxxxx1	All read CAS (w/ and w/out auto-pre) Counts the total number or DRAM Read CAS commands issued on this channel. This includes both regular RD CAS commands as well as those with implicit Precharge. AutoPre is only used in systems that are using closed page policy. We do not filter based on major mode, as RD_CAS is not issued during WMM (with the exception of underfills).
RD_UNDERFILL	bxxxxxx1x	Underfill Read Issued Counts the number of underfill reads that are issued by the memory controller. This will generally be about the same as the number of partial writes, but may be slightly less because of partials hitting in the WPQ. While it is possible for underfills to be issed in both WMM and RMM, this event counts both.
RD	b00000011	All DRAM Reads (includes underfills) Counts the total number of DRAM Read CAS commands issued on this channel (including underfills).
WR_WMM	bxxxxx1xx	DRAM WR_CAS (w/ and w/out auto-pre) in Write Major Mode Counts the total number or DRAM Write CAS commands issued on this channel while in Write-Major-Mode.
WR_RMM	bxxxx1xxx	DRAM WR_CAS (w/ and w/out auto-pre) in Read Major Mode Counts the total number of Opportunistic" DRAM Write CAS commands issued on this channel while in Read-Major-Mode.
WR	b00001100	All DRAM WR_CAS (both Modes) Counts the total number of DRAM Write CAS commands issued on this channel.
ALL	b00001111	All CASes issued. Counts the total number of DRAM CAS commands issued on this channel.
RD_WMM	bxxx1xxxx	Read CAS issued in WMM
RD_RMM	bxx1xxxxx	Read CAS issued in RMM
RD_ISOCH	bx1xxxxxx	Read CAS issued in Read ISOCH Mode
WR_ISOCH	b1xxxxxxx	Read CAS issued in Write ISOCH Mode

CLOCKTICKS

- **Title:** DRAM Clockticks
- **Category:** DCLK Events
- **Event Code:** 0x00
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

DRAM_PRE_ALL

- **Title:** DRAM Precharge All Commands
- **Category:** DRAM_PRE_ALL Events
- **Event Code:** 0x06
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that the precharge all command was sent.

DRAM_REFRESH

- **Title:** Number of DRAM Refreshes Issued
- **Category:** DRAM_REFRESH Events
- **Event Code:** 0x05
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of refreshes issued.

Table 2-109. Unit Masks for DRAM_REFRESH

Extension	umask [15:8]	Description
PANIC	bxxxxxx1x	
HIGH	bxxxxx1xx	

ECC_CORRECTABLE_ERRORS

- **Title:** ECC Correctable Errors
- **Category:** ECC Events
- **Event Code:** 0x09
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of ECC errors detected and corrected by the iMC on this channel. This counter is only useful with ECC DRAM devices. This count will increment one time for each correction regardless of the number of bits corrected. The iMC can correct up to 4 bit errors in independent channel mode and 8 bit erros in lockstep mode.

MAJOR_MODES

- **Title:** Cycles in a Major Mode
- **Category:** MAJOR_MODES Events
- **Event Code:** 0x07
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the total number of cycles spent in a major mode (selected by a filter) on the given channel. Major modea are channel-wide, and not a per-rank (or dimm or bank) mode.

Table 2-110. Unit Masks for MAJOR_MODES

Extension	umask [15:8]	Description
READ	bxxxxxxx1	Read Major Mode Read Major Mode is the default mode for the iMC, as reads are generally more critical to forward progress than writes.
WRITE	bxxxxxx1x	Write Major Mode This mode is triggered when the WPQ hits high occupancy and causes writes to be higher priority than reads. This can cause blips in the available read bandwidth in the system and temporarily increase read latencies in order to achieve better bus utilizations and higher bandwidth.
PARTIAL	bxxxxx1xx	Partial Major Mode This major mode is used to drain starved underfill reads. Regular reads and writes are blocked and only underfill reads will be processed.
ISOCH	bxxxx1xxx	Isoch Major Mode We group these two modes together so that we can use four counters to track each of the major modes at one time. These major modes are used whenever there is an ISOCH txn in the memory controller. In these mode, only ISOCH transactions are processed.

POWER_CHANNEL_DLLOFF

- **Title:** Channel DLLOFF Cycles
- **Category:** POWER Events
- **Event Code:** 0x84
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles when all the ranks in the channel are in CKE Slow (DLLOFF) mode.
- **NOTE:** IBT = Input Buffer Termination = Off

POWER_CHANNEL_PPD

- **Title:** Channel PPD Cycles
- **Category:** POWER Events
- **Event Code:** 0x85
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles when all the ranks in the channel are in PPD mode. If IBT=off is enabled, then this can be used to count those cycles. If it is not enabled, then this can count the number of cycles when that could have been taken advantage of.
- **NOTE:** IBT = Input Buffer Termination = On

POWER_CKE_CYCLES

- **Title:** CKE_ON_CYCLES by Rank
- **Category:** POWER Events
- **Event Code:** 0x83
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles spent in CKE ON mode. The filter allows you to select a rank to monitor. If multiple ranks are in CKE ON mode at one time, the counter will ONLY increment by one rather than doing accumulation. Multiple counters will need to be used to track multiple ranks simultaneously. There is no distinction between the different CKE modes (APD, PPDS, PPDF). This can be determined based on the system programming. These events should commonly be used with Invert to get the number of cycles in power saving mode. Edge Detect is also useful here. Make sure that you do NOT use Invert with Edge Detect (this just confuses the system and is not necessary).

Table 2-111. Unit Masks for POWER_CKE_CYCLES

Extension	umask [15:8]	Description
RANK0	b00000001	DIMM ID
RANK1	b00000010	DIMM ID
RANK2	b00000100	DIMM ID
RANK3	b00001000	DIMM ID
RANK4	b00010000	DIMM ID
RANK5	b00100000	DIMM ID
RANK6	b01000000	DIMM ID
RANK7	b10000000	DIMM ID

POWER_CRITICAL_THROTTLE_CYCLES

- **Title:** Critical Throttle Cycles
- **Category:** POWER Events
- **Event Code:** 0x86
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the iMC is in critical thermal throttling. When this happens, all traffic is blocked. This should be rare unless something bad is going on in the platform. There is no filtering by rank for this event.

POWER_PCU_THROTTLING

- **Title:**
- **Category:** POWER Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

POWER_SELF_REFRESH

- **Title:** Clock-Enabled Self-Refresh
- **Category:** POWER Events
- **Event Code:** 0x43
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the iMC is in self-refresh and the iMC still has a clock. This happens in some package C-states. For example, the PCU may ask the iMC to enter self-refresh even though some of the cores are still processing. One use of this is for Monroe technology. Self-refresh is required during package C3 and C6, but there is no clock in the iMC at this time, so it is not possible to count these cases.

POWER_THROTTLE_CYCLES

- **Title:** Throttle Cycles for Rank 0
- **Category:** POWER Events
- **Event Code:** 0x41
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles while the iMC is being throttled by either thermal constraints or by the PCU throttling. It is not possible to distinguish between the two. This can be filtered by rank. If multiple ranks are selected and are being throttled at the same time, the counter will only increment by 1.

Table 2-112. Unit Masks for POWER_THROTTLE_CYCLES

Extension	umask [15:8]	Description
RANK0	bxxxxxxx1	DIMM ID Thermal throttling is performed per DIMM. We support 3 DIMMs per channel. This ID allows us to filter by ID.
RANK1	bxxxxxx1x	DIMM ID
RANK2	bxxxxx1xx	DIMM ID
RANK3	bxxxx1xxx	DIMM ID
RANK4	bxxx1xxxx	DIMM ID
RANK5	bxx1xxxxx	DIMM ID
RANK6	bx1xxxxxx	DIMM ID
RANK7	b1xxxxxxx	DIMM ID

PREEMPTION

- **Title:** Read Preemption Count
- **Category:** PREEMPTION Events
- **Event Code:** 0x08
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times a read in the iMC preempts another read or write. Generally reads to an open page are issued ahead of requests to closed pages. This improves the page hit rate of the system. However, high priority requests can cause pages of active requests to be closed in order to get them out. This will reduce the latency of the high-priority request at the expense of lower bandwidth and increased overall average latency.

Table 2-113. Unit Masks for PREEMPTION

Extension	umask [15:8]	Description
RD_PREEMPT_RD	bxxxxxxx1	Read over Read Preemption Filter for when a read preempts another read.
RD_PREEMPT_WR	bxxxxxx1x	Read over Write Preemption Filter for when a read preempts a write.

PRE_COUNT

- **Title:** DRAM Precharge commands.
- **Category:** PRE Events
- **Event Code:** 0x02
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of DRAM Precharge commands sent on this channel.

Table 2-114. Unit Masks for PRE_COUNT

Extension	umask [15:8]	Description
PAGE_MISS	bxxxxxxx1	Precharges due to page miss Counts the number of DRAM Precharge commands sent on this channel as a result of page misses. This does not include explicit precharge commands sent with CAS commands in Auto-Precharge mode. This does not include PRE commands sent as a result of the page close counter expiration.
PAGE_CLOSE	bxxxxxx1x	Precharge due to timer expiration Counts the number of DRAM Precharge commands sent on this channel as a result of the page close counter expiring. This does not include implicit precharge commands sent in auto-precharge mode.
RD	bxxxxx1xx	Precharge due to read
WR	bxxxx1xxx	Precharge due to write
BYP	bxxx1xxxx	Precharge due to bypass

RD_CAS_PRIO

- **Title:**
- **Category:** CAS Events
- **Event Code:** 0xa0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-115. Unit Masks for RD_CAS_PRIO

Extension	umask [15:8]	Description
LOW	bxxxxxxx1	Read CAS issued with LOW priority
MED	bxxxxxx1x	Read CAS issued with MEDIUM priority
HIGH	bxxxxx1xx	Read CAS issued with HIGH priority
PANIC	bxxxx1xxx	Read CAS issued with PANIC NON ISOCH priority (starved)

RD_CAS_RANK0

- **Title:** RD_CAS Access to Rank 0
- **Category:** CAS Events
- **Event Code:** 0xb0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-116. Unit Masks for RD_CAS_RANK0

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK1

- **Title:** RD_CAS Access to Rank 1
- **Category:** CAS Events
- **Event Code:** 0xb1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-117. Unit Masks for RD_CAS_RANK1

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK2

- **Title:** RD_CAS Access to Rank 2
- **Category:** CAS Events
- **Event Code:** 0xb2
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-118. Unit Masks for RD_CAS_RANK2

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2

Table 2-118. Unit Masks for RD_CAS_RANK2

Extension	umask [15:8]	Description
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK3

- **Title:** RD_CAS Access to Rank 3
- **Category:** CAS Events
- **Event Code:** 0xb3
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-119. Unit Masks for RD_CAS_RANK3

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12

Table 2-119. Unit Masks for RD_CAS_RANK3

Extension	umask [15:8]	Description
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK4

- **Title:** RD_CAS Access to Rank 4
- **Category:** CAS Events
- **Event Code:** 0xb4
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-120. Unit Masks for RD_CAS_RANK4

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK5

- **Title:** RD_CAS Access to Rank 5
- **Category:** CAS Events
- **Event Code:** 0xb5
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-121. Unit Masks for RD_CAS_RANK5

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK6

- **Title:** RD_CAS Access to Rank 6
- **Category:** CAS Events
- **Event Code:** 0xb6
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-122. Unit Masks for RD_CAS_RANK6

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2

Table 2-122. Unit Masks for RD_CAS_RANK6

Extension	umask [15:8]	Description
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RD_CAS_RANK7

- **Title:** RD_CAS Access to Rank 7
- **Category:** CAS Events
- **Event Code:** 0xb7
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-123. Unit Masks for RD_CAS_RANK7

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12

Table 2-123. Unit Masks for RD_CAS_RANK7

Extension	umask [15:8]	Description
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

RPQ_CYCLES_FULL

- **Title:** Read Pending Queue Full Cycles
- **Category:** RPQ Events
- **Event Code:** 0x12
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the Read Pending Queue is full. When the RPQ is full, the HA will not be able to issue any additional read requests into the iMC. This count should be similar count in the HA which tracks the number of cycles that the HA has no RPQ credits, just somewhat smaller to account for the credit return overhead. We generally do not expect to see RPQ become full except for potentially during Write Major Mode or while running with slow DRAM. This event only tracks non-ISOC queue entries.

RPQ_CYCLES_NE

- **Title:** Read Pending Queue Not Empty
- **Category:** RPQ Events
- **Event Code:** 0x11
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles that the Read Pending Queue is not empty. This can then be used to calculate the average occupancy (in conjunction with the Read Pending Queue Occupancy count). The RPQ is used to schedule reads out to the memory controller and to track the requests. Requests allocate into the RPQ soon after they enter the memory controller, and need credits for an entry in this buffer before being sent from the HA to the iMC. They deallocate after the CAS command has been issued to memory. This filter is to be used in conjunction with the occupancy filter so that one can correctly track the average occupancies for schedulable entries and scheduled requests.

RPQ_INSERTS

- **Title:** Read Pending Queue Allocations
- **Category:** RPQ Events
- **Event Code:** 0x10
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of allocations into the Read Pending Queue. This queue is used to schedule reads out to the memory controller and to track the requests. Requests allocate into the RPQ soon after they enter the memory controller, and need credits for an entry in this buffer before being sent from the HA to the iMC. They deallocate after the CAS command has been issued to memory. This includes both ISOCH and non-ISOCH requests.

RPQ_OCCUPANCY

- **Title:** Read Pending Queue Occupancy
- **Category:** RPQ Events
- **Event Code:** 0x80
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Accumulates the occupancies of the Read Pending Queue each cycle. This can then be used to calculate both the average occupancy (in conjunction with the number of cycles not empty) and the average latency (in conjunction with the number of allocations). The RPQ is used to schedule reads out to the memory controller and to track the requests. Requests allocate into the RPQ soon after they enter the memory controller, and need credits for an entry in this buffer before being sent from the HA to the iMC. They deallocate after the CAS command has been issued to memory.

WMM_TO_RMM

- **Title:** Transition from WMM to RMM because of low threshold
- **Category:** MAJOR_MODES Events
- **Event Code:** 0xc0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-124. Unit Masks for WMM_TO_RMM

Extension	umask [15:8]	Description
LOW_THRESH	bxxxxxxx1	Transition from WMM to RMM because of starve counter
STARVE	bxxxxxx1x	
VMSE_RETRY	bxxxxx1xx	

WPQ_CYCLES_FULL

- **Title:** Write Pending Queue Full Cycles
- **Category:** WPQ Events
- **Event Code:** 0x22
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the Write Pending Queue is full. When the WPQ is full, the HA will not be able to issue any additional write requests into the iMC. This count should be similar count in the CHA which tracks the number of cycles that the CHA has no WPQ credits, just somewhat smaller to account for the credit return overhead.

WPQ_CYCLES_NE

- **Title:** Write Pending Queue Not Empty
- **Category:** WPQ Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles that the Write Pending Queue is not empty. This can then be used to calculate the average queue occupancy (in conjunction with the WPQ Occupancy Accumulation count). The WPQ is used to schedule write out to the memory controller and to track the writes. Requests allocate into the WPQ soon after they enter the memory controller, and need credits for an entry in this buffer before being sent from the CHA to the iMC. They deallocate after being issued to DRAM. Write requests themselves are able to complete (from the perspective of the rest of the system) as soon they have "posted" to the iMC. This is not to be confused with actually performing the write to DRAM. Therefore, the average latency for this queue is actually not useful for deconstruction intermediate write latencies.

WPQ_INSERTS

- **Title:** Write Pending Queue Allocations
- **Category:** WPQ Events
- **Event Code:** 0x20
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of allocations into the Write Pending Queue. This can then be used to calculate the average queuing latency (in conjunction with the WPQ occupancy count). The WPQ is used to schedule write out to the memory controller and to track the writes. Requests allocate into the WPQ soon after they enter the memory controller, and need credits for an entry in this buffer before being sent from the CHA to the iMC. They deallocate after being issued to DRAM. Write requests themselves are able to complete (from the perspective of the rest of the system) as soon they have "posted" to the iMC.

WPQ_READ_HIT

- **Title:** Write Pending Queue CAM Match
- **Category:** WPQ Events
- **Event Code:** 0x23
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times a request hits in the WPQ (write-pending queue). The iMC allows writes and reads to pass up other writes to different addresses. Before a read or a write is issued, it will first CAM the WPQ to see if there is a write pending to that address. When reads hit, they are able to directly pull their data from the WPQ instead of going to memory. Writes that hit will overwrite the existing data. Partial writes that hit will not need to do underfill reads and will simply update their relevant sections.

WPQ_WRITE_HIT

- **Title:** Write Pending Queue CAM Match
- **Category:** WPQ Events
- **Event Code:** 0x24
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times a request hits in the WPQ (write-pending queue). The iMC allows writes and reads to pass up other writes to different addresses. Before a read or a write is issued, it will first CAM the WPQ to see if there is a write pending to that address. When reads hit, they are able to directly pull their data from the WPQ instead of going to memory. Writes that hit will overwrite the existing data. Partial writes that hit will not need to do underfill reads and will simply update their relevant sections.

WRONG_MM

- **Title:** Not getting the requested Major Mode
- **Category:** MAJOR_MODES Events
- **Event Code:** 0xc1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

WR_CAS_RANK0

- **Title:** WR_CAS Access to Rank 0
- **Category:** CAS Events
- **Event Code:** 0xb8
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-125. Unit Masks for WR_CAS_RANK0

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK1

- **Title:** WR_CAS Access to Rank 1
- **Category:** CAS Events
- **Event Code:** 0xb9
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-126. Unit Masks for WR_CAS_RANK1

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8

Table 2-126. Unit Masks for WR_CAS_RANK1

Extension	umask [15:8]	Description
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK2

- **Title:** WR_CAS Access to Rank 2
- **Category:** CAS Events
- **Event Code:** 0xba
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-127. Unit Masks for WR_CAS_RANK2

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)

Table 2-127. Unit Masks for WR_CAS_RANK2

Extension	umask [15:8]	Description
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK3

- **Title:** WR_CAS Access to Rank 3
- **Category:** CAS Events
- **Event Code:** 0xbb
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-128. Unit Masks for WR_CAS_RANK3

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK4

- **Title:** WR_CAS Access to Rank 4
- **Category:** CAS Events
- **Event Code:** 0xbc
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-129. Unit Masks for WR_CAS_RANK4

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK5

- **Title:** WR_CAS Access to Rank 5
- **Category:** CAS Events
- **Event Code:** 0xbd
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-130. Unit Masks for WR_CAS_RANK5

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8

Table 2-130. Unit Masks for WR_CAS_RANK5

Extension	umask [15:8]	Description
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK6

- **Title:** WR_CAS Access to Rank 6
- **Category:** CAS Events
- **Event Code:** 0xbe
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-131. Unit Masks for WR_CAS_RANK6

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)

Table 2-131. Unit Masks for WR_CAS_RANK6

Extension	umask [15:8]	Description
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

WR_CAS_RANK7

- **Title:** WR_CAS Access to Rank 7
- **Category:** CAS Events
- **Event Code:** 0xbf
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-132. Unit Masks for WR_CAS_RANK7

Extension	umask [15:8]	Description
BANK0	b00000000	Bank 0
BANK1	b00000001	Bank 1
BANK2	b00000010	Bank 2
BANK3	b00000011	Bank 3
BANK4	b00000100	Bank 4
BANK5	b00000101	Bank 5
BANK6	b00000110	Bank 6
BANK7	b00000111	Bank 7
BANK8	b00001000	Bank 8
BANK9	b00001001	Bank 9
BANK10	b00001010	Bank 10
BANK11	b00001011	Bank 11
BANK12	b00001100	Bank 12
BANK13	b00001101	Bank 13
BANK14	b00001110	Bank 14
BANK15	b00001111	Bank 15
ALLBANKS	b00010000	All Banks
BANKG0	b00010001	Bank Group 0 (Banks 0-3)
BANKG1	b00010010	Bank Group 1 (Banks 4-7)
BANKG2	b00010011	Bank Group 2 (Banks 8-11)
BANKG3	b00010100	Bank Group 3 (Banks 12-15)

2.4 IIO Performance Monitoring

IIO stacks are responsible for managing traffic between the PCIe domain and the Mesh domain. The IIO PMON block is situated near the IIO stack's traffic controller capturing traffic controller as well as PCIe root port information. The traffic controller is responsible for translating traffic coming in from the Mesh (through M2PCIe) and processed by IRP into the PCIe domain to IO agents such as CBDMA, PCIe and MCP.

2.4.1 IIO Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each IIO Box, which sits near the IIO stack's Traffic Controller, supports event monitoring through four 48b wide counters (IIO{5-0}_MSR_PMON_CTR/CTL{3:0}). Each of these counters can be programmed to count any IIO event. The IIO counters can increment by a maximum of 7b per cycle.

IIO0 is part of the CBDMA Stack.

IIO1-3 is part of PCIe stack 0-2.

IIO4 is part of the MCP stack 0

2.4.2 Additional IIO Performance Monitoring

2.4.2.1 IIO PMON Counter Control - Difference from Baseline

IIO performance monitoring control registers provide a small amount of additional functionality. The following table defines those cases.

Figure 2-7. IIO Counter Control Register for Intel® Xeon® Processor Scalable Memory Family

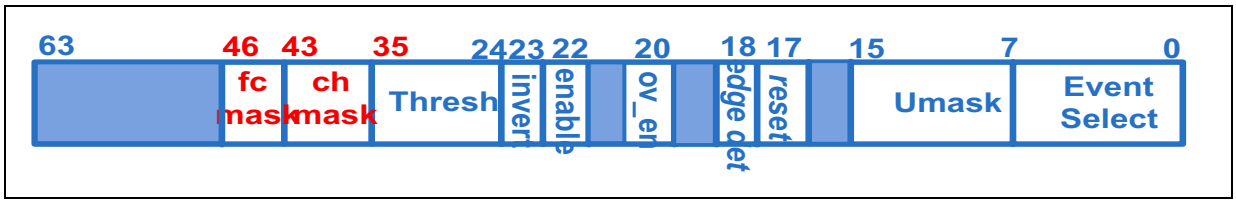


Table 2-133. IIOn_MSR_PMON_CTL{3-0} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	63:47	RV	0	Reserved. SW must write to 0 else behavior is undefined.
fc_mask	46:44	RW-V	0	FC Mask - applicable to certain events (Filter - fc) 0 - Posted Requests 1 - Non-posted Requests 2 - Completions
ch_mask	43:36	RW-V	0	Channel Mask Filter - applicable to certain events (Filter - chnl) PStacks: 0 - PCIe Port 0 1 - PCIe Port 1 2 - PCIe Port 2 3 - PCIe Port 3 4 - Intel VT-d 5-7 --- CStack: 0 - CBDMA 1 - DMI VC0 2 - DMI VC1 3 - DMI VCm 4 - Intel VT-d Non-Isoch 5 - Intel VT-d Isoch 6-7 ---
thresh	35:24	RW-V	0	Threshold used in counter comparison.

As of Intel® Xeon® Processor Scalable Memory Family, there are a number of free-running counters, providing information highly valuable to a wide array of customers, in each IIO Stack that collect counts for Input/Output x BW/Utilization for each Port. 'Free Running' counters cannot be changed by SW operating in a normal environment. SW cannot write them, cannot stop them and cannot reset the values.

Note: Counting will be suspended when the IIO stack is powered down.

There is one register per stack to track the number of IIO cycles

Table 2-134. IIO_MSR_PMON_FRCTR_IOCLK Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:36	RV	0	Ignored
event_count	35:0	RO-V	0	36-bit running count of IO clocks

One extra register per port to track

- **Inbound (PCIe -> CPU) bandwidth** - counts DWs (4 bytes) of data, associated with writes and completions, transmitted from the IO stack to the traffic controller

Table 2-135. IIO_MSR_PMON_FRCTR_BW_IN_P{0-3} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:36	RV	0	Ignored
event_count	35:0	RO-V	0	36-bit running count of data bytes transmitted from link for this port.

Note: On Intel® Xeon® Processor Scalable Memory Family ES2, these counters only count Write traffic. They are slated to be fixed in official PRQ part.

- **Output (CPU -> PCIe) bandwidth** - counts DWs (4 bytes) of data, associated with writes and completions, transmitted from the traffic controller to the IO stack

Table 2-136. IIO_MSR_PMON_FRCTR_BW_OUT_P{0-3} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:36	RV	0	Ignored
event_count	35:0	RO-V	0	36-bit running count of data bytes transmitted link for this port.

Note: On Intel® Xeon® Processor Scalable Memory Family ES2, these counters only count Write traffic. They are slated to be fixed in official PRQ part.

- **Input utilization** - the number of cycles in which the port from the IO stack was utilized. When subtracted from IO clocks, the resulting value tells SW how many cycles the attached device *could* have initiated new transactions. Utilization includes cycles for any use of the port (data, commands, housekeeping, powered down, etc)

Table 2-137. IIO_MSR_PMON_FRCTR_UTIL_IN_P{0-3} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:36	RV	0	Ignored
event_count	35:0	RO-V	0	36-bit running count of cycles the input link to this port was utilized.

- **Output utilization** - the number of cycles in which the port to the IO stack was utilized. When subtracted from IO clocks, the resulting value tells SW how many cycles it *could* have initiated new transactions to the attached device. Utilization includes cycles for any use of the port (data, commands, housekeeping, powered down, and so forth)

Table 2-138. IIO_MSR_PMON_FRCTR_UTIL_OUT_P{0-3} Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:36	RV	0	Ignored
event_count	35:0	RO-V	0	36-bit running count of cycles the input link to this port was utilized.

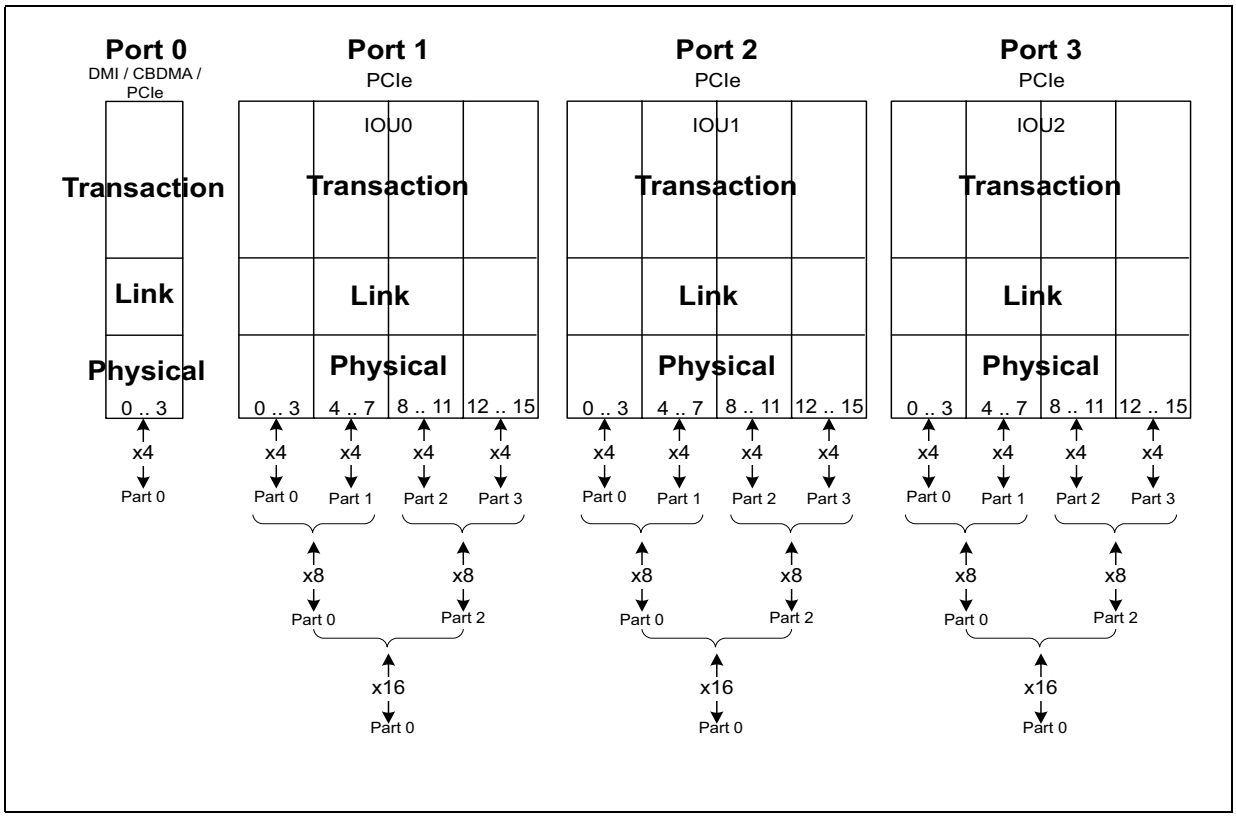
2.4.3 IIO Performance Monitoring Events

IIO provides events to track information related to all the traffic passing through it's boundaries.

- Bandwidth consumed and Transactions processed broken down by transaction type
- Per Port Utilization
- Link Power States
- Completion Buffer tracking

Some information provided through the IIO stack is highly dependent on what's attached to the stack. Following is a quick illustration for how information such as IIO Bandwidth and number of Transactions are rolled up.

Figure 2-8. How IIO BW/TXNs are rolled up relative to width of attached Device(s)



Events such as DATA_REQ_OF_CPU.MEM_READ and TXN_REQ_BY_CPU.PEER_WRITE are further subdivided into '.PART's. For instance, if this is a IIO stack supporting up to x16 PCIe and a x16 Card is plugged into the stack, .PART0 will contain all the relevant information. If 2 x8 Cards are plugged into the stack, .PART0 will contain the contribution of the first x8 Card's traffic and .PART2 will contain the contribution of the second x8 Card's traffic. etc.

2.4.4 IIO Box Events Ordered By Code

The following table summarizes the directly measured IIO Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
NOTHING	0x00		0	
CLOCKTICKS	0x01	0-3	0	Traffic Controller Clocks
MASK_MATCH_AND	0x02	0-3	0	AND Mask/match for debug bus
MASK_MATCH_OR	0x03	0-3	0	OR Mask/match for debug bus
LINK_NUM_RETRIES	0x0e		0	Num Link Retries
LINK_NUM_CORR_ERR	0x0f		0	Num Link Correctable Errors
MASK_MATCH	0x21		0	Number packets that passed the Mask/Match Filter
VTD_OCCUPANCY	0x40	0-3	0	Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) Occupancy
VTD_ACCESS	0x41	0-3	0	Intel VT-d Access
SYMBOL_TIMES	0x82		0	Symbol Times on Link
DATA_REQ_OF_CPU	0x83	0-1	0	Data requested of the CPU
TXN_REQ_OF_CPU	0x84	0-3	0	Number Transactions requested of the CPU
DATA_REQ_BY_CPU	0xc0	2-3	0	Data requested by the CPU
TXN_REQ_BY_CPU	0xc1	0-3	0	Number Transactions requested by the CPU
COMP_BUF_INSERTS	0xc2		0	PCIe Completion Buffer Inserts
COMP_BUF_OCCUPANCY	0xd5		0	PCIe Completion Buffer Occupancy

2.4.5 IIO Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the IIO Box.

CLOCKTICKS

- **Title:** Traffic Controller Clocks
- **Category:** Debug Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

COMP_BUF_INSERTS

- **Title:** PCIe Completion Buffer Inserts
- **Category:** PCIe Completion Buffer Events
- **Event Code:** 0xc2
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-139. Unit Masks for COMP_BUF_INSERTS

Extension	umask [15:8]	xtra [46:36]	Description
PORT0	bxxxxx1xx	b111 bxxxxxxx1	Port 0
PORT1	bxxxxx1xx	b111 bxxxxxx1x	Port 1

Table 2-139. Unit Masks for COMP_BUF_INSERTS

Extension	umask [15:8]	xtra [46:36]	Description
PORT2	bxxxxx1xx	b111 bxxxxx1xx	Port 2
PORT3	bxxxxx1xx	b111 bxxxx1xxx	Port 3

COMP_BUF_OCCUPANCY

- **Title:** PCIe Completion Buffer Occupancy
- **Category:** PCIe Completion Buffer Events
- **Event Code:** 0xd5
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DATA_REQ_BY_CPU

- **Title:** Data requested by the CPU
- **Category:** Payload Events
- **Event Code:** 0xc0
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 2-3
- **Definition:** Number of double word (4 bytes) requests initiated by the main die to the attached device.
- **NOTE:** Unlike free running counters, Mem Read and Peer read subevents count requests not completions. Peer R/W subevents do not include confined P2P traffic.

Table 2-141. Unit Masks for DATA_REQ_BY_CPU (Sheet 1 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
MEM_WRITE.PART0	bxxxxxxx1	b111 bxxxxxxx1	Core writing to Card's MMIO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_WRITE.VTD0	bxxxxxxx1	b111 bxxx1xxxx	Core writing to Card's MMIO space Intel VT-d - Type 0
MEM_WRITE.VTD1	bxxxxxxx1	b111 bxx1xxxxx	Core writing to Card's MMIO space Intel VT-d - Type 1
MEM_WRITE.PART3	bxxxxxxx1	b111 bxxxx1xxx	Core writing to Card's MMIO space x4 card is plugged in to slot 3
MEM_WRITE.PART2	bxxxxxxx1	b111 bxxxxx1xx	Core writing to Card's MMIO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MEM_WRITE.PART1	bxxxxxxx1	b111 bxxxxxx1x	Core writing to Card's MMIO space x4 card is plugged in to slot 1
PEER_WRITE.PART0	bxxxxxx1x	b111 bxxxxxxx1	Another card (different IIO stack) writing to this card. x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_WRITE.PART2	bxxxxxx1x	b111 bxxxxx1xx	Another card (different IIO stack) writing to this card. x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_WRITE.VTD1	bxxxxxx1x	b111 bxx1xxxxx	Another card (different IIO stack) writing to this card. F - Type 1
PEER_WRITE.PART1	bxxxxxx1x	b111 bxxxxxx1x	Another card (different IIO stack) writing to this card. x4 card is plugged in to slot 1

Table 2-141. Unit Masks for DATA_REQ_BY_CPU (Sheet 2 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
PEER_WRITE.PART3	bxxxxxx1x	b111 bxxxx1xxx	Another card (different IIO stack) writing to this card. x4 card is plugged in to slot 3
PEER_WRITE.VTD0	bxxxxxx1x	b111 bxxx1xxxx	Another card (different IIO stack) writing to this card. Intel VT-d - Type 0
MEM_READ.PART0	bxxxxx1xx	b111 bxxxxxxx1	Core reading from Card's MMIO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_READ.PART3	bxxxxx1xx	b111 bxxxx1xxx	Core reading from Card's MMIO space x4 card is plugged in to slot 3
MEM_READ.VTD0	bxxxxx1xx	b111 bxxx1xxxx	Core reading from Card's MMIO space Intel VT-d - Type 0
MEM_READ.PART1	bxxxxx1xx	b111 bxxxxxx1x	Core reading from Card's MMIO space x4 card is plugged in to slot 1
MEM_READ.VTD1	bxxxxx1xx	b111 bxx1xxxxx	Core reading from Card's MMIO space Intel VT-d - Type 1
MEM_READ.PART2	bxxxxx1xx	b111 bxxxxx1xx	Core reading from Card's MMIO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_READ.PART0	bxxxx1xxx	b111 bxxxxxxx1	Another card (different IIO stack) reading from this card. x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_READ.VTD1	bxxxx1xxx	b111 bxx1xxxxx	Another card (different IIO stack) reading from this card. Intel VT-d - Type 1
PEER_READ.PART1	bxxxx1xxx	b111 bxxxxxx1x	Another card (different IIO stack) reading from this card. x4 card is plugged in to slot 1
PEER_READ.PART3	bxxxx1xxx	b111 bxxxx1xxx	Another card (different IIO stack) reading from this card. x4 card is plugged in to slot 3
PEER_READ.VTD0	bxxxx1xxx	b111 bxxx1xxxx	Another card (different IIO stack) reading from this card. Intel VT-d - Type 0
PEER_READ.PART2	bxxxx1xxx	b111 bxxxxx1xx	Another card (different IIO stack) reading from this card. x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
CFG_WRITE.VTD0	bxxx1xxxx	b111 bxxx1xxxx	Core writing to Card's PCICFG space Intel VT-d - Type 0
CFG_WRITE.PART0	bxxx1xxxx	b111 bxxxxxxx1	Core writing to Card's PCICFG space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
CFG_WRITE.PART2	bxxx1xxxx	b111 bxxxxx1xx	Core writing to Card's PCICFG space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
CFG_WRITE.PART3	bxxx1xxxx	b111 bxxxx1xxx	Core writing to Card's PCICFG space x4 card is plugged in to slot 3
CFG_WRITE.PART1	bxxx1xxxx	b111 bxxxxxx1x	Core writing to Card's PCICFG space x4 card is plugged in to slot 1
CFG_WRITE.VTD1	bxxx1xxxx	b111 bxx1xxxxx	Core writing to Card's PCICFG space Intel VT-d - Type 1
IO_WRITE.PART1	bxx1xxxxx	b111 bxxxxxx1x	Core writing to Card's IO space x4 card is plugged in to slot 1
IO_WRITE.PART3	bxx1xxxxx	b111 bxxxx1xxx	Core writing to Card's IO space x4 card is plugged in to slot 3

Table 2-141. Unit Masks for DATA_REQ_BY_CPU (Sheet 3 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
IO_WRITE.VTD1	bxx1xxxxx	b111 bxx1xxxxx	Core writing to Card's IO space Intel VT-d - Type 1
IO_WRITE.PART0	bxx1xxxxx	b111 bxxxxxxx1	Core writing to Card's IO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
IO_WRITE.PART2	bxx1xxxxx	b111 bxxxxx1xx	Core writing to Card's IO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
IO_WRITE.VTD0	bxx1xxxxx	b111 bxxx1xxxx	Core writing to Card's IO space Intel VT-d - Type 0
CFG_READ.PART0	bx1xxxxxx	b111 bxxxxxxx1	Core reading from Card's PCICFG space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
CFG_READ.VTD0	bx1xxxxxx	b111 bxxx1xxxx	Core reading from Card's PCICFG space Intel VT-d - Type 0
CFG_READ.PART3	bx1xxxxxx	b111 bxxxx1xxx	Core reading from Card's PCICFG space x4 card is plugged in to slot 3
CFG_READ.PART2	bx1xxxxxx	b111 bxxxxx1xx	Core reading from Card's PCICFG space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
CFG_READ.VTD1	bx1xxxxxx	b111 bxx1xxxxx	Core reading from Card's PCICFG space Intel VT-d - Type 1
CFG_READ.PART1	bx1xxxxxx	b111 bxxxxxx1x	Core reading from Card's PCICFG space x4 card is plugged in to slot 1
IO_READ.VTD1	b1xxxxxxx	b111 bxx1xxxxx	Core reading from Card's IO space Intel VT-d - Type 1
IO_READ.PART0	b1xxxxxxx	b111 bxxxxxxx1	Core reading from Card's IO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
IO_READ.PART2	b1xxxxxxx	b111 bxxxxx1xx	Core reading from Card's IO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
IO_READ.VTD0	b1xxxxxxx	b111 bxxx1xxxx	Core reading from Card's IO space Intel VT-d - Type 0
IO_READ.PART1	b1xxxxxxx	b111 bxxxxxx1x	Core reading from Card's IO space x4 card is plugged in to slot 1
IO_READ.PART3	b1xxxxxxx	b111 bxxxx1xxx	Core reading from Card's IO space x4 card is plugged in to slot 3

DATA_REQ_OF_CPU

- **Title:** Data requested of the CPU
- **Category:** Payload Events
- **Event Code:** 0x83
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Number of double word (4 bytes) requests the attached device made of the main die.
- **NOTE:** Multiply value by 4 to convert to number of Bytes. Unlike free running counters, Mem Read and Peer read subevents count requests not completions. Peer R/W subevents do not include confined P2P traffic. Counts are incremented on the request path rather than the completion path. Expect to change back on next product.

Table 2-142. Unit Masks for DATA_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
MEM_WRITE.PART0	bxxxxxxx1	b111 bxxxxxxx1	Card writing to DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_WRITE.VTD1	bxxxxxxx1	b111 bxx1xxxxx	Card writing to DRAM Intel VT-d - Type 1
MEM_WRITE.VTD0	bxxxxxxx1	b111 bxxx1xxxx	Card writing to DRAM Intel VT-d - Type 0
MEM_WRITE.PART2	bxxxxxxx1	b111 bxxxxx1xx	Card writing to DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MEM_WRITE.PART1	bxxxxxxx1	b111 bxxxxxx1x	Card writing to DRAM x4 card is plugged in to slot 1
MEM_WRITE.PART3	bxxxxxxx1	b111 bxxxx1xxx	Card writing to DRAM x4 card is plugged in to slot 3
PEER_WRITE.PART0	bxxxxxx1x	b111 bxxxxxxx1	Card writing to another Card (same or different stack) x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_WRITE.VTD1	bxxxxxx1x	b111 bxx1xxxxx	Card writing to another Card (same or different stack) Intel VT-d - Type 1
PEER_WRITE.PART2	bxxxxxx1x	b111 bxxxxx1xx	Card writing to another Card (same or different stack) x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_WRITE.PART3	bxxxxxx1x	b111 bxxxx1xxx	Card writing to another Card (same or different stack) x4 card is plugged in to slot 3
PEER_WRITE.VTD0	bxxxxxx1x	b111 bxxx1xxxx	Card writing to another Card (same or different stack) Intel VT-d - Type 0
PEER_WRITE.PART1	bxxxxxx1x	b111 bxxxxxx1x	Card writing to another Card (same or different stack) x4 card is plugged in to slot 1
MEM_READ.PART3	bxxxxx1xx	b111 bxxxx1xxx	Card reading from DRAM x4 card is plugged in to slot 3
MEM_READ.PART0	bxxxxx1xx	b111 bxxxxxxx1	Card reading from DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_READ.VTD0	bxxxxx1xx	b111 bxxx1xxxx	Card reading from DRAM Intel VT-d - Type 0
MEM_READ.VTD1	bxxxxx1xx	b111 bxx1xxxxx	Card reading from DRAM Intel VT-d - Type 1
MEM_READ.PART1	bxxxxx1xx	b111 bxxxxxx1x	Card reading from DRAM x4 card is plugged in to slot 1

Table 2-142. Unit Masks for DATA_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
MEM_READ.PART2	bxxxxx1xx	b111 bxxxxx1xx	Card reading from DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_READ.PART0	bxxxx1xxx	b111 bxxxxxxx1	Card reading from another Card (same or different stack) x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_READ.PART1	bxxxx1xxx	b111 bxxxxxx1x	Card reading from another Card (same or different stack) x4 card is plugged in to slot 1
PEER_READ.PART3	bxxxx1xxx	b111 bxxxx1xxx	Card reading from another Card (same or different stack) x4 card is plugged in to slot 3
PEER_READ.VTD1	bxxxx1xxx	b111 bxx1xxxxx	Card reading from another Card (same or different stack) Intel VT-d - Type 1
PEER_READ.VTD0	bxxxx1xxx	b111 bxxx1xxxx	Card reading from another Card (same or different stack) Intel VT-d - Type 0
PEER_READ.PART2	bxxxx1xxx	b111 bxxxxx1xx	Card reading from another Card (same or different stack) x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
ATOMIC.PART2	bxxx1xxxx	b111 bxxxxx1xx	Atomic requests targeting DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
ATOMIC.PART0	bxxx1xxxx	b111 bxxxxxxx1	Atomic requests targeting DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
ATOMIC.VTD0	bxxx1xxxx	b111 bxxx1xxxx	Atomic requests targeting DRAM Intel VT-d - Type 0
ATOMIC.PART1	bxxx1xxxx	b111 bxxxxxx1x	Atomic requests targeting DRAM x4 card is plugged in to slot 1
ATOMIC.PART3	bxxx1xxxx	b111 bxxxx1xxx	Atomic requests targeting DRAM x4 card is plugged in to slot 3
ATOMIC.VTD1	bxxx1xxxx	b111 bxx1xxxxx	Atomic requests targeting DRAM Intel VT-d - Type 1
ATOMICCMP.PART2	bxx1xxxxx	b111 bxxxxx1xx	Completion of atomic requests targeting DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
ATOMICCMP.PART0	bxx1xxxxx	b111 bxxxxxxx1	Completion of atomic requests targeting DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
ATOMICCMP.PART1	bxx1xxxxx	b111 bxxxxxx1x	Completion of atomic requests targeting DRAM x4 card is plugged in to slot 1
ATOMICCMP.PART3	bxx1xxxxx	b111 bxxxx1xxx	Completion of atomic requests targeting DRAM x4 card is plugged in to slot 3
MSG.PART0	bx1xxxxxx	b111 bxxxxxxx1	Messages x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MSG.VTD0	bx1xxxxxx	b111 bxxx1xxxx	Messages Intel VT-d - Type 0
MSG.VTD1	bx1xxxxxx	b111 bxx1xxxxx	Messages Intel VT-d - Type 1
MSG.PART1	bx1xxxxxx	b111 bxxxxxx1x	Messages x4 card is plugged in to slot 1

Table 2-142. Unit Masks for DATA_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
MSG.PART3	bx1xxxxxx	b111 bxxxx1xxx	Messages x4 card is plugged in to slot 3
MSG.PART2	bx1xxxxxx	b111 bxxxxx1xx	Messages x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2

LINK_NUM_CORR_ERR

- **Title:** Num Link Correctable Errors
- **Category:** Link Events
- **Event Code:** 0x0f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

LINK_NUM_RETRIES

- **Title:** Num Link Retries
- **Category:** Link Events
- **Event Code:** 0x0e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

MASK_MATCH

- **Title:** Number packets that passed the Mask/Match Filter
- **Category:** Miscellaneous Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

MASK_MATCH_AND

- **Title:** AND Mask/match for debug bus
- **Category:** Debug Events
- **Event Code:** 0x02
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Asserted if all bits specified by mask match

Table 2-143. Unit Masks for MASK_MATCH_AND

Extension	umask [15:8]	Description
BUS0	bxxxxxxx1	Non-PCIE bus
BUS1	bxxxxxx1x	PCIE bus
BUS0_NOT_BUS1	bxxxxx1xx	Non-PCIE bus and !(PCIE bus)
BUS0_BUS1	bxxxx1xxx	Non-PCIE bus and PCIE bus
NOT_BUS0_BUS1	bxxx1xxxx	!(Non-PCIE bus) and PCIE bus
NOT_BUS0_NOT_BUS1	bxx1xxxxx	

MASK_MATCH_OR

- **Title:** OR Mask/match for debug bus
- **Category:** Debug Events
- **Event Code:** 0x03
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Asserted if any bits specified by mask match

Table 2-144. Unit Masks for MASK_MATCH_OR

Extension	umask [15:8]	Description
BUS0	bxxxxxxx1	Non-PCIE bus
BUS1	bxxxxxx1x	PCIE bus
BUS0_NOT_BUS1	bxxxxx1xx	Non-PCIE bus and !(PCIE bus)
BUS0_BUS1	bxxxx1xxx	Non-PCIE bus and PCIE bus
NOT_BUS0_BUS1	bxxx1xxxx	!(Non-PCIE bus) and PCIE bus
NOT_BUS0_NOT_BUS1	bxx1xxxxx	!(Non-PCIE bus) and !(PCIE bus)

NOTHING

- **Title:**
- **Category:** CLOCK Events
- **Event Code:** 0x00
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

SYMBOL_TIMES

- **Title:** Symbol Times on Link
- **Category:** Miscellaneous Events
- **Event Code:** 0x82
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Gen1 - increment once every 4nS, Gen2 - increment once every 2 nS, Gen3 - increment once every 1nS

TXN_REQ_BY_CPU

- **Title:** Number Transactions requested by the CPU
- **Category:** Transaction Events
- **Event Code:** 0xc1
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Also known as Outbound. Number of requests, to the attached device, initiated by the main die.
- **NOTE:** Unlike free running counters, Mem Read and Peer read subevents count requests not completions. Peer R/W subevents do not include confined P2P traffic.

Table 2-145. Unit Masks for TXN_REQ_BY_CPU (Sheet 1 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
MEM_WRITE.PART0	bxxxxxxx1	b111 bxxxxxxx1	Core writing to Card's MMIO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_WRITE.VTD0	bxxxxxxx1	b111 bxxx1xxxx	Core writing to Card's MMIO space Intel VT-d - Type 0
MEM_WRITE.VTD1	bxxxxxxx1	b111 bxx1xxxxx	Core writing to Card's MMIO space Intel VT-d - Type 1
MEM_WRITE.PART2	bxxxxxxx1	b111 bxxxxx1xx	Core writing to Card's MMIO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MEM_WRITE.PART1	bxxxxxxx1	b111 bxxxxxx1x	Core writing to Card's MMIO space x4 card is plugged in to slot 1
MEM_WRITE.PART3	bxxxxxxx1	b111 bxxxx1xxx	Core writing to Card's MMIO space x4 card is plugged in to slot 3
PEER_WRITE.PART0	bxxxxxx1x	b111 bxxxxxxx1	Another card (different IIO stack) writing to this card. x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_WRITE.VTD1	bxxxxxx1x	b111 bxx1xxxxx	Another card (different IIO stack) writing to this card. Intel VT-d - Type 1
PEER_WRITE.PART2	bxxxxxx1x	b111 bxxxxx1xx	Another card (different IIO stack) writing to this card. x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_WRITE.VTD0	bxxxxxx1x	b111 bxxx1xxxx	Another card (different IIO stack) writing to this card. Intel VT-d - Type 0
PEER_WRITE.PART3	bxxxxxx1x	b111 bxxxx1xxx	Another card (different IIO stack) writing to this card. x4 card is plugged in to slot 3
PEER_WRITE.PART1	bxxxxxx1x	b111 bxxxxxx1x	Another card (different IIO stack) writing to this card. x4 card is plugged in to slot 1
MEM_READ.PART3	bxxxxx1xx	b111 bxxxx1xxx	Core reading from Card's MMIO space x4 card is plugged in to slot 3
MEM_READ.PART0	bxxxxx1xx	b111 bxxxxxxx1	Core reading from Card's MMIO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_READ.VTD0	bxxxxx1xx	b111 bxxx1xxxx	Core reading from Card's MMIO space Intel VT-d - Type 0
MEM_READ.VTD1	bxxxxx1xx	b111 bxx1xxxxx	Core reading from Card's MMIO space Intel VT-d - Type 1
MEM_READ.PART1	bxxxxx1xx	b111 bxxxxxx1x	Core reading from Card's MMIO space x4 card is plugged in to slot 1
MEM_READ.PART2	bxxxxx1xx	b111 bxxxxx1xx	Core reading from Card's MMIO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_READ.PART0	bxxxx1xxx	b111 bxxxxxxx1	Another card (different IIO stack) reading from this card. x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_READ.PART3	bxxxx1xxx	b111 bxxxx1xxx	Another card (different IIO stack) reading from this card. x4 card is plugged in to slot 3
PEER_READ.PART1	bxxxx1xxx	b111 bxxxxxx1x	Another card (different IIO stack) reading from this card. x4 card is plugged in to slot 1

Table 2-145. Unit Masks for TXN_REQ_BY_CPU (Sheet 2 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
PEER_READ.VTD1	bxxxx1xxx	b111 bxx1xxxxx	Another card (different IIO stack) reading from this card. Intel VT-d - Type 1
PEER_READ.PART2	bxxxx1xxx	b111 bxxxxx1xx	Another card (different IIO stack) reading from this card. x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_READ.VTD0	bxxxx1xxx	b111 bxxx1xxxx	Another card (different IIO stack) reading from this card. Intel VT-d - Type 0
CFG_WRITE.VTD0	bxxx1xxxx	b111 bxxx1xxxx	Core writing to Card's PCICFG space Intel VT-d - Type 0
CFG_WRITE.PART2	bxxx1xxxx	b111 bxxxxx1xx	Core writing to Card's PCICFG space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
CFG_WRITE.PART0	bxxx1xxxx	b111 bxxxxxxx1	Core writing to Card's PCICFG space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
CFG_WRITE.PART3	bxxx1xxxx	b111 bxxxx1xxx	Core writing to Card's PCICFG space x4 card is plugged in to slot 3
CFG_WRITE.PART1	bxxx1xxxx	b111 bxxxxxx1x	Core writing to Card's PCICFG space x4 card is plugged in to slot 1
CFG_WRITE.VTD1	bxxx1xxxx	b111 bxx1xxxxx	Core writing to Card's PCICFG space Intel VT-d - Type 1
IO_WRITE.PART1	bxx1xxxxx	b111 bxxxxxx1x	Core writing to Card's IO space x4 card is plugged in to slot 1
IO_WRITE.PART0	bxx1xxxxx	b111 bxxxxxxx1	Core writing to Card's IO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
IO_WRITE.VTD1	bxx1xxxxx	b111 bxx1xxxxx	Core writing to Card's IO space Intel VT-d - Type 1
IO_WRITE.PART3	bxx1xxxxx	b111 bxxxx1xxx	Core writing to Card's IO space x4 card is plugged in to slot 3
IO_WRITE.VTD0	bxx1xxxxx	b111 bxxx1xxxx	Core writing to Card's IO space Intel VT-d - Type 0
IO_WRITE.PART2	bxx1xxxxx	b111 bxxxxx1xx	Core writing to Card's IO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
CFG_READ.PART0	bx1xxxxxx	b111 bxxxxxxx1	Core reading from Card's PCICFG space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
CFG_READ.VTD0	bx1xxxxxx	b111 bxxx1xxxx	Core reading from Card's PCICFG space Intel VT-d - Type 0
CFG_READ.PART1	bx1xxxxxx	b111 bxxxxxx1x	Core reading from Card's PCICFG space x4 card is plugged in to slot 1
CFG_READ.VTD1	bx1xxxxxx	b111 bxx1xxxxx	Core reading from Card's PCICFG space Intel VT-d - Type 1
CFG_READ.PART3	bx1xxxxxx	b111 bxxxx1xxx	Core reading from Card's PCICFG space x4 card is plugged in to slot 3
CFG_READ.PART2	bx1xxxxxx	b111 bxxxxx1xx	Core reading from Card's PCICFG space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
IO_READ.VTD1	b1xxxxxxx	b111 bxx1xxxxx	Core reading from Card's IO space Intel VT-d - Type 1

Table 2-145. Unit Masks for TXN_REQ_BY_CPU (Sheet 3 of 3)

Extension	umask [15:8]	xtra [46:36]	Description
IO_READ.PART2	b1xxxxxxx	b111 bxxxxx1xx	Core reading from Card's IO space x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
IO_READ.PART0	b1xxxxxxx	b111 bxxxxxxx1	Core reading from Card's IO space x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
IO_READ.VTD0	b1xxxxxxx	b111 bxxx1xxxx	Core reading from Card's IO space Intel VT-d - Type 0
IO_READ.PART1	b1xxxxxxx	b111 bxxxxxx1x	Core reading from Card's IO space x4 card is plugged in to slot 1
IO_READ.PART3	b1xxxxxxx	b111 bxxxx1xxx	Core reading from Card's IO space x4 card is plugged in to slot 3

TXN_REQ_OF_CPU

- **Title:** Number Transactions requested of the CPU
- **Category:** Transaction Events
- **Event Code:** 0x84
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Also known as Inbound. Number of 64 byte cache line requests initiated by the attached device.
- **NOTE:** Unlike free running counters, Mem Read and Peer read subevents count requests not completions. Peer R/W subevents do not include confined P2P traffic.

Table 2-146. Unit Masks for TXN_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
MEM_WRITE.PART1	bxxxxxxx1	b111 bxxxxxx1x	Card writing to DRAM x4 card is plugged in to slot 1
MEM_WRITE.PART2	bxxxxxxx1	b111 bxxxxx1xx	Card writing to DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MEM_WRITE.PART3	bxxxxxxx1	b111 bxxxx1xxx	Card writing to DRAM x4 card is plugged in to slot 3
MEM_WRITE.VTD0	bxxxxxxx1	b111 bxxx1xxxx	Card writing to DRAM Intel VT-d - Type 0
MEM_WRITE.VTD1	bxxxxxxx1	b111 bxx1xxxxx	Card writing to DRAM Intel VT-d - Type 1
MEM_WRITE.PART0	bxxxxxxx1	b111 bxxxxxxx1	Card writing to DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_WRITE.PART3	bxxxxxx1x	b111 bxxxx1xxx	Card writing to another Card (same or different stack) x4 card is plugged in to slot 3
PEER_WRITE.VTD0	bxxxxxx1x	b111 bxxx1xxxx	Card writing to another Card (same or different stack) Intel VT-d - Type 0
PEER_WRITE.PART1	bxxxxxx1x	b111 bxxxxxx1x	Card writing to another Card (same or different stack) x4 card is plugged in to slot 1
PEER_WRITE.VTD1	bxxxxxx1x	b111 bxx1xxxxx	Card writing to another Card (same or different stack) Intel VT-d - Type 1
PEER_WRITE.PART2	bxxxxxx1x	b111 bxxxxx1xx	Card writing to another Card (same or different stack) x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2

Table 2-146. Unit Masks for TXN_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
PEER_WRITE.PART0	bxxxxxx1x	b111 bxxxxxxx1	Card writing to another Card (same or different stack) x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MEM_READ.PART2	bxxxxx1xx	b111 bxxxxx1xx	Card reading from DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MEM_READ.PART1	bxxxxx1xx	b111 bxxxxxx1x	Card reading from DRAM x4 card is plugged in to slot 1
MEM_READ.VTD1	bxxxxx1xx	b111 bxx1xxxxx	Card reading from DRAM Intel VT-d - Type 1
MEM_READ.VTD0	bxxxxx1xx	b111 bxxx1xxxx	Card reading from DRAM Intel VT-d - Type 0
MEM_READ.PART3	bxxxxx1xx	b111 bxxxx1xxx	Card reading from DRAM x4 card is plugged in to slot 3
MEM_READ.PART0	bxxxxx1xx	b111 bxxxxxxx1	Card reading from DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
PEER_READ.PART2	bxxxx1xxx	b111 bxxxxx1xx	Card reading from another Card (same or different stack) x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
PEER_READ.VTD0	bxxxx1xxx	b111 bxxx1xxxx	Card reading from another Card (same or different stack) Intel VT-d - Type 0
PEER_READ.PART3	bxxxx1xxx	b111 bxxxx1xxx	Card reading from another Card (same or different stack) x4 card is plugged in to slot 3
PEER_READ.PART1	bxxxx1xxx	b111 bxxxxxx1x	Card reading from another Card (same or different stack) x4 card is plugged in to slot 1
PEER_READ.VTD1	bxxxx1xxx	b111 bxx1xxxxx	Card reading from another Card (same or different stack) Intel VT-d - Type 1
PEER_READ.PART0	bxxxx1xxx	b111 bxxxxxxx1	Card reading from another Card (same or different stack) x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
ATOMIC.VTD1	bxxx1xxxx	b111 bxx1xxxxx	Atomic requests targeting DRAM Intel VT-d - Type 1
ATOMIC.PART3	bxxx1xxxx	b111 bxxxx1xxx	Atomic requests targeting DRAM x4 card is plugged in to slot 3
ATOMIC.VTD0	bxxx1xxxx	b111 bxxx1xxxx	Atomic requests targeting DRAM Intel VT-d - Type 0
ATOMIC.PART1	bxxx1xxxx	b111 bxxxxxx1x	Atomic requests targeting DRAM x4 card is plugged in to slot 1
ATOMIC.PART0	bxxx1xxxx	b111 bx1xxxxxx	Atomic requests targeting DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
ATOMIC.PART2	bxxx1xxxx	b111 bxxxxx1xx	Atomic requests targeting DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
ATOMICCMP.PART3	bxx1xxxxx	b111 bxxxx1xxx	Completion of atomic requests targeting DRAM x4 card is plugged in to slot 3
ATOMICCMP.PART1	bxx1xxxxx	b111 bxxxxxx1x	Completion of atomic requests targeting DRAM x4 card is plugged in to slot 1

Table 2-146. Unit Masks for TXN_REQ_OF_CPU

Extension	umask [15:8]	xtra [46:36]	Description
ATOMICCMP.PART0	bxx1xxxxx	b111 bxxxxxxx1	Completion of atomic requests targeting DRAM x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
ATOMICCMP.PART2	bxx1xxxxx	b111 bxxxxx1xx	Completion of atomic requests targeting DRAM x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MSG.PART2	bx1xxxxxx	b111 bxxxxx1xx	Messages x8 card plugged in to Lane 2/3, Or x4 card is plugged in to slot 2
MSG.PART3	bx1xxxxxx	b111 bxxxx1xxx	Messages x4 card is plugged in to slot 3
MSG.PART0	bx1xxxxxx	b111 bxxxxxxx1	Messages x16 card plugged in to stack, Or x8 card plugged in to Lane 0/1, Or x4 card is plugged in to slot 0
MSG.PART1	bx1xxxxxx	b111 bxxxxxx1x	Messages x4 card is plugged in to slot 1
MSG.VTD1	bx1xxxxxx	b111 bxx1xxxxx	Messages Intel VT-d - Type 1
MSG.VTD0	bx1xxxxxx	b111 bxxx1xxxx	Messages Intel VT-d - Type 0

VTD_ACCESS

- **Title:** Intel VT-d Access
- **Category:** Intel VT-d Events
- **Event Code:** 0x41
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**
- **NOTE:** .L4_PAGE_HIT + L4_MISS == # VT-D Lookups? .CTXT_MISS + .L1_MISS + .L2_MISS + .L3_MISS + .L4_MISS == # Page Walker Reads Issued by VT-d?

Table 2-147. Unit Masks for VTD_ACCESS

Extension	umask [15:8]	Description
L4_PAGE_HIT	bxxxxxxx1	Vtd hit
CTXT_MISS	bxxxxxx1x	context cache miss
L1_MISS	bxxxxx1xx	L1 miss
L2_MISS	bxxxx1xxx	L2 miss
L3_MISS	bxxx1xxxx	L3 miss
TLB_MISS	bxx1xxxxx	TLB miss
TLB_FULL	bx1xxxxxx	TLB is full
TLB1_MISS	b1xxxxxxx	TLB miss

VTD_OCCUPANCY

- **Title:** Intel VT-d Occupancy
- **Category:** Intel VT-d Events
- **Event Code:** 0x40
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

2.5 IRP Performance Monitoring

IRP is responsible for maintaining coherency for IIO traffic targeting coherent memory.

2.5.1 IRP Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each IRP Box supports event monitoring through two 48b wide counters (IRP{5-0}_MSR_PMON_CTR/CTL{1:0}). Each of these counters can be programmed to count any IRP event. The IRP counters can increment by a maximum of 7b per cycle.

IRP0 is part of the CBDMA Stack.

IRP1-3 is part of PCIe stack 0-2.

IRP4 is part of the MCP stack 0

2.5.2 IRP Performance Monitoring Events

IRP provides events to track information related to all the traffic passing through it's boundaries.

- Write Cache Occupancy
- Ingress/Egress Traffic - by Ring Type
- Stalls awaiting Credits

2.5.3 IRP Box Events Ordered By Code

The following table summarizes the directly measured IRP Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CLOCKTICKS	0x01		0	IRP Clocks
TxC_BL_DRS_INSERTS	0x02	0-1	0	BL DRS Egress Inserts
TxC_BL_NCB_INSERTS	0x03	0-1	0	BL NCB Egress Inserts
TxC_BL_NCS_INSERTS	0x04	0-1	0	BL NCS Egress Inserts
TxC_BL_DRS_CYCLES_FULL	0x05	0-1	0	BL DRS Egress Cycles Full
TxC_BL_NCB_CYCLES_FULL	0x06	0-1	0	BL NCB Egress Cycles Full
TxC_BL_NCS_CYCLES_FULL	0x07	0-1	0	BL NCS Egress Cycles Full
TxC_BL_DRS_OCCUPANCY	0x08	0-1	0	BL DRS Egress Occupancy
TxC_BL_NCB_OCCUPANCY	0x09	0-1	0	BL NCB Egress Occupancy
TxC_BL_NCS_OCCUPANCY	0x0a	0-1	0	BL NCS Egress Occupancy
TxC_AK_INSERTS	0x0b	0-1	0	AK Egress Allocations
TxS_REQUEST_OCCUPANCY	0x0c	0-1	0	Outbound Request Queue Occupancy
TxS_DATA_INSERTS_NCB	0x0d	0-1	0	Outbound Read Requests
TxS_DATA_INSERTS_NCS	0x0e	0-1	0	Outbound Read Requests

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CACHE_TOTAL_OCCUPANCY	0x0f	0-1	0	Total Write Cache Occupancy
COHERENT_OPS	0x10	0-1	0	Coherent Ops
TRANSACTIONS	0x11	0-1	0	Inbound Transaction Count
SNOOP_RESP	0x12	0-1	0	Snoop Responses
P2P_TRANSACTIONS	0x13		0	P2P Transactions
P2P_INSERTS	0x14		0	P2P Requests
P2P_OCCUPANCY	0x15		0	P2P Occupancy
FAF_TRANSACTIONS	0x16		0	FAF allocation -- sent to ADQ
FAF_FULL	0x17		0	FAF RF full
FAF_INSERTS	0x18		0	FAF - request insert from TC.
FAF_OCCUPANCY	0x19		0	FAF occupancy
TxR2_AD_STALL_CREDIT_CYCLES	0x1a	0-1	0	No AD Egress Credit Stalls
TxR2_BL_STALL_CREDIT_CYCLES	0x1b	0-1	0	No BL Egress Credit Stalls
MISC0	0x1c	0-1	0	Misc Events - Set 0
MISC1	0x1d	0-1	0	Misc Events - Set 1
IRP_ALL	0x1e		0	

2.5.4 IRP Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the IRP Box.

CACHE_TOTAL_OCCUPANCY

- **Title:** Total Write Cache Occupancy
- **Category:** WRITE_CACHE Events
- **Event Code:** 0x0f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Accumulates the number of reads and writes that are outstanding in the uncore in each cycle. This is effectively the sum of the READ_OCCUPANCY and WRITE_OCCUPANCY events.

Table 2-148. Unit Masks for CACHE_TOTAL_OCCUPANCY

Extension	umask [15:8]	Description
ANY	b00000001	Any Source Tracks all requests from any source port.
IV_Q	b00000010	Snoops
MEM	b00000100	Mem

CLOCKTICKS

- **Title:** IRP Clocks
- **Category:** CLOCK Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

COHERENT_OPS

- **Title:** Coherent Ops
- **Category:** Coherency Events
- **Event Code:** 0x10
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Counts the number of coherency related operations servied by the IRP

Table 2-149. Unit Masks for COHERENT_OPS

Extension	umask [15:8]	Description
PCIRDCUR	bxxxxxxx1	PCIRdCur
CRD	bxxxxxx1x	CRd
DRD	bxxxxx1xx	DRd
RFO	bxxxx1xxx	RFO
PCITOM	bxxx1xxxx	PCIItoM
PCIDCAHINT	bxx1xxxxx	PCIDCAHin5t
WBMTOI	bx1xxxxxx	WbMtoI
CLFLUSH	b1xxxxxxx	CLFlush

FAF_FULL

- **Title:** FAF RF full
- **Category:** FAF Events
- **Event Code:** 0x17
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

FAF_INSERTS

- **Title:** FAF - request insert from TC.
- **Category:** FAF Events
- **Event Code:** 0x18
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

FAF_OCCUPANCY

- **Title:** FAF occupancy
- **Category:** FAF Events
- **Event Code:** 0x19
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

FAF_TRANSACTIONS

- **Title:** FAF allocation -- sent to ADQ
- **Category:** FAF Events
- **Event Code:** 0x16
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

IRP_ALL

- **Title:**
- **Category:** IRP Buffer Events
- **Event Code:** 0x1e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-150. Unit Masks for IRP_ALL

Extension	umask [15:8]	Description
INBOUND_INSERTS	b00000001	All Inserts Inbound (p2p + faf + cset)
OUTBOUND_INSERTS	b00000010	All Inserts Outbound (BL, AK, Snoops)

MISC0

- **Title:** Misc Events - Set 0
- **Category:** MISC Events
- **Event Code:** 0x1c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

Table 2-151. Unit Masks for MISC0

Extension	umask [15:8]	Description
FAST_REQ	b000000x1	Fastpath Requests
FAST_REJ	b0000001x	Fastpath Rejects
2ND_RD_INSERT	bx00xx100	Cache Inserts of Read Transactions as Secondary
2ND_WR_INSERT	bx00x1x00	Cache Inserts of Write Transactions as Secondary
2ND_ATOMIC_INSERT	bx001xx00	Cache Inserts of Atomic Transactions as Secondary
FAST_XFER	bxx100000	Fastpath Transfers From Primary to Secondary
PF_ACK_HINT	bx1x00000	Prefetch Ack Hints From Primary to Secondary
UNKNOWN	b1xx00000	

MISC1

- **Title:** Misc Events - Set 1
- **Category:** MISC Events
- **Event Code:** 0x1d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

Table 2-152. Unit Masks for MISC1

Extension	umask [15:8]	Description
SLOW_I	b000xxxx1	Slow Transfer of I Line Snoop took cacheline ownership before write from data was committed.
SLOW_S	b000xxx1x	Slow Transfer of S Line Secondary received a transfer that did not have sufficient MESI state

Table 2-152. Unit Masks for MISC1

Extension	umask [15:8]	Description
SLOW_E	b000xx1xx	Slow Transfer of E Line Secondary received a transfer that did have sufficient MESI state
SLOW_M	b000x1xxx	Slow Transfer of M Line Snoop took cacheline ownership before write from data was committed.
LOST_FWD	b0001xxxx	Lost Forward Snoop pulled away ownership before a write was committed
SEC_RCVD_INVLD	bxx1x0000	Received Invalid Secondary received a transfer that did not have sufficient MESI state
SEC_RCVD_VLD	bx1xx0000	Received Valid Secondary received a transfer that did have sufficient MESI state

P2P_INSERTS

- **Title:** P2P Requests
- **Category:** P2P Events
- **Event Code:** 0x14
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** P2P requests from the ITC

P2P_OCCUPANCY

- **Title:** P2P Occupancy
- **Category:** P2P Events
- **Event Code:** 0x15
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** P2P B & S Queue Occupancy

P2P_TRANSACTIONS

- **Title:** P2P Transactions
- **Category:** P2P Events
- **Event Code:** 0x13
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-153. Unit Masks for P2P_TRANSACTIONS

Extension	umask [15:8]	Description
RD	bxxxxxxx1	P2P reads
WR	bxxxxxx1x	P2P Writes
MSG	bxxxxx1xx	P2P Message
CMPL	bxxxx1xxx	P2P completions
REM	bxxx1xxxx	Match if remote only
REM_AND_TGT_MATCH	bxx1xxxxx	match if remote and target matches
LOC	bx1xxxxxx	match if local only
LOC_AND_TGT_MATCH	b1xxxxxxx	match if local and target matches

SNOOP_RESP

- **Title:** Snoop Responses
- **Category:** TRANSACTIONS Events
- **Event Code:** 0x12
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**
- **NOTE:** The first 4 subevent bits are the Responses to the Code/Data/Invalid Snoops represented by the last 3 subevent bits. At least 1 of the bottom 4 bits must be combined with 1 of the top 3 bits to obtain counts. Unsure which combinations are possible.

Table 2-154. Unit Masks for SNOOP_RESP

Extension	umask [15:8]	Description
MISS	bxxxxxxx1	Miss
HIT_I	bxxxxxx1x	Hit I
HIT_ES	bxxxxx1xx	Hit E or S
HIT_M	bxxxx1xxx	Hit M
SNPCODE	bxxx1xxxx	SnpCode
SNPDATA	bxx1xxxxx	SnpData
SNPINV	bx1xxxxxx	SnpInv

TRANSACTIONS

- **Title:** Inbound Transaction Count
- **Category:** TRANSACTIONS Events
- **Event Code:** 0x11
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Counts the number of "Inbound" transactions from the IRP to the Uncore. 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.
- **NOTE:** Bit 7 is a filter that can be applied to the other subevents. Meaningless by itself.

Table 2-155. Unit Masks for TRANSACTIONS

Extension	umask [15:8]	Filter Dep	Description
READS	bxxxxxxx1		Reads Tracks only read requests (not including read prefetches).
WRITES	bxxxxxx1x		Writes Trackes only write requests. Each write request should have a prefetch, so there is no need to explicitly track these requests. For writes that are tickled and have to retry, the counter will be incremented for each retry.
RD_PREF	bxxxxx1xx		Read Prefetches Tracks the number of read prefetches.
WR_PREF	bxxxx1xxx		Write Prefetches Tracks the number of write prefetches.

Table 2-155. Unit Masks for TRANSACTIONS

Extension	umask [15:8]	Filter Dep	Description
ATOMIC	bxxx1xxxx		Atomic Tracks the number of atomic transactions
OTHER	bxx1xxxxx		Other Tracks the number of 'other' kinds of transactions.

TxC_AK_INSERTS

- **Title:** AK Egress Allocations
- **Category:** AK Egress Events
- **Event Code:** 0x0b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_DRS_CYCLES_FULL

- **Title:** BL DRS Egress Cycles Full
- **Category:** BL Egress Events
- **Event Code:** 0x05
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_DRS_INSERTS

- **Title:** BL DRS Egress Inserts
- **Category:** BL Egress Events
- **Event Code:** 0x02
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_DRS_OCCUPANCY

- **Title:** BL DRS Egress Occupancy
- **Category:** BL Egress Events
- **Event Code:** 0x08
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCB_CYCLES_FULL

- **Title:** BL NCB Egress Cycles Full
- **Category:** BL Egress Events
- **Event Code:** 0x06
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCB_INSERTS

- **Title:** BL NCB Egress Inserts
- **Category:** BL Egress Events
- **Event Code:** 0x03
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCB_OCCUPANCY

- **Title:** BL NCB Egress Occupancy
- **Category:** BL Egress Events
- **Event Code:** 0x09
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCS_CYCLES_FULL

- **Title:** BL NCS Egress Cycles Full
- **Category:** BL Egress Events
- **Event Code:** 0x07
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCS_INSERTS

- **Title:** BL NCS Egress Inserts
- **Category:** BL Egress Events
- **Event Code:** 0x04
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxC_BL_NCS_OCCUPANCY

- **Title:** BL NCS Egress Occupancy
- **Category:** BL Egress Events
- **Event Code:** 0x0a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:**

TxR2_AD_STALL_CREDIT_CYCLES

- **Title:** No AD Egress Credit Stalls
- **Category:** STALL_CYCLES Events
- **Event Code:** 0x1a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Counts the number times when it is not possible to issue a request to the R2PCIe because there are no AD Egress Credits available.

TxR2_BL_STALL_CREDIT_CYCLES

- **Title:** No BL Egress Credit Stalls
- **Category:** STALL_CYCLES Events
- **Event Code:** 0x1b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Counts the number times when it is not possible to issue data to the R2PCIe because there are no BL Egress Credits available.

TxS_DATA_INSERTS_NCB

- **Title:** Outbound Read Requests
- **Category:** OUTBOUND_REQUESTS Events
- **Event Code:** 0x0d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1

- **Definition:** Counts the number of requests issued to the switch (towards the devices).

TxS_DATA_INSERTS_NCS

- **Title:** Outbound Read Requests
- **Category:** OUTBOUND_REQUESTS Events
- **Event Code:** 0x0e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Counts the number of requests issued to the switch (towards the devices).

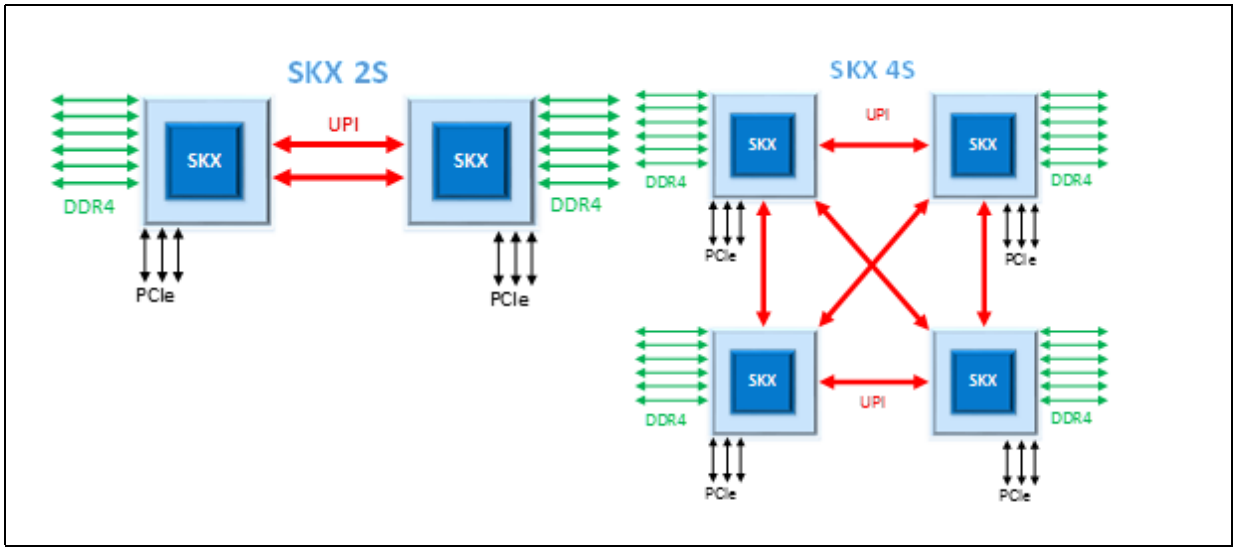
TxS_REQUEST_OCCUPANCY

- **Title:** Outbound Request Queue Occupancy
- **Category:** OUTBOUND_REQUESTS Events
- **Event Code:** 0x0c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-1
- **Definition:** Accumulates the number of outstanding outbound requests from the IRP to the switch (towards the devices). This can be used in conjunction with the allocations event in order to calculate average latency of outbound requests.

2.6 Intel® UPI Link Layer Performance Monitoring

Intel® Xeon® Processor Scalable Memory Family uses a new coherent interconnect for scaling to multiple sockets known as Intel® Ultra Path Interconnect (Intel UPI). Intel® UPI technology provides a cache coherent socket to socket external communication interface between processors. The processor implements 2 Intel® UPI links on -EP or 3 Intel® UPI links on -EX. Figures below show a 2-socket or 4-socket server systems, as examples. Intel® UPI is also used as a coherent communication interface between processors and OEM 3rd party Node Controllers (XNC).

Figure 2-9. Intel® Xeon® Processor Scalable Memory Family Intel UPI for Basic Multi Socket Configurations



There are two Intel® UPI agents that share a single mesh stop and a third agent in the EX part with its own mesh stop. These links can be connected to a single destination (such as in DP), or can be connected to two separate destinations (4s Ring or sDP). Therefore, it will be necessary to count Intel® UPI statistics for each agent separately.

The Intel® UPI module supports up to 2 Intel® UPI links (per mesh stop) and is comprised of the following layers for each Intel® UPI link:

- **Physical Layer** - The Intel® UPI Physical layer (PHY) is a hardware layer that lies between the Link layer above it, and the physical wires that connect to other devices. The Physical layer is further sub-divided into the logical and electrical sub-blocks.
- **Link Layer** - The Intel® UPI link layer bi-directionally converts between protocol layer messages and Link layer Flits, passes them through shared buffers, and manages the flow control information per virtual channel. The link layer also detects errors and retransmits packets on errors.
- **Routing Layer** - The Routing Layer is distributed among all agents that send Intel® UPI messages on the mesh (Intel® UPI, CHA, PCIe, IMC). The Intel® UPI Module provides a routing function to determine the correct mesh stop from which to forward a given packet.
- **Protocol Layer** - The Intel® UPI module does not implement the Protocol Layer. A protocol agent is a proxy for some entity which injects, generates, or services Intel® UPI transactions such as memory requests, interrupts, etc. The Protocol Layer is implemented in the following modules: Coherency Home Agent(CHA), PCIe, Configuration Agent (Ubox). A Coherency agent (CA) in the CHA both generates requests and services snoops. A Home Agent (HA) in the CHA services requests, generates snoops, and resolves conflicts. CHA will sometimes behave as CA, sometimes as a HA, and sometimes both at the same time. The PCIe module handles most IO proxy responsibilities. The Ubox handles internal configuration space and some other interrupt and messaging flows. A HA acts as proxy for DRAM, while the PCIe/Ubox handle all non-DRAM (NCB and NCS) requests.

The Intel UPI Subsystem implements a UPI port as a bi-directional interface, where each direction is 20 lanes wide. As such, the UPI Subsystem implements both transmit and receive interfaces and functionality. The Intel® Xeon® Processor Scalable Memory Family supports two UPI operational speeds - 9.6 GT/s or 10.4 GT/s.

The Intel® UPI Link Layer is responsible for packetizing requests from the caching agent on the way out to the system interface. The UPI link layer processes information at a flit granularity.

On Intel® Xeon® Processor Scalable Memory Family, Intel® UPI is split into two layers – M3UPI and the UPI Link Layer. M3UPI (Section 2.8, “M3UPI Performance Monitoring”) provides the interface to the Mesh for the Link Layer. M3UPI converts mesh packets (received from CHA) into UPI flits and vice-versa. M3UPI ingress is the point where remote UPI VNA/VN0 link credits are acquired. UPI Link Layer passes flits through shared buffers, and manages their flow control. The Link layer also detects errors, and on their occurrence, retransmits affected packets (corrected errors). Finally, the Link layer delivers packets to the caching agent.

A single Intel UPI flit can pack up to 3 mesh packets in 3 slots. The Intel® UPI Link Layer has the ability to transmit up to 3 mesh packets per cycle in each direction. In order to accommodate this, many of the events in the Link Layer can increment by 0, 1, or 2 in each cycle. It is not possible to monitor Rx (received) and Tx (transmitted) flit information at the same time on the same counter.

Note: Flit slots are not symmetric in their ability to relay flit traffic. Any analysis of UPI BW should keep this in mind.

2.6.1 Intel® UPI Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each Intel® UPI Link supports event monitoring through four 48b wide counters (U_Ly_PCI_PMON_CTR/CTL{3:0}). Each of these four counters can be programmed to count any Intel® UPI event. The Intel® UPI counters can increment by a maximum of 9b per cycle.

Each Intel® UPI Link also includes a mask/match register that allows a user to match packets, according to various standard packet fields such as message class, opcode, etc, as they leave the UPI Link.

2.6.2 Additional Intel® UPI Performance Monitoring

2.6.2.1 Intel® UPI PMON Counter Control - Difference From Baseline

The following table defines the difference in the layout of the Intel® UPI performance monitor control registers from the baseline presented in Chapter 1.

Figure 2-10. Intel UPI Counter Control Register for Intel® Xeon® Processor Scalable Memory Family

Table 2-156. U_Ly_PCI_PMON_CTL{3-0} Difference from Baseline – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
ig	63:56	RV	0	Reserved
umask_ext	55:32	RW-V	0	Umask Extension - Used for TxL/RxL_BASIC_HDR_MATCH events - allows a opcode, message class, local/remote and a couple other types of matching. - See Table 2-158, "UmaskExt Field Details for TxL/RxL_BASIC_HDR_MATCH.* events"

Table 2-157. Additional Intel® UPI Performance Monitoring Registers (PCICFG) (Sheet 1 of 2)

Register Name	PCICFG Address	Size (bits)	Description
PCICFG Base Address	Dev:Func DeviceID		
UPI0 Link 0 PMON Registers	B3:D14:F0 0x2058		
UPI0 Link 1 PMON Registers	B3:D15:F0 0x2058		

Table 2-157. Additional Intel® UPI Performance Monitoring Registers (PCICFG) (Sheet 2 of 2)

Register Name	PCICFG Address	Size (bits)	Description
UPI1 Link 2 PMON Registers	B3:D16:F0 0x2058		

2.6.2.2 Intel® UPI Basic Header Packet Matching

In Intel® Xeon® Processor Scalable Memory Family, the ability to more precisely breakdown traffic by traffic type has been moved to the individual counter control registers.

The matching extension is enabled when measuring events 0x04 (TxL_BASIC_HDR_MATCH) and 0x05 (RxL_BASIC_HDR_MATCH). Filtering can be performed according to the packet's Opcode, Message Class, DNID, RCSNID, etc. Details below.

Table 2-158. UmaskExt Field Details for TxL/RxL_BASIC_HDR_MATCH.* events (Sheet 1 of 2)

Field	Bits	Description
llcrd_implnull	55	LLCRD Implied Null - NOTE: Only applies to Slot 2 MC + Opcode Match
llcrd_non0	54	LLCRD Nonzero - NOTE: Only applies to Slot 2 MC + Opcode Match
slot2	53	Slot 2 - NOTE: bits 53:51 == 0x0 is equivalent to bits 53:51 == 0x7. i.e. if NO slot bits are set, all slots are enabled and used for the match.
slot1	52	Slot 1
slot0	51	Slot 0
en_rcsnid	50	Enable RCS Node ID Match - NOTE: Assumes matching applied to MC+Opcode combinations that have an R/C/S NID
rcsnid	49:46	RCS Node ID
en_dnidd	45	Enable Destination Node ID Match
ig	44	Reserved
dnid	43:40	Destination Node ID
isoch	39	Isoch
sglslot	38	Basic Header - Single Slot - Single / Dual are mutually exclusive. Only valid to set 1 bit.
dualslot	37	Basic Header - Dual Slot
nondata	36	Basic Header - NonData - Data / NonData are mutually exclusive. Only valid to set 1 bit.
data	35	Basic Header - Data

Table 2-158. UmaskExt Field Details for TxL/RxL_BASIC_HDR_MATCH.* events (Sheet 2 of 2)

Field	Bits	Description
rem	34	Remote - received packet is targeting a CHA in a different socket. - NOTE: Only valid for Rx event. - Local / Remote are mutually exclusive. Only valid to set 1 bit.
loc	33	Local - received packet is targeting a CHA in this socket. - NOTE: Only valid for Rx event.
opc	32	Enable Opcode Matching - See Table 3-6, "Intel UPI Opcode Match by Message Class" for available opcodes - The 4b opcode encoding should be written to the upper 4b of the umask field. Counter control bits 15:12

2.6.3 Intel® UPI LL Performance Monitoring Events

The Intel® UPI Link Layer provides events to gather information on topics such as:

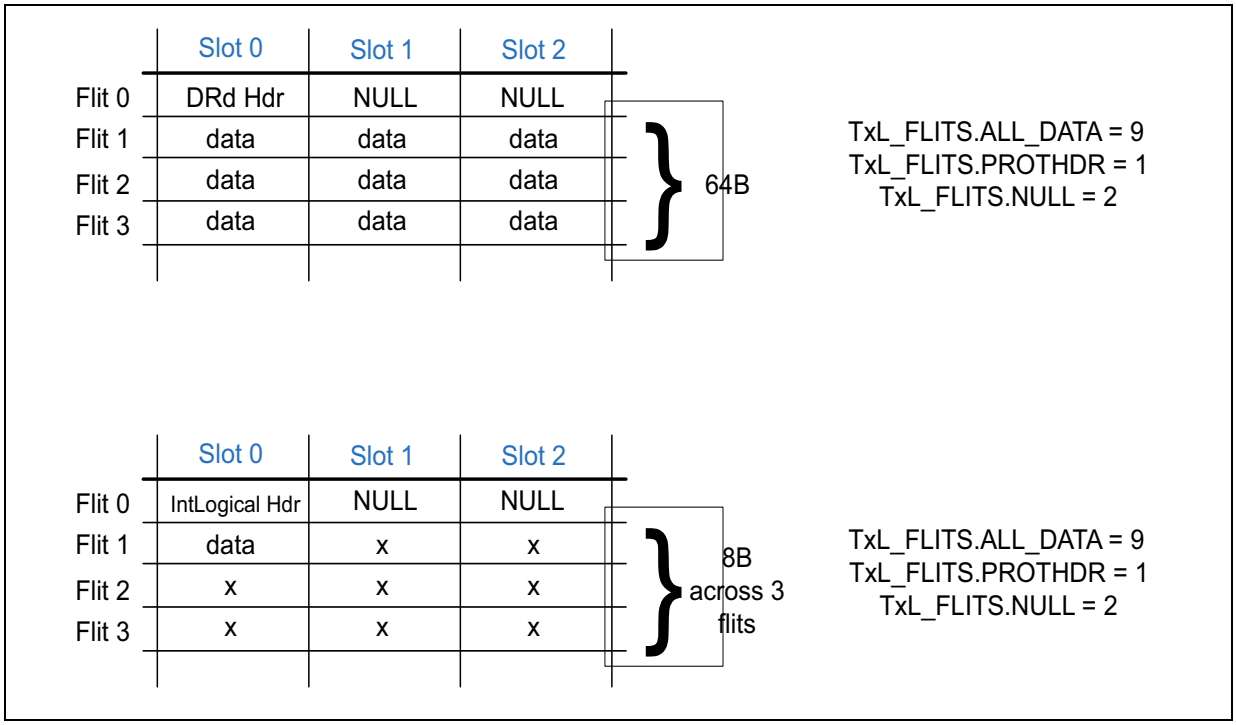
- Tracking incoming (mesh bound)/outgoing (system bound) transactions,
- Various queues that track those transactions,
- The Link Layer's power consumption as expressed by the time spent in the Link power states L0p (60% of lanes are disabled, i.e. it takes 2 of 6 cycles to transmit a flit instead of 5 of 6 cycles).
- A variety of static events such as Direct2Core and Direct2UPI statistics and when output credit is unavailable.
- Of particular interest, total link utilization may be calculated by capturing and subtracting transmitted/received idle flits from Intel® UPI clocks.

Many of these events can be further broken down by message class, including link utilization.

A quick illustration on calculating UPI Bandwidth. Here are two basic examples. The first is a typical DRd (data read) packet and the other is an IntLogical (logically addressed interrupt) packet. The point is , in both these cases, the number of flits sent are the same even in the rare case a full cacheline's worth of data isn't transmitted.

When measuring the amount of bandwidth consumed by transmission of the data (i.e. NOT including the header), it should be .ALL_DATA / 9 * 64B. .

Figure 2-11. Calculating UPI Bandwidth. Quick example



Note: If the DRd header takes up more than one slot or some other packet header (one without data) is packed in with the DRd Hdr flit, then .PROTHDR >1 and .NULL <2.

2.6.4 UPI LL Box Events Ordered By Code

The following table summarizes the directly measured UPI LL Box events.

Symbol Name	Event Code	Ctrs	Extra Select Bit	Max Inc/ Cyc	Description
CLOCKTICKS	0x01	0-3	0	0	Number of kfclks
TxL_FLITS	0x02	0-3	0	0	Valid Flits Sent
RxL_FLITS	0x03	0-3	0	0	Valid Flits Received
TxL_BASIC_HDR_MATCH	0x04	0-3	0	0	Matches on Transmit path of a UPI Port
RxL_BASIC_HDR_MATCH	0x05	0-3	0	0	Matches on Receive path of a UPI Port
DIRECT_ATTEMPTS	0x12	0-3	0	0	Direct packet attempts
M3_BYP_BLOCKED	0x14	0-3	0	0	
M3_RXQ_BLOCKED	0x15	0-3	0	0	
M3_CRD_RETURN_BLOCKED	0x16	0-3	0	0	
FLOWQ_NO_VNA_CRD	0x18	0-3	0	0	
PHY_INIT_CYCLES	0x20	0-3	0	0	Cycles where phy is not in L0, L0c, L0p, L1
L1_POWER_CYCLES	0x21	0-3	0	0	Cycles in L1
POWER_L1_REQ	0x22	0-3	0	0	L1 Req (same as L1 Ack).

Symbol Name	Event Code	Ctrs	Extra Select Bit	Max Inc/ Cyc	Description
POWER_L1_NACK	0x23	0-3	0	0	L1 Req Nack
RxL0_POWER_CYCLES	0x24	0-3	0	0	Cycles in L0. Receive side.
RxL0P_POWER_CYCLES	0x25	0-3	0	0	Cycles link in L0p. Receive side.
TxL0_POWER_CYCLES	0x26	0-3	0	0	Cycles in L0. Transmit side.
TxL0P_POWER_CYCLES	0x27	0-3	0	0	Cycles in L0p. Transmit side.
TxL0P_POWER_CYCLES_LL_ENTER	0x28	0-3	0	0	
TxL0P_POWER_CYCLES_M3_EXIT	0x29	0-3	0	0	
TxL0P_CLK_ACTIVE	0x2a	0-3	0	0	
RxL_INSERTS	0x30	0-3	0	0	RxQ Flit Buffer Allocations
RxL_BYPASSED	0x31	0-3	0	0	RxQ Flit Buffer Bypassed
RxL_OCCUPANCY	0x32	0-3	0	0	RxQ Occupancy - All Packets
RxL_SLOT_BYPASS	0x33	0-3	0	0	
RxL_CREDITS_CONSUMED_VNA	0x38	0-3	0	0	VNA Credit Consumed
RxL_CREDITS_CONSUMED_VN0	0x39	0-3	0	0	VN0 Credit Consumed
RxL_CREDITS_CONSUMED_VN1	0x3a	0-3	0	0	VN1 Credit Consumed
TxL_INSERTS	0x40	0-3	0	0	Tx Flit Buffer Allocations
TxL_BYPASSED	0x41	0-3	0	0	Tx Flit Buffer Bypassed
TxL_OCCUPANCY	0x42	0-3	0	0	Tx Flit Buffer Occupancy
VNA_CREDIT_RETURN_OCCUPANCY	0x44	0-3	0	0	VNA Credits Pending Return - Occupancy
VNA_CREDIT_RETURN_BLOCKED_VN01	0x45	0-3	0	0	
REQ_SLOT2_FROM_M3	0x46	0-3	0	0	

2.6.5 Intel UPI LL Box Common Metrics (Derived Events)

The following table summarizes metrics commonly calculated from Intel UPI LL Box events.

Symbol Name: Definition	Equation
DRS_E_FROM_UPI: DRS response in F or E states received from Intel UPI in bytes. To calculate the total data response for each cache line state, it's necessary to add the contribution from three flavors {DataC, DataC_FrcAckCnflt, DataC_Cmp} of data response packets for each cache line state.	RxL_BASIC_HDR_MATCH.{umask,opc}={0x1C,1} * 64
DRS_M_FROM_UPI: Data Response DataM packets received from Intel UPI. Expressed in bytes	RxL_BASIC_HDR_MATCH.{umask,opc}={0x0C,1} * 64
DRS_WB_FROM_UPI: DRS writeback packets received from Intel UPI in bytes. This is the sum of Wb{I,S,E} DRS packets	DRS_WbI_FROM_UPI + DRS_WbS_FROM_UPI + DRS_WbE_FROM_UPI
DRS_WbE_FROM_UPI: DRS writeback 'change M to E state' packets received from Intel UPI in bytes	RxL_BASIC_HDR_MATCH.{umask,opc}={0x2D,1} *64

Symbol Name: Definition	Equation
DRS_WbI_FROM_UPI: DRS writeback 'change M to I state' packets received from Intel UPI in bytes	RxL_BASIC_HDR_MATCH.{umask,opc}={0x0D,1} *64
DRS_WbS_FROM_UPI: DRS writeback 'change M to S state' packets received from Intel UPI in bytes	RxL_BASIC_HDR_MATCH.{umask,opc}={0x1D,1} *64
NCB_DATA_FROM_UPI_TO_NODEx: NCB Data packets (Any - Interrupts) received from Intel UPI sent to Node ID 'x'. Expressed in bytes	RxL_BASIC_HDR_MATCH.{umask,endnid,dnid} = {0xE,1,x} * 64
PCT_LINK_CRC_RETRY_CYCLES: Percent of Cycles the Intel UPI link layer is in retry mode due to CRC errors	RxL_CRC_CYCLES_IN_LLR / CLOCKTICKS
PCT_LINK_FULL_POWER_CYCLES: Percent of Cycles the Intel UPI link is at Full Power	RxL0_POWER_CYCLES / CLOCKTICKS
PCT_LINK_HALF_DISABLED_CYCLES: Percent of Cycles the Intel UPI link in power mode where 60% of the lanes are disabled.	RxL0P_POWER_CYCLES / CLOCKTICKS
PCT_LINK_SHUTDOWN_CYCLES: Percent of Cycles the Intel UPI link is Shutdown	L1_POWER_CYCLES / CLOCKTICKS
UPI_SPEED: Intel UPI Speed - In GT/s (GigaTransfers / Second) - Max Intel UPI Bandwidth is 2 * ROUND (Intel UPI Speed , 0)	ROUND ((CLOCKTICKS / TSC) * TSC_SPEED, 0) * (8 / 1000)

2.6.6 Intel UPI LL Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the Intel UPI LL Box.

CLOCKTICKS

- **Title:** Number of kfclks
- **Category:** CFCLK Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of clocks in the Intel UPI LL. This clock runs at 1/8th the "GT/s" speed of the Intel UPI link. For example, a 8GT/s link will have qfclk or 1GHz. Current products do not support dynamic link speeds, so this frequency is fixed.

DIRECT_ATTEMPTS

- **Title:** Direct packet attempts
- **Category:** DIRECT2CORE Events
- **Event Code:** 0x12
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of Data Response(DRS) packets Intel UPI attempted to send directly to the core or to a different Intel UPI link. Note: This only counts attempts on valid candidates such as DRS packets destined for CHAs.

Table 2-159. Unit Masks for DIRECT_ATTEMPTS

Extension	umask [15:8]	Description
D2C	bxxxxxxx1	Direct 2 Core
D2U	bxxxxxx1x	Direct 2 Intel UPI

FLOWQ_NO_VNA_CRD

- **Title:**
- **Category:** LL to M3 Events
- **Event Code:** 0x18
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-160. Unit Masks for FLOWQ_NO_VNA_CRD

Extension	umask [15:8]	Description
AD_VNA_EQ0	bxxxxxxx1	
AD_VNA_EQ1	bxxxxxx1x	
AD_VNA_EQ2	bxxxxx1xx	
BL_VNA_EQ0	bxxxx1xxx	
AK_VNA_EQ0	bxxx1xxxx	
AK_VNA_EQ1	bxx1xxxxx	
AK_VNA_EQ2	bx1xxxxxx	
AK_VNA_EQ3	b1xxxxxxx	

L1_POWER_CYCLES

- **Title:** Cycles in L1
- **Category:** POWER Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of Intel UPI qfclk cycles spent in L1 power mode. L1 is a mode that totally shuts down an Intel UPI link. Use edge detect to count the number of instances when the Intel UPI link entered L1. Link power states are per link and per direction, so for example the Tx direction could be in one state while Rx was in another. Because L1 totally shuts down the link, it takes a good amount of time to exit this mode.

M3_BYP_BLOCKED

- **Title:**
- **Category:** LL to M3 Events
- **Event Code:** 0x14
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-161. Unit Masks for M3_BYP_BLOCKED

Extension	umask [15:8]	Description
FLOWQ_AD_VNA_LE2	bxxxxxxx1	
FLOWQ_BL_VNA_EQ0	bxxxxxx1x	
FLOWQ_AK_VNA_LE3	bxxxxx1xx	
BGF_CRD	bxxxx1xxx	
GV_BLOCK	bxxx1xxxx	

M3_CRD_RETURN_BLOCKED

- **Title:**
- **Category:** LL to M3 Events
- **Event Code:** 0x16
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

M3_RXQ_BLOCKED

- **Title:**
- **Category:** LL to M3 Events
- **Event Code:** 0x15
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-162. Unit Masks for M3_RXQ_BLOCKED

Extension	umask [15:8]	Description
FLOWQ_AD_VNA_LE2	bxxxxxxx1	
FLOWQ_AD_VNA_BTW_2_THRESH	bxxxxxx1x	
FLOWQ_BL_VNA_EQ0	bxxxxx1xx	
FLOWQ_BL_VNA_BTW_0_THRESH	bxxxx1xxx	
FLOWQ_AK_VNA_LE3	bxxx1xxxx	
BGF_CRD	bxx1xxxxx	
GV_BLOCK	bx1xxxxxx	

PHY_INIT_CYCLES

- **Title:** Cycles where phy is not in L0, L0c, L0p, L1
- **Category:** POWER Events
- **Event Code:** 0x20
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

POWER_L1_NACK

- **Title:** L1 Req Nack
- **Category:** POWER Events
- **Event Code:** 0x23
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times a link sends/receives a LinkReqNAck. When the Intel UPI links would like to change power state, the Tx side initiates a request to the Rx side requesting to change states. This requests can either be accepted or denied. If the Rx side replies with an Ack, the power mode will change. If it replies with NAck, no change will take place. This can be filtered based on Rx and Tx. An Rx LinkReqNAck refers to receiving an NAck (meaning this agent's Tx originally requested the power change). A Tx LinkReqNAck refers to sending this command (meaning the peer agent's Tx originally requested the power change and this agent accepted it).
- **NOTE:** L1 only

POWER_L1_REQ

- **Title:** L1 Req (same as L1 Ack).
- **Category:** POWER Events
- **Event Code:** 0x22
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times a link sends/receives a LinkReqAck. When the Intel UPI links would like to change power state, the Tx side initiates a request to the Rx side requesting to change states. This requests can either be accepted or denied. If the Rx side replies with an Ack, the power mode will change. If it replies with NAck, no change will take place. This can be filtered based on Rx and Tx. An Rx LinkReqAck refers to receiving an Ack (meaning this agent's Tx originally requested the power change). A Tx LinkReqAck refers to sending this command (meaning the peer agent's Tx originally requested the power change and this agent accepted it).
- **NOTE:** L1 only

REQ_SLOT2_FROM_M3

- **Title:**
- **Category:** VNA_CREDIT_RETURN Events
- **Event Code:** 0x46
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-163. Unit Masks for REQ_SLOT2_FROM_M3

Extension	umask [15:8]	Description
VNA	bxxxxxxx1	
VN0	bxxxxxx1x	
VN1	bxxxxx1xx	
ACK	bxxxx1xxx	

RxL0P_POWER_CYCLES

- **Title:** Cycles link in L0p. Receive side.
- **Category:** POWER_RX Events
- **Event Code:** 0x25
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3

- **Definition:** Number of Intel UPI qfclk cycles spent in L0p power mode. L0p is a mode where we disable 60% of the Intel UPI lanes, decreasing our bandwidth in order to save power. It increases snoop and data transfer latencies and decreases overall bandwidth. This mode can be very useful in NUMA optimized workloads that largely only utilize Intel UPI for snoops and their responses. Use edge detect to count the number of instances when the Intel UPI link entered L0p. Link power states are per link and per direction, so for example the Tx direction could be in one state while Rx was in another.
- **NOTE:** Using .edge_det to count transitions does not function if L1_POWER_CYCLES > 0.

RxL0_POWER_CYCLES

- **Title:** Cycles in L0. Receive side.
- **Category:** POWER_RX Events
- **Event Code:** 0x24
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of Intel UPI qfclk cycles spent in L0 power mode in the Link Layer. L0 is the default mode which provides the highest performance with the most power. Use edge detect to count the number of instances that the link entered L0. Link power states are per link and per direction, so for example the Tx direction could be in one state while Rx was in another. The phy layer sometimes leaves L0 for training, which will not be captured by this event.

RxL_BASIC_HDR_MATCH

- **Title:** Matches on Receive path of an Intel UPI Port
- **Category:** FLIT match Events
- **Event Code:** 0x05
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Filter Dependency:** CtrCtrl[55:32]
- **Definition:**
- **NOTE:** This event is subject to finer grain filtering. See doc for more information. Filters available in the counter control's umask extention field b[55:32] - message class, opcode, local, remote, datahdr, ndatahdr, dual slot header, single slot header and pe) ANDed per Slot. Then slots are ORed.

Table 2-164. Unit Masks for RxL_BASIC_HDR_MATCH

Extension	umask [15:8]	xtra [32]	Description
REQ_OPC	bXXXX1000	0x1	Request Opcode Match REQ Opcodes - Specified in Umask[7:4]
REQ	bxxxx1000	0x0	Request REQ Message Class
SNP_OPC	bXXXX1001	0x1	Snoop Opcode Match SNP Opcodes - Specified in Umask[7:4]
SNP	bxxxx1001	0x0	Snoop SNP Message Class
RSP_NODATA_OPC	bXXXX1010	0x1	Response - No Data Match Message Class - RSP
RSP_NODATA	bxxxx1010	0x0	Response - No Data Match Message Class - RSP
RSP_DATA	bxxxx1100	0x0	Response - Data Match Message Class -WB
RSP_DATA_OPC	bXXXX1100	0x1	Response - Data Match Message Class -WB
WB_OPC	bxxxx1101	0x1	Writeback Match Message Class -WB

Table 2-164. Unit Masks for RxL_BASIC_HDR_MATCH

Extension	umask [15:8]	xtra [32]	Description
WB	bxxxx1101	0x0	Writeback Match Message Class -WB
NCB	bxxxx1110	0x0	Non-Coherent Bypass Match Message Class - NCB
NCB_OPC	bxxxx1110	0x1	Non-Coherent Bypass Match Message Class - NCB
NCS	bxxxx1111	0x0	Non-Coherent Standard Match Message Class - NCS
NCS_OPC	bxxxx1111	0x1	Non-Coherent Standard Match Message Class - NCS
RSPI	b00101010	0x1	Response - Invalid
RSPCNFLT	b10101010	0x1	Response - Conflict

RxL_BYPASSED

- **Title:** RxQ Flit Buffer Bypassed
- **Category:** RXQ Events
- **Event Code:** 0x31
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that an incoming flit was able to bypass the flit buffer and pass directly and into the Egress. This is a latency optimization, and should generally be the common case. If this value is less than the number of flits transfered, it implies that there was queueing getting onto the ring, and thus the transactions saw higher latency.

Table 2-165. Unit Masks for RxL_BYPASSED

Extension	umask [15:8]	Description
SLOT0	bxxxxxxx1	Slot 0
SLOT1	bxxxxxx1x	Slot 1
SLOT2	bxxxxx1xx	Slot 2

RxL_CREDITS_CONSUMED_VN0

- **Title:** VN0 Credit Consumed
- **Category:** RX_CREDITS_CONSUMED Events
- **Event Code:** 0x39
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that an RxQ VN0 credit was consumed (i.e. message uses a VN0 credit for the Rx Buffer). This includes packets that went through the RxQ and those that were bypassed.

RxL_CREDITS_CONSUMED_VN1

- **Title:** VN1 Credit Consumed
- **Category:** RX_CREDITS_CONSUMED Events
- **Event Code:** 0x3a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that an RxQ VN1 credit was consumed (i.e. message uses a VN1 credit for the Rx Buffer). This includes packets that went through the RxQ and those that were bypassed.

RxL_CREDITS_CONSUMED_VNA

- **Title:** VNA Credit Consumed
- **Category:** RX_CREDITS_CONSUMED Events
- **Event Code:** 0x38
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that an RxQ VNA credit was consumed (i.e. message uses a VNA credit for the Rx Buffer). This includes packets that went through the RxQ and those that were bypassed.

RxL_FLITS

- **Title:** Valid Flits Received
- **Category:** Flit Events
- **Event Code:** 0x03
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Shows legal flit time (hides impact of L0p and L0c).
- **NOTE:** When Umask is set to all 1's then all Flits should be counted as 3 since a full flit is counted for each valid slot. By counting all legal flit time we exclude impact of L0p, L0c, and the 5/6 ratio in L0. Slot 0 Dual is counted in slot 0 and slot 1 (as a protocol header)

Table 2-166. Unit Masks for RxL_FLITS

Extension	umask [15:8]	Description
SLOT0	bxxxxxxx1	Slot 0 Count Slot 0 - Other mask bits determine types of headers to count.
SLOT1	bxxxxxx1x	Slot 1 Count Slot 1 - Other mask bits determine types of headers to count.
SLOT2	bxxxxx1xx	Slot 2 Count Slot 2 - Other mask bits determine types of headers to count.
DATA	bxxxx1xxx	Data Count Data Flits (which consume all slots), but how much to count is based on Slot0-2 mask, so count can be 0-3 depending on which slots are enabled for counting..
ALL_DATA	b00001111	All Data
LLCRD	bxxx1xxxx	LLCRD Not Empty Enables counting of LLCRD (with non-zero payload). This only applies to slot 2 since LLCRD is only allowed in slot 2
NULL	bxx1xxxxx	Slot NULL or LLCRD Empty LLCRD with all zeros is treated as NULL. Slot 1 is not treated as NULL if slot 0 is a dual slot. This can apply to slot 0,1, or 2.
ALL_NULL	b00100111	All Null Slots
LLCTRL	bx1xxxxxx	LLCTRL Equivalent to an idle packet. Enables counting of slot 0 LLCTRL messages.
IDLE	b01000111	Idle
PROTHDR	b1xxxxxxx	Protocol Header Enables count of protocol headers in slot 0,1,2 (depending on slot uMask bits)
NON_DATA	b10010111	All Non Data

RxL_INSERTS

- **Title:** RxQ Flit Buffer Allocations
- **Category:** RXQ Events
- **Event Code:** 0x30
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of allocations into the Intel UPI Rx Flit Buffer. Generally, when data is transmitted across Intel UPI, it will bypass the RxQ and pass directly to the ring interface. If things back up getting transmitted onto the ring, however, it may need to allocate into this buffer, thus increasing the latency. This event can be used in conjunction with the Flit Buffer Occupancy event in order to calculate the average flit buffer lifetime.

Table 2-167. Unit Masks for RxL_INSERTS

Extension	umask [15:8]	Description
SLOT0	bxxxxxxx1	Slot 0
SLOT1	bxxxxxx1x	Slot 1
SLOT2	bxxxxx1xx	Slot 2

RxL_OCCUPANCY

- **Title:** RxQ Occupancy - All Packets
- **Category:** RXQ Events
- **Event Code:** 0x32
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Accumulates the number of elements in the Intel UPI RxQ in each cycle. Generally, when data is transmitted across Intel UPI, it will bypass the RxQ and pass directly to the ring interface. If things back up getting transmitted onto the ring, however, it may need to allocate into this buffer, thus increasing the latency. This event can be used in conjunction with the Flit Buffer Not Empty event to calculate average occupancy, or with the Flit Buffer Allocations event to track average lifetime.

Table 2-168. Unit Masks for RxL_OCCUPANCY

Extension	umask [15:8]	Description
SLOT0	bxxxxxxx1	Slot 0
SLOT1	bxxxxxx1x	Slot 1
SLOT2	bxxxxx1xx	Slot 2

RxL_SLOT_BYPASS

- **Title:**
- **Category:** RXQ Events
- **Event Code:** 0x33
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-169. Unit Masks for RxL_SLOT_BYPASS

Extension	umask [15:8]	Description
S0_RXQ1	bxxxxxxx1	
S0_RXQ2	bxxxxxx1x	
S1_RXQ0	bxxxxx1xx	
S1_RXQ2	bxxxx1xxx	
S2_RXQ0	bxxx1xxxx	
S2_RXQ1	bxx1xxxxx	

TxL0P_CLK_ACTIVE

- **Title:**
- **Category:** POWER_TX Events
- **Event Code:** 0x2a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

Table 2-170. Unit Masks for TxL0P_CLK_ACTIVE

Extension	umask [15:8]	Description
CFG_CTL	bxxxxxxx1	
RXQ	bxxxxxx1x	
RXQ_BYPASS	bxxxxx1xx	
RXQ_CRED	bxxxx1xxx	
TXQ	bxxx1xxxx	
RETRY	bxx1xxxxx	
DFX	bx1xxxxxx	
SPARE	b1xxxxxxx	

TxL0P_POWER_CYCLES

- **Title:** Cycles in L0p. Transmit side.
- **Category:** POWER_TX Events
- **Event Code:** 0x27
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of Intel UPI qfclk cycles spent in L0p power mode. L0p is a mode where we disable 60% of the Intel UPI lanes, decreasing our bandwidth in order to save power. It increases snoop and data transfer latencies and decreases overall bandwidth. This mode can be very useful in NUMA optimized workloads that largely only utilize Intel UPI for snoops and their responses. Use edge detect to count the number of instances when the Intel UPI link entered L0p. Link power states are per link and per direction, so for example the Tx direction could be in one state while Rx was in another.
- **NOTE:** Using .edge_det to count transitions does not function if L1_POWER_CYCLES > 0.

TxL0P_POWER_CYCLES_LL_ENTER

- **Title:**
- **Category:** POWER_TX Events
- **Event Code:** 0x28
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

TxL0P_POWER_CYCLES_M3_EXIT

- **Title:**
- **Category:** POWER_TX Events
- **Event Code:** 0x29
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

TxL0_POWER_CYCLES

- **Title:** Cycles in L0. Transmit side.
- **Category:** POWER_TX Events
- **Event Code:** 0x26
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of Intel UPI qfclk cycles spent in L0 power mode in the Link Layer. L0 is the default mode which provides the highest performance with the most power. Use edge detect to count the number of instances that the link entered L0. Link power states are per link and per direction, so for example the Tx direction could be in one state while Rx was in another. The phy layer sometimes leaves L0 for training, which will not be captured by this event.
- **NOTE:** Includes L0p cycles. To get just L0, subtract TxL0P_POWER_CYCLES

TxL_BASIC_HDR_MATCH

- **Title:** Matches on Transmit path of an Intel UPI Port
- **Category:** FLIT match Events
- **Event Code:** 0x04
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Filter Dependency:** CtrCtrl[55:32]
- **Definition:**
- **NOTE:** This event is subject to finer grain filtering. See doc for more information. Filters available in the counter control's umask extension field b[55:32] - message class, opcode, local, remote, datahdr, ndatahdr, dual slot header, single slot header and pe) ANDed per Slot. Then slots are ORed.

Table 2-171. Unit Masks for TxL_BASIC_HDR_MATCH (Sheet 1 of 2)

Extension	umask [15:8]	xtra [32]	Description
REQ	bxxxx1000	0x0	Request REQ Message Class
REQ_OPC	bXXXX1000	0x1	Request Opcode Match REQ Opcodes - Specified in Umask[7:4]
SNP	bxxxx1001	0x0	Snoop SNP Message Class
SNP_OPC	bXXXX1001	0x1	Snoop Opcode Match SNP Opcodes - Specified in Umask[7:4]
RSP_NODATA_OPC	bXXXX1010	0x1	Response - No Data Match Message Class - RSP

Table 2-171. Unit Masks for TxL_BASIC_HDR_MATCH (Sheet 2 of 2)

Extension	umask [15:8]	xtra [32]	Description
RSP_NODATA	bxxxx1010	0x0	Response - No Data Match Message Class - RSP
RSP_DATA	bxxxx1100	0x0	Response - Data Match Message Class -WB
RSP_DATA_OPC	bXXXX1100	0x1	Response - Data Match Message Class -WB
WB_OPC	bxxxx1101	0x1	Writeback Match Message Class -WB
WB	bxxxx1101	0x0	Writeback Match Message Class -WB
NCB_OPC	bxxxx1110	0x1	Non-Coherent Bypass Match Message Class - NCB
NCB	bxxxx1110	0x0	Non-Coherent Bypass Match Message Class - NCB
NCS_OPC	bxxxx1111	0x1	Non-Coherent Standard Match Message Class - NCS
NCS	bxxxx1111	0x0	Non-Coherent Standard Match Message Class - NCS
RSPI	b00101010	0x1	Response - Invalid
RSPCNFLT	b10101010	0x1	Response - Conflict

TxL_BYPASSED

- **Title:** Tx Flit Buffer Bypassed
- **Category:** TXQ Events
- **Event Code:** 0x41
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of times that an incoming flit was able to bypass the Tx flit buffer and pass directly out the Intel UPI Link. Generally, when data is transmitted across Intel UPI, it will bypass the TxQ and pass directly to the link. However, the TxQ will be used with L0p and when LLR occurs, increasing latency to transfer out to the link.

TxL_FLITS

- **Title:** Valid Flits Sent
- **Category:** Flit Events
- **Event Code:** 0x02
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Shows legal flit time (hides impact of L0p and L0c).
- **NOTE:** You can OR any of 5 MSB together and apply against any combination of slots and they will be added together, but a slot MUST be selected.

Table 2-172. Unit Masks for TxL_FLITS

Extension	umask [15:8]	Description
SLOT0	bxxxxxxx1	Slot 0 Count Slot 0 - Other mask bits determine types of headers to count.
SLOT1	bxxxxxx1x	Slot 1 Count Slot 1 - Other mask bits determine types of headers to count.
SLOT2	bxxxxx1xx	Slot 2 Count Slot 2 - Other mask bits determine types of headers to count.

Table 2-172. Unit Masks for TxL_FLITS

Extension	umask [15:8]	Description
DATA	bxxxx1xxx	Data Count Data Flits (which consume all slots), but how much to count is based on Slot0-2 mask, so count can be 0-3 depending on which slots are enabled for counting..
ALL_DATA	b00001111	All Data
LLCRD	bxxx1xxxx	LLCRD Not Empty Enables counting of LLCRD (with non-zero payload). This only applies to slot 2 since LLCRD is only allowed in slot 2
NULL	bxx1xxxxx	Slot NULL or LLCRD Empty LLCRD with all zeros is treated as NULL. Slot 1 is not treated as NULL if slot 0 is a dual slot. This can apply to slot 0,1, or 2.
ALL_NULL	b00100111	All Null Slots
LLCTRL	bx1xxxxxx	LLCTRL Equivalent to an idle packet. Enables counting of slot 0 LLCTRL messages.
IDLE	b01000111	Idle
PROTHDR	b1xxxxxxx	Protocol Header Enables count of protocol headers in slot 0,1,2 (depending on slot uMask bits)
NON_DATA	b10010111	All Non Data

TxL_INSERTS

- **Title:** Tx Flit Buffer Allocations
- **Category:** TXQ Events
- **Event Code:** 0x40
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of allocations into the Intel UPI Tx Flit Buffer. Generally, when data is transmitted across Intel UPI, it will bypass the TxQ and pass directly to the link. However, the TxQ will be used with L0p and when LLR occurs, increasing latency to transfer out to the link. This event can be used in conjunction with the Flit Buffer Occupancy event in order to calculate the average flit buffer lifetime.

TxL_OCCUPANCY

- **Title:** Tx Flit Buffer Occupancy
- **Category:** TXQ Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Accumulates the number of flits in the TxQ. Generally, when data is transmitted across Intel UPI, it will bypass the TxQ and pass directly to the link. However, the TxQ will be used with L0p and when LLR occurs, increasing latency to transfer out to the link. This can be used with the cycles not empty event to track average occupancy, or the allocations event to track average lifetime in the TxQ.

VNA_CREDIT_RETURN_BLOCKED_VN01

- **Title:**
- **Category:** VNA_CREDIT_RETURN Events
- **Event Code:** 0x45
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

VNA_CREDIT_RETURN_OCCUPANCY

- **Title:** VNA Credits Pending Return - Occupancy
- **Category:** VNA_CREDIT_RETURN Events
- **Event Code:** 0x44
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of VNA credits in the Rx side that are waitng to be returned back across the link.

2.7 M2M Performance Monitoring

M2M blocks manage the interface between the Mesh (operating on both Mesh and the SMI3 protocol) and the Memory Controllers. M2M acts as intermediary between the local CHA issuing memory transactions to its attached Memory Controller. Commands from M2M to the MC are serialized by a scheduler and only one can cross the interface at a time.

2.7.1 M2M Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each M2M Box supports event monitoring through four 48b wide counters (M2Mn_PCI_PMON_CTR/CTL{3:0}). Each of these four counters can be programmed to count almost any M2M event (see NOTE for exceptions). the M2M counters can increment by a maximum of 5b per cycle.

M2M PMON also includes mask/match registers that allow a user to match packets of traffic heading to DRAM or heading to the Mesh, according to various standard packet fields such as message class, opcode, etc.

2.7.2 Additional M2M Performance Monitoring

Table 2-173. Additional M2M Performance Monitoring Registers (PCICFG) (Sheet 1 of 2)

Register Name	PCICFG Address	Size (bits)	Description
PCICFG Base Address	Dev:Func DeviceID		
M2M 0 PMON Registers	B2:D8:F0 0x2066		
M2M 1 PMON Registers	B2:D9:F0 0x2066		
Box-Level Filters			
M2Mn_PCI_PMON_OPCODE_MM	278	64	M2M Opcode Mask/Match
M2Mn_PCI_PMON_ADDRMASK0	270	64	M2M PMON Address Mask 0 - b[31:0]

Table 2-173. Additional M2M Performance Monitoring Registers (PCICFG) (Sheet 2 of 2)

Register Name	PCICFG Address	Size (bits)	Description
M2Mn_PCI_PMON_ADDRMASK1	274	64	M2M PMON Address Mask 1 - b[45:32]
M2Mn_PCI_PMON_ADDRMATCH0	268	64	M2M PMON Address Match 0 - b[31:0]
M2Mn_PCI_PMON_ADDRMATCH1	26C	64	M2M PMON Address Match 1 - b[45:32]

2.7.2.1 M2M Filter Registers

In addition to generic event counting, each M2M has several registers that allow a user to opcode match and/or address match packet traffic into and out of M2M. The filter registers report the number of matches through the PKT_MATCH event. The opcode mask/match register provides the ability to simultaneously filter incoming and outgoing traffic.

Figure 2-12. M2M PMON Opcode Filter Register

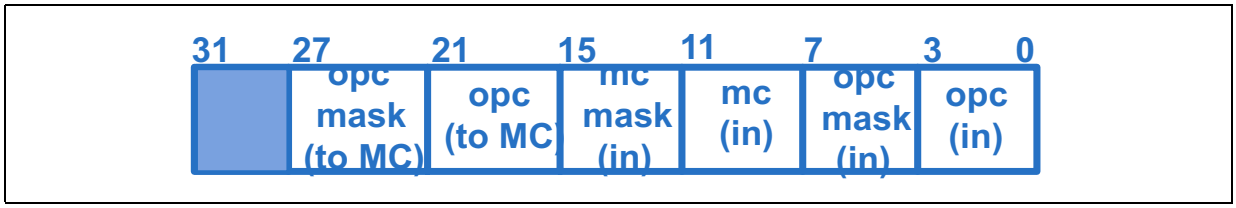


Table 2-174. M2Mn_PCI_PMON_OPCODE_MM Register – Field Definitions

Field	Bits	Atrtr	HW Reset Val	Description
rsv	31:28	RV	0	Reserved SW must set to 0 else behavior is undefined
mcopc_mask	27:22	RW	0	Outgoing Opcode Mask encoding - See the SMI3 encodings in the Packet Match table Table 3-8, "SMI3 Opcode Match by Message Class" on page 234
mcopc	21:16	RW	0	Outgoing Opcode Match encoding - See the SMI3 encodings in the Packet Match table Table 3-8, "SMI3 Opcode Match by Message Class" on page 234
mc_mask	15:12	RW	0	Incoming Message Class Mask
mc	11:8	RW	0	Incoming Message Class Match 4'b0000: REQ (AD) 4'b0010: RSP (AD) 4'b1010: NCB (BL) 4'b1011: NCS (BL) 4'b1100: WB (BL)
opc_mask	7:4	RW	0	Incoming Opcode Mask encoding - See the SMI3 encodings in the Packet Match table Table 3-8, "SMI3 Opcode Match by Message Class" on page 234
opc	3:0	RW	0	Incoming Opcode Match encoding - See the SMI3 encodings in the Packet Match table Table 3-8, "SMI3 Opcode Match by Message Class" on page 234

Table 2-175. M2Mn_PCI_PMON_ADDR{MASK,MATCH}0 Register – Field Definitions

Field	Bits	Atrtr	HW Reset Val	Description
addr	31:0	RW	0	LSB [31:0] of transaction address to mask/match on

Table 2-176. M2Mn_PCI_PMON_ADDR{MASK,MATCH}1 Register – Field Definitions

Field	Bits	Atrtr	HW Reset Val	Description
rsv	31:15	RV	0	Reserved SW must set to 0 else behavior is undefined
addr	14:0	RW	0	MSB [45:32] of transaction address to mask/match on

2.7.3 M2M Performance Monitoring Events

M2M provides events to track information related to all the traffic passing through it's boundaries.

Mesh Section 2.2, "Mesh Performance Monitoring"

- IMC credit tracking - credits rejected, acquired and used all broken down by message Class.
- Reads and Writes issued to the iMC

2.7.4 M2M Box Events Ordered By Code

The following table summarizes the directly measured M2M Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CLOCKTICKS	0x00		0	Cycles - at UCLK
RxC_AD_INSERTS	0x01		0	AD Ingress (from CMS) Allocations
RxC_AD_OCCUPANCY	0x02		0	AD Ingress (from CMS) Occupancy
RxC_AD_CYCLES_NE	0x03		0	AD Ingress (from CMS) Not Empty
RxC_AD_CYCLES_FULL	0x04		0	AD Ingress (from CMS) Full
RxC_BL_INSERTS	0x05		0	BL Ingress (from CMS) Allocations
RxC_BL_OCCUPANCY	0x06		0	BL Ingress (from CMS) Occupancy
RxC_BL_CYCLES_NE	0x07		0	BL Ingress (from CMS) Not Empty
RxC_BL_CYCLES_FULL	0x08		0	BL Ingress (from CMS) Full
TxC_AD_INSERTS	0x09		0	AD Egress (to CMS) Allocations
TxC_AD_OCCUPANCY	0x0a		0	AD Egress (to CMS) Occupancy
TxC_AD_CYCLES_NE	0x0b		0	AD Egress (to CMS) Not Empty

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
TxC_AD_CYCLES_FULL	0x0c		0	AD Egress (to CMS) Full
TxC_AD_CREDITS_ACQUIRED	0x0d		0	AD Egress (to CMS) Credit Acquired
TxC_AD_CREDIT_OCCUPANCY	0x0e		0	AD Egress (to CMS) Credits Occupancy
TxC_AD_NO_CREDIT_CYCLES	0x0f		0	Cycles with No AD Egress (to CMS) Credits
TxC_AD_NO_CREDIT_STALLED	0x10		0	Cycles Stalled with No AD Egress (to CMS) Credits
TxC_AK_INSERTS	0x11		0	AK Egress (to CMS) Allocations
TxC_AK_OCCUPANCY	0x12		0	AK Egress (to CMS) Occupancy
TxC_AK_CYCLES_NE	0x13		0	AK Egress (to CMS) Not Empty
TxC_AK_CYCLES_FULL	0x14		0	AK Egress (to CMS) Full
TxC_BL_INSERTS	0x15		0	BL Egress (to CMS) Allocations
TxC_BL_OCCUPANCY	0x16		0	BL Egress (to CMS) Occupancy
TxC_BL_CYCLES_NE	0x17		0	BL Egress (to CMS) Not Empty
TxC_BL_CYCLES_FULL	0x18		0	BL Egress (to CMS) Full
TxC_BL_CREDITS_ACQUIRED	0x19		0	BL Egress (to CMS) Credit Acquired
TxC_BL_CREDIT_OCCUPANCY	0x1a		0	BL Egress (to CMS) Credits Occupancy
TxC_BL_NO_CREDIT_CYCLES	0x1b		0	Cycles with No BL Egress (to CMS) Credits
TxC_BL_NO_CREDIT_STALLED	0x1c		0	Cycles Stalled with No BL Egress (to CMS) Credits
TxC_AK_CREDITS_ACQUIRED	0x1d		0	AK Egress (to CMS) Credit Acquired
TxC_AK_CREDIT_OCCUPANCY	0x1e		0	AK Egress (to CMS) Credits Occupancy
TxC_AK_NO_CREDIT_CYCLES	0x1f		0	Cycles with No AK Egress (to CMS) Credits
TxC_AK_NO_CREDIT_STALLED	0x20		0	Cycles Stalled with No AK Egress (to CMS) Credits
BYPASS_M2M_INGRESS	0x21		0	M2M to iMC Bypass
BYPASS_M2M_EGRESS	0x22		0	M2M to iMC Bypass
DIRECT2CORE_TAKEN	0x23		0	Direct2Core Messages Sent
DIRECT2CORE_NOT_TAKEN_DIRSTATE	0x24		0	Cycles when Direct2Core was Disabled
DIRECT2CORE_TXN_OVERRIDE	0x25		0	Number of Reads that had Direct2Core Overridden
DIRECT2UPI_TAKEN	0x26		0	Direct2UPI Messages Sent
DIRECT2UPI_NOT_TAKEN_DIRSTATE	0x27		0	Cycles when Direct2UPI was Disabled
DIRECT2UPI_NOT_TAKEN_CREDITS	0x28		0	Number of Reads that had Direct2UPI Overridden
DIRECT2UPI_TXN_OVERRIDE	0x29		0	Number of Reads that had Direct2UPI Overridden
DIRECTORY_HIT	0x2a		0	Directory Hit
DIRECTORY_MISS	0x2b		0	Directory Miss
DIRECTORY_LOOKUP	0x2d		0	Directory Lookups
DIRECTORY_UPDATE	0x2e		0	Directory Updates
IMC_READS	0x37		0	M2M Reads Issued to iMC
IMC_WRITES	0x38		0	M2M Writes Issued to iMC
TxC_AK	0x39		0	Outbound Ring Transactions on AK

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
TxC_BL	0x40		0	Outbound DRS Ring Transactions to Cache
TGR_AD_CREDITS	0x41		0	Number AD Ingress Credits
TGR_BL_CREDITS	0x42		0	Number BL Ingress Credits
RPQ_CYCLES_REG_CREDITS	0x43		0	M2M to iMC RPQ Cycles w/Credits - Regular
RPQ_CYCLES_SPEC_CREDITS	0x44		0	M2M to iMC RPQ Cycles w/Credits - Special
TRACKER_CYCLES_FULL	0x45		0	Tracker Cycles Full
TRACKER_CYCLES_NE	0x46		0	Tracker Cycles Not Empty
TRACKER_OCCUPANCY	0x47		0	Tracker Occupancy
TRACKER_PENDING_OCCUPANCY	0x48		0	Data Pending Occupancy
TRACKER_INSERTS	0x49		0	Tracker Inserts
WRITE_TRACKER_CYCLES_FULL	0x4a		0	Write Tracker Cycles Full
WRITE_TRACKER_CYCLES_NE	0x4b		0	Write Tracker Cycles Not Empty
PKT_MATCH	0x4c		0	Number Packet Header Matches
WPQ_CYCLES_REG_CREDITS	0x4d		0	M2M->iMC WPQ Cycles w/Credits - Regular
WPQ_CYCLES_SPEC_CREDITS	0x4e		0	M2M->iMC WPQ Cycles w/Credits - Special
PREFCAM_CYCLES_FULL	0x53		0	Prefetch CAM Cycles Full
PREFCAM_CYCLES_NE	0x54		0	Prefetch CAM Cycles Not Empty
PREFCAM_OCCUPANCY	0x55		0	Prefetch CAM Occupancy
PREFCAM_DEMAND_PROMOTIONS	0x56		0	Data Pending Occupancy
PREFCAM_INSERTS	0x57		0	Prefetch CAM Inserts
WRITE_TRACKER_OCCUPANCY	0x60		0	Write Tracker Occupancy
WRITE_TRACKER_INSERTS	0x61		0	Write Tracker Inserts
TxC_AK_SIDEBAND	0x6b		0	AK Egress (to CMS) Sideband

2.7.5 M2M Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the M2M Box.

BYPASS_M2M_EGRESS

- **Title:** M2M to iMC Bypass
- **Category:** BL Egress Events
- **Event Code:** 0x22
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-177. Unit Masks for BYPASS_M2M_EGRESS

Extension	umask [15:8]	Description
TAKEN	bxxxxxxx1	Taken
NOT_TAKEN	bxxxxxx1x	Not Taken

BYPASS_M2M_INGRESS

- **Title:** M2M to iMC Bypass
- **Category:** BL Ingress Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-178. Unit Masks for BYPASS_M2M_INGRESS

Extension	umask [15:8]	Description
TAKEN	bxxxxxxx1	Taken
NOT_TAKEN	bxxxxxx1x	Not Taken

CLOCKTICKS

- **Title:** Cycles - at UCLK
- **Category:** UCLK Events
- **Event Code:** 0x00
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2CORE_NOT_TAKEN_DIRSTATE

- **Title:** Cycles when Direct2Core was Disabled
- **Category:** DIRECT2CORE Events
- **Event Code:** 0x24
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2CORE_TAKEN

- **Title:** Direct2Core Messages Sent
- **Category:** DIRECT2CORE Events
- **Event Code:** 0x23
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2CORE_TXN_OVERRIDE

- **Title:** Number of Reads that had Direct2Core Overridden
- **Category:** DIRECT2CORE Events
- **Event Code:** 0x25
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2UPI_NOT_TAKEN_CREDITS

- **Title:** Number of Reads that had Direct2UPI Overridden
- **Category:** DIRECT2UPI Events
- **Event Code:** 0x28
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2UPI_NOT_TAKEN_DIRSTATE

- **Title:** Cycles when Direct2UPI was Disabled
- **Category:** DIRECT2UPI Events
- **Event Code:** 0x27
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2UPI_TAKEN

- **Title:** Direct2UPI Messages Sent
- **Category:** DIRECT2UPI Events
- **Event Code:** 0x26
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECT2UPI_TXN_OVERRIDE

- **Title:** Number of Reads that had Direct2UPI Overridden
- **Category:** DIRECT2UPI Events
- **Event Code:** 0x29
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DIRECTORY_HIT

- **Title:** Directory Hit
- **Category:** Directory State Events
- **Event Code:** 0x2a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Covers NearMem Reads (Demand and Underfill).

Table 2-179. Unit Masks for DIRECTORY_HIT

Extension	umask [15:8]	Description
DIRTY_I	bxxxxxxx1	On Dirty Line in I State
DIRTY_S	bxxxxxx1x	On Dirty Line in S State
DIRTY_P	bxxxxx1xx	On Dirty Line in L State
DIRTY_A	bxxxx1xxx	On Dirty Line in A State
CLEAN_I	bxxx1xxxx	On NonDirty Line in I State
CLEAN_S	bxx1xxxxx	On NonDirty Line in S State
CLEAN_P	bx1xxxxxx	On NonDirty Line in L State
CLEAN_A	b1xxxxxxx	On NonDirty Line in A State

DIRECTORY_LOOKUP

- **Title:** Directory Lookups
- **Category:** DIRECTORY Events
- **Event Code:** 0x2d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-180. Unit Masks for DIRECTORY_LOOKUP

Extension	umask [15:8]	Description
ANY	bxxxxxxx1	Any state
STATE_I	bxxxxxx1x	I State
STATE_S	bxxxxx1xx	S State
STATE_A	bxxxx1xxx	A State

DIRECTORY_MISS

- **Title:** Directory Miss
- **Category:** Directory State Events
- **Event Code:** 0x2b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Covers NearMem Reads (Demand and Underfill).

Table 2-181. Unit Masks for DIRECTORY_MISS

Extension	umask [15:8]	Description
DIRTY_I	bxxxxxxx1	On Dirty Line in I State
DIRTY_S	bxxxxxx1x	On Dirty Line in S State
DIRTY_P	bxxxxx1xx	On Dirty Line in L State
DIRTY_A	bxxxx1xxx	On Dirty Line in A State
CLEAN_I	bxxx1xxxx	On NonDirty Line in I State
CLEAN_S	bxx1xxxxx	On NonDirty Line in S State
CLEAN_P	bx1xxxxxx	On NonDirty Line in L State
CLEAN_A	b1xxxxxxx	On NonDirty Line in A State

DIRECTORY_UPDATE

- **Title:** Directory Updates
- **Category:** DIRECTORY Events
- **Event Code:** 0x2e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Tie to Packet Mask/Match?

Table 2-182. Unit Masks for DIRECTORY_UPDATE

Extension	umask [15:8]	Description
ANY	bxxxxxxx1	Any
I2S	bxxxxxx1x	I2S
I2A	bxxxxx1xx	I2A
S2I	bxxxx1xxx	S2I
S2A	bxxx1xxxx	S2A

Table 2-182. Unit Masks for DIRECTORY_UPDATE

Extension	umask [15:8]	Description
A2I	bxx1xxxxx	A2I
A2S	bx1xxxxxx	A2S

IMC_READS

- **Title:** M2M Reads Issued to iMC
- **Category:** IMC Events
- **Event Code:** 0x37
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Scrub Reads due to ECC errors not currently included

Table 2-183. Unit Masks for IMC_READS

Extension	umask [15:8]	Description
NORMAL	bxxxxxxx1	Normal Priority
ISOCH	bxxxxxx1x	Critical Priority
ALL	bxxxxx1xx	All, regardless of priority.
FROM_TRANSGRESS	bxxx1xxxx	All, regardless of priority.

IMC_WRITES

- **Title:** M2M Writes Issued to iMC
- **Category:** IMC Events
- **Event Code:** 0x38
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Scrub Writes due to ECC errors not currently included

Table 2-184. Unit Masks for IMC_WRITES

Extension	umask [15:8]	Description
FULL	bxxxxxxx1	Full Line Non-ISOCH
PARTIAL	bxxxxxx1x	Partial Non-ISOCH
FULL_ISOCH	bxxxxx1xx	ISOCH Full Line
PARTIAL_ISOCH	bxxxx1xxx	ISOCH Partial
ALL	bxxx1xxxx	All Writes
FROM_TRANSGRESS	bx1xxxxxx	All, regardless of priority.
NI	b1xxxxxxx	All, regardless of priority.

PKT_MATCH

- **Title:** Number Packet Header Matches
- **Category:** PACKET MATCH Events
- **Event Code:** 0x4c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**

Table 2-185. Unit Masks for PKT_MATCH

Extension	umask [15:8]	Description
MESH	bxxxxxxx1	Mesh Match
MC	bxxxxxx1x	MC Match

PREFCAM_CYCLES_FULL

- **Title:** Prefetch CAM Cycles Full
- **Category:** CAM Prefetch Events
- **Event Code:** 0x53
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

PREFCAM_CYCLES_NE

- **Title:** Prefetch CAM Cycles Not Empty
- **Category:** CAM Prefetch Events
- **Event Code:** 0x54
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

PREFCAM_DEMAND_PROMOTIONS

- **Title:** Data Pending Occupancy
- **Category:** CAM Prefetch Events
- **Event Code:** 0x56
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

PREFCAM_INSERTS

- **Title:** Prefetch CAM Inserts
- **Category:** CAM Prefetch Events
- **Event Code:** 0x57
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

PREFCAM_OCCUPANCY

- **Title:** Prefetch CAM Occupancy
- **Category:** CAM Prefetch Events
- **Event Code:** 0x55
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RPQ_CYCLES_REG_CREDITS

- **Title:** M2M to iMC RPQ Cycles w/Credits - Regular
- **Category:** RPQ CREDIT Events
- **Event Code:** 0x43
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**
- **NOTE:** To Count # cycles w/o credits, either set .thresh to 1 and .invert to 1 OR subtract this from total cycles

Table 2-186. Unit Masks for RPQ_CYCLES_REG_CREDITS

Extension	umask [15:8]	Description
CHN0	bxxxxxxx1	Channel 0
CHN1	bxxxxxx1x	Channel 1
CHN2	bxxxxx1xx	Channel 2

RPQ_CYCLES_SPEC_CREDITS

- **Title:** M2M to iMC RPQ Cycles w/Credits - Special
- **Category:** RPQ CREDIT Events
- **Event Code:** 0x44
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** To Count # cycles w/o credits, either set .thresh to 1 and .invert to 1 OR subtract this from total cycles

Table 2-187. Unit Masks for RPQ_CYCLES_SPEC_CREDITS

Extension	umask [15:8]	Description
CHN0	bxxxxxxx1	Channel 0
CHN1	bxxxxxx1x	Channel 1
CHN2	bxxxxx1xx	Channel 2

RxC_AD_CYCLES_FULL

- **Title:** AD Ingress (from CMS) Full
- **Category:** AD Ingress Events
- **Event Code:** 0x04
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_AD_CYCLES_NE

- **Title:** AD Ingress (from CMS) Not Empty
- **Category:** AD Ingress Events
- **Event Code:** 0x03
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_AD_INSERTS

- **Title:** AD Ingress (from CMS) Allocations
- **Category:** AD Ingress Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_AD_OCCUPANCY

- **Title:** AD Ingress (from CMS) Occupancy
- **Category:** AD Ingress Events
- **Event Code:** 0x02
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_BL_CYCLES_FULL

- **Title:** BL Ingress (from CMS) Full
- **Category:** BL Ingress Events
- **Event Code:** 0x08
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_BL_CYCLES_NE

- **Title:** BL Ingress (from CMS) Not Empty
- **Category:** BL Ingress Events
- **Event Code:** 0x07
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_BL_INSERTS

- **Title:** BL Ingress (from CMS) Allocations
- **Category:** BL Ingress Events
- **Event Code:** 0x05
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_BL_OCCUPANCY

- **Title:** BL Ingress (from CMS) Occupancy
- **Category:** BL Ingress Events
- **Event Code:** 0x06
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TGR_AD_CREDITS

- **Title:** Number AD Ingress Credits
- **Category:** Transgress/M2MIngress Credit Events
- **Event Code:** 0x41
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TGR_BL_CREDITS

- **Title:** Number BL Ingress Credits
- **Category:** Transgress/M2MIngress Credit Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TRACKER_CYCLES_FULL

- **Title:** Tracker Cycles Full
- **Category:** TRACKER Events
- **Event Code:** 0x45
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-188. Unit Masks for TRACKER_CYCLES_FULL

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

TRACKER_CYCLES_NE

- **Title:** Tracker Cycles Not Empty
- **Category:** TRACKER Events
- **Event Code:** 0x46
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-189. Unit Masks for TRACKER_CYCLES_NE

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

TRACKER_INSERTS

- **Title:** Tracker Inserts
- **Category:** TRACKER Events
- **Event Code:** 0x49
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-190. Unit Masks for TRACKER_INSERTS

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

TRACKER_OCCUPANCY

- **Title:** Tracker Occupancy
- **Category:** TRACKER Events
- **Event Code:** 0x47
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Tie to Packet Mask/Match?

Table 2-191. Unit Masks for TRACKER_OCCUPANCY

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

TRACKER_PENDING_OCCUPANCY

- **Title:** Data Pending Occupancy
- **Category:** TRACKER Events
- **Event Code:** 0x48
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_CREDITS_ACQUIRED

- **Title:** AD Egress (to CMS) Credit Acquired
- **Category:** AD CMS/Mesh Egress Credit Events
- **Event Code:** 0x0d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_CREDIT_OCCUPANCY

- **Title:** AD Egress (to CMS) Credits Occupancy
- **Category:** AD CMS/Mesh Egress Credit Events
- **Event Code:** 0x0e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_CYCLES_FULL

- **Title:** AD Egress (to CMS) Full
- **Category:** AD Egress Events
- **Event Code:** 0x0c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_CYCLES_NE

- **Title:** AD Egress (to CMS) Not Empty
- **Category:** AD Egress Events
- **Event Code:** 0x0b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**

TxC_AD_INSERTS

- **Title:** AD Egress (to CMS) Allocations
- **Category:** AD Egress Events
- **Event Code:** 0x09
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_NO_CREDIT_CYCLES

- **Title:** Cycles with No AD Egress (to CMS) Credits
- **Category:** AD CMS/Mesh Egress Credit Events
- **Event Code:** 0x0f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_NO_CREDIT_STALLED

- **Title:** Cycles Stalled with No AD Egress (to CMS) Credits
- **Category:** AD CMS/Mesh Egress Credit Events
- **Event Code:** 0x10
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AD_OCCUPANCY

- **Title:** AD Egress (to CMS) Occupancy
- **Category:** AD Egress Events
- **Event Code:** 0x0a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

TxC_AK

- **Title:** Outbound Ring Transactions on AK
- **Category:** OUTBOUND_TX Events
- **Event Code:** 0x39
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-192. Unit Masks for TxC_AK

Extension	umask [15:8]	Description
NDR	bxxxxxxx1	NDR Transactions
CRD_CBO	bxxxxxx1x	CRD Transactions to Cbo

TxC_AK_CREDITS_ACQUIRED

- **Title:** AK Egress (to CMS) Credit Acquired
- **Category:** AK CMS/Mesh Egress Credit Events
- **Event Code:** 0x1d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**

Table 2-193. Unit Masks for TxC_AK_CREDITS_ACQUIRED

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_AK_CREDIT_OCCUPANCY

- **Title:** AK Egress (to CMS) Credits Occupancy
- **Category:** AK CMS/Mesh Egress Credit Events
- **Event Code:** 0x1e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-194. Unit Masks for TxC_AK_CREDIT_OCCUPANCY

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_AK_CYCLES_FULL

- **Title:** AK Egress (to CMS) Full
- **Category:** AK Egress Events
- **Event Code:** 0x14
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** Some extra wild guesses as to what the subevents count - and why does it seem to track 0 credits for each CMS agent, but the other related events don't?

Table 2-195. Unit Masks for TxC_AK_CYCLES_FULL

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All
RDCRD0	b0xxx1xxx	Read Credit Request
WRCRD0	b0xx1xxxx	Write Credit Request
WRCMP0	b0x1xxxxx	Write Compare Request
RDCRD1	b1xxx1xxx	Read Credit Request
WRCRD1	b1xx1xxxx	Write Credit Request
WRCMP1	b1x1xxxxx	Write Compare Request

TxC_AK_CYCLES_NE

- **Title:** AK Egress (to CMS) Not Empty
- **Category:** AK Egress Events
- **Event Code:** 0x13
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-196. Unit Masks for TxC_AK_CYCLES_NE

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All
RDCRD	bxxxx1xxx	Read Credit Request
WRCRD	bxxx1xxxx	Write Credit Request
WRCMP	bxx1xxxxx	Write Compare Request

TxC_AK_INSERTS

- **Title:** AK Egress (to CMS) Allocations
- **Category:** AK Egress Events
- **Event Code:** 0x11
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-197. Unit Masks for TxC_AK_INSERTS

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All
RDCRD	bxxxx1xxx	Read Credit Request
WRCRD	bxxx1xxxx	Write Credit Request
WRCMP	bxx1xxxxx	Write Compare Request
PREF_RD_CAM_HIT	bx1xxxxxx	Prefetch Read Cam Hit

TxC_AK_NO_CREDIT_CYCLES

- **Title:** Cycles with No AK Egress (to CMS) Credits
- **Category:** AK CMS/Mesh Egress Credit Events
- **Event Code:** 0x1f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-198. Unit Masks for TxC_AK_NO_CREDIT_CYCLES

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_AK_NO_CREDIT_STALLED

- **Title:** Cycles Stalled with No AK Egress (to CMS) Credits
- **Category:** AK CMS/Mesh Egress Credit Events
- **Event Code:** 0x20
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-199. Unit Masks for TxC_AK_NO_CREDIT_STALLED

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_AK_OCCUPANCY

- **Title:** AK Egress (to CMS) Occupancy
- **Category:** AK Egress Events
- **Event Code:** 0x12
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-200. Unit Masks for TxC_AK_OCCUPANCY

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All
RDCRD	bxxxx1xxx	Read Credit Request
WRCRD	bxxx1xxxx	Write Credit Request
WRCMP	bxx1xxxxx	Write Compare Request

TxC_AK_SIDEBAND

- **Title:** AK Egress (to CMS) Sideband
- **Category:** AK Egress Events
- **Event Code:** 0x6b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-201. Unit Masks for TxC_AK_SIDEBAND

Extension	umask [15:8]	Description
RD	bxxxxxxx1	
WR	bxxxxxx1x	

TxC_BL

- **Title:** Outbound DRS Ring Transactions to Cache
- **Category:** OUTBOUND_TX Events
- **Event Code:** 0x40
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-202. Unit Masks for TxC_BL

Extension	umask [15:8]	Description
DRS_CACHE	bxxxxxxx1	Data to Cache
DRS_CORE	bxxxxxx1x	Data to Core
DRS_UPI	bxxxxx1xx	Data to QPI

TxC_BL_CREDITS_ACQUIRED

- **Title:** BL Egress (to CMS) Credit Acquired
- **Category:** BL CMS/Mesh Egress Credit Events
- **Event Code:** 0x19
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-203. Unit Masks for TxC_BL_CREDITS_ACQUIRED

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_BL_CREDIT_OCCUPANCY

- **Title:** BL Egress (to CMS) Credits Occupancy
- **Category:** BL CMS/Mesh Egress Credit Events
- **Event Code:** 0x1a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-204. Unit Masks for TxC_BL_CREDIT_OCCUPANCY

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_BL_CYCLES_FULL

- **Title:** BL Egress (to CMS) Full
- **Category:** BL Egress Events
- **Event Code:** 0x18
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-205. Unit Masks for TxC_BL_CYCLES_FULL

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All

TxC_BL_CYCLES_NE

- **Title:** BL Egress (to CMS) Not Empty
- **Category:** BL Egress Events
- **Event Code:** 0x17
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-206. Unit Masks for TxC_BL_CYCLES_NE

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All

TxC_BL_INSERTS

- **Title:** BL Egress (to CMS) Allocations
- **Category:** BL Egress Events
- **Event Code:** 0x15
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-207. Unit Masks for TxC_BL_INSERTS

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All

TxC_BL_NO_CREDIT_CYCLES

- **Title:** Cycles with No BL Egress (to CMS) Credits
- **Category:** BL CMS/Mesh Egress Credit Events
- **Event Code:** 0x1b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-208. Unit Masks for TxC_BL_NO_CREDIT_CYCLES

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_BL_NO_CREDIT_STALLED

- **Title:** Cycles Stalled with No BL Egress (to CMS) Credits
- **Category:** BL CMS/Mesh Egress Credit Events
- **Event Code:** 0x1c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-209. Unit Masks for TxC_BL_NO_CREDIT_STALLED

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side

TxC_BL_OCCUPANCY

- **Title:** BL Egress (to CMS) Occupancy
- **Category:** BL Egress Events
- **Event Code:** 0x16
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-210. Unit Masks for TxC_BL_OCCUPANCY

Extension	umask [15:8]	Description
CMS0	bxxxxxxx1	Common Mesh Stop - Near Side
CMS1	bxxxxxx1x	Common Mesh Stop - Far Side
ALL	bxxxxxx11	All

WPQ_CYCLES_REG_CREDITS

- **Title:** M2M->iMC WPQ Cycles w/Credits - Regular
- **Category:** WPQ_CREDITS Events
- **Event Code:** 0x4d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**
- **NOTE:** To Count # cycles w/o credits, either set .thresh to 1 and .invert to 1 OR subtract this from total cycles

Table 2-211. Unit Masks for WPQ_CYCLES_REG_CREDITS

Extension	umask [15:8]	Description
CHN0	bxxxxxxx1	Channel 0
CHN1	bxxxxxx1x	Channel 1
CHN2	bxxxxx1xx	Channel 2

WPQ_CYCLES_SPEC_CREDITS

- **Title:** M2M->iMC WPQ Cycles w/Credits - Special
- **Category:** WPQ_CREDITS Events
- **Event Code:** 0x4e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**
- **NOTE:** To Count # cycles w/o credits, either set .thresh to 1 and .invert to 1 OR subtract this from total cycles

Table 2-212. Unit Masks for WPQ_CYCLES_SPEC_CREDITS

Extension	umask [15:8]	Description
CHN0	bxxxxxxx1	Channel 0
CHN1	bxxxxxx1x	Channel 1
CHN2	bxxxxx1xx	Channel 2

WRITE_TRACKER_CYCLES_FULL

- **Title:** Write Tracker Cycles Full
- **Category:** TRACKER Events
- **Event Code:** 0x4a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-213. Unit Masks for WRITE_TRACKER_CYCLES_FULL

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

WRITE_TRACKER_CYCLES_NE

- **Title:** Write Tracker Cycles Not Empty
- **Category:** TRACKER Events
- **Event Code:** 0x4b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-214. Unit Masks for WRITE_TRACKER_CYCLES_NE

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

WRITE_TRACKER_INSERTS

- **Title:** Write Tracker Inserts
- **Category:** TRACKER Events
- **Event Code:** 0x61
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-215. Unit Masks for WRITE_TRACKER_INSERTS

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

WRITE_TRACKER_OCCUPANCY

- **Title:** Write Tracker Occupancy
- **Category:** TRACKER Events
- **Event Code:** 0x60
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-216. Unit Masks for WRITE_TRACKER_OCCUPANCY

Extension	umask [15:8]	Description
CH0	bxxxxxxx1	Channel 0
CH1	bxxxxxx1x	Channel 1
CH2	bxxxxx1xx	Channel 2

2.8 M3UPI Performance Monitoring

M3UPI is the interface between the mesh and the Intel® UPI Link Layer. It is responsible for translating between mesh protocol packets and flits that are used for transmitting data across the Intel® UPI interface. It performs credit checking between the local Intel® UPI LL, the remote Intel® UPI LL and other agents on the local mesh.

The M3UPI agent provides several functions:

- Interface between Mesh and Intel® UPI:
  One of the primary attributes of the mesh is its ability to convey Intel® UPI semantics with no translation. For example, this architecture enables initiators to

communicate with a local Home agent in exactly the same way as a remote Home agent on another Intel® Xeon® Processor Scalable Memory Family socket. With this philosophy, the M3UPI block is lean and does very little with regards to the Intel® UPI protocol aside from mirror the request between the mesh and the Intel® UPI interface.

- Intel® UPI routing:
  In order to optimize layout and latency, both full width Intel® UPI interfaces share the same mesh stop. Therefore, an Intel® UPI packet might be received on one interface and simply forwarded along on the other Intel® UPI interface. The M3UPI has sufficient routing logic to determine if a request, snoop or response is targeting the local socket or if it should be forwarded along to the other interface. This routing remains isolated to M3UPI and does not impede traffic on the Mesh.
- Intel® UPI Home Snoop Protocol (with early snoop optimizations for DP):
  The M3UPI agent implements a latency-reducing optimization for dual sockets which issues snoops within the socket for incoming requests as well as a latency-reducing optimization to return data satisfying Direct2Core (D2C) requests.

2.8.1 M3UPI Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

Each M3UPI Link in supports event monitoring through three 48b wide counters (M3_Ly_PCI_PMON_CTR/CTL{2:0}). Each of these three counters can be programmed to count almost any M3UPI event (see NOTE for exceptions). The M3UPI counters can increment by a maximum of 6b per cycle.

Note: Only counter 0 can be used for tracking occupancy events. Only counter 2 can be used to count mesh events.

Note: There is a fourth counters that's broken?

2.8.2 M3UPI Performance Monitoring Events

M3UPI provides events to track information related to all the traffic passing through it's boundaries.

- VN/IIO credit tracking - in addition to tracking the occupancy of the full VNA queue, M3UPI provides a great deal of additional information: credits rejected, acquired and used often broken down by Message Class.
  Mesh Section 2.2, "Mesh Performance Monitoring"

2.8.3 M3UPI Box Events Ordered By Code

The following table summarizes the directly measured M3UPI Box events.

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
CLOCKTICKS	0x01	0-3	0	Number of uclks in domain
TxC_AD_FLQ_OCCUPANCY	0x1c	0	0	AD Flow Q Occupancy

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
TxC_BL_FLQ_OCCUPANCY	0x1d	0	0	BL Flow Q Occupancy
TxC_AK_FLQ_OCCUPANCY	0x1e	0	0	AK Flow Q Occupancy
UPI_PEER_AD_CREDITS_EMPTY	0x20	0-3	0	UPI0 AD Credits Empty
UPI_PEER_BL_CREDITS_EMPTY	0x21	0-3	0	UPI0 BL Credits Empty
CHA_AD_CREDITS_EMPTY	0x22	0-3	0	CBox AD Credits Empty
M2_BL_CREDITS_EMPTY	0x23	0-3	0	M2 BL Credits Empty
TxC_AD_FLQ_CYCLES_NE	0x27	0-3	0	AD Flow Q Not Empty
TxC_BL_FLQ_CYCLES_NE	0x28	0-3	0	BL Flow Q Not Empty
UPI_PREFETCH_SPAWN	0x29	0-3	0	FlowQ Generated Prefetch
D2U_SENT	0x2a	0-3	0	D2U Sent
D2C_SENT	0x2b	0-3	0	D2C Sent
TxC_AD_FLQ_BYPASS	0x2c	0-3	0	AD FlowQ Bypass
TxC_AD_FLQ_INSERTS	0x2d	0-3	0	AD Flow Q Inserts
TxC_BL_FLQ_INSERTS	0x2e	0-3	0	BL Flow Q Inserts
TxC_AK_FLQ_INSERTS	0x2f	0-3	0	AK Flow Q Inserts
TxC_AD_ARB_FAIL	0x30		0	Failed ARB for AD
TxC_AD_SPEC_ARB_NO_OTHER_PEND	0x32		0	Speculative ARB for AD - No Credit
TxC_AD_SPEC_ARB_NEW_MSG	0x33		0	Speculative ARB for AD - New Message
TxC_AD_SPEC_ARB_CRD_AVAIL	0x34		0	Speculative ARB for AD - Credit Available
TxC_BL_ARB_FAIL	0x35		0	Failed ARB for BL
TxC_BL_SPEC_ARB_NO_OTHER_PEND	0x37		0	Speculative ARB for AD Failed - No Credit
TxC_BL_SPEC_ARB_NEW_MSG	0x38		0	Speculative ARB for BL - New Message
TxC_AD_SNPF_GRP1_VN1	0x3c	0	0	Number of Snoop Targets
TxC_AD_SNPF_GRP2_VN1	0x3d		0	Snoop Arbitration
MULTI_SLOT_RCVD	0x3e	0-3	0	Multi Slot Flit Received
RxC_BYPASSED	0x40	0-2	0	Ingress Queue Bypasses
RxC_INSERTS_VN0	0x41		0	VN0 Ingress (from CMS) Queue - Inserts
RxC_INSERTS_VN1	0x42		0	VN1 Ingress (from CMS) Queue - Inserts
RxC_CYCLES_NE_VN0	0x43		0	VN0 Ingress (from CMS) Queue - Cycles Not Empty
RxC_CYCLES_NE_VN1	0x44		0	VN1 Ingress (from CMS) Queue - Cycles Not Empty
RxC_OCCUPANCY_VN0	0x45		0	VN0 Ingress (from CMS) Queue - Occupancy
RxC_OCCUPANCY_VN1	0x46		0	VN1 Ingress (from CMS) Queue - Occupancy
RxC_ARB_NOCRED_VN0	0x47		0	No Credits to Arb for VN0
RxC_ARB_NOCRED_VN1	0x48		0	No Credits to Arb for VN1
RxC_ARB_NOAD_REQ_VN0	0x49		0	Can't Arb for VN0
RxC_ARB_NOAD_REQ_VN1	0x4a		0	Can't Arb for VN1
RxC_ARB_LOST_VN0	0x4b		0	Lost Arb for VN0
RxC_ARB_LOST_VN1	0x4c		0	Lost Arb for VN1
RxC_ARB_MISC	0x4d		0	Arb Miscellaneous

Symbol Name	Event Code	Ctrs	Max Inc/ Cyc	Description
RxC_PACKING_MISS_VN0	0x4e	0-2	0	VN0 message can't slot into flit
RxC_PACKING_MISS_VN1	0x4f	0-2	0	VN1 message can't slot into flit
RxC_COLLISION_VN0	0x50	0-2	0	VN0 message lost contest for flit
RxC_COLLISION_VN1	0x51	0-2	0	VN1 message lost contest for flit
RxC_HELD	0x52	0-2	0	Message Held
RxC_FLIT_GEN_HDR1	0x53		0	Flit Gen - Header 1
RxC_FLIT_GEN_HDR2	0x54		0	Flit Gen - Header 2
RxC_FLIT_NOT_SENT	0x55		0	Header Not Sent
RxC_FLITS_SENT	0x56		0	Sent Header Flit
RxC_FLITS_DATA_NOT_SENT	0x57		0	Data Flit Not Sent
RxC_FLITS_SLOT_BL	0x58		0	Slotting BL Message Into Header Flit
RxC_FLITS_GEN_BL	0x59		0	Generating BL Data Flit Sequence
RxC_FLITS_MISC	0x5a		0	
RxC_VNA_CRD	0x5b		0	Remote VNA Credits
VN0_CREDITS_USED	0x5c		0	VN0 Credit Used
VN1_CREDITS_USED	0x5d		0	VN1 Credit Used
VN0_NO_CREDITS	0x5e		0	VN0 No Credits
VN1_NO_CREDITS	0x5f		0	VN1 No Credits
RxC_CRD_MISC	0x60		0	Miscellaneous Credit Events
RxC_CRD_OCC	0x61		0	Credit Occupancy
RxC_SMI3_PFTCH	0x62		0	SMI3 Prefetch Messages

2.8.4 M3UPI Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the M3UPI Box.

CHA_AD_CREDITS_EMPTY

- **Title:** CBox AD Credits Empty
- **Category:** EGRESS Credit Events
- **Event Code:** 0x22
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** No credits available to send to Cbox on the AD Ring (covers higher CBoxes)

Table 2-217. Unit Masks for CHA_AD_CREDITS_EMPTY

Extension	umask [15:8]	Description
VNA	bxxxxxxx1	VNA Messages
WB	bxxxxxx1x	Writebacks
REQ	bxxxxx1xx	Requests
SNP	bxxxx1xxx	Snoops

CLOCKTICKS

- **Title:** Number of uclks in domain
- **Category:** UCLK Events
- **Event Code:** 0x01
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of uclks in the M3 uclk domain. This could be slightly different than the count in the Ubox because of enable/freeze delays. However, because the M3 is close to the Ubox, they generally should not diverge by more than a handful of cycles.

D2C_SENT

- **Title:** D2C Sent
- **Category:** Special Egress Events
- **Event Code:** 0x2b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Count cases BL sends direct to core

D2U_SENT

- **Title:** D2U Sent
- **Category:** Special Egress Events
- **Event Code:** 0x2a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Cases where SMI3 sends D2U command
- **NOTE:** NOT required anymore

M2_BL_CREDITS_EMPTY

- **Title:** M2 BL Credits Empty
- **Category:** EGRESS Credit Events
- **Event Code:** 0x23
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** No vn0 and vna credits available to send to M2

Table 2-218. Unit Masks for M2_BL_CREDITS_EMPTY

Extension	umask [15:8]	Description
IIO0_IIO1_NCB	bxxxxxxx1	IIO0 and IIO1 share the same ring destination. (1 VN0 credit only)
IIO2_NCB	bxxxxxx1x	IIO2
IIO3_NCB	bxxxxx1xx	IIO3
IIO4_NCB	bxxxx1xxx	IIO4
IIO5_NCB	bxxx1xxxx	IIO5
NCS	bxx1xxxxx	All IIO targets for NCS are in single mask. ORs them together
NCS_SEL	bx1xxxxxx	Selected M2p BL NCS credits

MULTI_SLOT_RCVD

- **Title:** Multi Slot Flit Received
- **Category:** Special Egress Events
- **Event Code:** 0x3e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3

- **Definition:** Multi slot flit received - S0, S1 and/or S2 populated (can use AK S0/S1 masks for AK allocations)

Table 2-219. Unit Masks for MULTI_SLOT_RCVD

Extension	umask [15:8]	Description
AD_SLOT0	bxxxxxxx1	AD - Slot 0
AD_SLOT1	bxxxxxx1x	AD - Slot 1
AD_SLOT2	bxxxxx1xx	AD - Slot 2
BL_SLOT0	bxxxx1xxx	BL - Slot 0
AK_SLOT0	bxxx1xxxx	AK - Slot 0
AK_SLOT2	bxx1xxxxx	AK - Slot 2

RxC_ARB_LOST_VN0

- **Title:** Lost Arb for VN0
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x4b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VN0 message requested but lost arbitration

Table 2-220. Unit Masks for RxC_ARB_LOST_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_ARB_LOST_VN1

- **Title:** Lost Arb for VN1
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x4c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VN1 message requested but lost arbitration

Table 2-221. Unit Masks for RxC_ARB_LOST_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_ARB_MISC

- **Title:** Arb Miscellaneous
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x4d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-222. Unit Masks for RxC_ARB_MISC

Extension	umask [15:8]	Description
PAR_BIAS_VN0	bxxxxxxx1	Parallel Bias to VN0 VN0/VN1 arbiter gave second, consecutive win to vn0, delaying vn1 win, because vn0 offered parallel ad/bl
PAR_BIAS_VN1	bxxxxxx1x	Parallel Bias to VN1 VN0/VN1 arbiter gave second, consecutive win to vn1, delaying vn0 win, because vn1 offered parallel ad/bl
NO_PROG_AD_VN0	bxxxxx1xx	No Progress on Pending AD VN0 Arbitration stage made no progress on pending ad vn0 messages because slotting stage cannot accept new message
NO_PROG_AD_VN1	bxxxx1xxx	No Progress on Pending AD VN1 Arbitration stage made no progress on pending ad vn1 messages because slotting stage cannot accept new message
NO_PROG_BL_VN0	bxxx1xxxx	No Progress on Pending BL VN0 Arbitration stage made no progress on pending bl vn0 messages because slotting stage cannot accept new message
NO_PROG_BL_VN1	bxx1xxxxx	No Progress on Pending BL VN1 Arbitration stage made no progress on pending bl vn1 messages because slotting stage cannot accept new message
ADBL_PARALLEL_WIN	bx1xxxxxx	AD, BL Parallel Win AD and BL messages won arbitration concurrently / in parallel

RxC_ARB_NOAD_REQ_VN0

- **Title:** Can't Arb for VN0
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x49
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VNo message was not able to request arbitration while some other message won arbitration

Table 2-223. Unit Masks for RxC_ARB_NOAD_REQ_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_ARB_NOAD_REQ_VN1

- **Title:** Can't Arb for VN1
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x4a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VN1 message was not able to request arbitration while some other message won arbitration

Table 2-224. Unit Masks for RxC_ARB_NOAD_REQ_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).

Table 2-224. Unit Masks for RxC_ARB_NOAD_REQ_VN1

Extension	umask [15:8]	Description
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_ARB_NOCRED_VN0

- **Title:** No Credits to Arb for VN0
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x47
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VN0 message is blocked from requesting arbitration due to lack of remote UPI credits

Table 2-225. Unit Masks for RxC_ARB_NOCRED_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_ARB_NOCRED_VN1

- **Title:** No Credits to Arb for VN1
- **Category:** INGRESS Arbitration Events
- **Event Code:** 0x48
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** VN1 message is blocked from requesting arbitration due to lack of remote UPI credits

Table 2-226. Unit Masks for RxC_ARB_NOCRED_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_BYPASSED

- **Title:** Ingress Queue Bypasses
- **Category:** INGRESS Events
- **Event Code:** 0x40
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:** Number ot times message is bypassed around the Ingress Queue

Table 2-227. Unit Masks for RxC_BYPASSED

Extension	umask [15:8]	Description
AD_S0_IDLE	bxxxxxxx1	AD to Slot 0 on Idle AD is taking bypass to slot 0 of independent flit while pipeline is idle
AD_S0_BL_ARB	bxxxxxx1x	AD to Slot 0 on BL Arb AD is taking bypass to slot 0 of independent flit while bl message is in arbitration
AD_S1_BL_SLOT	bxxxxx1xx	AD + BL to Slot 1 AD is taking bypass to flit slot 1 while merging with bl message in same flit
AD_S2_BL_SLOT	bxxxx1xxx	AD + BL to Slot 2 AD is taking bypass to flit slot 2 while merging with bl message in same flit

RxC_COLLISION_VN0

- **Title:** VN0 message lost contest for flit
- **Category:** INGRESS Slotting Events
- **Event Code:** 0x50
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:** Count cases where Ingress VN0 packets lost the contest for Flit Slot 0.

Table 2-228. Unit Masks for RxC_COLLISION_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_COLLISION_VN1

- **Title:** VN1 message lost contest for flit
- **Category:** INGRESS Sloting Events
- **Event Code:** 0x51
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:** Count cases where Ingress VN1 packets lost the contest for Flit Slot 0.

Table 2-229. Unit Masks for RxC_COLLISION_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_CRD_MISC

- **Title:** Miscellaneous Credit Events
- **Category:** INGRESS Credit Events
- **Event Code:** 0x60
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-230. Unit Masks for RxC_CRD_MISC

Extension	umask [15:8]	Description
ANY_BGF_FIFO	bxxxxxxx1	Any In BGF FIFO Indication that at least one packet (flit) is in the bgf (fifo only)
ANY_BGF_PATH	bxxxxxx1x	Any in BGF Path Indication that at least one packet (flit) is in the bgf path (i.e. pipe to fifo)
NO_D2K_FOR_ARB	bxxxxx1xx	No D2K For Arb VN0 or VN1 BL RSP message was blocked from arbitration request due to lack of D2K CMP credits

RxC_CRD_OCC

- **Title:** Credit Occupancy
- **Category:** INGRESS Credit Events
- **Event Code:** 0x61
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-231. Unit Masks for RxC_CRD_OCC

Extension	umask [15:8]	Description
VNA_IN_USE	bxxxxxxx1	VNA In Use Remote UPI VNA credit occupancy (number of credits in use), accumulated across all cycles
FLITS_IN_FIFO	bxxxxxx1x	Packets in BGF FIFO Occupancy of m3upi ingress -> upi link layer bgf; packets (flits) in fifo
FLITS_IN_PATH	bxxxxx1xx	Packets in BGF Path Occupancy of m3upi ingress -> upi link layer bgf; packets (flits) in path (i.e. pipe to fifo or fifo)
TxQ_CRD	bxxxx1xxx	Transmit Credits Link layer transmit queue credit occupancy (credits in use), accumulated across all cycles
D2K_CRD	bxxx1xxxx	D2K Credits D2K completion fifo credit occupancy (credits in use), accumulated across all cycles
P1P_TOTAL	bxx1xxxxx	count of bl messages in pump-1-pending state, in marker table and in fifo
P1P_FIFO	bx1xxxxxx	count of bl messages in pump-1-pending state, in completion fifo only

RxC_CYCLES_NE_VN0

- **Title:** VN0 Ingress (from CMS) Queue - Cycles Not Empty
- **Category:** INGRESS Events
- **Event Code:** 0x43
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of cycles when the UPI Ingress is not empty. This tracks one of the three rings that are used by the UPI agent. This can be used in conjunction with the UPI Ingress Occupancy Accumulator event in order to calculate average queue occupancy. Multiple ingress buffers can be tracked at a given time using multiple counters.

Table 2-232. Unit Masks for RxC_CYCLES_NE_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_CYCLES_NE_VN1

- **Title:** VN1 Ingress (from CMS) Queue - Cycles Not Empty
- **Category:** INGRESS Events
- **Event Code:** 0x44
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of allocations into the UPI VN1 Ingress. This tracks one of the three rings that are used by the UPI agent. This can be used in conjunction with the UPI VN1 Ingress Occupancy Accumulator event in order to calculate average queue latency. Multiple ingress buffers can be tracked at a given time using multiple counters.

Table 2-233. Unit Masks for RxC_CYCLES_NE_VN1 (Sheet 1 of 2)

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.

Table 2-233. Unit Masks for RxC_CYCLES_NE_VN1 (Sheet 2 of 2)

Extension	umask [15:8]	Description
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_FLITS_DATA_NOT_SENT

- **Title:** Data Flit Not Sent
- **Category:** INGRESS Flit Events
- **Event Code:** 0x57
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Data flit is ready for transmission but could not be sent

Table 2-234. Unit Masks for RxC_FLITS_DATA_NOT_SENT

Extension	umask [15:8]	Description
ALL	bxxxxxxx1	All
NO_BGF	bxxxxxx1x	No BGF Credits
NO_TXQ	bxxxxx1xx	No TxQ Credits

RxC_FLITS_GEN_BL

- **Title:** Generating BL Data Flit Sequence
- **Category:** INGRESS Flit Events
- **Event Code:** 0x59
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-235. Unit Masks for RxC_FLITS_GEN_BL

Extension	umask [15:8]	Description
P0_WAIT	bxxxxxxx1	Wait on Pump 0 generating bl data flit sequence; waiting for data pump 0
P1_WAIT	bxxxxxx1x	Wait on Pump 1 generating bl data flit sequence; waiting for data pump 1
P1P_TO_LIMBO	bxxxxx1xx	a bl message finished but is in limbo and moved to pump-1-pending logic
P1P_BUSY	bxxxx1xxx	pump-1-pending logic is tracking at least one message

Table 2-235. Unit Masks for RxC_FLITS_GEN_BL

Extension	umask [15:8]	Description
P1P_AT_LIMIT	bxxx1xxxx	pump-1-pending logic is at capacity (pending table plus completion FIFO at limit)
P1P_HOLD_P0	bxx1xxxxx	pump-1-pending logic is at or near capacity, such that pump-0-only bl messages are getting stalled in slotting stage
P1P_FIFO_FULL	bx1xxxxxx	pump-1-pending completion FIFO is full

RxC_FLITS_MISC

- **Title:**
- **Category:** INGRESS Flit Events
- **Event Code:** 0x5a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

RxC_FLITS_SENT

- **Title:** Sent Header Flit
- **Category:** INGRESS Flit Events
- **Event Code:** 0x56
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-236. Unit Masks for RxC_FLITS_SENT

Extension	umask [15:8]	Description
1_MSG	bxxxxxxx1	One Message One message in flit; VNA or non-VNA flit
2_MSGS	bxxxxxx1x	Two Messages Two messages in flit; VNA flit
3_MSGS	bxxxxx1xx	Three Messages Three messages in flit; VNA flit
1_MSG_VNX	bxxxx1xxx	One Message in non-VNA One message in flit; non-VNA flit
SLOTS_1	bxxx1xxxx	
SLOTS_2	bxx1xxxxx	
SLOTS_3	bx1xxxxxx	

RxC_FLITS_SLOT_BL

- **Title:** Slotting BL Message Into Header Flit
- **Category:** INGRESS Flit Events
- **Event Code:** 0x58
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-237. Unit Masks for RxC_FLITS_SLOT_BL

Extension	umask [15:8]	Description
ALL	bxxxxxxx1	All
NEED_DATA	bxxxxxx1x	Needs Data Flit BL message requires data flit sequence
P0_WAIT	bxxxxx1xx	Wait on Pump 0 Waiting for header pump 0
P1_WAIT	bxxxx1xxx	Wait on Pump 1 Waiting for header pump 1
P1_NOT_REQ	bxxx1xxxx	Don't Need Pump 1 Header pump 1 is not required for flit
P1_NOT_REQ_BUT_BUBBLE	bxx1xxxxx	Don't Need Pump 1 - Bubble Header pump 1 is not required for flit but flit transmission delayed
P1_NOT_REQ_NOT_AVAIL	bx1xxxxxx	Don't Need Pump 1 - Not Avail Header pump 1 is not required for flit and not available

RxC_FLIT_GEN_HDR1

- **Title:** Flit Gen - Header 1
- **Category:** INGRESS Flit Events
- **Event Code:** 0x53
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Events related to Header Flit Generation - Set 1

Table 2-238. Unit Masks for RxC_FLIT_GEN_HDR1

Extension	umask [15:8]	Description
ACCUM	bxxxxxxx1	Accumulate Header flit slotting control state machine is in any accumulate state; multi-message flit may be assembled over multiple cycles
ACCUM_READ	bxxxxxx1x	Accumulate Ready header flit slotting control state machine is in accum_ready state; flit is ready to send but transmission is blocked; more messages may be slotted into flit
ACCUM_WASTED	bxxxxx1xx	Accumulate Wasted Flit is being assembled over multiple cycles, but no additional message is being slotted into flit in current cycle; accumulate cycle is wasted
AHEAD_BLOCKED	bxxxx1xxx	Run-Ahead - Blocked Header flit slotting entered run-ahead state; new header flit is started while transmission of prior, fully assembled flit is blocked
AHEAD_MSG	bxxx1xxxx	Run-Ahead - Message Header flit slotting is in run-ahead to start new flit, and message is actually slotted into new flit
PAR	bxx1xxxxx	Parallel Ok New header flit construction may proceed in parallel with data flit sequence
PAR_MSG	bx1xxxxxx	Parallel Message Message is slotted into header flit in parallel with data flit sequence
PAR_FLIT	b1xxxxxxx	Parallel Flit Finished Header flit finished assembly in parallel with data flit sequence

RxC_FLIT_GEN_HDR2

- **Title:** Flit Gen - Header 2
- **Category:** INGRESS Flit Events
- **Event Code:** 0x54
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Events related to Header Flit Generation - Set 2

Table 2-239. Unit Masks for RxC_FLIT_GEN_HDR2

Extension	umask [15:8]	Description
RMSTALL	bxxxxxxx1	Rate-matching Stall Rate-matching stall injected
RMSTALL_NOMSG	bxxxxxx1x	Rate-matching Stall - No Message Rate matching stall injected, but no additional message slotted during stall cycle

RxC_FLIT_NOT_SENT

- **Title:** Header Not Sent
- **Category:** INGRESS Flit Events
- **Event Code:** 0x55
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** header flit is ready for transmission but could not be sent

Table 2-240. Unit Masks for RxC_FLIT_NOT_SENT

Extension	umask [15:8]	Description
ALL	bxxxxxxx1	All
NO_BGF_CRD	bxxxxxx1x	No BGF Credits No BGF credits available
NO_TXQ_CRD	bxxxxx1xx	No TxQ Credits No TxQ credits available
NO_BGF_NO_MSG	bxxxx1xxx	No BGF Credits + No Extra Message Slotted No BGF credits available; no additional message slotted into flit
NO_TXQ_NO_MSG	bxxx1xxxx	No TxQ Credits + No Extra Message Slotted No TxQ credits available; no additional message slotted into flit
ONE_TAKEN	bxx1xxxxx	Sent - One Slot Taken sending header flit with only one slot taken (two slots free)
TWO_TAKEN	bx1xxxxxx	Sent - Two Slots Taken sending header flit with only two slots taken (one slots free)
THREE_TAKEN	b1xxxxxxx	Sent - Three Slots Taken sending header flit with three slots taken (no slots free)

RxC_HELD

- **Title:** Message Held
- **Category:** INGRESS Sloting Events
- **Event Code:** 0x52
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:**

Table 2-241. Unit Masks for RxC_HELD

Extension	umask [15:8]	Description
VN0	bxxxxxxx1	VN0 vn0 message(s) that couldn't be slotted into last vn0 flit are held in slotting stage while processing vn1 flit
VN1	bxxxxxx1x	VN1 vn1 message(s) that couldn't be slotted into last vn1 flit are held in slotting stage while processing vn0 flit
PARALLEL_ATTEMPT	bxxxxx1xx	Parallel Attempt ad and bl messages attempted to slot into the same flit in parallel
PARALLEL_SUCCESS	bxxxx1xxx	Parallel Success ad and bl messages were actually slotted into the same flit in paralle
PARALLEL_AD_LOST	bxxx1xxxx	Parallel AD Lost some AD message lost contest for slot 0 (logical OR of all AD events under INGR_SLOT_LOST_MC_VN{0,1})
PARALLEL_BL_LOST	bxx1xxxxx	Parallel BL Lost some BL message lost contest for slot 0 (logical OR of all BL events under INGR_SLOT_LOST_MC_VN{0,1})
CANT_SLOT_AD	bx1xxxxxx	Can't Slot AD some AD message could not be slotted (logical OR of all AD events under INGR_SLOT_CANT_MC_VN{0,1})
CANT_SLOT_BL	b1xxxxxxx	Can't Slot BL some BL message could not be slotted (logical OR of all BL events under INGR_SLOT_CANT_MC_VN{0,1})

RxC_INSERTS_VN0

- **Title:** VN0 Ingress (from CMS) Queue - Inserts
- **Category:** INGRESS Events
- **Event Code:** 0x41
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of allocations into the UPI Ingress. This tracks one of the three rings that are used by the UPI agent. This can be used in conjunction with the UPI Ingress Occupancy Accumulator event in order to calculate average queue latency. Multiple ingress buffers can be tracked at a given time using multiple counters.

Table 2-242. Unit Masks for RxC_INSERTS_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.

Table 2-242. Unit Masks for RxC_INSERTS_VN0

Extension	umask [15:8]	Description
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_INSERTS_VN1

- **Title:** VN1 Ingress (from CMS) Queue - Inserts
- **Category:** INGRESS Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Counts the number of allocations into the UPI VN1 Ingress. This tracks one of the three rings that are used by the UPI agent. This can be used in conjunction with the UPI VN1 Ingress Occupancy Accumulator event in order to calculate average queue latency. Multiple ingress buffers can be tracked at a given time using multiple counters.

Table 2-243. Unit Masks for RxC_INSERTS_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_OCCUPANCY_VN0

- **Title:** VN0 Ingress (from CMS) Queue - Occupancy
- **Category:** INGRESS Events
- **Event Code:** 0x45
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Accumulates the occupancy of a given UPI VN1 Ingress queue in each cycle. This tracks one of the three ring Ingress buffers. This can be used with the UPI VN1 Ingress Not Empty event to calculate average occupancy or the UPI VN1 Ingress Allocations event in order to calculate average queuing latency.

Table 2-244. Unit Masks for RxC_OCCUPANCY_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_OCCUPANCY_VN1

- **Title:** VN1 Ingress (from CMS) Queue - Occupancy
- **Category:** INGRESS Events
- **Event Code:** 0x46
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Accumulates the occupancy of a given UPI VN1 Ingress queue in each cycle. This tracks one of the three ring Ingress buffers. This can be used with the UPI VN1 Ingress Not Empty event to calculate average occupancy or the UPI VN1 Ingress Allocations event in order to calculate average queuing latency.

Table 2-245. Unit Masks for RxC_OCCUPANCY_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.

Table 2-245. Unit Masks for RxC_OCCUPANCY_VN1

Extension	umask [15:8]	Description
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_PACKING_MISS_VN0

- **Title:** VN0 message can't slot into flit
- **Category:** INGRESS Sloting Events
- **Event Code:** 0x4e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:** Count cases where Ingress has packets to send but did not have time to pack into flit before sending to Agent so slot was left NULL which could have been used.

Table 2-246. Unit Masks for RxC_PACKING_MISS_VN0

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_PACKING_MISS_VN1

- **Title:** VN1 message can't slot into flit
- **Category:** INGRESS Sloting Events
- **Event Code:** 0x4f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-2
- **Definition:** Count cases where Ingress has packets to send but did not have time to pack into flit before sending to Agent so slot was left NULL which could have been used.

Table 2-247. Unit Masks for RxC_PACKING_MISS_VN1

Extension	umask [15:8]	Description
AD_REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
AD_SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
AD_RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_RSP	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
BL_WB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
BL_NCB	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.
BL_NCS	bx1xxxxxx	NCS on BL Non-Coherent Standard (NCS) messages on BL.

RxC_SMI3_PFTCH

- **Title:** SMI3 Prefetch Messages
- **Category:** INGRESS Credit Events
- **Event Code:** 0x62
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-248. Unit Masks for RxC_SMI3_PFTCH

Extension	umask [15:8]	Description
ARRIVED	bxxxxxxx1	Arrived
ARB_LOST	bxxxxxx1x	Lost Arbitration
SLOTTED	bxxxxx1xx	Slotted
DROP_OLD	bxxxx1xxx	Dropped - Old
DROP_WRAP	bxxx1xxxx	Dropped - Wrap Dropped because it was overwritten by new message while prefetch queue was full

RxC_VNA_CRD

- **Title:** Remote VNA Credits
- **Category:** INGRESS Credit Events
- **Event Code:** 0x5b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

Table 2-249. Unit Masks for RxC_VNA_CRD

Extension	umask [15:8]	Description
USED	bxxxxxxx1	Used Number of remote vna credits consumed per cycle
CORRECTED	bxxxxxx1x	Corrected Number of remote vna credits corrected (local return) per cycle
LT1	bxxxxx1xx	Level < 1 Remote vna credit level is less than 1 (i.e. no vna credits available)
LT4	bxxxx1xxx	Level < 4 Remote vna credit level is less than 4; bl (or ad requiring 4 vna) cannot arb on vna
LT5	bxxx1xxxx	Level < 5 Remote vna credit level is less than 5; parallel ad/bl arb on vna not possible
ANY_IN_USE	bxx1xxxxx	Any In Use At least one remote vna credit is in use

TxC_AD_ARB_FAIL

- **Title:** Failed ARB for AD
- **Category:** ARB Events
- **Event Code:** 0x30
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** AD arb but no win; arb request asserted but not won

Table 2-250. Unit Masks for TxC_AD_ARB_FAIL

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_AD_FLQ_BYPASS

- **Title:** AD FlowQ Bypass
- **Category:** Special Egress Events
- **Event Code:** 0x2c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts cases when the AD flowQ is bypassed (S0, S1 and S2 indicate which slot was bypassed with S0 having the highest priority and S2 the least)

Table 2-251. Unit Masks for TxC_AD_FLQ_BYPASS

Extension	umask [15:8]	Description
AD_SLOT0	bxxxxxxx1	
AD_SLOT1	bxxxxxx1x	
AD_SLOT2	bxxxxx1xx	
BL_EARLY_RSP	bxxxx1xxx	

TxC_AD_FLQ_CYCLES_NE

- **Title:** AD Flow Q Not Empty
- **Category:** FlowQ Events
- **Event Code:** 0x27
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles the AD Egress queue is Not Empty
- **NOTE:** Counts the number of cycles when the QPI FlowQ is not empty. This tracks one of the three rings that are used by the QPI agent. This can be used in conjunction with the QPI FlowQ Occupancy Accumulator event in order to calculate average queue occupancy. Only a single FlowQ queue can be tracked at any given time. It is not possible to filter based on direction or polarity.

Table 2-252. Unit Masks for TxC_AD_FLQ_CYCLES_NE

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_AD_FLQ_INSERTS

- **Title:** AD Flow Q Inserts
- **Category:** FlowQ Events
- **Event Code:** 0x2d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of allocations into the QPI FlowQ. This can be used in conjunction with the QPI FlowQ Occupancy Accumulator event in order to calculate average queue latency. Only a single FlowQ queue can be tracked at any given time. It is not possible to filter based on direction or polarity.

Table 2-253. Unit Masks for TxC_AD_FLQ_INSERTS

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages

Table 2-253. Unit Masks for TxC_AD_FLQ_INSERTS

Extension	umask [15:8]	Description
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages

TxC_AD_FLQ_OCCUPANCY

- **Title:** AD Flow Q Occupancy
- **Category:** FlowQ Events
- **Event Code:** 0x1c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:**

Table 2-254. Unit Masks for TxC_AD_FLQ_OCCUPANCY

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages

TxC_AD_SNPF_GRP1_VN1

- **Title:** Number of Snoop Targets
- **Category:** ARB Events
- **Event Code:** 0x3c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:** Number of snpfanout targets and non-idle cycles can be used to calculate average snpfanout latency

Table 2-255. Unit Masks for TxC_AD_SNPF_GRP1_VN1

Extension	umask [15:8]	Description
VN0_PEER_UPI0	bxxxxxxx1	Peer UPI0 on VN0 Number of VN0 Snpf to peer UPI0
VN0_PEER_UPI1	bxxxxxx1x	Peer UPI1 on VN0 Number of VN0 Snpf to peer UPI1
VN0_CHA	bxxxxx1xx	CHA on VN0 Number of VN0 Snpf to CHA
VN1_PEER_UPI0	bxxxx1xxx	Peer UPI0 on VN1 Number of VN1 Snpf to peer UPI0
VN1_PEER_UPI1	bxxx1xxxx	Peer UPI1 on VN1 Number of VN1 Snpf to peer UPI1

Table 2-255. Unit Masks for TxC_AD_SNPF_GRP1_VN1

Extension	umask [15:8]	Description
VN1_CHA	bxx1xxxxx	CHA on VN1 Number of VN1 Snpf to CHA
VN0_NON_IDLE	bx1xxxxxx	Non Idle cycles on VN0 Number of non-idle cycles in issuing Vn0 Snpf
VN1_NON_IDLE	b1xxxxxxx	Non Idle cycles on VN1 Number of non-idle cycles in issuing Vn1 Snpf

TxC_AD_SNPF_GRP2_VN1

- **Title:** Snoop Arbitration
- **Category:** ARB Events
- **Event Code:** 0x3d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Outcome of SnpF pending arbitration

Table 2-256. Unit Masks for TxC_AD_SNPF_GRP2_VN1

Extension	umask [15:8]	Description
VN0_SNPFP_NONSNP	bxxxxxxx1	FlowQ Won FlowQ txn issued when SnpF pending on Vn0
VN1_SNPFP_NONSNP	bxxxxxx1x	FlowQ Won FlowQ txn issued when SnpF pending on Vn1
VN0_SNPFP_VN2SNP	bxxxxx1xx	FlowQ SnpF Won FlowQ Vn0 SnpF issued when SnpF pending on Vn1
VN1_SNPFP_VN0SNP	bxxxx1xxx	FlowQ SnpF Won FlowQ Vn1 SnpF issued when SnpF pending on Vn0

TxC_AD_SPEC_ARB_CRD_AVAIL

- **Title:** Speculative ARB for AD - Credit Available
- **Category:** ARB Events
- **Event Code:** 0x34
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** AD speculative arb request with prior cycle credit check complete and credit avail

Table 2-257. Unit Masks for TxC_AD_SPEC_ARB_CRD_AVAIL

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_AD_SPEC_ARB_NEW_MSG

- **Title:** Speculative ARB for AD - New Message
- **Category:** ARB Events
- **Event Code:** 0x33
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** AD speculative arb request due to new message arriving on a specific channel (MC/VN)

Table 2-258. Unit Masks for TxC_AD_SPEC_ARB_NEW_MSG

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_AD_SPEC_ARB_NO_OTHER_PEND

- **Title:** Speculative ARB for AD - No Credit
- **Category:** ARB Events
- **Event Code:** 0x32
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** AD speculative arb request asserted due to no other channel being active (have a valid entry but don't have credits to send)

Table 2-259. Unit Masks for TxC_AD_SPEC_ARB_NO_OTHER_PEND

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_AK_FLQ_INSERTS

- **Title:** AK Flow Q Inserts
- **Category:** FlowQ Events
- **Event Code:** 0x2f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

TxC_AK_FLQ_OCCUPANCY

- **Title:** AK Flow Q Occupancy
- **Category:** FlowQ Events
- **Event Code:** 0x1e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:**

TxC_BL_ARB_FAIL

- **Title:** Failed ARB for BL
- **Category:** ARB Events
- **Event Code:** 0x35
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** BL arb but no win; arb request asserted but not won

Table 2-260. Unit Masks for TxC_BL_ARB_FAIL

Extension	umask [15:8]	Description
VN0_RSP	bxxxxxxx1	VN0 RSP Messages
VN0_WB	bxxxxxx1x	VN0 WB Messages
VN0_NCB	bxxxxx1xx	VN0 NCB Messages
VN0_NCS	bxxxx1xxx	VN0 NCS Messages
VN1_RSP	bxxx1xxxx	VN1 RSP Messages
VN1_WB	bxx1xxxxx	VN1 WB Messages
VN1_NCB	bx1xxxxxx	VN1 NCS Messages
VN1_NCS	b1xxxxxxx	VN1 NCB Messages

TxC_BL_FLQ_CYCLES_NE

- **Title:** BL Flow Q Not Empty
- **Category:** FlowQ Events
- **Event Code:** 0x28
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles the BL Egress queue is Not Empty
- **NOTE:** Counts the number of cycles when the QPI FlowQ is not empty. This tracks one of the three rings that are used by the QPI agent. This can be used in conjunction with the QPI FlowQ Occupancy Accumulator event in order to calculate average queue occupancy. Only a single FlowQ queue can be tracked at any given time. It is not possible to filter based on direction or polarity.

Table 2-261. Unit Masks for TxC_BL_FLQ_CYCLES_NE

Extension	umask [15:8]	Description
VN0_REQ	bxxxxxxx1	VN0 REQ Messages
VN0_SNP	bxxxxxx1x	VN0 SNP Messages
VN0_RSP	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 WB Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages

Table 2-261. Unit Masks for TxC_BL_FLQ_CYCLES_NE

Extension	umask [15:8]	Description
VN1_RSP	bx1xxxxxx	VN1 RSP Messages
VN1_WB	b1xxxxxxx	VN1 WB Messages

TxC_BL_FLQ_INSERTS

- **Title:** BL Flow Q Inserts
- **Category:** FlowQ Events
- **Event Code:** 0x2e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of allocations into the QPI FlowQ. This can be used in conjunction with the QPI FlowQ Occupancy Accumulator event in order to calculate average queue latency. Only a single FlowQ queue can be tracked at any given time. It is not possible to filter based on direction or polarity.

Table 2-262. Unit Masks for TxC_BL_FLQ_INSERTS

Extension	umask [15:8]	Description
VN0_NCB	bxxxxxxx1	VN0 RSP Messages
VN0_NCS	bxxxxxx1x	VN0 WB Messages
VN0_WB	bxxxxx1xx	VN0 NCB Messages
VN0_RSP	bxxxx1xxx	VN0 NCS Messages
VN1_NCB	bxxx1xxxx	VN1 RSP Messages
VN1_NCS	bxx1xxxxx	VN1 WB Messages
VN1_WB	bx1xxxxxx	VN1_NCS Messages
VN1_RSP	b1xxxxxxx	VN1_NCB Messages

TxC_BL_FLQ_OCCUPANCY

- **Title:** BL Flow Q Occupancy
- **Category:** FlowQ Events
- **Event Code:** 0x1d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0
- **Definition:**

Table 2-263. Unit Masks for TxC_BL_FLQ_OCCUPANCY

Extension	umask [15:8]	Description
VN0_RSP	bxxxxxxx1	VN0 RSP Messages
VN0_WB	bxxxxxx1x	VN0 WB Messages
VN0_NCB	bxxxxx1xx	VN0 NCB Messages
VN0_NCS	bxxxx1xxx	VN0 NCS Messages
VN1_RSP	bxxx1xxxx	VN1 RSP Messages
VN1_WB	bxx1xxxxx	VN1 WB Messages
VN1_NCB	bx1xxxxxx	VN1_NCS Messages
VN1_NCS	b1xxxxxxx	VN1_NCB Messages

TxC_BL_SPEC_ARB_NEW_MSG

- **Title:** Speculative ARB for BL - New Message
- **Category:** ARB Events
- **Event Code:** 0x38
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** BL speculative arb request due to new message arriving on a specific channel (MC/VN)

Table 2-264. Unit Masks for TxC_BL_SPEC_ARB_NEW_MSG

Extension	umask [15:8]	Description
VN0_WB	bxxxxxxx1	VN0 WB Messages
VN0_NCB	bxxxxxx1x	VN0 WB Messages
VN0_NCS	bxxxx1xxx	VN0 NCS Messages
VN1_WB	bxxx1xxxx	VN1 RSP Messages
VN1_NCB	bxx1xxxxx	VN1 WB Messages
VN1_NCS	b1xxxxxxx	VN1 NCB Messages

TxC_BL_SPEC_ARB_NO_OTHER_PEND

- **Title:** Speculative ARB for AD Failed - No Credit
- **Category:** ARB Events
- **Event Code:** 0x37
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** BL speculative arb request asserted due to no other channel being active (have a valid entry but don't have credits to send)

Table 2-265. Unit Masks for TxC_BL_SPEC_ARB_NO_OTHER_PEND

Extension	umask [15:8]	Description
VN0_RSP	bxxxxxxx1	VN0 RSP Messages
VN0_WB	bxxxxxx1x	VN0 WB Messages
VN0_NCB	bxxxxx1xx	VN0 NCB Messages
VN0_NCS	bxxxx1xxx	VN0 NCS Messages
VN1_RSP	bxxx1xxxx	VN1 RSP Messages
VN1_WB	bxx1xxxxx	VN1 WB Messages
VN1_NCB	bx1xxxxxx	VN1 NCS Messages
VN1_NCS	b1xxxxxxx	VN1 NCB Messages

UPI_PEER_AD_CREDITS_EMPTY

- **Title:** UPI0 AD Credits Empty
- **Category:** EGRESS Credit Events
- **Event Code:** 0x20
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** No credits available to send to UPIs on the AD Ring
- **NOTE:** 2 cases for non-smi3 mode and 3 cases for smi3 mode

Table 2-266. Unit Masks for UPI_PEER_AD_CREDITS_EMPTY

Extension	umask [15:8]	Description
VNA	bxxxxxxx1	VNA
VN0_REQ	bxxxxxx1x	VN0 REQ Messages
VN0_SNP	bxxxxx1xx	VN0 SNP Messages
VN0_RSP	bxxxx1xxx	VN0 RSP Messages
VN1_REQ	bxxx1xxxx	VN1 REQ Messages
VN1_SNP	bxx1xxxxx	VN1 SNP Messages
VN1_RSP	bx1xxxxxx	VN1 RSP Messages

UPI_PEER_BL_CREDITS_EMPTY

- **Title:** UPI0 BL Credits Empty
- **Category:** EGRESS Credit Events
- **Event Code:** 0x21
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** No credits available to send to UPI on the BL Ring (diff between non-SMI and SMI mode)
- **NOTE:** smi and non-smi modes

Table 2-267. Unit Masks for UPI_PEER_BL_CREDITS_EMPTY

Extension	umask [15:8]	Description
VNA	bxxxxxxx1	VNA
VN0_RSP	bxxxxxx1x	VN0 REQ Messages
VN0_NCS_NCB	bxxxxx1xx	VN0 RSP Messages
VN0_WB	bxxxx1xxx	VN0 SNP Messages
VN1_RSP	bxxx1xxxx	VN1 REQ Messages
VN1_NCS_NCB	bxx1xxxxx	VN1 RSP Messages
VN1_WB	bx1xxxxxx	VN1 SNP Messages

UPI_PREFETCH_SPAWN

- **Title:** FlowQ Generated Prefetch
- **Category:** Special Egress Events
- **Event Code:** 0x29
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Count cases where FlowQ causes spawn of Prefetch to iMC/SMI3 target

VN0_CREDITS_USED

- **Title:** VN0 Credit Used
- **Category:** Link VN Credit Events
- **Event Code:** 0x5c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a VN0 credit was used on the DRS message channel. In order for a request to be transferred across UPI, it must be guaranteed to have a flit buffer on the remote socket to sink into. There are two credit pools, VNA and VN0. VNA is a shared pool used to achieve high performance. The VN0 pool has

reserved entries for each message class and is used to prevent deadlock. Requests first attempt to acquire a VNA credit, and then fall back to VN0 if they fail. This counts the number of times a VN0 credit was used. Note that a single VN0 credit holds access to potentially multiple flit buffers. For example, a transfer that uses VNA could use 9 flit buffers and in that case uses 9 credits. A transfer on VN0 will only count a single credit even though it may use multiple buffers.

Table 2-268. Unit Masks for VN0_CREDITS_USED

Extension	umask [15:8]	Description
REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
WB	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
NCB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
NCS	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.

VN0_NO_CREDITS

- **Title:** VN0 No Credits
- **Category:** Link VN Credit Events
- **Event Code:** 0x5e
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of Cycles there were no VN0 Credits

Table 2-269. Unit Masks for VN0_NO_CREDITS

Extension	umask [15:8]	Description
REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
WB	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).

Table 2-269. Unit Masks for VN0_NO_CREDITS

Extension	umask [15:8]	Description
NCB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
NCS	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.

VN1_CREDITS_USED

- **Title:** VN1 Credit Used
- **Category:** Link VN Credit Events
- **Event Code:** 0x5d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of times a VN1 credit was used on the WB message channel. In order for a request to be transferred across QPI, it must be guaranteed to have a flit buffer on the remote socket to sink into. There are two credit pools, VNA and VN1. VNA is a shared pool used to achieve high performance. The VN1 pool has reserved entries for each message class and is used to prevent deadlock. Requests first attempt to acquire a VNA credit, and then fall back to VN1 if they fail. This counts the number of times a VN1 credit was used. Note that a single VN1 credit holds access to potentially multiple flit buffers. For example, a transfer that uses VNA could use 9 flit buffers and in that case uses 9 credits. A transfer on VN1 will only count a single credit even though it may use multiple buffers.

Table 2-270. Unit Masks for VN1_CREDITS_USED

Extension	umask [15:8]	Description
REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
WB	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
NCB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
NCS	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.

VN1_NO_CREDITS

- **Title:** VN1 No Credits
- **Category:** Link VN Credit Events
- **Event Code:** 0x5f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:** Number of Cycles there were no VN1 Credits

Table 2-271. Unit Masks for VN1_NO_CREDITS

Extension	umask [15:8]	Description
REQ	bxxxxxxx1	REQ on AD Home (REQ) messages on AD. REQ is generally used to send requests, request responses, and snoop responses.
SNP	bxxxxxx1x	SNP on AD Snoops (SNP) messages on AD. SNP is used for outgoing snoops.
RSP	bxxxxx1xx	RSP on AD Response (RSP) messages on AD. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
WB	bxxxx1xxx	RSP on BL Response (RSP) messages on BL. RSP packets are used to transmit a variety of protocol flits including grants and completions (CMP).
NCB	bxxx1xxxx	WB on BL Data Response (WB) messages on BL. WB is generally used to transmit data with coherency. For example, remote reads and writes, or cache to cache transfers will transmit their data using WB.
NCS	bxx1xxxxx	NCB on BL Non-Coherent Broadcast (NCB) messages on BL. NCB is generally used to transmit data without coherency. For example, non-coherent read data returns.

2.9 Power Control (PCU) Performance Monitoring

The PCU is the primary Power Controller for the Intel® Xeon® Processor Scalable Memory Family die, responsible for distributing power to core/uncore components and thermal management. It runs in firmware on an internal micro-controller and coordinates the socket's power states.

The PCU algorithmically governs the P-state of the processor, C-state of the core and the package C-state of the socket. It enables the core to go to a higher performance state ("turbo mode") when the proper set of conditions are met. Conversely, the PCU will throttle the processor to a lower performance state when a thermal violation occurs.

Through specific events, the OS and the PCU will either promote or demote the C-State of each core by altering the voltage and frequency. The system power state (S-state) of all the sockets in the system is managed by the server legacy bridge in coordination with all socket PCUs.

The PCU communicates to all the other units through multiple PMLink interfaces on-die and Message Channel to access their registers. The OS and BIOS communicates to the PCU thru standardized MSR registers and ACPI.

The PCU also acts as the interface to external management controllers via PECI and voltage regulators (NPTM). The DMI2 interface is the communication path from the southbridge for system power management.

Note: Power management is not completely centralized. Many units employ their own power saving features. Events that provide information about those features are captured in the PMON bocks of those units. For example, Intel® UPI Link Power saving states and Memory CKE statistics are captured in the Intel® UPI Perfmon and IMC Perfmon respectively.

2.9.1 PCU Performance Monitoring Overview

See Section 1.4, "Unit Level PMON State" for details about the basic unit level performance monitoring state upon which all PMON blocks are built. Extensions will be outlined in subsequent sections.

The uncore PCU supports event monitoring through four 48-bit wide counters (PCU_MSR_PMON_CTR{3:0}). Each of these counters can be programmed (PCU_MSR_PMON_CTL{3:0}) to monitor any PCU event. The PCU counters can increment by a maximum of 5b per cycle.

Four extra 64-bit counters are provided by the PCU to track P and C-State Residence. Although documented in this manual for reference, these counters exist outside of the PMON infrastructure.

2.9.2 Additional PCU Performance Monitoring

2.9.2.1 PCU PMON Counter Control - Difference from Baseline

The following table defines the difference in the layout of the PCU performance monitor control registers from the baseline presented in Chapter 1.

Since much of the PCU's functionality is provided by an embedded microcontroller, many of the available events are generated by the microcontroller and handed off to the hardware for PMON capture. Among the events generated by the microcontroller are occupancy events allowing a user to measure the number of cores in a given C-state per-cycle. Given this unique situation, extra control bits are provided to filter the output of the these special occupancy events.

- *.occ_invert* - Changes the *.thresh* test condition to '<' for the occupancy events (when .ev_sel[7] is set to 1)
- *.occ_edge_det* - Rather than accumulating the raw count each cycle (for events that can increment by 1 per cycle), the register can capture transitions from no event to an event incoming for the PCU's occupancy events (when .ev_sel[7] is set to 1).

Figure 2-13. PCU Counter Control Register for Intel® Xeon® Processor Scalable Memory Family

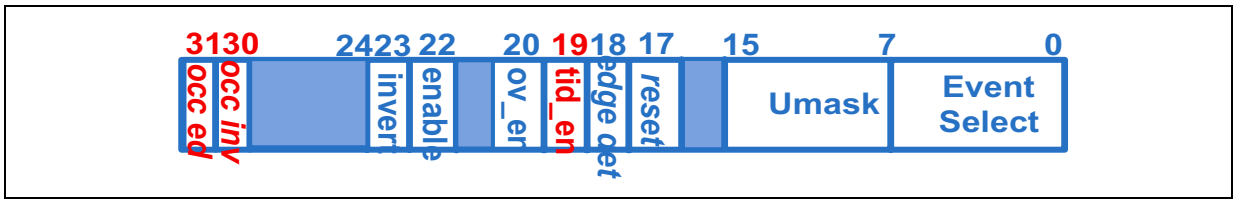


Table 2-272. PCU_MSR_PMON_CTL{3-0} Difference from Baseline – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
occ_edge_det	31	RW-V	0	Enables edge detect for occupancy events (.ev_sel[7] is 1) When set to 1, rather than measuring the event in each cycle it is active, the corresponding counter will increment when a 0 to 1 transition (i.e. rising edge) is detected. When 0, the counter will increment in each cycle that the event is asserted. NOTE: .edge_det is in series following .thresh, Due to this, the .thresh field must be set to a non-0 value. For events that increment by no more than 1 per cycle, set .thresh to 0x1.
occ_invert	30	RW-V	0	Invert comparison against Threshold for the PCU Occupancy events (.ev_sel[7] is 1) 0 - comparison will be 'is event increment >= threshold?'. 1 - comparison is inverted - 'is event increment < threshold?' NOTE: .invert is in series following .thresh, Due to this, the .thresh field must be set to a non-0 value. For events that increment by no more than 1 per cycle, set .thresh to 0x1. Also, if .edge_det is set to 1, the counter will increment when a 1 to 0 transition (i.e. falling edge) is detected.

Context sensitive filtering is provided for through the PCU_MSR_PMON_BOX_FILTER register.

- For frequency/voltage band filters, the multiplier is at 100MHz granularity. So, a value of 32 (0x20) would represent a frequency of 3.2GHz.
- Support for limited Frequency/Voltage Band histogramming. Each of the four bands provided for in the filter may be simultaneous tracked by the corresponding event

Note: Since use of the register as a filter is heavily overloaded, simultaneous application of this filter to additional events in the same run is severely limited

Table 2-273. Additional PCU Performance Monitoring Registers (MSR)

MSR Name	MSR Address	Size (bits)	Description
Fixed (Non-PMON) Counters			
PCU_MSR_CORE_P6_CTR	0x03F9	64	Fixed P-State Residency Counter
PCU_MSR_CORE_P3_CTR	0x03F8	64	Fixed P-State Residency Counter
PCU_MSR_CORE_C6_CTR	0x03FD	64	Fixed C-State Residency Counter
PCU_MSR_CORE_C3_CTR	0x03FC	64	Fixed C-State Residency Counter

Note: Address to the PCU specific filtering register can be found in Chapter 1. But there are some additional pieces of state of relevance to performance monitoring uses.

Table 2-274. PCU_MSR_PMON_BOX_FILTER Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
rsv	63:48	RV	0	Reserved
filt31_24	31:24	RW-V	0	Band 3 - For Voltage/Frequency Band Event
filt23_16	23:16	RW-V	0	Band 2 - For Voltage/Frequency Band Event
filt15_8	15:8	RW-V	0	Band 1 - For Voltage/Frequency Band Event
filt7_0	7:0	RW-V	0	Band 0 - For Voltage/Frequency Band Event

The PCU includes two extra MSRs that track the number of reference cycles a core (any core) is in either the C3 or C6 state. And the PCU includes two extra MSRs that track the number of reference cycles the package is in either the C3 or C6 state. As mentioned before, these counters are not part of the PMON infrastructure so they can't be frozen or reset with the otherwise controlled by the PCU PMON control registers.

Note: To be clear, these counters track the number of cycles **some** core is in C3/6 state. It does not track the total number of cores in the C3/6 state in any cycle. For that, a user should refer to the regular PCU event list.

Table 2-275. PCU_MSR_CORE_{C3,C6,P3,P6}_CTR Register – Field Definitions

Field	Bits	Attr	HW Reset Val	Description
event_count	63:0	RO	0	64-bit performance event counter

2.9.3 PCU Performance Monitoring Events

The PCU provides the ability to capture information covering a wide range of the PCU's functionality including:

- Number of cores in a given C-state per-cycle
- Core State Transitions - there are a larger number of events provided to track when cores transition C-state, when the enter/exit specific C-states, when they receive a C-state demotion, etc.
- Package State Transitions
- Frequency/Voltage Banding - ability to measure the number of cycles the uncore was operating within a frequency or voltage 'band' that can be specified in a separate filter register.

Note: Given the nature of many of the PCU events, a great deal of additional information can be measured by setting the *.edge_det* bit. By doing so, an event such as "Cycles Changing Frequency" becomes "Number of Frequency Transitions.

On Occupancy Events:

Because it is not possible to "sync" the PCU occupancy counters by employing tricks such as bus lock before the events start incrementing, the PCU has provided fixed occupancy counters to track the major queues.

1. Cores in C0 (4 bits)
2. Cores in C3 (4 bits)

3. Cores in C6 (4 bits)

The PCU perfmon implementation/programming is more complicated than many of the other units. As such, it is best to describe how to use them with a couple examples.

- Case 1: Cycles there was a Voltage Transition (Simple Event)
- Case 2: Cores in C0 (Occupancy Accumulation)
- Case 3: Cycles w/ more than 4 cores in C0 (Occupancy Thresholding)
- Case 4: Transitions into more than 4 cores in C0 (Thresholding + Edge Detect)
- Case 5: Cycles a) w/ > 4 Cores in C0 and b) there was a Voltage Transition
- Case 6: Cycles a) w/ <4 Cores in C0 and b) Freq < 2.0GHz

Table 2-276. PCU Configuration Examples

	Case					
Config	**1**	**2**	**3**	**4**	**5**	**6**
Counter Control 0						
.ev_sel		0x80	0x80	0x80	0x80	0x80
.occ_sel		0x1	0x1	0x1	0x1	0x1
.thresh		0x0	0x5	0x5	0x5	0x4
.invert		0	0	0	0	1
.occ_invert		0	0	0	0	1
.occ_edge_det		0	0	1	0	0
Counter Control 1						
.ev_sel	0x03				0x03	0x0B
Filter	0x00	0x00	0x00	0x00	0x00	0x14

2.9.4 PCU Box Events Ordered By Code

The following table summarizes the directly measured PCU Box events.

Symbol Name	Event Code	Ctrs	Extra Select Bit	Max Inc/ Cyc	Description
CLOCKTICKS	0x00	0-3	0	0	pclk Cycles
FREQ_MAX_LIMIT_THERMAL_CYCLES	0x04	0-3	0	0	Thermal Strongest Upper Limit Cycles
FREQ_MAX_POWER_CYCLES	0x05	0-3	0	0	Power Strongest Upper Limit Cycles
MCP_PROCHOT_CYCLES	0x06		0	0	
PMAX_THROTTLED_CYCLES	0x07		0	0	
PROCHOT_INTERNAL_CYCLES	0x09	0-3	0	0	Internal Prochot
PROCHOT_EXTERNAL_CYCLES	0x0a	0-3	0	0	External Prochot
CTS_EVENT0	0x11		0	0	
CTS_EVENT1	0x12		0	0	
PKG_RESIDENCY_C0_CYCLES	0x2a	0-3	0	0	Package C State Residency - C0
PKG_RESIDENCY_C2E_CYCLES	0x2b	0-3	0	0	Package C State Residency - C2E
PKG_RESIDENCY_C3_CYCLES	0x2c	0-3	0	0	Package C State Residency - C3

Symbol Name	Event Code	Ctrs	Extra Select Bit	Max Inc/ Cyc	Description
PKG_RESIDENCY_C6_CYCLES	0x2d	0-3	0	0	Package C State Residency - C6
MEMORY_PHASE_SHEDDING_CYCLES	0x2f	0-3	0	0	Memory Phase Shedding Cycles
DEMOTIONS	0x30		0	0	
VR_HOT_CYCLES	0x42	0-3	0	0	VR Hot
CORE_TRANSITION_CYCLES	0x60	0-3	0	0	
TOTAL_TRANSITION_CYCLES	0x72	0-3	0	0	Total Core C State Transition Cycles
FREQ_MIN_IO_P_CYCLES	0x73	0-3	0	0	IO P Limit Strongest Lower Limit Cycles
FREQ_TRANS_CYCLES	0x74	0-3	0	0	Cycles spent changing Frequency
FIVR_PS_PS0_CYCLES	0x75	0-3	0	0	Phase Shed 0 Cycles
FIVR_PS_PS1_CYCLES	0x76	0-3	0	0	Phase Shed 1 Cycles
FIVR_PS_PS2_CYCLES	0x77	0-3	0	0	Phase Shed 2 Cycles
FIVR_PS_PS3_CYCLES	0x78	0-3	0	0	Phase Shed 3 Cycles

2.9.5 PCU Box Common Metrics (Derived Events)

The following table summarizes metrics commonly calculated from PCU Box events.

Symbol Name: Definition	Equation
PCT_CYC_FREQ_CURRENT_LTD: Percentage of Cycles the Max Frequency is limited by current	FREQ_MAX_CURRENT_CYCLES / CLOCKTICKS
PCT_CYC_FREQ_POWER_LTD: Percentage of Cycles the Max Frequency is limited by power	FREQ_MAX_POWER_CYCLES / CLOCKTICKS
PCT_CYC_FREQ_THERMAL_LTD: Percentage of Cycles the Max Frequency is limited by thermal issues	FREQ_MAX_CURRENT_CYCLES / CLOCKTICKS
s: Percentage of Cycles the Max Frequency is limited by the OS	FREQ_MAX_OS_CYCLES / CLOCKTICKS

2.9.6 PCU Box Performance Monitor Event List

The section enumerates Intel® Xeon® Processor Scalable Memory Family performance monitoring events for the PCU Box.

CLOCKTICKS

- **Title:** pclk Cycles
- **Category:** MEMORY_PHASE_SHEDDING Events
- **Event Code:** 0x00
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** The PCU runs off a fixed 1 GHz clock. This event counts the number of pclk cycles measured while the counter was enabled. The pclk, like the Memory Controller's dclk, counts at a constant rate making it a good measure of actual wall time.

CORE_TRANSITION_CYCLES

- **Title:**
- **Category:** CORE_C_STATE_TRANSITION Events
- **Event Code:** 0x60
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

CTS_EVENT0

- **Title:**
- **Category:** Misc Events
- **Event Code:** 0x11
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

CTS_EVENT1

- **Title:**
- **Category:** Misc Events
- **Event Code:** 0x12
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

DEMOTIONS

- **Title:**
- **Category:** CORE_C_STATE_TRANSITION Events
- **Event Code:** 0x30
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

FIVR_PS_PS0_CYCLES

- **Title:** Phase Shed 0 Cycles
- **Category:** FIVR Events
- **Event Code:** 0x75
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Cycles spent in phase-shedding power state 0

FIVR_PS_PS1_CYCLES

- **Title:** Phase Shed 1 Cycles
- **Category:** FIVR Events
- **Event Code:** 0x76
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Cycles spent in phase-shedding power state 1

FIVR_PS_PS2_CYCLES

- **Title:** Phase Shed 2 Cycles
- **Category:** FIVR Events
- **Event Code:** 0x77
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Cycles spent in phase-shedding power state 2

FIVR_PS_PS3_CYCLES

- **Title:** Phase Shed 3 Cycles
- **Category:** FIVR Events
- **Event Code:** 0x78
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Cycles spent in phase-shedding power state 3

FREQ_MAX_LIMIT_THERMAL_CYCLES

- **Title:** Thermal Strongest Upper Limit Cycles
- **Category:** FREQ_MAX_LIMIT Events
- **Event Code:** 0x04
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when thermal conditions are the upper limit on frequency. This is related to the THERMAL_THROTTLE CYCLES_ABOVE_TEMP event, which always counts cycles when we are above the thermal temperature. This event (STRONGEST_UPPER_LIMIT) is sampled at the output of the algorithm that determines the actual frequency, while THERMAL_THROTTLE looks at the input.

FREQ_MAX_POWER_CYCLES

- **Title:** Power Strongest Upper Limit Cycles
- **Category:** FREQ_MAX_LIMIT Events
- **Event Code:** 0x05
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when power is the upper limit on frequency.

FREQ_MIN_IO_P_CYCLES

- **Title:** IO P Limit Strongest Lower Limit Cycles
- **Category:** FREQ_MIN_LIMIT Events
- **Event Code:** 0x73
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when IO P Limit is preventing us from dropping the frequency lower. This algorithm monitors the needs to the IO subsystem on both local and remote sockets and will maintain a frequency high enough to maintain good IO BW. This is necessary for when all the IA cores on a socket are idle but a user still would like to maintain high IO Bandwidth.

FREQ_TRANS_CYCLES

- **Title:** Cycles spent changing Frequency
- **Category:** FREQ_TRANS Events
- **Event Code:** 0x74
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the system is changing frequency. This can not be filtered by thread ID. One can also use it with the occupancy counter that monitors number of threads in C0 to estimate the performance impact that frequency transitions had on the system.

MCP_PROCHOT_CYCLES

- **Title:**
- **Category:** PROCHOT Events
- **Event Code:** 0x06
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**
- **Definition:**

MEMORY_PHASE_SHEDDING_CYCLES

- **Title:** Memory Phase Shedding Cycles
- **Category:** MEMORY_PHASE_SHEDDING Events
- **Event Code:** 0x2f
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles that the PCU has triggered memory phase shedding. This is a mode that can be run in the iMC physicals that saves power at the expense of additional latency.
- **NOTE:** Package C1

PKG_RESIDENCY_C0_CYCLES

- **Title:** Package C State Residency - C0
- **Category:** PKG_C_STATE_RESIDENCY Events
- **Event Code:** 0x2a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the package was in C0. This event can be used in conjunction with edge detect to count C0 entrances (or exits using invert). Residency events do not include transition times.

PKG_RESIDENCY_C2E_CYCLES

- **Title:** Package C State Residency - C2E
- **Category:** PKG_C_STATE_RESIDENCY Events
- **Event Code:** 0x2b
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the package was in C2E. This event can be used in conjunction with edge detect to count C2E entrances (or exits using invert). Residency events do not include transition times.

PKG_RESIDENCY_C3_CYCLES

- **Title:** Package C State Residency - C3
- **Category:** PKG_C_STATE_RESIDENCY Events
- **Event Code:** 0x2c
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the package was in C3. This event can be used in conjunction with edge detect to count C3 entrances (or exits using invert). Residency events do not include transition times.

PKG_RESIDENCY_C6_CYCLES

- **Title:** Package C State Residency - C6
- **Category:** PKG_C_STATE_RESIDENCY Events
- **Event Code:** 0x2d
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles when the package was in C6. This event can be used in conjunction with edge detect to count C6 entrances (or exits using invert). Residency events do not include transition times.

PMAX_THROTTLED_CYCLES

- **Title:**
- **Category:** FREQ_MAX_LIMIT Events
- **Event Code:** 0x07
- **Max. Inc/Cyc:.** 0, **Register Restrictions:**

- **Definition:**

PROCHOT_EXTERNAL_CYCLES

- **Title:** External Prochot
- **Category:** PROCHOT Events
- **Event Code:** 0x0a
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles that we are in external PROCHOT mode. This mode is triggered when a sensor off the die determines that something off-die (like DRAM) is too hot and must throttle to avoid damaging the chip.

PROCHOT_INTERNAL_CYCLES

- **Title:** Internal Prochot
- **Category:** PROCHOT Events
- **Event Code:** 0x09
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Counts the number of cycles that we are in Interal PROCHOT mode. This mode is triggered when a sensor on the die determines that we are too hot and must throttle to avoid damaging the chip.

TOTAL_TRANSITION_CYCLES

- **Title:** Total Core C State Transition Cycles
- **Category:** CORE_C_STATE_TRANSITION Events
- **Event Code:** 0x72
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:** Number of cycles spent performing core C state transitions across all cores.

VR_HOT_CYCLES

- **Title:** VR Hot
- **Category:** VR_HOT Events
- **Event Code:** 0x42
- **Max. Inc/Cyc:.** 0, **Register Restrictions:** 0-3
- **Definition:**

§

3 Reference for PMON Filtering

3.1 Packet Matching Reference(s)

3.1.1 Reference for CHA Packet Matching

In the CHA, the component responsible for managing the last-level cache and maintaining coherency, the performance monitoring infrastructure allows a user to filter IDI packet traffic tracked in the TOR according to certain fields. The Message Class/ Opcode fields have been summarized in the following tables.

Note that the TOR is comprised of different logical queues managing different classes of requests. The opcodes relevant to each logical queue class are presented in separate tables.

The following tables list the IDI opcodes, broken down by queue, that can be matched on with the above filter. The two opcode match fields operate independently and the results are ORed together. It is not possible to measure events filtered by opcode that match in different fields

IDI opcodes relevant to the IRQ (Ingress Request Queue)

Table 3-1. Opcode Match by IDI Packet Type (relevant to IRQ) for Cn_MSR_PMON_BOX_FILTER.opc (Sheet 1 of 2)

opc Value	Opcode	Defn
0x200	RFO	Demand Data RFO - Read for Ownership requests from core for lines to be cached in E
0x201	CRd	Demand Code Read - Full cache-line read requests from core for lines to be cached in S, typically for code
0x202	DRd	Demand Data Read - Full cache-line read requests from core for lines to be cached in S or E, typically for data
0x207	PRd	Partial Reads (UC) - Partial read requests of 0-32B (IIO can be up to 64B). Uncacheable.
0x20C	WCiLF	Streaming Store - Full - Write invalidate for full cache line of write combining stores
0x20D	WCiL	Streaming Store - Partial - Write invalidate for partial cache line of write combining stores
0x20E	UCRdF	Uncacheable Reads - Full - Full-line uncachable read requests.
0x20F	WiL	Write Invalidate Line - Partial
0x21E	RdCur	Read current - Read Current requests from IIO. Used to read data without changing state.
0x243	WbPushHint	
0x244	WbMtoI	Request writeback Modified invalidate line - Evict full M-state cache line from core. Guarantees core has no cached copies.

Table 3-1. Opcode Match by IDI Packet Type (relevant to IRQ) for Cn_MSR_PMON_BOX_FILTER.opc (Sheet 2 of 2)

opc Value	Opcode	Defn
0x245	WbMtoE	Request writeback Modified set to Exclusive - Evict full M-state cache line from core.
0x246	WbEFtoI	Request "clean" (E or F -state line) writeback - Core guarantees it will no longer retain ownership of the line when the write-back completes
0x247	WbEFtoE	Request "clean" (E or F -state line) writeback - Core may retain ownership of the line when the write-back completes.
0x248	ItoM	Request Invalidate Line - Request Exclusive Ownership of cache line
0x258	LlcPrefRFO	LLC Prefetch RFO - Uncore will first look up the line in the LLC; for a cache hit, the LRU will be updated, on a miss, the RFO will be initiated
0x259	LlcPrefCode	LLC Prefetch Code - Uncore will first look up the line in the LLC; for a cache hit, the LRU will be updated, on a miss, the CRd will be initiated
0x25A	LlcPrefData	LLC Prefetch Data - Uncore will first look up the line in the LLC; for a cache hit, the LRU will be updated, on a miss, the DRd will be initiated
0x279	IntLog	Interrupt (Logically Addressed)
0x27A	IntPhy	Interrupt (Physically Addressed)
0x27B	IntPriUp	Interrupt Priority Update
0x27E	SplitLock	SplitLock - Request to start split lock sequence
0x27F	Lock	Lock - Request to start IDI lock sequence

IDI opcodes relevant to the IPQ (Ingress Probe Queue)

Table 3-2. Opcode Match by IDI Packet Type (relevant to IPQ) for Cn_MSR_PMON_BOX_FILTER.opc

opc Value	Opcode	Defn
0x180	SnpCur	Snoop Current - Snoop to get uncacheable ‘snapshot’ of data
0x181	SnpCode	Snoop Code - Snoop requests from the uncore for lines intended to be cached in S at requester
0x182	SnpData	Snoop Data - Snoop requests from the uncore for lines intended to be cached in S or E state at the requester (the E state can be cached at requester if all cores respond with RspI)
0x183	SnpDataMig	Snoop Data Migratory - Snoop to get data in M, E, or S
0x184	SnpInvOwn	Snoop Invalidate Own - Snoop Invalidate Own - get data in M or E
0x185	SnpInv	Snoop Invalidate - Snoop requests from the uncore for lines intended to be cached in E state at the requester

IDI opcodes relevant to the RRQ (Remote Request Queue)

Table 3-3. Opcode Match by IDI Packet Type (relevant to RRQ) for Cn_MSR_PMON_BOX_FILTER.opc

opc Value	Opcode	Defn
0x080	RdCur	Read Current - Request cache line in I. Typically issued by I/O proxy entities, RdCur is used to obtain a coherent snapshot of an uncached line
0x081	RdCode	Read Code - Read cache line in S
0x082	RdData	Read Data - Request cache line in either E or S. The choice between S and E is determined by whether or not per caching agent has cache line in S state
0x083	RdDataMig	Read Data Migratory - Same as RdData, except that peer cache can forward requested cache line in M state without any writeback to memory.
0x084	RdInvOwn	Read Invalidate Own - Read Invalidate Own requests a cache line in M or E state. M or E is determined by whether requester is forwarded an M copy by a peer caching agent or sent an E copy by home agent
0x085	RdInvXtoI	Read Invalidate X to I -
0x086	RdPushHint	Read Push Hint -
0x087	RdInvItoE	Read Invalidate I to E -
0x08C	RdInv	Read Invalidate - Request cache line in E from the home agent; any modified copy is committed to memory before receiving the data.
0x08F	RdInvItoM	Read Invalidate I to M -

IDI opcodes relevant to the WBQ (Writeback Queue)

Table 3-4. Opcode Match by IDI Packet Type (relevant to WBQ) for Cn_MSR_PMON_BOX_FILTER.opc (Sheet 1 of 2)

opc Value	Opcode	Defn
0x000	WbMtoI	Writeback M to I - Evict full M-state cache line from core. Guarantees core has no cached copies. Write a cache line in M state back to memory and invalidate the line in the cache.
0x001	WbMtoS	Writeback M to S - Write a cache line in M state back to memory and transition its state to S.
0x002	WbMtoE	Writeback M to E - Evict full M-state cache line from core. Write a cache line in M state back to memory and transition its state to E
0x003	NonsnpWr	Non-Snoop Write - Write a line to memory.
0x004	WbMtoIPtl	Writeback M to I Partial - Write a cache line in M state back to memory, according to a byte-enable mask, and transition its state to I.

Table 3-4. Opcode Match by IDI Packet Type (relevant to WBQ) for Cn_MSR_PMON_BOX_FILTER.opc (Sheet 2 of 2)

opc Value	Opcode	Defn
0x006	WbMtoEPtl	Writeback M to E Partial - Write a cache line in M state back to memory, according to a byte-enable mask, transition the line to E, and clear the line's mask in the cache.
0x007	NonsnpWrPtl	Non-Snoop Write Partial - Write a line to memory according to byte-enable mask.
0x008	WbPushMtoI	Writeback Push M to I - Push cache line in M state to the HA; HA may push data to a local cache (in M state) or write the data to memory. Transition cache line to I.
0x00B	WbFlush	Writeback Flush - Hint for flushing writes in memory hierarchy. No data is sent with the request.
0x00C	EvctCln	Evict Clean - Notification to home that a cache line in E state was invalidated in the cache
0x00D	NonSnpRd	Non-Snoop Read - Request a read only line (i.e. an uncacheable 'snapshot') from memory.

3.1.2 Reference for Intel UPI LL Packet Matching

In the Intel® UPI Link Layer, the component responsible for transmitting and receiving traffic crossing between sockets in a multi-socket machine, the performance monitoring infrastructure allows a user to filter UPI packet traffic according to certain fields. A couple common fields, the Message Class/Opcode fields, have been summarized in the following tables.

Table 3-5. Intel® UPI Interconnect Packet Message Classes

Code	Name	Definition
b0000	REQ	Requests
b0001	SNP	Snoop
b0010	RSP - NoData	Non-Data Responses
b0011	---	
b0100	RSP - Data	Data Response
b0101	WB	Writebacks
b0110	NCB	Non-Coherent Bypass
b0111	NCS	Non-Coherent Standard

Table 3-6. Intel UPI Opcode Match by Message Class (Sheet 1 of 2)

Opc	REQ	SNP	RSP2- NoData	
0000	RdCur	SnpCur	CmpU	
0001	RdCode	SnpCode	P2PCmpU	
0010	RdData	SnpData	RspI	
0011	RdDataMig	SnpDataMig	RspS	
0100	RdInvOwn	SnpInvOwn	RspFwd	
0101	InvXtoI	SnpInv	RspFwdI	

Table 3-6. Intel UPI Opcode Match by Message Class (Sheet 2 of 2)

Opc	REQ	SNP	RSP2- NoData	
0110	---	---	RspFwdS	
0111	InvItoE	---	---	
1000	---	SnpFCur	MirCmpU	
1001	---	SnpFCode	---	
1010	---	SnpFData	RspCnflt	
1011	---	SnpFDataMig	---	
1100	RdInv	SnpFInvOwn	CmpO	
1101	---	SnpFInv	FwdCnfltO	
1110	---	---	---	
1111	InvItoM	---	---	
Opc	**RSP4 - Data**	**WB**	**NCB**	**NCS**
0000	Data_M	WbMtoI	NcWr	NcRd
0001	Data_E	WbMtoS	WcWr	IntAck
0010	Data_SI	WbMtoE	---	---
0011	---	NonSnpWr	---	---
0100	Data_M_CmpO	WbMtoIPtl	---	NcRdPtl
0101	Data_E_CmpO	---	---	NcCfgRd
0110	Data_SI_CmpO	WbMtoEPtl	---	NcLTRd
0111	---	NonSnpWrPtl	---	NcIORd
1000	---	WbPushMtoI	NcMsgB	NcMsgS
1001	---	---	IntLogical	NcCfgWr
1010	RspFwdIWb	---	IntPhysical	NcLTWr
1011	RspFwdSWb	WBFlush	IntPrioUpd	NcIOWr
1100	RspIWb	EvctCln	NcWrPtl	---
1101	RspSWb	NonSnpRd	WcWrPtl	---
1110	---	---	---	---
1111	DebugData	---	NcP2PB	NcP2PS

Note: Opcodes in Italics aren't implemented in Intel® Xeon® Processor Scalable Memory Family.

Table 3-7. Intel UPI Opcodes (Alphabetical Listing) (Sheet 1 of 4)

Name	Opc	Msg Class	Gen By	Desc
CmpO	1100	RSP2		Completion message with no ordering requirements
CmpU	0000	RSP2		Completion message that must be ordered with forward responses.
DataE	0001	RSP4		Data in E
DataE_CmpO	0101	RSP4		Data in E with an ordered completion response
DataM	0000	RSP4		Data in M
DataM_CmpO	0100	RSP4		Data in M with an ordered completion response
DataSI	0010	RSP4		Depending on request, data in S or uncacheable 'snapshot' of data

Table 3-7. Intel UPI Opcodes (Alphabetical Listing) (Sheet 2 of 4)

Name	Opc	Msg Class	Gen By	Desc
DataSI_CmpO	0110	RSP4		Depending on request, data in S or uncacheable 'snapshot' of data; with an ordered completion response
DebugData	1111	RSP4		Debug Data
EvctCln	1100	SNP		Notification to home that a cache line in E state was invalidated in the cache
FwdCnfltO	1101	WB		Ordered response from home agent to resolve conflict situation and let receiver properly process original snoop request. There is always a pre-allocated resource to sink the FwdCnfltO in the coherence agent.
IntAck	0001	NCS		Interrupt acknowledge to legacy 8259 interrupt controller
IntLogical	1001	NCB		Logical mode interrupt to processor
IntPhysical	1010	NCB		Physical mode interrupt to processor
IntPrioUpd	1011	NCB		Interrupt priority update message to source interrupt agents.
InvItoE	0111	REQ		Invalidate to E state. Requests exclusive ownership of a cache line without receiving data.
InvItoM	1111	REQ		Invalidate to M state. Requests exclusive ownership of a cache line without receiving data and with the intent of performing a writeback soon afterword.
InvXtoI	0101	REQ		Flush a cache line from all caches (that is, downgrade all clean copies to I and cause any dirty copy to be written back to memory). Requesting agent must invalidate the line in its cache before issuing this request
NcCfgRd	0101	NCS		Configuration read from configuration space
NcCfgWr	1001	NCS		Configuration write to configuration space
NcIORd	0111	NCS		Read from legacy I/O space
NcIOWr	1011	NCS		Write to legacy I/O space
NcMsgB	1000	NCB		Non-coherent Message (non-coherent bypass channel)
NcMsgS	1000	NCS		Non-coherent Message (Non-coherent standard channel)
NcP2PB	1111	NCB		Peer-to-peer transaction between I/O entities (non-coherent bypass channel)
NcP2PS	1111	NCS		Peer-to-peer transaction between I/O entities. (Non-coherent standard channel)
NcRd	0000	NCS		Read from non-coherent memory mapped I/O space
NcRdPtl	0100	NCS		Partial read from non-coherent memory mapped I/O space
NcWr	0000	NCB		Write to non-coherent memory mapped I/O space
NcWrPtl	1100	NCB		Partial write to non-coherent memory mapped I/O space
NonSnpRd	1101	WB		Request a read only line (i.e. an uncacheable 'snapshot') from memory.
NonSnpWr	0011	WB		Write a line to memory.
NonSnpWrPtl	0111	WB		Write a line to memory according to byte-enable mask.
P2PCmpU	0001	RSP2		Peer-to-peer codmpletion message that must be ordered with forward responses.

Table 3-7. Intel UPI Opcodes (Alphabetical Listing) (Sheet 3 of 4)

Name	Opc	Msg Class	Gen By	Desc
RdCode	0001	REQ		Read cache line in S
RdCur	0000	REQ		Request cache line in I. Typically issued by I/O proxy entities, RdCur is used to obtain a coherent snapshot of an uncached line.
RdData	0010	REQ		Request cache line in either E or S. The choice between S and E is determined by whether or not per caching agent has cache line in S state.
RdDataMig	0011	REQ		Same as RdData, except that peer cache can forward requested cache line in M state without any writeback to memory.
RdInv	1100	REQ		Request cache line in E from the home agent; any modified copy is committed to memory before receiving the data.
RdInvOwn	0100	REQ		Read Invalidate Own requests a cache line in M or E state. M or E is determined by whether requester is forwarded an M copy by a peer caching agent or sent an E copy by home agent.
RspCnflt	1010	RSP2		Peer has outstanding request to same address, is requesting an ordered forward response, and has allocated a resource for the forward.
RspFwd	0100	RSP2		Copy of cache line was sent to requesting agent, cache state did not change
RspFwdI	0101	RSP2		Copy of cache line was sent to requesting agent, cache state was downgraded to I
RspFwdIWb	1010	RSP4		Modified line is being implicitly written back to memory, a copy of cache line was sent to requesting agent and the line was downgraded to I
RspFwdS	0110	RSP2		Copy of cache line was sent to requesting agent, cache state was downgraded to S
RspFwdSWb	1011	RSP4		Modified line is being implicitly written back to memory, a copy of cache line was sent to requesting agent and the line was downgraded to S
RspI	0010	RSP2		Cache is in I
RspIWb	1100	RSP4		Modified line is being implicitly written back to memory, cache line was downgraded to I
RspS	0011	RSP2		Cache is in S
RspSWb	1101	RSP4		Modified line is being implicitly written back to memory, cache line was downgraded to S
SnpCode	0001	SNP		Snoop Code - get data in S
SnpCur	0000	SNP		Snoop to get uncacheable ‘snapshot’ of data
SnpData	0010	SNP		Snoop Data - get data in E or S
SnpDataMig	0011	SNP		Snoop to get data in M, E, or S
SnpFCode	1001	SNP		Snoop Code - get data in S; Routing layer will handle distribution to all fanout peers
SnpFCur	1000	SNP		Snoop to get uncacheable ‘snapshot’ of data; Routing layer will handle distribution to all fanout peers
SnpFData	1010	SNP		Snoop Data - get data in E or S; Routing layer will handle distribution to all fanout peers
SnpFDataMig	1011	SNP		Snoop to get data in M, E, or S; Routing layer will handle distribution to all fanout peers
SnpFInv	1101	SNP		Snoop to invalidate peer’s cache, flushing any M copy to memory; Routing layer will handle distribution to all fanout peers

Table 3-7. Intel UPI Opcodes (Alphabetical Listing) (Sheet 4 of 4)

Name	Opc	Msg Class	Gen By	Desc
SnpFInvOwn	1100	SNP		Snoop Invalidate Own - get data in M or E; Routing layer will handle distribution to all fanout peers
SnpInv	0101	SNP		Snoop to invalidate peer's cache, flushing any M copy to memory
SnpInvOwn	0100	SNP		Snoop Invalidate Own - get data in M or E
SnpInvXtoI	1100	SNP		Snoop Invalidate Writeback M to I state. To invalidate peer caching agent, flushing any M state data to home.
WBFlush	1011	WB		Hint for flushing writes in memory hierarchy. No data is sent with the request.
WbMtoE	0010	WB		Write a cache line in M state back to memory and transition its state to E.
WbMtoEPtl	0110	WB		Write a cache line in M state back to memory, according to a byte-enable mask, transition the line to E, and clear the line's mask in the cache.
WbMtoI	0000	WB		Write a cache line in M state back to memory and invalidate the line in the cache.
WbMtoIPtl	0100	WB		Write a cache line in M state back to memory, according to a byte-enable mask, and transition its state to I.
WbMtoS	0001	WB		Write a cache line in M state back to memory and transition its state to S.
WbPushMtoI	1000	WB		Push cache line in M state to the HA; HA may push data to a local cache (in M state) or write the data to memory. Transition cache line to I.
WcWr	0001	NCB		Write combinable write to non-coherent memory mapped I/O space
WcWrPtl	1101	NCB		Partial write combinable write to non-coherent memory mapped I/O space

3.1.3 Reference for M2M Packet Matching

In M2M, the component that connects the Memory Controller to the Mesh, the performance monitoring infrastructure allows a user to filter SMI3 packet traffic according to certain fields. The Message Class/Opcode fields have been summarized in the following tables.

Table 3-8. SMI3 Opcode Match by Message Class (Sheet 1 of 2)

Opc	REQ	SNP	RSP2- NoData	
0000	MemRd	---	CmpU	
0001	MemSpecRd	---	---	
0010	MemRdData	---	---	
0011	---	---	---	
0100	MemRdXtoS	---	---	
0101	MemRdXtoI	---	---	
0110	MemRdXtoA	---	---	
0111	---	---	---	
1000	MemInv	---	---	

Table 3-8. SMI3 Opcode Match by Message Class (Sheet 2 of 2)

Opc	REQ	SNP	RSP2- NoData	
1001	MemInvXtoI	---	---	
1010	MemInvXtoA	---	---	
1011	---	---	---	
1100	MemInvItoX	---	---	
1101	---	---	---	
1110	---	---	---	
1111	---	---	MultCmpUd	
Opc	**RSP4 - Data**	**WB**	**NCB**	**NCS**
0000	---	MemWr	---	---
0001	---	---	---	---
0010	---	---	---	---
0011	---	MemWrNI	---	---
0100	---	MemWrPtl	---	---
0101	---	---	---	---
0110	MemData	---	---	---
0111	---	MemWrPtlNI	---	---
1000	---	---	---	---
1001	---	---	---	---
1010	---	---	---	---
1011	---	MemWrFlush	---	---
1100	---	---	---	---
1101	---	---	---	---
1110	---	---	---	---
1111	---	---	---	---

Note: Opcodes in Italics aren't implemented in Intel® Xeon® Processor Scalable Memory Family.

Table 3-9. SMI3 Opcodes (Alphabetical Listing) (Sheet 1 of 2)

Name	Opc	Msg Class	Gen By	Desc
CmpU	0000	RSP2		Completion for MemWr, MemWrNI, MemWrPtl, MemWrPtlNI and all MemInv* commands. For write completions, it is sent when the write becomes globally visible.
MemRd	0000	REQ		Normal memory read. Directory information is left unmodified.
MemSpecRd	0001	REQ		Speculative Read. This is a hint to the memory controller that a read to this address will likely occur. It allows the memory controller to get a head start on the read to reduce average latency. If the demand read is received by the time the data comes back. It will be used. Otherwise the data may be dropped.
MemRdData	0010	REQ		Memory Read, rewrite with directory (if necessary) as follows: I to A S has no change A has no change. If a change is required from A, the host must specifically change it.

Table 3-9. SMI3 Opcodes (Alphabetical Listing) (Sheet 2 of 2)

Name	Opc	Msg Class	Gen By	Desc
MemRdXtoS	0100	REQ		Memory read, directory result is Shared. If the directory indicates anything other than S, it is written back as S
MemRdXtoI	0101	REQ		Memory read, Directory result is Invalid. If the directory indicates anything other than I, it is written back as I
MemRdXtoA	0110	REQ		Memory read, Directory result is Any. If the directory indicates anything other than A, it is written back as A
MemInv	1000	REQ		Memory Invalidate. The command reads and returns the directory state. No directory update is done.
MemInvXtoI	1001	REQ		Memory Invalidate. Directory result is Invalid. No read data is returned. The memory controller reads the cache line and rewrites it with the directory set to I if not already that way.
MemInvXtoA	1010	REQ		Memory Invalidate. Directory result is Any. No read data is returned. The memory controller reads the cache line and rewrites it with the directory set to A if not already that way.
MemInvItoX	1100	REQ		Memory controller reads Near Memory. Tag is not updated in NM regardless of hit or miss If it a NM miss: - No Far Memory read is performed - if NM was Modified, then the data is written to Far Memory since a later write to a different address within this set may overwrite it. If it is an NM hit: - No change to NM or FM
MultCmpU	1111	RSP2		Multiple Completion for MemWr, MemWrNI, MemWrPtl, MemWrPtlNI and all MemInv* commands. Up to 8 completions can be sent in this flit.
MemData	0110	RSP4		Memory Read Data. Returned in response to all MemRd commands (but not MemSpecRd). These commands may also use the headerless data return, however there are cases where the headered version must be used.
MemWr	0000	WB		Memory Write. Used for full line writes in 1LM and for Inclusive full line writes in 2LM
MemWrNI	0011	WB		Memory Write Non-Inclusive. Write checks NM tag and if it is a hit in NM, then updates data/directory NM. If a miss in NM then writes to FM only.
MemWrPtl	0100	WB		Memory Write Partial. Used for partial line writes in 1LM and for Inclusive full line writes in 2LM
MemWrPtlNI	0011	WB		Memory Write Partial Non-Inclusive. Write checks NM tag and if it is a hit in NM, then updates data/directory NM. If a miss in NM then writes to FM only. Not used for 1LM.
MemWrFlush	1011	WB		Memory Write Flush. This command is sent as a result of a PCommit (Persistent Commit) in the host. The memory controller should flush all writes to persistent memory before returning the completion.

§