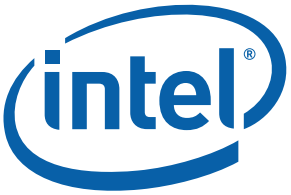


Intel® Itanium® Processor 9500 Series Reference Manual

Software Development and Optimization Guide

July 2012

Reference Number: 327132-001

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Contents

Figures

Tables

Revision History

Document Number	Revision Number	Description	Date
327132	001	• Initial release	July 2012

§

1 Introduction

This document provides microarchitectural description of the Intel® Itanium® processor 9500 series (formerly code-named Poulson), performance monitor information, and other guidance to assist software developers in optimizing for the this Intel® Itanium® processor.

1.1 Terminology

The following definitions are for terms that will be used throughout this document:

Term	Definition
Dispersal	The process of mapping instructions within bundles to functional units
Bundle rotation	The process of bringing new bundles into the two-bundle issue window
Split issue	Instruction execution when an instruction does not issue at the same time as the instruction immediately before it.
Advanced load address table (ALAT)	The ALAT holds the state necessary for advanced load and check operations.
Translation lookaside buffer (TLB)	The TLB holds virtual to physical address mappings
Virtual hash page table (VHPT)	The VHPT is an extension of the TLB hierarchy, which resides in the virtual memory space, is designed to enhance virtual address translation performance.
Hardware page walker (HPW)	The HPW is the third level of address translation. It is an engine that performs page look-ups from the VHPT and seeks opportunities to insert translations into the processor TLBs.
Register stack engine (RSE)	The RSE moves registers between the register stack and the backing store in memory.
Event address registers (EARs)	The EARs record the instruction and data addresses of data cache misses.

1.2 Related Documentation

The reader of this document should also be familiar with the material and concepts presented in the following documents:

- 2.3 *Intel® Itanium® Architecture Software Developer's Manual, Volume 1: Application Architecture*
- 2.3 *Intel® Itanium® Architecture Software Developer's Manual, Volume 2: System Architecture*
- 2.3 *Intel® Itanium® Architecture Software Developer's Manual, Volume 3: Instruction Set Reference*
- *Intel® Itanium® Architecture Software Developer's Manual Specification Update*
- *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization*
- *Dual-Core Update to the Intel® Itanium® 2 Processor Reference Manual*

1.3 Identifying Intel® Itanium® Processors

The eight generations of the Intel® Itanium® processor can be identified by their unique CPUID values. For simplicity of documentation, throughout this document we will group all processors of like model together. Table 1-1 lists the CPUID values of all of the Intel® Itanium® processors. Table 1-2 lists Intel® Itanium® processors and their grouping.

Table 1-1. Intel® Itanium® Processor Family and Model Values

Family	Model	Description
0x07	0x00	Intel® Itanium® Processor
0x1f	0x00	Intel® Itanium® Processor (up to 3 MB L3 cache)
0x1f	0x01	Intel® Itanium® Processor (up to 6 MB L3 cache)
0x1f	0x02	Intel® Itanium® Processor (up to 9 MB L3 cache)
0x20	0x00	Intel® Itanium® Processor 9000 Series
0x20	0x01	Intel® Itanium® Processor 9100 Series
0x20	0x02	Intel® Itanium® Processor 9300 Series
0x21	0x00	Intel® Itanium® Processor 9500 Series

Table 1-2. Definition Table (Sheet 1 of 2)

Processor	Abbreviation
Intel® Itanium® Processor 900 MHz with 1.5 MB L3 Cache	Intel® Itanium® Processor (up to 3 MB L3 cache)
Intel® Itanium® Processor 1.0 GHz with 3 MB L3 Cache	
Low Voltage Intel® Itanium® Processor 1.0 GHz with 1.5 MB L3 Cache	Intel® Itanium® Processor (up to 6 MB L3 cache)
Intel® Itanium® Processor 1.40 GHz with 1.5 MB L3 Cache	
Intel® Itanium® Processor 1.40 GHz with 3 MB L3 Cache	
Intel® Itanium® Processor 1.60 GHz with 3 MB L3 Cache	
Intel® Itanium® Processor 1.30 GHz with 3 MB L3 Cache	
Intel® Itanium® Processor 1.40 GHz with 4 MB L3 Cache	
Intel® Itanium® Processor 1.50 GHz with 6 MB L3 Cache	
Low Voltage Intel® Itanium® Processor 1.30 GHz with 3 MB L3 Cache	Intel® Itanium® Processor (up to 9 MB L3 cache)
Intel® Itanium® Processor 1.60 GHz with 3 MB L3 Cache at 400 and 533 MHz System Bus (DP Optimized)	
Intel® Itanium® Processor 1.50 GHz with 4 MB L3 Cache	
Intel® Itanium® Processor 1.60 GHz with 6 MB L3 Cache	
Intel® Itanium® Processor 1.60 GHz with 9 MB L3 Cache	
Intel® Itanium® Processor 1.66 GHz with 6 MB L3 Cache	
Intel® Itanium® Processor 1.66 GHz with 9 MB L3 Cache	
Intel® Itanium® Processor 9010	Intel® Itanium® Processor 9000 Series (dual-core)
Intel® Itanium® Processor 9015	
Intel® Itanium® Processor 9020	
Intel® Itanium® Processor 9030	
Intel® Itanium® Processor 9040	
Intel® Itanium® Processor 9050	

Table 1-2. Definition Table (Sheet 2 of 2)

Processor	Abbreviation
Intel® Itanium® Processor 9110N	Intel® Itanium® Processor 9100 Series (dual-core)
Intel® Itanium® Processor 9120N	
Intel® Itanium® Processor 9130M	
Intel® Itanium® Processor 9140N	
Intel® Itanium® Processor 9140M	
Intel® Itanium® Processor 9150N	
Intel® Itanium® Processor 9150M	
Intel® Itanium® Processor 9310	Intel® Itanium® Processor 9300 Series (quad-core)
Intel® Itanium® Processor 9320	
Intel® Itanium® Processor 9330	
Intel® Itanium® Processor 9340	
Intel® Itanium® Processor 9350	
Intel® Itanium® Processor 9520	Intel® Itanium® Processor 9500 Series (eight-core)
Intel® Itanium® Processor 9540	
Intel® Itanium® Processor 9550	
Intel® Itanium® Processor 9560	

§

2 The Intel Itanium Processor 9500 series Core

The Intel Itanium processor 9500 series introduces a redesigned Itanium architecture core, which significantly improves both frequency and power efficiency through many micro-architectural and technology advancements. Although the processor core can trace its origin to the Intel Itanium processor 9300 series core, it is also the first major revamp of the Intel Itanium core microarchitecture since the original Itanium® core microarchitecture. The Intel Itanium processor 9500 series core has been re-optimized from the ground-up for the current process technology and includes the elimination of global stalls, re-design of most known critical paths, reduction in overall wiring delays through better floor planning and power efficiency improvements through elimination of dynamic circuits and improved clock gating.

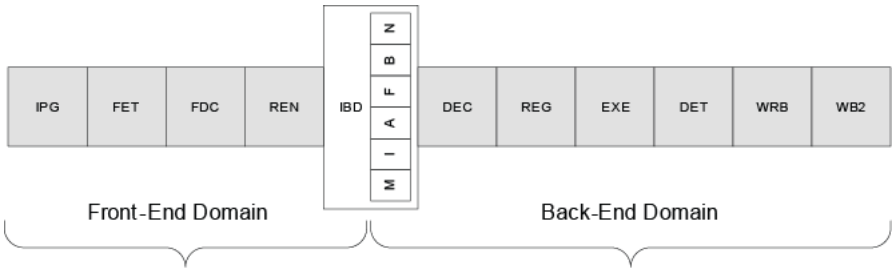
2.1 Overview

The Intel Itanium processor 9500 series core provides a 12-wide issue (4 bundles) 11 stage deep pipeline that can fetch up to six instructions per cycle and retire up to 12 (4 bundles) instructions per cycle. The following is a summary of the key features supported in the processor core:

- Use pipeline replays for pipeline control instead of global stall mechanisms.
- In-order 12 wide instruction (4 bundles) issue and retirement.
- A 96 entry instruction buffer per thread for decoupling the Front End (FE) and Back End (BE).
- Execution Units:
  - 6 integer ALU units
  - 4 multimedia units
  - 2 load/store units
  - 1 integer multiply unit
  - 2 floating point units capable of extended, double and single-precision arithmetic with hardware support for denormal, unnormal and pseudo-normal operands.
  - 3 branch units
  - support for nop squashing through a dedicated nop pipeline
- Independent FE and BE thread domains providing hardware support for 2 threads per core.
- Duplicated first-level and second-level Data TLBs for the 2 threads.
- 32 extra General Registers increasing the stacked physical registers to 128.
- Dedicated load return paths from the MLD to the Integer Register File.
- Intel® Virtualization Technology (Intel® VT) for Intel® 64 or Itanium® architecture (Intel® VT-i) 3 support - virtualization support extensions.
- New Data Prefetcher unit for improved software and hardware-initiated data prefetching.

- Improved control speculation support via spontaneous deferral, multiple outstanding page walks.
- Expanded memory operation hints for software to communicate cache locality, cache allocation, prefetch hints and deferral information to hardware.

Figure 2-1. Core Pipeline



IPG – Instruction *P*ointer *Generate*
FET – Instruction *Fet*ch
FDC – *F*ront-end *De*code
REN – Register Rename

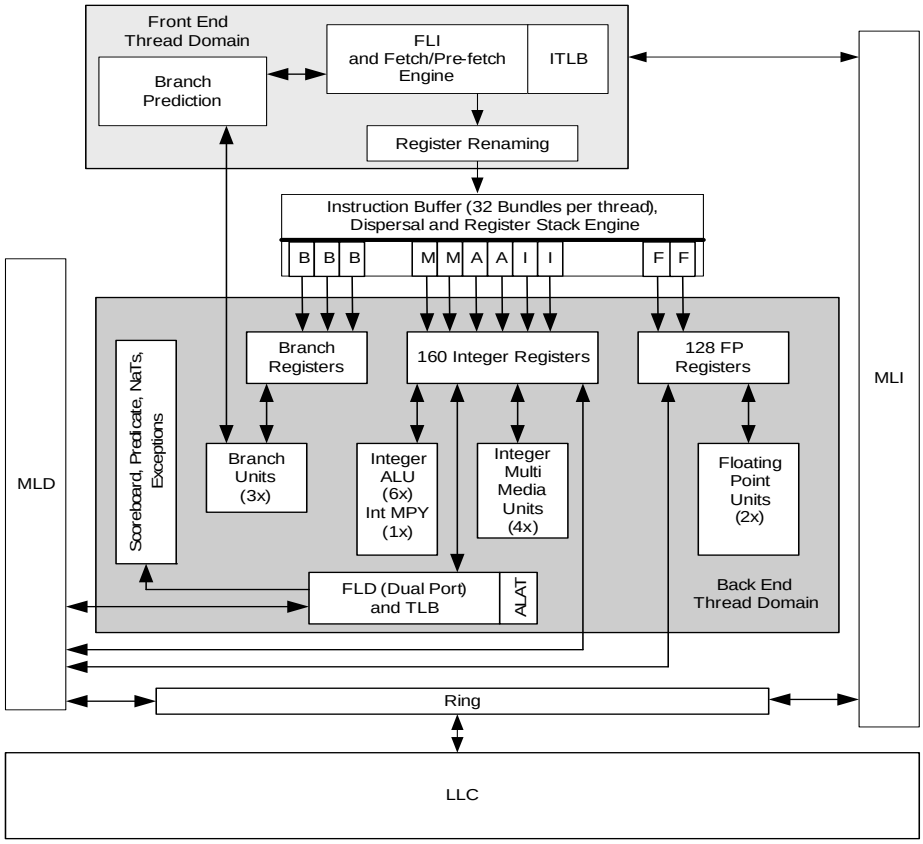
IBD – Instruction Buffer and Dispersal
DEC – Instruction *De*code
REG – *Reg*ister Access
EXE – Instruction Execute
DET – *Detect Exceptions*
WRB – *W*riteback Commit
WB2 – *Writeback-2 Retire*

The first stage initiates the instruction fetch (IPG) of two bundles. The second stage (FET) completes the FLI cache lookup and delivers the two bundles to the front end decoding (FDC) stage which then passes them on to the register renaming (REN) stage. The six instructions in the two bundles with their renamed register identifiers are inserted into an appropriate memory (M), ALU (A), integer (I), floating point (F), branch (B) and NOP (N) queues in the IBD stage. The first three stages operate on one of two stages independently of the last 6 stages. Up to 12 instructions can be issued from the instruction buffer queues in the IBD stage to 2 memory (M), 2 ALU (A), 2 integer (I), 2 FPU (F) and 3 branch (B) pipelines. NOP instructions do not use execution pipelines and are managed within the control portion of the core pipeline. The instructions issued by the instruction buffer are distributed and decoded in the DEC stage, operand bypass occurs in the REG stage and execution starts in the EXE stage. Exception information is coalesced in the DET stage, instruction commit vectors are created in the WRB stage. The WB2 stage folds in late arriving error detection information that may retry instruction execution.

Table 2-1. Core Pipelines

Pipeline	Pipe Stages														
Main	IPG	FET	FDC	REN	IBD	DEC	REG	EXE	DET	WRB	WB2				
MLD									L1A	L1T	L1H	L1M	L1D	L1C	L1W
FPU								FP1	FP2	FP3	FP4	FP5	FP6		

Figure 2-2. Core Block Diagram



- Mid-level Instruction cache (MLI) — 512 kB, 8 way, 64B line x2 sectored, 9 cycle latency, 1 read, 1 fill port, NRU replacement. States: Invalid, Valid, Killed
- Instruction paging cache (IPC) — 128 entries, 50 physical address bit support, RID and Key prevalidated
- First-level Instruction cache (FLI) — 16 kB, 4 way, 64B line, single cycle latency, 1 read, 1 fill port, LRU replacement. States: Invalid, Valid
- First-level TLB (FLITLB) — 32 entry, fully associative, NRU replacement
- Instruction buffer (IBQ) — 96 instructions per thread, 32 instruction MQ (memory), 32 instruction AQ (alu), 32 instruction IQ (integer), 32 instruction FQ (floating point) and 16 bundle BQ (bru)
- Execution ports: 2 M/A (memory/alu), 2 A (alu), 2 I/A (integer/alu), 2 F (fpu), 3 B (branch) ports and 1 N (nop)
- Integer registers (IRF) — 32 static, 16 banked, 128 stacked 64 bit registers plus 1 NaT bit per thread. 12 read ports, 10 write ports

- Integer execution unit (IEU) — 2 M/A, 2 A and 2 I/A units. Most I and A instructions are one cycle latency; some I and multi-media instructions are two cycle latency; the one 32x32 multiplier is 4 cycle latency.
- Floating point register file (FRF) — 32 static, 96 rotating 80 bit registers per thread, 6 read ports, 6 write ports
- Floating point execution unit (FPU) — 2 fused multiply-accumulate and 2 misc execution units. 6 cycle latency
- Branch resolution execution unit (BRU) — 3 branch execution units, 8 branch registers per thread. Generates a bruFlushWrb signal on a branch mispredict, 10 cycle penalty.
- Predicate delivery unit (PDU) — 16 static and 48 rotating one-bit registers per thread.
- ALAT (Advanced load address table) — 32 entry, 24-bit physical address support, fully associative per thread
- First level data cache (FLD) — 16 kB, 4 way, 64B line, LRU replacement, single cycle latency, 2 read, 2 write ports, write-through. TLB prevalidated
- First level data TLB (FLDTLB) — 32 entries per thread, fully associative, 4 kB, 8 kB and 16 kB variable page size,
- Second level data TLB (DTB) - 128 entries per thread

2.2 The Intel Itanium Processor 9500 Series

2.2.1 Implementation Specific Behavior

2.2.1.1 Virtual and Physical Addressing

The processor core supports the full 64 bit virtual address space with 224 regions, 50 bits of physical address and 24-bit protection keys. Since the processor implements the PSR.vm bit, the number of implemented virtual address bits is reduced by 1, that is, from 61 to 60 when PSR.vm=1. This allows for creating a protected memory address space under processor virtualization for the host or virtual machine manager.

2.2.1.2 Region Registers

The processor implements the full size of region register IDs (24b). Instead of looking up the entire virtual address and the region, the region ID of a TLB entry is prevalidated. A single bit is kept in the TLB (called RR match) which is set whenever there is an insert operation or whenever there is a mov to RR operation to the corresponding region register (selected by VA bits 63:61) with a RID that matches the TLB entry's RID. The bit is cleared when a mov to RR occurs to the region register with a different RID. This bit is used as an additional valid bit during a TLB lookup.

2.2.1.3 Protection Key Registers

The processor supports 16 protection key registers, with 24b of key. The keys are kept prevalidated in the TLB. A PKR index field is added to the TLB entries. Whenever there is a mov to PKR operation (to any PKR), if the key matches a TLB entry's key, then the entry is prevalidated with the incoming key valid bit and the index field is written with the index used in the mov to PKR operation. If the key does not match but the index of the op matches with the PKR index in a TLB entry, then the key valid bit is cleared.

2.2.1.4 RSE stacked physical register file size

The processor adds 32 integer registers to the RSE stacked physical register file size for a total of 128. These registers are referenced from the virtual register id space of r32–r127, which is a 96 entry window that moves across the 128 entry stacked physical register file. These additional registers improve performance by reducing the number of RSE injected fill and spill operations. Intel Itanium processor 9300 series has 96 stacked physical registers.

2.2.1.5 Reversibility of ttag and thash

The VHPT hash is implementation specific. The processors hash is defined as:

```
if hpn = va{60:0} >> RR[va{63:61}].ps
 tag = hpn ^ (RR[va{63:61}].rid << 39)
 hash = (PTA.base{63:15} & (1 << 64 – 1<<PTA.size))
 | (hpn^RR[va{63:61}].rid << 5) & (1 << PTA.size –1)
```

From the values of hash, tag, RR[va{63:61}].ps and PTA.size:

```
va{50:12+x} = tag{38-x:0} where RR.ps = (12 + x)
 rid{23:10} = tag{62:49}
 rid{9:0} = hash{14:5}^va{21+x:12+x}
 va{60:50} = tag{48–x:39–x}^rid{9–x:0}. if x > 9, then the rid term is ignored
```

2.2.1.6 Purge Behavior

The processor supports the following page sizes for purges or inserts: 4K, 8K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M, 1G, 4G. A purge operation which uses a page size that is not supported will take a reserved register field fault. The only exception is an incoming shootdown operation which can specify a page size that's not supported by the processor. The processor will then round the page size to the nearest supported page size above the incoming page size and it will use that to execute the purge.

2.2.1.7 ptc.e — Purge translation cache entries

This is an architected instruction of type M28 with some implementation dependent behavior. See Volume 3 of the "Intel IA-64 Architecture Software Developer's Manual" for the architected description and encoding. In the processor, one application of this instruction will purge all translation cache (TC) entries from both the instruction and data translation caches. This instruction purges TC entries from only the current thread; Intel Itanium processor 9300 series purged both thread's entries. The "parameter" given in r3 is not used.

2.2.1.8 fc.i — Flush Cache, Instruction Cache Coherence

The processor does not flush the last level cache (LLC) on a fc.i instruction. This is to assist with dynamic instruction translation performance by not moving the cache line to memory to make it visible to the I-side. All previous Itanium CPUs flushed all caches with this instruction.

2.2.1.9 NaT'd ld and st8.spill Behavior

The st8.spill will write the register's 64-bit data portion to memory. The processor returns a zero into the target register of all NaTed speculative loads, and also ensures that all NaT propagating instructions perform all computations as specified by the instruction pages.

2.2.1.10 hint @ pause / priority

Hint @pause signals to hardware that the executing thread in the BE pipeline is willing to give away execution resources and should be switched to the background. Hint @priority signals to hardware that the executing thread in the BE pipeline is a high priority thread and should generally not be switched away. Both of these hint instructions are supported on all M, I and F execution units. The processor does not support the hint.b or B-unit hint instructions i.e. hint.b will be ignored.

2.2.1.11 Unsupported Accesses and alignment

The following operations will generate unsupported reference faults:

- ld16/st16 to UC and WC space
- ldfe/stfe to UC and WC space
- Architecturally, semaphore operations are not allowed to UC and WC space.

The following are supported references subject to alignment requirements:

- ldf.fill, stf.spill are supported to UC and WC space

The following operations and their respective data alignments are supported when PSR.ac=0:

- All semaphores operations {xchg, cmpxchg, fetchadd} that are naturally aligned
- All 16 byte accesses that are naturally aligned
- All 10 byte accesses (ldfe/stfe) that do not cross a 16B boundary
- All FP 4, 8 byte accesses that do not cross a 16B boundary except tldfps which must be naturally (8B) aligned
- All Integer 2, 4 and 8 byte accesses that do not cross an 8B boundary

2.2.1.12 Floating Point Software Assist (FPSWA) Faults

The processor adds the following instructions to the FPSWA faulting list:

- SIMD FMAC ops: fpma, fpms, fpnma, fpcvt
- SIMD mins and maxes: fpmax, fpmin, fpamin, fpamax
- fprcpa, fprqsrta
- fpcmp

The Intel Itanium processor 9500 series and Intel Itanium processor 9300 series also take a FPSWA on the following instructions:

- frcpa
- frsqrta

The processor does not FPSWA on denormal, unnormal and pseudo-denormal operands. These operand types are automatically normalized by the hardware.

2.2.1.13 Processor Instruction Issue Port Mapping

As mentioned previously, the processor core BE has 12 execution pipelines. The instruction issue port map table below outlines which instructions can execute on which pipelines and their execution latency. The Instruction Buffer (IBL) unit which houses the instruction dispersal logic is responsible for ensuring this issue port map. The twelve execution pipelines in the BE are:

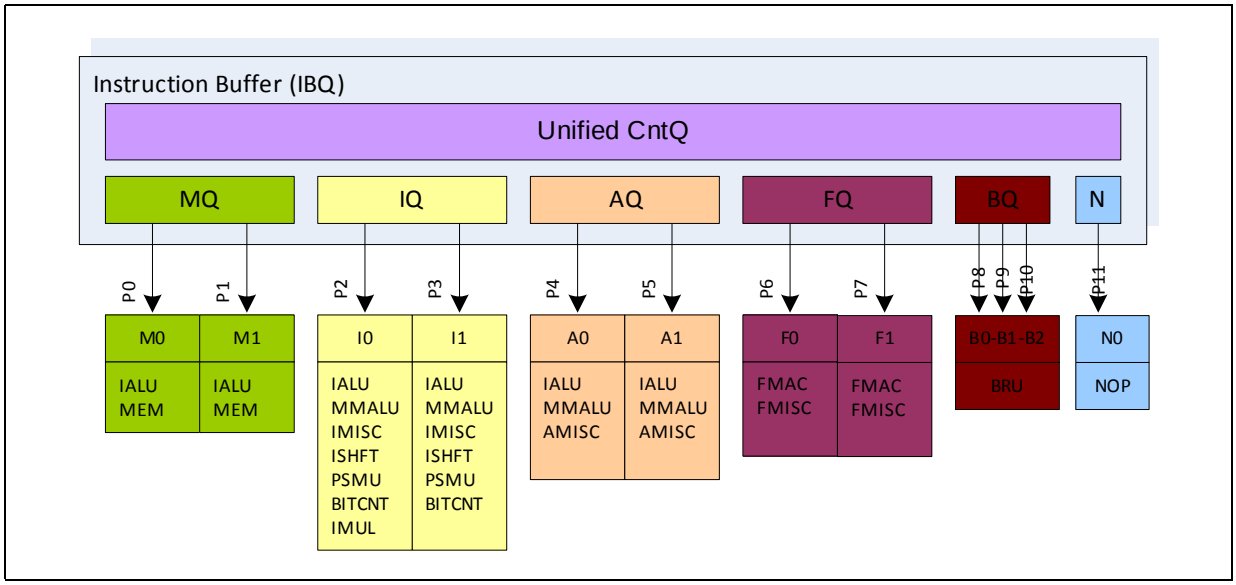
- 2 M-ports — M0 (or P0), M1 (or P1)
- 2 I-ports — I0 (or P2), I1 (or P3)
- 2 A-ports — A0 (or P4), A1 (or P5)
- 2 F-ports — F0 (or P6), F1 (or P7)
- 3 B-ports — B0 (or P8), B1 (or P9), B2 (or P10)
- 1 N-port — capable of executing 12 nops per cycle

The table below lists various functional units that are available on each of the twelve execution pipelines:

Table 2-2. Functional Unit Support on Ports

Port Type	Pipelines	Functional Capabilities
M-port	M0, M1	1-cycle ALU, Memory Address support
I-port	I0	1-cycle ALU, 2-cycle MM ALU, Integer Shifter, MM Shifter, Misc I-unit, Integer Multiplier
	I1	1-cycle ALU, 2-cycle MM ALU, Integer Shifter, MM Shifter, Misc I-unit
A-port	A0, A1	1-cycle ALU, 2-cycle MM ALU, Misc A-unit
F-port	F0, F1	6-cycle Fmac unit, 6-cycle Fmisc unit
B-port	B0, B1, B2	Branch resolution unit
N-port	N0	Handles up to 12 nop, brp instructions

Figure 2-3. BE Execution Pipelines



The tables below specify the instruction issue port map for A-type, I-type, M-type, F-type and B-type instructions.

Table 2-3. Issue Port Map for A-type Instructions

M		I		A		Format	Description	Examples
0	1	0	1	0	1			
1	1	1	1	1	1	A1	Integer ALU	add, sub, addp4, and, andcm, or, xor
1	1	1	1	1	1	A2	Shift Left and Add	shladd, shladdp4
1	1	1	1	1	1	A3	Integer ALU	sub, and, andcm, or, xor
1	1	1	1	1	1	A4	Add Imm	adds, addp4
1	1	1	1	1	1	A5	Add Imm	addl
1	1	1	1	1	1	A6-A8	Integer compare	cmp, cmp4
		2	2	2	2	A9	MM ALU	padd, psub, pavg, pavgsub, pcmp
		2	2	2	2	A10	MM shift and Add	pshladd, pshradd

Legend:
1 — 1-cycle latency, issues to that port
2 — 2-cycle latency, issues to that port

Note: A9, A10 issue width is different from Intel Itanium processor 9300 series. These could issue to all 6 ports on Intel Itanium processor 9300 series. Additional decode will be required in the processor to steer them away from M-ports.

Table 2-4. Issue Port Map for I/X-type Instructions (Sheet 1 of 2)

M		I		A		N	Format	Description	Examples
0	1	0	1	0	1				
		4					I1	MM Mult & Shift	pmpyshr
		4					I2	MM Mult	pmpy, mpy4, mpyshl4
		2	2					MM Mix/Pack/Unpack	mix, pack, unpack, pmin, pmax
		2	2						psad
		2	2				I3–4	MM Mux	mux1, mux2
		2	2				I5	Shift Right Variable	pshr, shr
		2	2				I6	MM Shift Right Fixed	pshr
		2	2				I7	Shift Left Variable	pshl, shl
		2	2				I8	MM Shift Left Fixed	pshl
		2	2				I9	Bit Strings	popcnt, clz
		1	1				I10	Shift Right Pair	shrp
		1	1				I11	Extract	extr
		1	1				I12–15	Deposit	dep, dep.z
		1	1				I16–17	Test bit/ Test NaT	tbit, tnat
						N	I18	Nop	nop.i
		1	1					Hint	hint.i
		1	1				I19	Break	break.i
		1	1	1_a	1_a		I20	Integer Spec Check	chk.s.i
		1	1				I21	Move BR	mov to BR
		2					I22	Move BR	mov from BR
		1	1				I23–24	Move PR	mov to PR
		2					I25	Move PR	mov from PR/IP
		1	1				I26–27	Move AR	mov to AR

Table 2-4. Issue Port Map for I/X-type Instructions (Sheet 2 of 2)

M		I		A		N	Format	Description	Examples
0	1	0	1	0	1				
		2					I28	Move AR	mov from AR
		1	1				I29	Sign/Zero Extend	sxt, zxt
		2	2					Compute Zero Index	czx
		1	1				I30	Test Feature	tf
		1	1				X1	Break	break.x
		1	1				X2	Move Long Imm	movl
		1	1				X5	Nop	nop.x
		1	1					Hint	hint.x

Notes:

1 — 1-cycle latency, issues to that port
2 — 2-cycle latency, issues to that port
4 — 4-cycle latency, issues to that port
1_a — 1-cycle latency, instructions can issue to A-port if I-ports are already subscribed
N — squashed nop

Multiply instructions are changing to 4-cycle latency (over 2 on Intel Itanium processor 9300 series).

The mov to br/pr/ar instructions may issue on either I0 or I1 but the same resource cannot be targeted in the same cycle. For example, a mov to br could be paired with a mov to pr but not with another mov to br. Note that it is possible to perform a mov to AR.EC and a mov to AR.LC in the same cycle, that is, each AR is considered to be a different resource.

Table 2-5. Issue Port Map for M-type Instructions (Sheet 1 of 2)

M		I		A		N	Format	Description	Examples
0	1	0	1	0	1				
X	X						M1–3	Integer Load	ld, ld.a, ld.c, ld.s, ld.sa, ld16
X	X						M4–5	Integer Store	st, st.rel, st.spill, st16
X	X						M6–8	FP Load	ldf, ldf.a, ldf.sa, ldf.c
X	X						M9–10	FP Store	stf, stf.spill
X	X						M11–12	FP Load Pair	ldfp, ldfp.a, ldfp.sa, ldfp.c
X	X			X_a	X_a		M13–15	Line Prefetch	lfetch
X	X								lfetch.fault, lfetch.excl
X	X			X_a	X_a		M51		lfetch
X	X						M52	Counted Line Prefetch	lfetch.count
X	X						M16–17	Semaphore	cmpxchg, xchg, fetchadd
X	X						M18	Set FR	setf
X	X						M19	Get FR	getf
X	X			X_a	X_a		M20	Int Spec Check	chk.s.m
X	X						M21	FP Spec Check	chk.s.m
X	X						M22–23	Adv Load Check	chk.a
X	X						M24	Inval ALAT	invala
X								Sync/Serialize/Fence	sync.i, srlz.d, srlz.i, mf, fwb

Table 2-5. Issue Port Map for M-type Instructions (Sheet 2 of 2)

M		I		A		N	Format	Description	Examples
0	1	0	1	0	1				
X							M25	RSE Control	flushrs, loadrs
X	X						M26–27	ALAT Entry Inval	invala.e
X	X						M28	Flush Cache	fc, fc.i
X							M29–33	Mov AR/CR	mov ar, mov cr
X							M34	Allocate Stack Frame	alloc
X							M35–36	Mov PSR	mov psr
X	X						M37	Break	break
X							M38–40	Probe	probe.r, probe.w
X							M41	TC insert	itc.d, itc.i
X							M42	TR insert	itr.d, itr.i
X							M43	Mov from Indirect	mov rr/dbr/ibr/pkr/pmc/pmd
X							M44	System Mask	sum, rum, ssm, rsm
X							M45	TC, TR purge	ptc.l, ptc.g, ptc.ga, ptr.d, ptr.i
X							M46	Translation access	thash, ttag, tpa, tak
X							M47	Purge Entry	ptc.e
						N	M48	Nop	nop.m
X	X						M49	Hint	hint.m
X	X						M50	Move to DAHR	mov dahr

Notes:

X — issues to port
X_a — Instructions can issue to A-port if M-ports are already subscribed
N — squashed nop

Table 2-6. Issue Port Map for F-type instructions

F		N	Format	Description	Examples
0	1				
6	6		F1	FP Multiply Add	fma, fnma, fms, fnms
6	6		F2	Fixed Multiply Add	xma
6	6		F3	FP Select	fselect
6	6		F4	FP Compare	fcmp
6	6		F5	FP Class	fclass
6	6		F6	FP Reciprocal Approx	frcpa
6	6		F7	FP Square Root	frsqrta
6	6		F8	FP Min/Max	fmin, famin, fmax, famax
6	6		F9	FP Merge/Logical	fmerge, fmix, fsxt, fand, fandcm, for, fxor
6	6		F10–11	FP Conversion	fcvt
6	6		F12–14	FP Status Field	fsetc, fclrf, fchkf
6	6		F15	Break	break.f
		N	F16	Nop	nop.f
6	6			Hint	hint.f

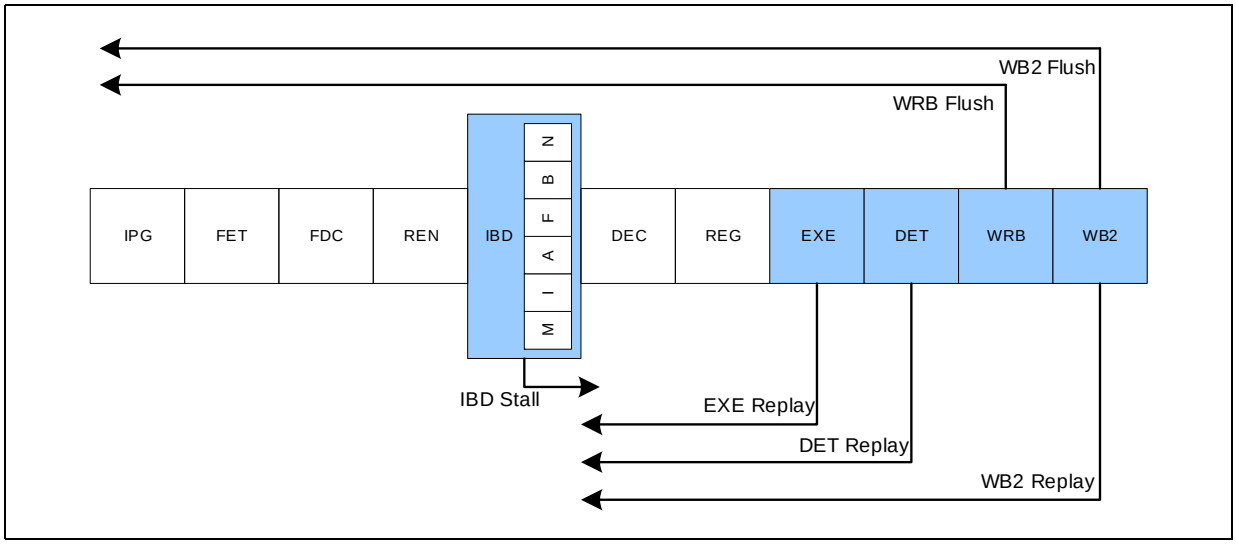
Table 2-7. Issue Port Map for B-type Instructions

B			N	Format	Description	Examples
0	1	2				
X	X	X		B1	IP-rel Branch	br.cond
		X				br.wexit, br.wtop
		X		B2	IP-rel Counted Branch	br.cloop, br.cexit, br.ctop
X	X	X		B3	IP-rel Call	br.call
X	X	X		B4–5	Indirect Branch	br.cond, br.call, br.ia, br.ret
			N	B6–7	Branch Predict	brp, brp.ret
		X		B8	Misc	cover, clrrrb, rfi, bsw, epc, vmsw
X	X	X		B9	Break	break.b
			N		Nop	nop.b
X	X	X			Hint	hint.b

2.2.2 Processor Core Pipeline

As discussed above, the processor core pipeline consists of a Front-end or FE pipeline and a Back-end or BE pipeline. The FE and the BE each have their own pipeline control mechanisms consisting of replays, flushes and stalls to account for various pipeline hazards. For a discussion of the FE pipeline control mechanisms, please refer to the IFR chapter in the Instruction Fetch section. The figure below shows the primary BE pipeline control mechanisms. Generally, the BE pipeline has been converted from a global stall based microarchitecture on previous Itanium processors to a replay based micro-architecture on the Intel Itanium processor 9500 series.

Figure 2-4. Back-End Pipeline Control Mechanisms



Appendix , "Data Access Hints"

2.2.3 New Instruction Support

The processor is fully compliant with latest revisions of the Intel Itanium Architecture SDM and adds support for the following new instructions:

Table 2-8. New Instruction Support

New Instruction	Description
clz	Counts the number of leading zeros in a 64-bit GR value
hint @priority	Indicates to processor that current thread is performing a high-priority task
lfetch.count	Counted line prefetch
mov dahr	Move Data Access Hint Register (refer to Chapter 2.3.1 for further details)
mov r_1 = dahr[r_3]	Move Indirect (from dahr) register (refer to Chapter 2.3.1 for further details)
mpy4	Unsigned 32x32 integer multiply
mpyshl4	Unsigned 32x32 integer shift left and multiply
tf	Test feature presence from features vector in CPUID[4]

Although tf is not a new instruction in the Intel Itanium architecture, the processor is the first implementation that can return a true predicate. All previous implementations have always returned a false predicate for all features.

2.2.4 Advanced Load Address Table (ALAT)

The ALAT is a 32 entry fully associative table that attempts to hold for the window of data speculation the set of advanced (data speculative) loads that have executed but whose target locations have not been modified by a store. The ALAT is mostly unchanged in the processor except for the changes listed in Table 2-9.

Table 2-9. ALAT Changes

	Intel Itanium Processor 9500 Series	Previous Intel Itanium Processors
number of ports	2	4
store precedes ld.c or chk.a by 2 or more cycles	24-bit comparison	20-bit comparison
store precedes ld.c or chk.a by 1 or cycle1	24-bit comparison	12-bit comparison
ld.c fail penalty	7 cycles (WB2 replay)	forced FLD miss
chk.a fail penalty	19 cycles	14 cycles
banked register aliasing	no	yes

2.2.5 Data Translation Lookaside Buffers (TLBs)

The processor has two data TLBs: the FLDTLB and DTB. The FLDTLB isn't really a complete TLB because it is missing some information associate with translations. It's entire purpose is to enable the FLD to have single cycle latency by pre-validating the FLD with respect to the FLDTLB. "Pre-validation" means that each line in the FLD is associated with an FLDTLB entry and that and access must hit on the associated entry in the FLDTLB to hit on line in the FLD. Likewise, in order to trigger an FLD fill, an access must hit in the FLDTLB. Independent of the outcome of the FLDTLB lookup, all data accesses access the DTB. Table 2-10 lists some important characteristics of these TLBs.

Table 2-10. Data TLB Characteristics

	FLDTLB	DTB
size	32 entries per thread	128 entries per thread, entries 64-127 can only be TCs
latency	1 cycle	5 cycle minimum non-blocking minimum DTB->FLDTLB transfer latency, 2 cycle lookup latency (part of MLD latency)
page sizes	4 KB, 8 KB, 16 KB	4KB-4GB
associativity	fully associative	fully associative
number of ports	2 read, 1 write	2 read, 1 write
replacment algorithm	perfect LRU	4 quadrant NRU
max # primary misses outstanding	1 DTB->FLDTLB transfer (non-blocking), otherwise unlimited by FLDTLB	1 blocking miss per thread; # of non-blocking misses unlimited by DTB
max # secondary misses outstanding	unlimited by FLDTLB	# of non-blocking secondary misses is unlimited by DTB

Table 2-11. Data TLB Differences from Previous Intel Itanium Processors

	FLDTLB	DTB
size	twice the previous size; statically split between threads vs. dynamically split previously	twice the previous size; statically split between threads vs. dynamically split previously
latency	unchanged	DTB->FLDTLB minimum latency is up 1 but has gone from blocking to non-blocking
page sizes	unchanged	unchanged
associativity	unchanged	unchanged
number of ports	down from 4	down from 4
replacment algorithm	improved from NRU	improved
max # primary misses outstanding	improved from 1 blocking miss	improved from 1 blocking miss
max # secondary misses outstanding	improved from 0	improved from 0

2.2.5.1 FLDTLB

The FLD Translation Lookaside Buffer (FLDTLB) is a fully associative TLB with 32 entries per thread that exists to speed access to the FLD. As discussed in the earlier FLD section of this document, every FLD entry is associated with an FLDTLB entry. All FLD accesses require an FLDTLB translation. The FLDTLB is not a complete TLB in the architectural sense because it does not contain all of the required architected values. Therefore, when the FLDTLB is used to do a lookup in the FLD, the translation must also be looked up in the DTB, a full architectural TLB. If the translation does not exist in the DTB, the results of the FLD lookup cannot be used.

2.2.5.1.1 Non-blocking FLDTLB Misses

On previous processors, FLDTLB misses were blocking. On the processor they are non-blocking. Operations that miss the FLDTLB and hit in the DTB are immediately sent to the MLD but are not able to fill the FLD. Typically, if such an operation is of the type that

would have filled the FLD (FLDCANFILL) and there is no DTB->FLDTLB transfer in progress, a new DTB->FLDTLB transfer will be initiated for the corresponding translation. The DTB->FLDTLB transfer process occurs in the background and in the common case (no PA purge) attempts not to interrupt the progress of the main pipeline. A cycle where nothing is inserted in the M-pipes is required to complete the transfer, but the transfer waits up to 16 cycles before requesting M0 and M1 in an attempt to use a cycle in which they happen to be unused. The theory is that there isn't typically much benefit in interrupting the main pipe if it is making progress. The reason the transfer waits only 16 cycles is that a pending transfer can block other things (e.g. the DPFQ) that shouldn't be held off indefinitely. Thread switches cause DTB->FLDTLB transfers to be aborted.

2.2.5.1.2 FLDTLB inserts

FLDTLB inserts occur only as DTB->FLDTLB transfers. FLDTLB inserts do not occur as part of DTB inserts. DTB->FLDTLB transfers are initiated when all the following are true:

- FLDCANFILL operation reaches DET stage
- operation is a virtual mode operation
- data access hint says allocate in FLD
- operation misses in FLDTLB
- operation hits in DTB
- DTB translation attributes: page present and memory attribute of WB
- there is no other outstanding DTB->FLDTLB transfer in progress

An outstanding DTB->FLDTLB transfer is cancelled when any of the following events occur:

- anything causes an entry to be purged or replaced in the DTB
- a back-end thread switch occurs

2.2.5.1.3 FLDTLB Replacement

The replacement algorithm is per-thread perfect LRU.

2.2.5.1.4 Virtual Aliasing

As with previous Intel Itanium processors, the FLD cannot handle an FLDWR that hits two virtually aliased (same PA different VA) cache lines because doing so would require the ability for a single store to simultaneously write to two different locations in the cache. To prevent this from happening, before inserting a new translation into the FLDTLB, any translations that are virtually aliased with the new translation are purged. A new translation (for example, 16K pages) could overlap with as many as 4 existing translations (for example, 4K pages) in the FLDTLB.

Due to the lack of physical tags, the processor FLD has a further restriction in that it doesn't allow locations in the FLD to be virtually aliased with any FLDWR. To prevent this from happening, stores that miss the FLDTLB purge any virtually aliased translations in the FLDTLB. An FLDWR could overlap with at most one existing translation in the FLDTLB. Any FLDLD that hits in the FLD after an FLDWR that needs to purge the FLDTLB but before the purge is completed will be WB2 replayed. If the FLDTLB missing FLDWR and FLD hitting FLDLD occur in the same cycle, a WB2 replay is triggered if comparing VA[11:0] and sizes suggest these two operations may overlap. (PMU event: CYC_BE_WB2_REPLAY.STORE_ALIAS)

2.2.5.2 DTB

The Data TLB (DTB) is a fully associative 128-entry per thread TLB. Entries 0 to 63 can be used as TR entries. Any entry not being used as a TR entry can be used as a TC entry. DTB misses can trigger the Hardware Page Walker (HPW). On previous Intel Itanium processors, a DTB miss that triggered a hardware page walk was always blocking. In other words, it stalled the main pipeline. On the processor, DTB misses that trigger hardware page walks can be either blocking or non-blocking. A DTB miss that triggers a hardware page walk will be non-blocking if all of the following conditions hold true and blocking otherwise:

- operation is a control speculative load, lfetch, or hardware data prefetch
- spontaneous deferral is allowed
- data access is hinted PIPE_DEFER

2.2.5.2.1 Blocking DTB Miss

(PMU events: CYC_BE_WB2_REPLAY.BLK_HPW, CYC_BE_IBD_STALL.HPW)

2.2.5.2.2 Non-blocking DTB Miss

When an lfetch triggers a non-blocking walk, it is placed in the Data Prefetch Queue (DPFQ - see data prefetching section) and then re-issued, if possible, when the walk is completed. When a ld.s triggers a non-blocking walk, it is deferred (NATed). In this case, NATed register value is available in the bypass network in WB2. The more common scenario where the integer speculative load missed the FLD, and unusual scenario where the integer speculative load hit in the FLD (and missed in the DTB).

(PMU events: CYC_BE_EXE_REPLAY_GR_LOAD_RAW, CYC_BE_WB2_REPLAY.NAT_HZRD)

2.2.5.2.3 Pre-validation With Respect to Region Register

The DTB is pre-validated with respect to the region registers. This means that each DTB entry is associated with a particular region register and the Region Identifier (RID) in that region register. If a new RID is written into a region register, then all of the entries associated with that RID and region register become invalid. However, if such an entry is still in the DTB when the original RID is written back to the same region register, then the entry becomes valid again.

2.2.5.2.4 Replacement Algorithm

The replacement algorithm for the DTB has been improved relative to previous Intel Itanium processors. The 128 DTB locations for each thread are split up into four 32 entry quadrants. Quadrant n includes location n, n +4, n+8, etc. to n+124. Each quadrant consists of half TR/TC entries and half TC only entries. Inserts rotate through the quadrants such that following an insert to quadrant 0, the next insert will be to quadrant 1, and so forth. Entries will be chosen for replacement in the following priority order:

- Use the Dtb entry location specified by the itr.d or ldat commands.
- Choose the first invalid entry in the following order: 64-127, 0-63.
- The first non TR location in the current quadrant without a currently valid RID entry
- The first non TR location in the current quad with the Recently Used (RU) bit equal to zero.

The RU bit for each entry starts off at 0, and whenever a location is inserted to or is used by a translation, the corresponding RU bit is set to 1. If all the non-TR RU bits for the quadrant are 1's, the RU bits are all forced to 0's.

2.2.5.3 Hardware Page Walker (HPW)

The Hardware Page Walker (HPW) can be used on a TLB miss (DTB on the data side) to look up a translation in the Virtual Hashed Page Table (VHPT). Previous processors could only do a single hardware page walk at a time, and data side hardware page walk always blocked the main pipeline. Both of these constraints have been removed on the processor. See the previous DTB section for a discussion of servicing DTB misses in a non-blocking manner.

2.2.5.3.1 Concurrent Walks

Any two walks from the following list can be executed concurrently:

- 1 i-walk (from either thread)
- 1 blocking d-walk from thread 0
- 1 blocking d-walk from thread 1
- 1 non-blocking d-walk from either thread

A Non-blocking d-walk is dropped whenever another d-walk is requested before it begins execution. New d-walk requests are coalesced with outstanding non-blocking d-walks to the same V[63:12] and thread.

2.2.5.3.2 Thread Switching

Hardware page walks are long latency events and thus can affect and interact with thread switching in the following ways:

- Blocking d-walk requests hold off thread switching only until VHPT load enters MLD OZQ.
- A switch away event occurs when a blocking VHPT load exceeds a certain latency.
- A switch back event occurs when a blocking VHPT load completes
- Off thread VHPT loads can occur without a thread switch.

2.2.6 Architectural Ordering

2.2.6.1 Acquire Semantics

On the processor, acquire semantics are handled cooperatively by the FLD and MLD. The FLD prevents servicing FLD and ALAT check hits whenever there is an older operation with acquire semantics that hasn't reached global visibility by replaying FLD hits, ld.c's, and chk.a's (PMU events: CYC_BE_WB2_REPLAY.LOAD_ACQ, CYC_BE_DET_REPLAY.LOAD_ACQ, CYC_BE_IBD_STALL.ACQ).

The MLD doesn't allow operations from same thread as an older operation with acquire semantics to issue from the OZQ until the older acquire operation has issued and has hit in the MLD or is no longer in the MLDFAB or SMQ. An exception to this rule is that lfetches that haven't requested an FLD fill are allowed to forgo the MLD acquire semantics restrictions. In an effort to improve performance in the presence of operations with acquire semantics, hardware will generate prefetches (via the DPFQ) for all loads sent to the MLD while an acquire operation is outstanding. These

prefetches do not request FLD fills, and so, if they can make it into the MLD OZQ before it fills up due to a long latency acquire operation, they can issue regardless of the state of the acquire operation.

2.2.6.2 Release Semantics

An operation with release semantics will not issue from OZQ until it is oldest in OZQ (from both threads) and all release dependencies from already issued operations have been met. This does imply that a long latency release operation from one thread can hold up a release operation from the other thread. In an effort to improve performance in the presence of stores with release semantics, hardware will generate a prefetch (via the DPFQ) for every st.rel that goes down the pipeline. Such a prefetch is not affected by release semantics, and so, if it can make it into the MLD OZQ while a long latency release operation is waiting to be issued, it can issue and potentially improve the performance of the release operation.

2.2.6.3 Memory Fences

Memory fence semantics is a combination of acquire and release semantics. Therefore, operations with memory fence semantics are handled by combining the acquire and release behaviors already discussed. The following restrictions are also implemented for memory fences:

- Operations with memory fence semantics do not allow lfetches to ignore their acquire behaviors. This implementation restriction is architecturally required by mf.a but is not required for mf.
- Operations with memory fence semantics wait for all outstanding memory accesses, including lfetches and operations from the other thread, to complete before they become globally visible. This implementation restriction (called a "super release") is not architecturally required.
- The semantic restrictions for mf.a are applied across both threads. This implementation restriction is not architecturally required.

2.2.7 Execution Latencies

The following tables show the processor core instruction execution latencies. Separate tables are presented for GRs, FRs, BRs, PRs, ARs and CRs.

Table 2-12. Integer Execution Latencies (Sheet 1 of 2)

Consumer (across) Producer (down)	IALU	ICMP	ISHFT	MMALU	PSMU	BITCNT	PSAD	IMULT	ICHK	IMOVTO	MADDR	STDATA
IALU	1	1	1	1	1	1	1	1	1	1	1	1
ISHFT	1	1	1	1	1	1	1	1	1	1	2	1
MMALU	2	2	2	2	2	2	2	2	2	2	3	2
PSMU	2	2	2	2	2	2	2	2	2	2	3	2
BITCNT	2	2	2	2	2	2	2	2	2	2	3	2
PSAD	2	2	2	2	2	2	2	2	2	2	3	2
IMULT	4	4	4	4	4	4	4	4	4	4	5	4
IMOVFM	2	2	2	2	2	2	2	2	2	2	3	2
FLDHIT	1	1	1	2	2	2	2	2	1	1	2	1

Table 2-12. Integer Execution Latencies (Sheet 2 of 2)

Consumer (across) Producer (down)	IALU	ICMP	ISHFT	MMALU	PSMU	BITCNT	PSAD	IMULT	ICHK	IMOVTO	MADDR	STDATA
FLDLDFILLHIT	2	2	2	2	2	2	2	2	2	2	2	2
MLDRTN	M	M	M	M	M	M	M	M	M	M	M+1	M
DCSRTN	C	C	C	C+1	C+1	C+1	C+1	C+1	C	C	C+1	C

Table 2-13. Floating-Point Execution Latencies

Consumer (across) Producer (down)	FMAC	FMISC	FPST	GETF
FMAC	6	6	6	6
FMISC	6	6	6	6
SETF	9+	9+	9+	9+
MLDRTN	M+1	M+1	M+1	M+1

Table 2-14. Predicate Execution Latencies

Consumer (across) Producer (down)	NONBRQP	BRQP	MOVFMPR
INTPREDWR	1	0	1
FPPREDWR	3	2	3
MODSCHBR	1	1	1
MOVTOPR	2	0	1

Table 2-15. Branch Execution Latencies

Consumer (across) Producer (down)	INDBR	MOVFMBR
BRCALL	1	1
MOVTOBR	0	1

Table 2-16. Instruction Type Details (Sheet 1 of 2)

Inst Type	Instructions
IALU	A1-5, M2-3, M5, M7-8, M10, M12, M14-15, M31(unat,rnat), I29(zxt)
ICMP	A6-8, I16-17
MMALU	A9-10,I2(pmin,pmax),M13-15(A-ports),I26(pfs)
ISHFT	I10-15,I29(sxt)
PSMU	I3-8,I2(mix,pack,unpack)
BITCNT	I9,I29(czx)
PSAD	I2(psad)
IMULT	I1,I2(mpy,pmpy)
ICHK	I20,M20
IMOVTO	I21,I23,I26
IMOVFM	I22,I25,I28

Table 2-16. Instruction Type Details (Sheet 2 of 2)

Inst Type	Instructions
MADDR	M1-12, M13-15(M-ports), M16-17, M28, M38-40, M42-43, M45-47, M51-52
STDATA	M4-5, M16, M18, M29, M32, M35, M38, M41-42, M45, M1003
FLDHIT	M1-3
FLDLDFILLHIT	M1-3(ld.fill)
MLDRTN (GR)	M1-3,M16-17,M19,M31(csd,ccv);
MLDRTN (FR)	M6-8, M11-12
DCSRTN	M31(excluding csd, ccv, unat, rnat), M33-34, M36, M38-39, M43, M46, M1002
FMAC	F1-2, F10-11
FMISC	F3-9
FPST	M9-10
GETF	M19
SETF	M18
INTPREDWR	A6-8, I16-17, I30
FPPREDWR	F4-7
MODSCHBR	B1-2
MOVTOPR	I23-24
MOVFMPR	I25
BRCALL	B3, B5, X4
MOVTOBR	I21
INDBR	B4, B5
MOVFMBR	I22

2.3 Data Access Hints, Fetch, Dispersal and Execution

2.3.1 Data Access Hints

2.3.1.1 Hint Architecture Overview

Up until now, loads and stores have had only a few bits in the instruction encoding to hint some attributes such as temporal locality and expected MESI state behavior. There are many other useful attributes associated with memory operations that could potentially be communicated by software in order to optimize the handling of those operations by the hardware. Towards this end the following extensions to the Instruction Set Architecture (ISA) have been made:

- All loads, stores, lfetches, and semaphore instructions specify a dynamic DAHR instead of a static hint, using the two encoding bits formerly used for the temporalness hints. Each of the 8 DAHRs can specify a unique set of memory attributes, including such things as speculation, prefetching and temporal hints. An additional “h” bit is defined in the encoding of lfetch instructions and all loads and stores except the FP load pair, register update, and immediate update forms, allowing those instructions to reference any of the 8 DAHRs. Semaphore instructions also do not have the "h" bit. The instructions without the "h" bit can only index DAHRs 0-3. Hardware sets the default DAHR values such that they mimic legacy hint bit behavior.

- A new instruction, mov-to-DAHR, allows software to specify a set of attribute values to use for subsequent memory operations (in the current procedure) which are tagged with that particular DAHR. This new instruction uses a formerly-unused encoding within the hint opcode space, so it is ignored by legacy processors. The mov-to-DAHR instruction has single cycle latency on the processor.
- Another new instruction, mov-from-DAHR, allows software to read the current contents of a DAHR. In general, however, software is expected to remember the values it wrote into the DAHRs, and to simply (re)write a DAHR before an important code sequence (rather than first reading the DAHR to see if it needs to be changed). Thus, the mov-from-DAHR operation is not optimized on the processor, and may result in many replay cycles.
- A Data Access Hint Stack (DAHS) preserves modified DAHR values across procedure calls and returns. Any br.call instruction or interruption will allocate a new stack frame, and set all of the DAHRs in that new frame to default values. Any br.ret or rfi instruction will throw away the current stack frame, and return to the previous frame. The processor implements 7 entries in its DAHS. If the stack overflows (as a result of nesting more than 7 procedure calls), the oldest stack frame is lost, and the DAHRs will revert to their default values when control eventually returns to its associated procedure. The DAHRs are also reinitialized to default values, and the entire DAHS is cleared, on a context switch (indicated by a mov-to-BSPSTORE instruction).

In the processor implementation of this hint architecture, the following will cause 7 cycle WB2 replays:

- a mov-to-DAHR and a DAHS push followed by a DAHS pop in a 6 cycle window
- multiple DAHS pushes followed by a DAHS pop in a 6 cycle window
- a mov-to-BSPSTORE instruction followed by a mov-to-DAHR or a DAHS pop in a 6 cycle window

2.3.1.2 Hint Definitions

The architecture supports up to 16 hint bits per DAHR. The processor implements 7 independent hint fields, using 11 of the available bits. The remaining hint bits are reserved: they should always be set to zero in mov-to-DAHR instructions to allow for additional capabilities in future products. It should be noted that while hardware strives to follow these hints as much as possible, there may be cases where the hints are ignored. The definitions of the implemented fields and their corresponding values are the following:

- FLD_LOC (bits [1:0])
  These hint bits affect the allocation of a line into the first-level data (FLD) cache.
  - FLD_NORMAL (value = 0)
    Meaning: This hint is really just the absence of the two other FLD_LOCALITY hints.
    HW behavior: FLD missing integer loads and data prefetches with this hint typically trigger FLD fills, and such fills are initially marked recently used (RU).
  - FLD_NRU (value = 1)
    Meaning: This hint tells hardware that the hinted access is accessing a line that is less likely than other cache lines to be re-used during its lifetime in the FLD.
    HW behavior: Whether this line is already in the FLD or being filled into the FLD because of the hinted access, the line is marked not recently used (NRU) instead of recently used (RU).

PMU Events: FLD_LINE_DEMOTE counts the demotion of a cache lines to not-recently-used due to this hint. FLD_FILL_NRU counts the allocation to not-recently-used due to this hint.

— FLD_NO_ALLOCATE (value = 2)
Meaning: This hint tells the hardware that the hinted access should not trigger an FLD fill.
HW behavior: Accesses with this hint do not trigger FLD fills.
PMU Events: FLD_HINT_NOALLOC counts the number of times this hint prevents an FLD fill request.

- MLD_LOC (bits [3:2])
These hint bits affect the allocation of a line into the mid-level data (MLD) cache.

— MLD_NORMAL (value = 0)
Meaning: This hint is really just the absence of the two other MLD_LOCALITY hints.
HW behavior: MLD misses with this hint typically trigger MLD fills, and such fills are initially marked recently used (RU).

— MLD_NRU (value = 1)
Meaning: This hint tells hardware that cache lines filled to the MLD as a result of the hinted access are less likely to be re-used during their lifetimes in the MLD than other cache lines filled to the MLD.
HW behavior: A cache line filled to the MLD by such an access is initially marked not recently used (NRU) instead of recently used (RU). Subsequent accesses to that line update it to the RU state.
PMU Events: MLD_HINT_NRU counts the allocation to not-recently-used due to this hint.

— MLD_NO_ALLOCATE (value = 2)
Meaning: This hint tells hardware that the hinted access should not trigger an MLD fill.
HW behavior: Accesses with this hint do not trigger MLD fills.
PMU Events: MLD_HINT_NOALLOC counts the number of times this hint prevents an MLD fill.

- LLC_LOC (bit [4])
These hint bits affect the allocation of a line into the last-level cache (LLC).

— LLC_NORMAL (value = 0)
Meaning: This hint is really just the absence of the LLC_LOCALITY.ALLOC_NRU hint.
HW behavior: LLC misses with this hint typically trigger LLC fills, and such fills are initially marked recently used (RU). Subsequent accesses to that line leave the line in the RU state.

— LLC_NRU (value = 1)
Meaning: This hint tells hardware that cache lines filled to the LLC as a result of the hinted access are less likely to be re-used during their lifetimes in the LLC than other cache lines filled to the LLC.
HW behavior: A cache line filled to the LLC by such an access is marked not recently used (RU) instead of recently used (RU).
PMU Events: RIL_REQ_HINT_NRU counts requests which allocate to not-recently-used due to this hint.

- PF (bits [6:5])

These hint bits affect which hardware data prefetchers can be triggered by an operation. Please see Section 2.3.4.1.3 for a description of all the types of the processor's hardware data prefetchers.

— PF_NORMAL(value = 0)
Meaning: This hint is really the absence of software hinted hardware prefetching restrictions.
HW behavior: All hardware prefetchers *may* use this operation as a trigger for more prefetches. In practice, various prefetchers ignore various operations and only trigger prefetches subject to their algorithms.

— PF_NO_FLD (value = 1)
Meaning: Any hardware prefetch mechanisms that prefetch into the FLD and can generate multiple cache lines of prefetches from a single access should ignore this access.
HW behavior: The FLD sequential (neighbor line) prefetcher will ignore accesses with this hint.

— PF_NO_MLD (value = 2)
Meaning: Any hardware prefetch mechanisms that prefetch into the FLD or MLD and can generate multiple cache lines of prefetches from a single access should ignore this access.
HW behavior: The FLD sequential (neighbor line) prefetcher and the MLD sequential prefetcher will ignore accesses with this hint.

— PF_NONE (value = 3)
Meaning: Any hardware prefetch mechanisms that prefetch into the FLD or MLD and can generate multiple cache lines of prefetches from a single access and any buddy line prefetchers should ignore this access.
HW behavior: The FLD sequential (neighbor line) prefetcher, the MLD sequential prefetcher, and the MLD buddy line prefetcher will ignore accesses with this hint.
PMU Events: FLD_HINT_NO_MULTI_HWPREF counts the number of times *any* setting of this hint (NO_FLD_MULTI, NO_FLD_MLD_MULTI, or NO_FLD_MLD_MULTI_NOBUDDY) prevents an operation from triggering an FLD sequential (neighbor line) hardware data prefetch. Similarly, MLD_HINT_NO_BUDDY counts the number of times any setting of this hint prevents an operation from triggering an MLD buddy line prefetch.

- PF_DROP (bits [8:7])
These hint bits cause data prefetches (lfetches and hardware data prefetches) to be dropped when various events occur.

— PFD_NORMAL(value = 0)
Meaning: This hint is really the absence of the other PF_DROP hints.
HW behavior: The lfetch is carried to completion (short of triggering a fault if not lfetch.fault).

— PFD_TLB(value = 1)
Meaning: This lfetch should be dropped if it doesn't hit in a data TLBs.
HW behavior: This lfetch is dropped if it misses the DTB.
PMU Events: PREF_DROP.DTLB_MISS counts the number of times this hint causes a prefetch to be dropped due to a DTB miss.

— PFD_TLB_MLD (value = 2)
Meaning: This lfetch should be dropped if it doesn't hit in any data TLB or it misses the MLD.
HW behavior: This lfetch is dropped if it misses the DTB or the MLD.

PMU Events: In addition to the PMU event for the ON_DTB_MISS setting, the PMU event MLD_HINT_PREF_DROP that counts the number of times this hint causes a prefetch to be dropped due to an MLD miss.

— PFD_ANY (value = 3)

Meaning: This lfetch should be dropped if anything happens that would increase the cost of this lfetch above the minimum possible cost.

HW behavior: This lfetch is dropped if it misses in any data TLB, or if it misses the MLD, or if there is some risk of overflowing the data prefetch queue.

PMU Events: In addition to the PMU events for the ON_DTB_MLD_MISS setting, the PMU event PREF_DROP.FLDTLB_MISS) that counts the number of times this hint causes a prefetch to be dropped due to an FLDTLB miss.

- PIPE(bit [9])

— PIPE_DEFER (value = 0)

Meaning: Lfetches with this hint should not block the pipeline while fetching their TLB translations, and speculative loads with this hint should spontaneously defer when when waiting for some long latency operation.

HW behavior: An lfetch with this hint that misses the DTB will initiate a hardware page walk and be placed in the data prefetch queue. A speculative load that misses the DTB or MLD will spontaneously defer (if architecturally allowed).

PMU Events: MLD_HINT_DEFER counts the number of times a ld.s is spontaneously deferred due to an MLD miss.

— PIPE_BLOCK (value = 1)

Meaning: Lfetches with this hint should block the pipeline until they are done fetching their TLB translations, and speculative loads with this hint should block uses of their target register until they have completed their fetch.

HW behavior: An lfetch with this hint that misses the DTB will block the pipeline until its hardware page walk is completed. The pipeline will be blocked on the use of a speculative load with this hint that misses the DTB or the MLD until the speculative load returns a value to the target register. These behaviors were the default behavior on previous Intel Itanium processors, and this hint is a way to force these behaviors on the processor. An lfetch or speculative load without this hint may not block the pipeline, and the speculative load may be spontaneously NATed on a DTB or MLD miss - subject to architectural restrictions for spontaneous deferral.

PMU Events: DTLB_HPWHINT_BLK counts the number of times this hint causes a ld.s not to be spontaneously deferred due to a DTB miss.

- BIAS (bit [10])

— BIAS_EXCL

Meaning: If hardware has a choice of getting a line in either the shared or exclusive MESI states, it should choose exclusive.

HW behavior: A DREAD access with this hint that causes an LLC fill will fill the line to the exclusive (E) MESI state if the line doesn't exist in any other cache.

— BIAS_SHARED

Meaning: If hardware has a choice of getting a line in either the shared or exclusive MESI states, it should choose shared.

HW behavior: A DREAD access with this hint that causes an LLC fill will fill the line to the shared (S) MESI state.

PMU Events: RIL_REQ_REF_DATA.WB_CRD counts the number of times this hint causes an LLC fill to the S state instead of the E state.

2.3.1.3 PMU DEAR Support for Data Access Hints

In addition to the PMU support via the hint related events mentioned in the previous section, the Data cache EAR captures the hint values that were associated with the captured access. This allows a dynamic optimizer to know for sure what hint values various accesses are actually using. This can be useful both in figuring out what the opportunities are for changing hint values and for verifying which accesses are affected by a change in hint values.

2.3.1.4 Backwards Compatibility of DAHR Hints with Temporal Hints

Two of the DAHR index bits were previously defined to be temporal hint bits. The processor will interpret legacy code temporal hint usage by using the default hint values of the associated DAHRs. The temporal hint to associated DAHR mappings are shown in Table 2-18. The default DAHR values are shown in Table 2-19. The way that previous Intel Itanium processors will interpret DAHRs is shown in Table 2-18.

Table 2-17. Legacy Code Temporal Hint Mapping to DAHRs by the Processor

Temporal Hint	DAHR
none	DAHR0
nt1	DAHR1
nt2	DAHR2
nta	DAHR3

Table 2-18. DAHR Mapping to Temporal Hints by Previous Intel Itanium Processors

DAHR	Temporal HInt
DAHR0	none
DAHR1	nt1
DAHR2	nt2
DAHR3	nta
DAHR4	none
DAHR5	nt1
DAHR6	nt2
DAHR7	nta

Table 2-19. Default DAHR Values

DAHR	fld_loc	mld_loc	llc_loc	pf	pf_drop	pipe	bias
DAHR0	fld_normal	mld_normal	llc_normal	pf_normal	pfd_normal	pipe_block	bias_excl
DAHR1	fld_no_allocate	mld_normal	llc_normal	pf_no_fld	pfd_normal	pipe_block	bias_excl
DAHR2	fld_no_allocate	mld_nru	llc_normal	pf_no_fld	pfd_normal	pipe_block	bias_excl
DAHR3	fld_no_allocate	mld_normal	llc_nru	pfd_no_fld	pfd_normal	pipe_block	bias_excl
DAHR4	fld_normal	mld_normal	llc_normal	pf_normal	pfd_normal	pipe_defer	bias_excl
DAHR5	fld_no_allocate	mld_normal	llc_normal	pf_no_fld	pfd_normal	pipe_defer	bias_excl
DAHR6	fld_no_allocate	mld_no_allocate	llc_normal	pf_no_mld	pfd_normal	pipe_defer	bias_excl
DAHR7	fld_no_allocate	mld_no_allocate	llc_nru	pf_none	pfd_normal	pipe_defer	bias_excl

2.3.2 Instruction Fetch

2.3.2.1 Intel Itanium Processor 9300 Series Processor Differences

The following table enumerates key differences between Intel Itanium Processor 9500 Series and Intel Itanium Processor 9300 Series.

Table 2-20. Intel Itanium Processor 9300 Series Differences

	Intel Itanium Processor 9500 Series	Intel Itanium Processor 9300 Series
Branch misprediction penalty	10 cycles	6 cycles
MLI (512k) nominal access time	9 cycles	6 cycles
MLITLB (128 entry shared) access time	3 cycles	2 cycles
MLI to LLC fetch size	2 64B cache lines	1 128B cache line
0-bubble resteers	4 short IP-relative branches	Any short IP-relative branch in the FLI
1-bubble resteers	Any short IP-relative branch in the FLI, br.ret	br.ret
2-bubble resteers	NA	Non-return indirect branch
3-bubble resteers	Non-return indirect branch, long branch	NA
Branch predictor	Long branch (brl) target predictor added	No long branch support

2.3.2.2 Memory Hierarchy

Table 2-21. Instruction Cache Characteristics

	FLI	MLI
Size	16k	512k
Latency	1 Cycle	9 Cycles
Line Size	64B	128B
Associativity	4-way	8-way
Shared Across Threads?	Yes	Yes
Number of Ports	2 read/1 write	1 read/1 write
Max # of misses outstanding	16 64B (8 entries restricted to MLI buddy line)	8 128B
Fill Bandwidth	32B/cycle	32B/cycle
Address Ordering Granularity	32B	32B
Address Type	Pre-validated	Physical

The FLI on the processor is nearly identical to the FLI on Intel Itanium Processor 9300 Series. Both use prevalidation and consequently an FLI TLB page invalidation will invalidate all FLI instruction data associated with the invalidated page. On Intel Itanium Processor 9300 Series, virtual address page aliasing, even between threads, would cause FLI TLB page invalidation. Therefore, two threads accessing the same physical page in memory would invalidate each others FLI cache data and FLITLB on a 4k page size granularity. On the processor, virtual address page aliasing between threads will install a TLB entry per thread into the FLI TLB. Therefore, two threads executing out of

the same physical address space will not cause FLI TLB and FLI victimizations. However, the instruction data will be duplicated in the FLI in this scenario. In addition, the default FLI replacement policy is to mark both demand and prefetch fills as most recently used. Another key difference is that the processor no longer has an ISB. On the processor, fills from MLI are immediately filled into the FLI.

2.3.2.3 Instruction Prefetch

Instruction prefetching is controlled primarily by special prefetching hint instructions, instruction completer hints on branch instructions, and move to BR instructions. The only exception to this is “next line” prefetching, which is controlled by the hardware. With "next line" prefetching, an FLI miss to an address with IP[5]=0 will cause a 64B miss to the cache line aligned with their current IP address as well as to IP[5]=1. That is, the frontend will fetch the 64B "buddy" address of the current miss. If the miss address is to the "buddy", then only one 64B miss will be issued.

The following table enumerates how software can initiate instruction prefetching.

Table 2-22. Instruction Prefetching Instructions

Instruction	Action
mov br[N] = r[N]	Prefetch 8 bundles. Target address is specified by the branch register
br.few target	No action taken. Next line prefetching is possible if IP[5]=0
br.many target	Engage the streaming prefetch engine. Fetch 32 bundles ahead of the 64B cache line pointed to by target. The prefetch target starts at the next cache line after the target address.
brp.few target	Prefetch 8 bundles at the specified target address. Instruction treated as NOP by execution stages.
brp.many target	Prefetch 16 bundles at the specified target address. Instruction treated as NOP by execution stages.
brp.exit target, count	Prefetch count bundles at specified target address. Instruction treated as NOP by execution stages.

The instruction prefetcher will terminate prematurely if a control flow change is detected at that current IP, not at the prefetch IP. This control flow change can occur either due to a predicted taken branch or a backend resteer. On a thread switch, instruction prefetching is halted and saved on the thread that is going inactive, and the instruction prefetcher restores prior state and continues prefetching on the thread that is becoming the active thread if the instruction prefetcher was previously active.

2.3.2.4 Instruction Prefetch - Prefetch Structures

- PVAB (prefetch virtual address buffer):

  The PVAB is an 8-entry buffer used to store prefetch requests from brp and mov-br instructions. The PVAB is needed because the front end can see more brp/mov to br instructions per clock than can be issued.

- SPE (streaming prefetch engine):

  The SPE spins on a sequence of prefetch addresses for large basic blocks.

2.3.2.5 Instruction Prefetch -brp prefetches

Only brp instructions that are in slot 2 of a bundle are recognized by the prefetch logic. The PVAB has two write ports for storing prefetch information. Since it is possible to have a bundle pair with brp instructions in slot 2 of both bundles and have a mov-br executing in the DET stage at the same time, one of the brp instructions will be dropped. In this case, the brp in bundle 1 will be dropped.

Table 2-23. brp and movbr: Contention for PVAB Write Ports

brp instruction on slot 2 of bundle 0 (Brp0)	brp instruction of slot 2 of bundle 1 (Brp1)	MovBr in DET? (MovBr)	PVAB Inserts	
y	n	n	Brp0	—
n	y	n	Brp1	—
n	n	y	MovBr	—
y	y	n	Brp0	Brp1
y	n	y	Brp0	MovBr
n	y	y	Brp1	MovBr
y	y	y	Brp0	MovBr
n	n	n	—	—

2.3.2.6 Instruction Prefetch - Prefetch Cancellation

Prefetches generate extra memory traffic. To minimize unnecessary prefetches, the processor may cancel certain prefetches that are in the PVAB or in the prefetch pipeline. Prefetches are not sent to the MLI if they result in an FLI hit. Prefetches are also canceled on FLITLB misses. Additionally, the branch hints may specify a trace vector (.tk.tk, .nt.nt, .dc.dc etc). The trace vector is used to cancel a prefetch in the PVAB or any of the prefetch pipestages if the execution trace does not match the trace vector specified in the branch hints.

The trace vector is implemented as a 4-bit shift register with aging vector. The trace vector is a two bit vector corresponding to the direction the next two branches are to be predicted before the prefetch becomes ready. The aging vector is a two bit vector that indicates whether the corresponding trace bits have been checked (0=not checked, 1=checked). When both bits of the aging vector are 1, the prefetch is ready to be issued.

Every time a branch is executed in the front end (FET stage), the trace vector LSB for all PVAB entries that have a 0 in the aging vector LSB are compared to the branch prediction. If they match, then both the aging and trace vectors are shifted to the right, and a '1 is shifted in from the left into the aging vector. For entries that have a "dc" hint for any of the bits in the trace vector, the aging bit is set to a '1 to indicate that the branch has been checked. The streaming prefetch engine also supports prefetch cancellation based on the prefetch vector for brp.many prefetches.

2.3.2.7 Instruction Prefetch - Guidelines

Efficient use of the instruction prefetcher can reduce instruction cache miss penalties. However, the danger of over prefetching is FLI and ideally to a lesser extent, MLI cache pollution.

- Guideline: Use br.many instead of br.few unless you know that the target code of the branch is going branch again in less than 8 bundles for a branch target with IP[5]=0 or 4 bundles with a branch target with IP[5]=1.
- Guideline: Brp instructions are lfetch's for the i-side. Brp instructions can be used to reduce i-cache miss penalties. However, only i-cache misses that cause empty cycles in the backend reduce performance. In particular, i-cache misses after branch mispredictions are always exposed, and this is one area where brp instructions can be utilized.

2.3.2.8 Branch Prediction - Guidelines

- .dptk vs .sptk

  Dynamic branches have better prediction accuracy at the expense of extra FET replays on tight loops during the first few iterations. Static branches do not incur FET replay penalties, but prediction accuracy suffers in many cases. Static branches can not detect patterns or loop exit conditions on small trip count loops.

  Guideline: Use .dptk except on unconditionals where .sptk should be used.

- Multiway Branch Bundles

  Perfect branch history is only stored in the FLI BHT for up to two branches per bundle. BBB templates use an encoded history that is imperfect has several detrimental effects.

  Guideline: Avoid using BBB bundles

- MLIBHT/MLB Oversubscription

  Once a branch is evicted from the FLI, we store its branch history in the second level branch history table (MLB). The MLB has a history capacity of approximately 12k entries. Each entry has enough storage for two branches per bundle pair or one BBB per bundle pair.

  Guideline: For branching bundle pairs whose prediction requires local branch history for prediction accuracy, try to keep branch density down to at most two branches per bundle pair.

  Guideline: If a particular branch is unconditional or only expected to be seen once in > ~12k-24k branches, use the .clr hint to free up MLB resources. The MLB aliases when there are capacity issues.

- Short vs Long IP-relative target prediction

Table 2-24. Short versus Long IP-relative Penalties and Target Accuracy

	Resteer Penalty	Target Accuracy
Short IP-relative	0-1 cycles	21b immediate offset, only VA[40:4] are predicted. If IP+offset_21b causes a carry out past VA[40], this branch will always mispredict
Long IP-relative	3 cycles	Perfect target prediction

  Guideline: Use brl if IP-rel target calculation will be mispredicted using a short IP-rel branch.

- Non-return indirect branches

  While trigger prediction accuracy is typically very high, target prediction accuracy is very low. The FE reads the branch register file when the branch is in the FDC pipestage. This requires a large distance in cycles between a mov-to-br and non-return indirect branch. Backend pipeline dispersal stalls and replays can cause the required distance to be increase. It is important to note that this distance is greater on the processor than for prior Itanium processors.

  Guideline: Avoid non-return indirect branches by converting most common targets to IP-rel. In addition, scatter branch register usage across all 8 branch registers. This may increase the probability of a branch register already having the correct target.

2.3.2.9 Branch Prediction - Zero-Bubble Buffer

On prior Intel Itanium processors, short IP-relative resteers occurred without incurring any penalty cycles. That is, taken IP-relative branches were able to resteer the instruction pointer through a 0-bubble resteer mechanism. The branch target buffer (BTB) for these IP-relative branches on the processor now takes an additional cycle to access. Therefore, short IP-relative resteers now occur with a 1-cycle penalty.

This additional bubble during an IP-relative resteer becomes material during tight loop execution with high trip counts where the backend is not encountering any long latency events. For this reason, a smaller faster BTB was added on the processor. The Zero-Bubble Buffer (ZBB) is a small 4-entry BTB that caches the target for specific IP-relative branches. The insertion, replacement, and invalidation of branches into the ZBB is controlled explicitly by the hardware.

For a branch to be inserted into the ZBB, the branch needs to be:

- Short IP-relative
- Predicted ’taken’ for the current instruction fet
- Have a history of ’taken’ for the last 4 iterations

This last condition is accelerated for branches that are fetched into the FLI with either a .sptk or .dptk completer.

When a new branch meets the criteria for insertion into the ZBB, the replacement algorithm first looks for an invalid entry. If all ZBB entries are allocated, then an LRU is used. The LRU is updated on ZBB hits and ZBB invalidates.

A branch is invalidated from the ZBB when it is mispredicted 'taken' by the ZBB. In addition, capacity issues naturally cause branch replacement.

2.3.3 Register Fetch

The Register Fetch section of the core supports the register renaming and register stack features of the Itanium architecture.

The Register Renaming (RNM) unit translates virtual integer, floating-point and predicate register identifiers into physical register identifiers. On the processor, register renaming is performed in the Front End (FE) of the pipeline on the 32-byte aligned bundle pair delivered by instruction fetch. The renamed register identifiers are stored in the Instruction Buffer Queues for direct dispersal to the register files and the execution pipelines in the BE. A few corner cases are introduced due to register renaming being performed on the fetched bundle pair in the FE as opposed to being performed in the BE after instruction dispersal on “pre” Intel Itanium processor 9500 series Differences processors:

- Multiple alloc instructions in bundle pair - since an alloc does not change BOF, it does not affect renaming in the current cycle. The last alloc in program order will simply update CFM.
- Multiple cover, clrrrb instructions in bundle pair - A cover or clrrrb in the even-addressed bundle will cause instruction fetch to invalidate the odd-addressed bundle in the bundle pair and present it to RNM again in the next cycle.
- Multiple bsw instructions in bundle pair - At most 2 bsw instructions can be present in a bundle pair. The RNM unit tracks the state of PSR.bn on a per bundle basis and thus accounts for this corner case.

- Multiple branch instructions in bundle pair - Instruction fetch can only mark 1 branch instruction as a predicted taken branch in a bundle pair. If the predicted taken branch is in the even-addressed bundle, the odd-addressed bundle is invalidated and the instructions at the predicted taken target are delivered in subsequent cycles.

The Register Stack Engine (RSE) manages the stacked register set and is responsible for spilling and filling registers using RSE st and ld operations. The processor RSE, just like all previous Intel Itanium processors, only operates in the Lazy Mode (AR.RSC=0), that is, every memory reference issued by the RSE is deemed "mandatory" and needed for program progress. The processor core does add an extra 32 general registers to increase the number of stacked registers to 128 from 96 on the processor. This should be completely invisible to software, in general, and lead to fewer pipeline stall cycles to accomplish the injection of RSE spill and fill operations. The pipeline penalties attributable to RSE operations are accounted by the CYC_BE_IBD_STALL.RSE_* PMU events.

In order to reduce the data access penalties associated with RSE spill and fill operations, the hardware data prefetcher on the processor does support data prefetching initiated by various RSE operations. For further details, related to RSE-based data prefetching, please refer to the Data Prefetching section.

2.3.4 Data Fetch

For examples, refer to Appendix C, "'Data Fetch Software Optimization Opportunities and Examples".

The processor has a three level cache hierarchy. All cache levels are on chip. Data and instructions are stored in separate caches at the first- and mid-level and in the same cache at the last level. The caches that store data are called First Level Data (FLD) cache, Mid Level Data (MLD) cache, and Last Level Cache (LLC). Unless stated otherwise, all data cache hierarchy resources are dynamically shared by different threads. The Last Level Cache (LLC) is also shared by all the cores on a die.

The processor includes both per-core caches and per-socket cache that is shared across all the cores on the socket. The processor, like previous Itanium processors, has a three-level cache hierarchy. The first two levels include separate I and D in each core. The last-level cache (LLC) is inclusive of all of the cores caches. Some high-level details of the cache hierarchy are shown in the table below.

Table 2-25. Core Cache Hierarchy Summary

Cache	Data Type Supported	WriteThrough / WriteBack	Size	Line Size	Ways	Index	Queueing	Latency
FLD	Integer	WT	16K	64B	4	VA[11:6]	8 fills	1
FLI	Instruction	NA	16k	64B	4	VA[11:6]	1 Dmnd + 7 Prefetch	1
MLD	Integer, FP	WB	256K	64B	8	PA[14:7]	16 Ozq/16 Fills	8
MLI	Instruction	NA	512K	128B	8	PA[15:7]	8 requests	9

Table 2-26. Data Cache Characteristics

	FLD	MLD	LLC
size	16 KB	256 KB	up to 32 MB shared
latency	1 cycle reads, 2 cycle writes[b]	8,9 cycles minimum read latency for int / fp loads respectively[c]	~50 cycles nominally
line size	64 bytes	128 byte sectored tags, 64 byte data	64 bytes
associativity	4-way	8-way	32 ways
number of ports	2 read/write[d]	2 read/write[e]	8 read/write[f]
replacement algorithm	NRU	NRU	NRU
max # primary misses outstanding	16 fill requests; otherwise unlimited by FLD	at least 16 independent and 16 buddies[g]	12 misses / writebacks per slice
max # secondary misses outstanding	unlimited by FLD	16 independent and 16 more to the same 64 byte lines as first 16	LLC doesn't see secondary misses from a single core; cross-core secondary misses have complex queuing behaviors
write policy	write through; no write allocate	write back; write allocate	write back; write allocate
store data buffering	n/a	32 16-byte entries	12 misses / writebacks per slice
fill bandwidth	64 bytes (1 line) per cycle[h]	128 bytes (1 line and buddy) every 5 cycles	dependent on system and memory configuration (LLC isn't limiter)
alignment	all alignments of 1, 2, 4, and 8 byte accesses that do not cross an 8 byte boundary are supported	generally operations must not cross 16 byte boundaries and semaphore operations must be naturally aligned; see MLD section for more detail.	n/a
address ordering granularity	overlap in VA[11:0]	overlap in VA[7:4]	entire cache lines

[b] FLD writes are speculative on the processor. See FLD section for more details..
[c] Minimum MLD hit latency is up from 5 on previous Intel Itanium processors, but average MLD hit latency is probably similar to previous Intel Itanium processors.
[d] There are some conflicts among stores but between loads and stores as on previous Intel Itanium processors. See FLD section for more details.
[e] These aren't true ports but actually banks instead. Bank conflicts can occur as in previous Intel Itanium processors. See MLD section for details.
[f] These aren't true ports but are slices instead. (Slices and banks are pretty similar, but in the LLC they are called slices.) Slice conflicts and many other types of conflicts can occur on the Ring.
[g] Because of the intricacies of MLD queueing, sometimes MLD will be non-blocking to some hits even with more this number of primary misses unresolved. However, at most the specified number can be simultaneously outstanding to the higher level caches.
[h] There are some conflicts between fills and stores, but less so that previous Intel Itanium processors. See FLD section for more details.

Table 2-27. Data Cache Differences from Previous Intel Itanium Processors

	FLD	MLD	LLC
Size	Unchanged	Unchanged	Less (4) MB per core than some previous processors but cache is shared among all cores on die
Latency	2 cycle (speculative) wr latency is down from 4 (non-speculative) previously	Minimum latency is up from 5,6 cycles previously; typical latency is close to unchanged	Latency is more than double previous Intel Itanium processor
Line Size	Unchanged	Down from 128 byte lines previously to 64- byte lines with buddy line prefetch	64B down from 128B
Associativity	Unchanged	Unchanged	up from 12 ways

Table 2-27. Data Cache Differences from Previous Intel Itanium Processors

	FLD	MLD	LLC
number of ports	down from 4 ports (2R, 2W); fill port is better on the processor	down from 4 ports	still a ratio of 1 port per core, but all 8 ports are accessible by all cores
replacment algorithm	unchanged	unchanged	unchanged
max # primary misses outstanding	up from 8 previously	unchanged	fewer misses per core, but all queueing available to all cores
max # secondary misses outstanding	unchanged	# unchanged, but different implementation might be detectable	n/a
write policy	unchanged	unchanged	unchanged
store data buffering	none needed on the processor due to lower speculative write latencies	up from 24 entries	less writeback buffering per core but total writeback buffering is available to all cores
fill bandwidth	up from 32 bytes / cycle	unchanged	system dependent
alignment	unchanged	unchanged	n/a
address ordering granularity	improved from overlap in VA[11:2]	less granular	unchanged

Each data access operation comes into the one of the main pipeline M-pipes from the instruction buffer, the data prefetch queue, or the register stack engine. Operations that continue on to the MLD exit the main pipeline and are inserted in the MLD OZQ. The MLD OZQ is a decoupling buffer between the main pipeline back-end and the MLD pipeline similar to the instruction buffer between the main pipeline front-end and back-end. Data access operations enter the MLD pipeline from the MLD OZQ (or bypass), the MLD SMQ, or the MLD FAB. Operations that MLD cannot service by itself are inserted into the MLD FAB and continue on to the LLC via the Ring Interface Layer (RIL) and the Ring that connects the multiple LLC cache slices. Operations that the LLC cannot service by itself continue on to socket local memory or over Intel QPI to off socket memory as appropriate.

Data returning from memory returns over the Ring to both the appropriate LLC slice and the requesting core. For loads returning from the Ring, as critical chunk arrives, it is forwarded to integer or floating point register file as appropriate. The register files have separate ports for register returns, so register return operations to NOT interrupt the main pipeline as they did in previous processors. Operations that fill the FLD forward a whole cache line of data to the FLD. The FLD has a (somewhat) separate fill port, so FLD fill operations do not typically interrupt the main pipeline. Operations returning to the MLD from the Ring must typically be issued down the MLD pipeline again whether or not they fill the MLD.

2.3.4.1 Data Prefetching

The processor has support for a variety of new kinds of hardware and software initiated data prefetching.

2.3.4.1.1 Data Prefetch Queue (DPFQ)

The DPFQ is a queue that hold data prefetches that are waiting to issue on a main pipeline M-port. All data prefetches, except for single count lfetches issued on an M-port and MLD buddy line prefetches, go through this queue. The DPFQ has 8 entries per thread and holds all the information needed to execute a data prefetch. The DPFQ is a FIFO, and when a prefetch is written into a full DPFQ, the oldest entry is dropped. A

variety of mechanisms are employed to avoid dropping software initiated prefetches and to intentionally drop old or bandwidth limited hardware initiated prefetches. See the following sections on software and hardware data prefetching for more details.

2.3.4.1.2 Software Initiated Data Prefetching

Lfetches are instructions that allow software to initiate a data prefetch. In previous processors, all lfetches were executed in the main pipeline in much the same way a load instruction would be. On the processor, lfetches make use of the DPFQ in the following cases:

- lfetch-on-A

  Previous Intel Itanium processors had 4 memory ports in the main pipeline. The processor has only 2. Thus, the available M-op issue bandwidth per cycle has decreased relative to previous processors. This decrease in M-op issue bandwidth makes it more costly for software to issue lfetches. In an attempt to offset this increase in cost and to encourage software to issue as many useful lfetches as it can, regular lfetches (those without .excl, .fault, or .count suffixes) will be allowed to issue on the A ports in addition to the M ports.

  One of the functions of the DPFQ is to temporarily hold lfetches from an A-port after they look up their virtual addresses (VAs). Lfetches in the DPFQ wait and issue in order to an M-port, on a cycle that it is not being used. Thus, the DPFQ is essentially providing out-of-order execution of lfetches and allowing these lfetches to access M-port bandwidth that is available dynamically (due to various pipeline replays, flushes, and issue stalls) but not previously available statically to software. To handle situations where the M-ports are completely utilized, the DPF can be configured to force an lfetch into the pipeline via preemption, if that lfetch has been waiting to issue for more than a certain number of cycles. The PMU event DPFQ_ENQ.LFETCH counts when lfetches from an A-port are inserted into the DPFQ.

- lfetch.count

  The DPFQ also supports a new counted variety of lfetch, named lfetch.count. A single lfetch.count instruction can represent up to 32 individual prefetches, and can specify various forward and backward strides between fetches. An lfetch.count occupies one entry in the DPFQ, and is expanded into multiple fetches as it is read out of the queue and sent into an M-port. The PMU event DPFQ_ENQ.LFETCH_COUNT counts when counted lfetches are inserted into the DPFQ.

- DTB missing lfetch

  In previous designs a static choice could be made that allowed lfetches that missed the DTB to either be immediately dropped or to stall the main pipeline waiting for the resulting hardware page walk (HPW) to complete. Each of these choices is not optimal sometimes. If the lfetches are really needed, dropping all of them that target a page lacking a translation in the DTB is not a good choice. If the lfetches are not needed (e.g. prefetching off the end of a loop), then repeatedly stalling (possibly repeatedly) the pipeline to do a useless HPW is not a good choice. The DPFQ allows the pipeline to continue executing after an lfetch encounters a DTB miss without dropping lfetches. Instead, lfetches missing the DTB are placed in the DPFQ, where they wait until a (non-blocking) HPW is completed and then issue to the M-ports. The PMU event FLD_HWPREF_INS.DTBMISS counts when these operations are inserted into the DPFQ.

2.3.4.1.3 Hardware Initiated Data Prefetching

Software initiated prefetches are data prefetches that are initiated by lfetch instructions. Hardware initiated prefetches are prefetches that are initiated by the hardware in response to something other than an lfetch instruction. Both software and hardware prefetching (except MLD buddy prefetching) make use of the DPFQ. The DPF block provides for several varieties of hardware initiated prefetching:

- FLD sequential (neighbor line) prefetching

  The goal of FLD sequential prefetching is to use a simple algorithm to take advantage of spatial locality in the FLD. The simple algorithm is as follows:

  — If an FLD load (FLDLD) misses the FLD and hits in the FLDTLB, insert a prefetch in the DPFQ that will prefetch N lines in the forward direction and M lines in the backward direction. The PMU event DPFQ_ENQ.FLD_BIDI counts these DPFQ insertions.

  — When the Nth line of a forward FLD sequential prefetch fills the FLD, mark that cache line FWD. When the Mth line of a backward FLD sequential prefetch fills the FLD, mark that cache line BWD.

  — When an FLDLD hits in the FLD on a line marked FWD, insert a prefetch in the DPFQ that will prefetch N more lines in the forward direction. When an FLDLD hits in the FLD on a line marked BWD, insert a prefetch in the DPFQ that will prefetch M more lines in the backward direction. The PMU events DPFQ_ENQ.FLD_FWD and DPFQ_ENQ.FLD_BWD count these DPFQ insertions.

  The parameters N and M will be fixed, but have not yet been determined.

- .rel op transform to prefetch

  A prefetch is inserted in the DPFQ when store with release semantics is observed in the main pipeline. This generated prefetch is not required to observe the release semantics and thus may be able to fetch the relevant cache line sooner than the original fetch. The prefetch is dropped if it hits in MLD. The PMU event FLD_HWPREF_INS.REL_OP) that counts these DPFQ insertions.

- .acq op prefetching

  When a data fetch instruction with acquire semantics is outstanding (observed by DPFQ in main pipe, but not yet globally visible), any MLD destined load instructions observed by the DPFQ will cause a data prefetch operation to inserted in the DPFQ. These data prefetches, if they reach the MLD OZQ while the instruction with acquire semantics is still outstanding, will be allowed to issue out of the OZQ while the instruction with acquire semantics is outstanding. The PMU event FLD_HWPREF_INS.ACQ_PEND counts these DPFQ insertions.

- FLD store alias replay case prefetching

  In the case where the instruction on M1 is WB2 replayed due to a possible store alias with the instruction on M0 (see FLD WB2 replay section), a prefetch of the line targeted by the store in M1 is inserted in the DPFQ. This is done because this WB2 replay is not exact, and moving the translation for the store into the FLDTLB will resolve the potential store alias in a way that will remove unneeded WB2 replays. The PMU event FLD_HWPREF_INS.STORE_ALIAS counts these DPFQ insertions.

- RSE prefetching

  Various types of RSE activity can trigger hardware prefetching, with the goal of prefetching lines that the RSE will soon need. The last of a sequence (or “episode”) of RSE fills will trigger prefetching backwards from the final load address. The last of a sequence/episode of RSE spills will trigger prefetching forwards from the final store address. A mov to ar.bspstore will begin prefetching backwards from the new bspstore address, with the expectation that new RSE fills are likely to follow soon. The number of lines prefetched (or whether no prefetching is done at all) is individually configurable for each of these three cases. There are PMU events

(DPFQ_ENQ.MOV_BSPST, DPFQ_ENQ.RSE_ANY, DPFQ_ENQ.RSE_LOAD, and DPFQ_ENQ.RSE_STORE) that count these DPFQ insertions.

- MLD sequential prefetching

  All demand data fetches that miss the MLD are tracked, and can trigger hardware prefetching if sequential behavior is detected. The MLD prefetcher can track accesses to 8 different 4 kB pages (per thread). Two accesses within the same address window on one of these pages indicate a possible sequential access pattern, and prefetch the next line in the appropriate direction. Further accesses that fall within the address window of the previous access indicate increasingly higher confidence of a sequential access pattern, and increase the number of lines that are prefetched ahead of the last access. An access to a page that falls outside the address window of the previous access indicates reduced confidence of a sequential access pattern, and decreases the number of lines that will be prefetched. The window size for determining sequential accesses, and the maximum number of lines that may be prefetched (or whether no prefetching is done at all), are both configurable. There is a PMU event (currently called DPFQ_ENQ.MLD) that counts these DPFQ insertions.

- MLD buddy prefetching

  Unless hinted otherwise (via PF_NONE), an operation that misses MLD typically triggers a prefetch of the other half of its aligned 128B chunk, in addition to the fetch of its 64B line. (PMU event: MLD_FILL_MESI_STATE_BUDDY.ANY)

2.3.4.1.4 DPFQ Data Fetch Retry

There are several cases where the DPFQ is used to retry a data fetch that would have been dropped or would have significantly delayed the main pipeline:

- MLD OZQ full DPFQ retry

  When a data fetch (either hardware or software prefetch) that was issued into the main pipe by the DPFQ is rejected by the MLD OZQ because no entries are available to hold the data fetch (looks like an lfetch to MLD), that data fetch is re-inserted into the DPFQ behind the other data fetches already in the DPFQ. The PMU events FLD_HWPREF_INS.OZQ_FULL and FLD_HWPREF_INS.OZQ_FULL_LFETCH counts these DPFQ insertions.

- FLD fill request retry

  FLD fill requests associated with a data fetch are dropped in several scenarios including the following:

  — data fetch misses the FLDTLB (but goes out to MLD, and possibly triggers a DTB-to-FLD transfer)

  — a store targeting the requested line reaches the DET stage of the pipeline before the outstanding FLD fill request has filled the FLD

  In these cases, a prefetch requesting an FLD fill is inserted in the DPFQ in place of the dropped FLD fill for the original data fetch. The prefetch is hinted to be dropped if it misses the MLD. Three PMU events (FLD_HWPREF_INS.FLDTLBMISS, FLD_HWPREF_INS.FLDTLBMISS_LFETCH, and FLD_HWPREF_INS.CNCLDFILL) count these DPFQ insertions.

2.3.4.1.5 Managing DPFQ Oversubscription

If prefetches are being inserted into the DPFQ at a higher rate than they are leaving the queue and entering the pipeline, then the DPFQ can fill up and overflow. When the DPFQ overflows, the older prefetches are dropped to make way for the newer prefetches. Dropping software initiated prefetches is typically highly undesirable from a performance perspective. Therefore, the DPFQ has several mechanisms to help it avoid dropping software initiated prefetches. It can drop hardware initiated prefetches. It can

force open slots on the M-pipes. It can continually replay the main pipeline. To facilitate the usage of these mechanisms, the DPFQ keeps track of how old the entry at the head of the queue is, how full it is, and how many software initiated prefetches it contains.

When the prefetch at the head of the DPFQ gets too old or the queue gets too full of software initiated prefetches, if that prefetch is a hardware initiated prefetch, it is dropped. If that prefetch is a software initiated prefetch and it is ready to be issued (not waiting on the DTB or HPW), then it holds off instruction issue on the M-ports for a cycle so it can be issued.

When the DPFQ gets full enough of software initiated prefetches that it is in danger of dropping some and it is unable to issue any prefetches because it is waiting on something (for example, DTB->FLDTLB transfer, HPW completion, available OZQ entry), it will start continuously WB2 replaying the main pipeline until this condition clears. (PMU event: BE_CYC_WB2_REPLAY.DAHR_HZRD)

When an lfetch instruction is hinted PFD_ANY (see Data Access Hints section) and it enters the DPFQ (for example, due to being issued on an A-port or being a counted lfetch), it will be treated as if it were a hardware initiated prefetch for the purpose of managing DPFQ oversubscription. The purpose of this capability is to provide software with a very low cost prefetch instruction.

2.3.4.1.6 Dropping Data Prefetches

The following are the events that cause data prefetches to be dropped:

- DPFQ Overflow

  When an prefetch is inserted in a full DPFQ, the oldest DPFQ entry is dropped. The PMU event DPFQ_ENQ_OVERFLOW.ANY counts these events.

- DPFQ Stale or Filling Up with Software Initiated Prefetches

  When hardware initiated prefetches have been in the DPFQ for too long or are at the head of the DPFQ and the queue is filling up with software initiated prefetches, they may be dropped. The PMU event DPFQ_DEQ_PREEMPT.TIMEOUT counts these events.

- FLD Hit

  Prefetches are dropped when they hit in the FLD. The PMU event PREF_DROP.FLD_HIT counts these events.

- Secondary FLD Miss or FLD FAB Full

  A prefetch whose only goal is to fill the FLD (hinted (PFD_TLB_MLD or PFD_ANY) and not FLD_NO_ALLOCATE) is dropped when it looks in the FLDFAB sees that it is full or sees there is an outstanding FLD fill request to the same line. The PMU event PREF_DROP.FLD_SECONDARY_MISS counts these events.

- FLDTLB Miss

  Prefetches hinted PFD_ANY are dropped when they miss the FLDTLB. The PMU event PREF_DROP.FLDTLB_MISS counts these events.

- DTB Miss

  Prefetches hinted PF_DROP!=PFD_NORMAL are dropped when they miss the DTB. Virtually addressed data prefetches, other than lfetch.fault, that have any issue that would cause an lfetch.fault to take an address related fault will be dropped. The PMU event PREF_DROP.DTB_MISS counts these events.

- MLD Hit

  Prefetches that don't request an FLD fill are dropped when they hit the MLD. The PMU events (MLD_REF.HIT + MLD_SMQ_REF.HIT) with a dataref filter of LFETCH and/or HWPF counts these events.

- MLD Miss

  Prefetches hinted PFD_TLB_MLD or PFD_ANY are dropped when they miss the MLD. The PMU event MLD_HINT_PREF_DROP counts these events.

- Secondary MLD Miss

  Prefetches that are secondary MLD misses are dropped. The PMU events (MLD_REF.SECONDARY_DROP + MLD_SMQ_REF.SECONDARY_DROP) count these events.

- FLD Fill Cancel

  The FLD fill portion of a data prefetch can be cancelled by anything that can cancel an FLD fill (see FLD fill section), even though that prefetch may still fill MLD and/or LLC. The PMU event FLD_FILL_CANCEL.ANY with a dataref filter of LFETCH and/or HWPF) that counts these events.

2.3.4.2 Data Cache

2.3.4.2.1 FLD

The First Level Data cache (FLD) is a 16 KB, 4-way set associative, single cycle, two-port data cache. It handles integer loads and stores and has a line size of 64B. As with previous Intel Itanium processors, floating point loads and stores and handled by the MLD.

- Pipelines

  In previous Intel Itanium processors, the FLD (called L1D), the FLDTLB, and the DTB were on a pipeline that was distinct from the main pipeline. It could recirculate, and might not always be in step with the main pipeline. On the processor, the FLD, FLDTLB, and DTB are all directly part of the main pipeline. On previous Intel Itanium processors, the main pipeline could stall. On the processor, the main pipeline cannot stall. Instead, hazards are handled with replays. In particular, FLD hazards are handled with DET and WB2 replays. Also, on previous Intel Itanium processors, some MLD (called L2D) related operations, including recirculates and returns to register files, had to go through the FLD (called L1D) pipeline. On the processor, once operations are sent to the MLD, they never have to go back through the FLD (that is, main) pipeline.

- Pre-validation (with respect to the FLDTLB) and Multi-threading

  As with previous processors, the FLD cache is pre-validated with respect to the FLDTLB. This means that every entry is associated with one and only one entry in the FLDTLB. Since entries in the FLDTLB are associated with only one thread, FLD cache lines are also associated with only one thread. (See FLDTLB section for a discussion of two threads accessing the same physical address and virtual aliasing in general.) If the FLDTLB entry that an FLD cache line is associated with is overwritten or become invalid, the corresponding lines in the FLD are no longer accessible.

- FLD Operations and Operation Types

  The next two tables define some FLD operations and types of operations that will be used in descriptions and figures that follow.

Table 2-28. Some FLD Asynchronous Operations (Sheet 1 of 2)

Name	Description	Injection Stage
a_snp	Snoop (not snoop to shared)	IBD

Table 2-28. Some FLD Asynchronous Operations (Sheet 2 of 2)

Name	Description	Injection Stage
a_snps	Snoop to shared	IBD
a_dpf	data prefetch	REG/IBD
a_flshd_stinv	Invalidation of an FLD cache line that was updated by a non-committing store	DEC/REG/ EXE;IBD

Table 2-29. FLD Operation Type Definitions

Op Type	Instructions	async ops
FLDLD	integer loads other than ld16 and ldc.acq; RSE loads	none
FLDPF	lfetch	a_dpf
FLDCANFILL	FLDLD + FLDPF instructions	FLDPF ops
FLDST	any integer store other than st16; RSE stores	none
FLDINVLDT	stf, st16, fc, xchg, cmpxchg, fetchadd	a_snp, a_snps, a_flshd_stinv
FLDWR	FLDST + FLDINVLDT instructions	FLDINVLDT ops

- FLD Hit: FLDLD

  The result of FLDLD that hits in the FLD is available to most uses the cycle following the load. An FLDLD result is not available to be used as an address until 2 cycles after the load. Any use of the FLDLD result as an address within a cycle of the FLDLD (independent of predicates) will result in a single cycle IBL issue stall (PMU event: CYC_BE_IBD_STALL.MTOM).

- FLD Hit: FLDWR

  The FLD implements no allocate on write and write-through policies. In other words, writes do not trigger FLD fills, and all writes are forwarded to the MLD. Writes to the FLD are somewhat speculative on the processor allowing a lower FLD write to read latency than on previous Intel Itanium processors. The result of an FLDWR that hits in the FLD is available to an FLDLD 2 cycles after the FLDWR. An FLDLD that follows an FLDWR in the same cycle or 1 cycle later with an overlapping virtual address (VA[63:12] ignored but truly overlapping with respect to VA[11:0]) will be DET replayed (PMU event: CYC_BE_DET_REPLAY.LOAD_AFTER_WRITE). Previous Intel Itanium processors had greater penalties, in general, for FLDLD after FLDWR hazards as shown in Table 2-30.

Table 2-30. FLD Hitting FLDLD after FLDWR Hazard Penalties

FLDWR - FLDLD Distance	Processor Penalty	Previous Intel Itanium Processor Penalty
0 cycles	5 cycles (VA[11:0] considered)	17 cycles (VA[11:2] considered)
1 cycle	5 cycles (VA[11:0] considered)	3 or 5 cycles (VA[11:2] considered)
2 cycles	no penalty	3 cycles (full VA considered)
3 cycles	no penalty	1 or 3 cycles (full VA considered)

The processor FLD does not have a store buffer. Instead, hitting FLDSTs write the FLD speculatively (DET stage). If they do not retire (WB2 stage) the speculatively updated cache lines must be invalidated. WB2 replays, faults/traps/interrupts, and branch target mispredictions can cause FLDSTs that hit in the FLD to not retire. Any FLD invalidates due to flushed stores will typically occur in the empty cycles following a replay or flush. However, snoops conflict with FLD flushed store invalidate operations, so snoops already in the pipeline could delay the FLD flushed

store invalidate operations. If an FLDLD beats any pending a_flshd_stinv operations down the pipeline, it will be DET replayed (PMU event: CYC_BE_DET_REPLAY.FLUSHED_STORE).

  - The FLD store ports on processor have been improved relative to previous Intel Itanium processors. Previously, some stores could conflict with other loads and stores. On the processor, there are not conflicts between loads and stores. However, two simultaneous hitting FLDSTs with the same VA[7:5] and different VA[11:8] conflict. The second store will be DET replayed (PMU event: CYC_BE_DET_REPLAY.STORE_VS_STORE).

- FLD Misses
  - Using the result of an FLDLD that misses the FLD will result in an EXE replay (PMU event: CYC_BE_EXE_REPLAY.GR_LOAD_RAW) if the load more than one cycle after the load or a DET replay (PMU event: CYC_ BE_DET_REPLAY.GR_LOAD).
- FLD Fills

  Moving a cache line into the FLD is called filling the FLD. Before requesting that the MLD return a cache line for an FLD fill, the FLD must allocate an entry in the 16 entry FLD Fill Address Buffer (FLDFAB). An operation cannot trigger an FLD fill request when the FLD FAB is full. The FLDFAB is used to store addressing information associated with the FLD fill. An FLD fill is requested and an entry allocated in the FLDFAB when all of the following are true:

  - operation type is FLDCANFILL (see Table 2-29)
  - operation is a virtual mode operation
  - data access hint says allocate in FLD (see section on data access hints)
  - operation misses the FLD
  - operation hits in the FLDTLB
  - there is an entry available in the FLDFAB
  - there is no outstanding FLD fill to the same line

  An FLDFAB entry is removed when any of the following events occur:

  - the operation is an instruction and that instruction fails to retire (MLD drops fill request)
  - the FLD receives an FLD fill operation from MLD

  An FLD fill is cancelled prior to or simultaneous with the removal of the FLDFAB entry when any of the following events occur:

  - an FLDWR operation targeting the fill cache line reaches the DET stage (A hardware initiated prefetch can occur in this case.)
  - the FLDTLB entry corresponding to the fill is invalidated
  - the MLD requests the FLD fill be dropped due to a prefetch that was hinted to be dropped on MLD hit or an off thread prefetch that passed a potentially overlapping (same VA[11:6]) on thread store
  - an FLDTLB insert occurs 1 cycle before the time at which the FLD fill was going to occur
  - an FLDINVLDT with the same VA[11:6] reaches DET 1 cycle before the fill
  - a snoop is lined up to occur 1 cycle after an FLD fill

  On previous Intel Itanium processors, all FLD (called L1D) fills conflicted with all FLD accesses. On the processor, conflicts between FLD fills and other FLD accesses in the main pipeline have been reduced to the following two DET replay causing conflicts:

— FLDWR hit vs. FLD fill - A hitting FLDST is replayed if it occurs simultaneously with a fill with same VA[7:6] and different VA[11:8] due to a structural hazard in the FLD data array. A hitting FLDWR is replayed if it occurs at the same time as or one cycle following a fill with the same VA[11:6] so that a write targeting a line being replaced will not incorrectly write to the line that is replacing it. (PMU event: CYC_BE_DET_REPLAY.WRITE_HIT_VS_FILL)

— FLDWR miss vs. FLD fill - An FLDST that misses the FLD and has the same VA[13:6] as a fill that occurs at the same time as or one cycle following the FLDST is DET replayed. An FLDWR that misses the FLD and has the same VA[13:6] as a fill that occurs one cycle before the FLDWR is DET replayed. These replays happen because there is not time for the FLDWR to cancel the fill in the case that the FLDWR and fill are associated with the same VA[63:6]. (PMU event: CYC_BE_DET_REPLAY.WRITE_MISS_VS_FILL)

- FLD replacement

  The FLD uses a Not Recently Used (NRU) replacement algorithm. In this algorithm, one bit of state is associated with each line in the cache. Each of these bits can be in a Recently Used (RU) state or a Not Recently Used (NRU) state.

  At the time of an FLD fill, one of the 4 ways in the appropriate set is selected for replacement in the following manner:

  — If all of the ways are marked NRU, the way pointed to by the FLD_random_way_ptr is chosen and the FLD_random_way_ptr is rotate by one way. Otherwise, the first way marked NRU is selected.

  The replacement state of the FLD is updated in the following manner:

  — When an FLDLD hits in the FLD...
    - Mark the accessed cache line RU.
    - If all the lines in the set are now marked RU, mark them all NRU.

  — When a cache line is filled to the FLD...
    - If the fill is hinted to be marked NRU, it is.
    - Otherwise, mark the filled line RU.
    - If all the lines in the set are now marked RU, mark them all NRU.

2.3.4.2.2 MLD

The Mid Level Data Cache (MLD) is a 256 KB, 8-way set associative, 2-ported data cache. The minimum integer load-use latency is 8 cycles. It handles all memory reference instructions save integer loads that can be satisfied by the First Level Data Cache (FLD). In addition to the large caching structure, the MLD also contains the Ordering Czar Queue (OZQ), which manages all architecturally required ordering constraints on memory references. The MLD is not guaranteed to be inclusive of the FLD, but the LLC will be guaranteed to be inclusive of the MLD.

- Pipelines

  The MLD implements a 9-stage pipeline that is independent from the main instruction pipeline. The MLD connects to the main pipeline where it receives memory operations either directly from the instruction stream or synthesized by the FLD. The OZQ forms the logical decoupling point between the main pipeline and the MLD pipeline (in much the same way the the Instruction Buffer forms a decoupling point between the Front-End and Back-End instruction pipelines). When the OZQ is empty, memory ops may bypass around the OZQ. In this case, the L1A stage of the MLD pipeline corresponds with the DET stage of the main pipeline. When the OZQ is not empty, or an operation is not allowed to bypass (due to asymmetry or semantic ordering constraints), the OZQ is written at the end of the DET stage. Memory ops in the OZQ are selected (nominated) in the first, or L1N, stage of the MLD pipeline.

Unlike prior processors, the MLD tag and data arrays are part of the same monolithic pipeline and logically occur after the OZQ. In addition, the MLD pipeline is a stall-based pipeline, not a replay-based pipeline. Once an op is issued into the MLD pipeline from the OZQ it is guaranteed to complete. The MLD pipeline stalls to reconcile structural and RAW hazards. The OZQ nomination mechanism handles semantic and address ordering constraints (and makes an attempt to prevent some structural hazards).

Like prior processors, the MLD data is pseudo-ported via banking. The data array is broken up into 16 banks. A given data value is mapped to a specific bank by address bits 7:4. Each bank is single-ported. If multiple operations need to read or write the same bank on the same cycle, a structural hazard occurs, see below. The banks are organized in such a way that a 64 B FLD fill operation, essentially a 64 B transfer from MLD to FLD, can complete in a single cycle. By the same token, a 64 B MLD fill operation (on an MLD miss) also completes in a single cycle. The MLD tag array has 2 true ports, supporting two independent reads or one write (for fills) per cycle. All structural hazards, therefore, are a consequence of the data array design.

- Hazards

  There are two types of hazards the MLD pipeline resolves: structural and RAW.

  There are several sub-types of structural hazards: RR bank, WW bank, RW bank, Fill-store, and fill port.

  — *RR bank* hazards occur when two memory ops flow down the MLD pipeline together that need to read the same bank (determined by address bits 7:4), but not the same full address. In this case, the MLD pipeline will stall for one cycle. Ops issued from the OZQ will never have a RR bank hazard and thus this hazard will only occur on bypassed ops.

  — *WW bank* hazards occur when two memory ops flow down the MLD pipeline together than need to write the same bank (address bits 7:4), but not the same full address. Again, the MLD pipeline will stall for one cycle. Note that if a pair of ops have both RR bank and WW bank hazards (that is, Read-Modify-Write (RMW) stores), only one pipeline stall will occur. Ops issued from the OZQ will never have a WW bank hazard and thus this hazard will only occur on bypassed ops.

  — *RW bank* hazards occur when a store that is writing a given bank is followed 4 cycles later by a memory op that is reading the same bank, but not the same full address. The pipeline will stall for one cycle. Note that the stall may cause a subsequent RW bank hazard with a following store. Because this is an inter-stage hazard, ops issued from the OZQ may have a RW bank hazard.

  — *Fill-store* hazards occur when a store is followed 4 cycles later by a fill operation, independent of address. The pipeline will stall for one cycle (and may cause a subsequent Fill-store hazard with a following store.)

  — *Fill port* hazards occur when two memory ops flow down the MLD pipeline together and both ops will perform an FLD fill. The pipeline will stall for one cycle. Note that if the ops also have a RR bank hazard, only one stall will occur. The OZQ will attempt to avoid these hazards but they are not completely prevented when issuing out of the OZQ.

  RAW hazards occur when a multi-bank read matches same physical address of an older store that has not completed. Matching in this case means matching down to address bit 6. Multi-bank reads are integer loads and prefetches performing an FLD fill, flushes and snoops. The MLD pipeline will keep the multi-bank read stalled in the L1M pipeline stage until the store reaches the L1X pipeline stage. An exception to this occurs when an integer load with FLD fill follows the store and both the load and store address match down to address bit 4. In this case, there will be no stalls. Because this is an inter-stage hazard, ops issued from the OZQ may have a RAW hazard.

- AR Hazards

  There is another form of hazard that is resolved by the OZQ nomination mechanism rather than the MLD pipeline - those involving the two ARs implemented by the MLD, namely CCV and CSD. RAW, WAR, and WAW AR hazards are detected between ops. AR writers are "move to" instructions, ld16 and cmp8xchg16. AR readers are "move from" instructions, st16, and cmpxchg. These hazards are resolved by holding off issue into the MLD pipeline until the previously issued operations that form the hazard are completed (possibly requiring waiting for a ld16, st16 or cmpxchg miss to complete). This hazard logic is thread aware (since the ARs are threaded), although there may be superfluous stalls when an AR reader or writer for the opposite thread is flowing down the MLD pipeline.

- Store-Store and Store-Load Bypassing

  Unlike prior processors, a store followed by another store to the same address suffers no penalty - if even the stores are RWM stores or are closer than 5 cycles. A RWM store is a 1 or 2 byte store or an unaligned store. Likewise, a load following a store to the same address will not have a penalty (provided they are either to the same address down to address bit 4 or the load is not requesting an FLD fill.

- OZQ Full and OZQ Data Buffer Full Replays

  The OZQ on the processor is smaller (16 entries) than on previous processors. This was enabled by two changes: 1) Secondary misses are placed into a separate structure - the SMQ. 2) The OZQ allocation and nomination mechanism doesn't use head and tail pointers but uses age vectors instead. This allows more efficient usage of it's entries - there are no "holes" in the OZQ. The OZData Buffer, which holds data for stores that have not yet completed, has been expanded from prior processors to 32 entries. However, the OZQ can still be filled completely as can the OZData Buffer. When the MLD OZQ is full, operations that need to go into the MLD OZQ are WB2 replayed to an IBL issue stall. When the MLD OZ Data Buffer is full, operations that need to put data into the MLD OZ Data buffer are WB2 replayed and then EXE replayed continuously until the MLD OZ Data Queue is no longer full.

- MLD Fill Policies

  Unlike prior processors, memory operations that miss the MLD are not required to fill into the MLD. This behavior can be controlled via Data Access Hints. The MLD line size was reduced from 128B in prior processors to 64B. This was implemented without increasing the tag array size by using a "buddy" line scheme (also known as sectoring). A single MLD tag covers 2 64B data lines. Each of the 64B lines contains its own, independent MESI state. Generally, on an MLD miss both 64B lines will be filled into the MLD. The non-critical 64B line (the buddy) is essentially prefetched by the MLD. Again, this can be controlled via Data Access Hints. The MLD is 8-way set associative, with the way selected for replacement at the time an MLD miss is detected. Note that there is no "pending" tag state for the MLD. Instead, the victim data is not read out until immediately prior to the corresponding fill.

- MLD Replacement

  The MLD uses a Not Recently Used (NRU) replacement algorithm. In this algorithm, one bit of state is associated with each line in the cache. Each of these bits can be in a Recently Used (RU) state or a Not Recently Used (NRU) state.

  At the time of an MLD fill, one of the 8 ways in the appropriate set is selected for replacement in the following manner:

  — The first way marked NRU that is not pointed at by the MLD_prohibited_way_ptr is selected. The MLD_prohibited_way_ptr is rotated by one way each cycle.

  The replacement state of the MLD is updated in the following manner:

  — When a data access hits in the MLD...

- Mark the accessed cache line RU.
- If all the lines in the set are now marked RU, mark them all NRU except for the line being accessed.

— When a cache line is filled to the MLD...

- If the fill is hinted to be marked NRU, it is.
- Otherwise, mark the filled line RU.
- If all the lines in the set are now marked RU, mark them all NRU except for the line being filled.

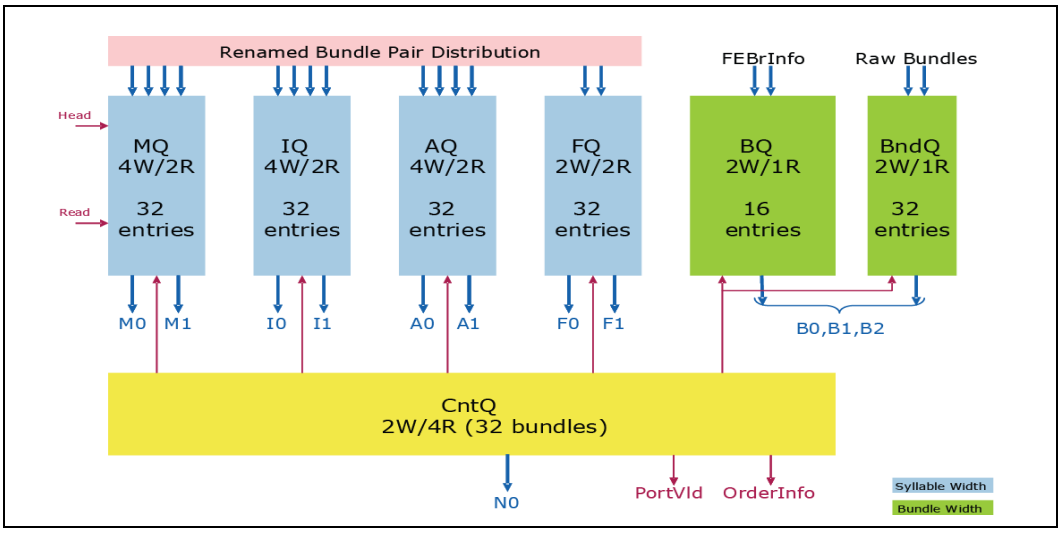
2.3.4.2.3 LLC

The LLC contains instructions and data and is shared among all 8 processor cores. It is inclusive of all of the lower level caches in all 8 processor cores. This means that an eviction from the LLC forces an eviction of the corresponding cache line in every lower level cache it is resident in. The LLC is arranged in 8 slices around a bi-directional ring shaped communication channel called the Ring. The entire LLC is a 32 MB, 32-way set associative cache. Each slice is a 4 MB 32-way subset of the entire LLC. The cache is indexed via a hash of PA[49:6] in order to better spread accesses around the cache physically to increase performance. The latency of the LLC is variable depending on distance between requesting core and responding LLC slice, frequency of the uncore relative to the core, and bandwith of accesses from all of the cores. However, 50 cycles would be a reasonable number to use for nominal load-to-use latency for a load hitting in the LLC. The LLC implements write back and allocate-on-write policies. The LLC can queue up 12 outstanding LLC misses / writebacks per slice. It implements a NRU replacement policy.

2.3.5 Instruction Execution

As shown in the unit level block diagram below, the IBL is comprised of several *syllable* and *bundle* wide queues. The queue storage elements and the associated pointers are duplicated for multi-threading in each queue while the write/read datapath and control logic are shared between the 2 threads. Two generic queue structures are used to build each of the queues in IBL. The next two sections describe a generic syllable queue structure and a generic bundle queue structure. Then the following sections list out the queue entry formats for each specific queue.

Figure 2-5. IBL Block Diagram



2.3.5.1 Instruction Dispersal

One of the important functions that IBL performs is that of Instruction Dispersal. Instruction dispersal refers to the process by which up to 2 bundles of instructions delivered by instruction fetch are issued to the twelve instruction execution pipelines in the BE. Instruction dispersal on the processor is split into two steps, namely *Instruction Insertion* and *Instruction Issue*. Instruction insertion is the process by which up to six instructions from the two bundles delivered by instruction fetch are distributed and inserted into the MQ, IQ, AQ, FQ or BQ each cycle in the FE pipeline. Instruction issue refers to the process of examining up to four bundles and issuing up to 12 instructions in the BE pipeline to the respective execution pipelines attached to each of the aforementioned IBL queues.

Frontend Instruction Insertion

The processor core implements the Template-based Queue Assignment (TQA) algorithm to accomplish instruction insertion. The TQA algorithm has two primary inputs which are the architectural bundle template field and instruction subtype generated from simple instruction decode. The essential idea of TQA is that for each architectural template, there is a default mapping for steering each instruction slot of the bundle to a particular queue. This default mapping is motivated by the desire to evenly distribute instructions across the BE execution pipelines. Since the BE provides 2M, 2I, 2A, 2F and 3B functional units, the TQA algorithm attempts to use 1 of each functional unit type for a given bundle. Furthermore, the algorithm assumes that within each instruction group and bundle, the more restrictive instruction subtype is earlier in the bundle slots. This assumption allows TQA to bias the second slot of the same type in a bundle to the AQ and thus achieve even distribution. The primary benefits of TQA are:

- predictable hardware dispersal behavior and software control since it is based on architectural templates
- independent of instruction address alignment and fetch patterns

As mentioned above, TQA requires instruction subtype generation. Simple instruction decode is performed in the FE pipeline in the FDC stage to classify each instruction slot of a bundle into 1 of 7 instruction subtypes. The 7 instruction subtypes are enumerated in the table below. For each instruction subtype, the table shows the bundle slot from which the instruction can originate, the legal queue assignment and a brief description of the instructions in that subtype. Legal queue assignment implies that a particular subtype must be inserted into a particular queue because instruction execution is only supported on the functional units attached to that queue.

Table 2-31. Instruction Subtypes for Dispersal

Subtype	Bundle Slot	Legal Queue Assignment	Description
M01	M-slot	MQ	M-format instructions except for chk.s, nop and some lfetch
MA	M-slot	MQ, AQ	A1-8 instructions and some M-format such as chk.s.m, nop.m, lfetch
IA	M-slot	AQ	A9-10 instructions
	I-slot	IQ, AQ	A1-10, chk.s.i instructions
I01	I-slot	IQ	I-format instructions except for chk.s.i, nop.i
NOP	any	NQ	Any nop instruction
FOP	F-slot	FQ	F-format instructions except for nop.f
BOP	B-slot	BQ	B-format instructions except for nop.b, brp

Given the architectural template and the instruction subtype information described above, a queue mapping table can be derived for queue assignment as per the TQA algorithm. In the queue mapping table shown below, each row of the table corresponds to a unique template type in the Itanium architecture. For convenience, the MLX template is shown in 2 rows (MLI and MLB) to explicitly indicate behavior when the X-slot is an I-unit or a B-unit instruction. The leftmost column in the table specifies the architectural template (in uppercase letters). The next six columns specify the rules of the TQA algorithm that are applied when a particular template is presented and combined with instruction subtype information. Further details on the rules column are:

- 2M/2I - applies when instruction subtype forces the queue assignment to 2 MQ or 2 IQ entries. Any given bundle can be labeled as a "2M", "2I" or neither. For example, in a MMI template, if the first 2 slots are M01 subtypes, then this bundle would be labeled as a 2M bundle.
- 2M Prev or 2I Prev - This denotes whether there were two MQ or two IQ assignments in the previous bundle. This is a simple history mechanism used to adjust queue assignments for the current bundle based on information from the previous bundle. The previous bundle information may be from a concurrently fetched bundle or fetched earlier. This previous bundle history is cleared if the previous bundle was terminated with a stop bit, or contained a stop bit, or if it was a branch bundle. There is one other case where the hardware will assert 2M Prev irrespective of history, and that is when hardware recognizes a 3rd bundle in an instruction group. This is an optimization to give compilation more scheduling options to allow more 3-bundle wide instruction groups to achieve 9-wide issue in the BE. For this purpose, an instruction group is considered terminated if a template stop bit is reached or a bundle with an actual branch instruction is reached. Lastly, there is a flaw to take note of. If an MI;;I template is used with an A9 or A10 format instruction in the M-slot, and a true I01 subtype in slot 1, the HW can count this as 2I Prev for the next bundle. This only occurs if the MI;;I bundle is at an even bundle address, and if there is no stop bit at the end of this bundle.
- 2nd M/I - This means that the 2nd M-slot or a 2nd I-slot is a M01 subtype or a I01 subtype respectively. So for MM* templates, if the 2nd M-slot is a M01 subtype, this bundle would be labeled a "2nd M" template. Likewise for *II templates, if the 2nd I-slot is a I01 subtype, this bundle would be labeled a "2nd I" template.
- Default - The default mapping is the queue assignment if none of the columns to the left apply and is set to achieve even distribution of instructions from a bundle across the BE execution pipelines.
- Fallback - This is the fall-back case and the rule applied if all of the other rules fail to create a legal queue assignment. As shown in the table, it is needed for only a couple of templates.

Each cell in the table specifies the queue assignment where the lower-case letters indicate the appropriate queue, that is, m=MQ, i=IQ, a=AQ, f=FQ and b=BQ. A blank cell implies that for the given row, that rule column does not apply. The queue mapping table is applied by starting with a bundle template, choosing an appropriate row of the table and applying the rules from left to right along the row skipping empty cells as follows:

- If current bundle is "2M" or a "2I" then use column 2M/2I or else
- If current bundle is ("2nd M/I" and "no mapping conflict") then use column 2nd M/I or else
- If previous bundle is ("2M" and "no mapping conflict") then use column 2M Prev or else

- If previous bundle is ("2I" and "no mapping conflict") then use column 2I Prev or else
- If "Default" column has ("no mapping conflict") then use Default or else
- Use the Fallback column.

A mapping conflict is when the queue assignment would conflict with the Legal Queue Assignment as defined in Table 2-31. Nops do not have mapping conflicts, do not count for 2M or 2I determination, and are not placed in a queue.

Table 2-32. Queue Mapping Table

Template	2M/2I	2nd M/I	2M Prev	2I Prev	Default	Fallback
MII	mii	mai	aia	maa	mia	
MI;I	mii		aai		mai	mii
MLI			aai		mai	
MLB			abb		mbb	
MMI	mmi	ami	aai	maa	mai	
M;MI	mmi				ami	mmi
MFI			afi	mfa	mfi	
MMF	mmf	amf	aaf		maf	
MIB			aib	mab	mib	
MBB			abb		mbb	
BBB					bbb	
MMB	mmb	amb	aab		mab	
MFB			afb		mfb	

Since the processor core IEU does not support execution of A9 and A10 format instructions on the M-port execution pipelines, these instructions must be dealt with as a special case in instruction dispersal. Instead of treating A9/A10 instructions in M-slots as a mapping conflict, it was found to be simpler to devise specific steering rules and they are presented in the table below. The queue assignments specified below do not attempt to optimize for performance but instead are designed to simplify hardware implementation. It is recommended to software that A9-A10 instructions be placed in I-slots in a bundle for optimal performance. This queue mapping table is read and interpreted just like the generic one above except that a A9-A10 instruction in an M-slot is called out as a capitol "A" in the second column from the left.

Table 2-33. Queue Mapping Table - A9/A10 in an M-slot Special Case (Sheet 1 of 2)

Template		2M/2I	2nd M/I	2M Prev	2I Prev	Default	Fallback
MII	AII					aai	aii
MI;I	AI;I					aai	aii
MLI	ALI					aai	
MLB	ALB					abb	
MMI	AMI		ami			aai	
	MAI			aai	maa	mai	
	AAI					aai	

Table 2-33. Queue Mapping Table - A9/A10 in an M-slot Special Case (Sheet 2 of 2)

Template		2M/2I	2nd M/I	2M Prev	2I Prev	Default	Fallback
M;MI	A;MI		ami			aai	
	M;AI			aai	maa	mai	
	A;AI					aai	
MFI	AFI					afi	
MMF	AMF		amf			aaf	
	MAF			aaf		maf	
	AAF					aaf	
MIB	AIB					aab	aib
MBB	ABB					abb	
BBB	BBB					bbb	
MMB	AMB		amb			aab	
	MAB			aab		mab	
	AAB					aab	
MFB	AFB					afb	

While writing instructions into the instruction queues, IBL also puts information into the control queue (CQ) about each bundle. The CQ entry is defined in ibldspq section above. For each slot in a bundle the CQ records which queue it went into (called here its "type") as a one-hot 6-bit field that is divided in two sections, one for Mq, Iq, Aq, or Fq, and one for Bq or Nq, where the Nq is the pseudo-queue for squashed nops. For each slot, the CQ entry also records the address or pointer in the corresponding queue in which the instruction will be stored. These pointers are referred to as the shadow write pointers, because they are copies of the pointers that live in the respective queue control blocks. Along with the "where" information, the CQ entry records per slot a stop bit, a port need for I0 or M0 (the asym bit), and a bit for a need for an IBD stall due to RSE behavior. For the whole bundle, the CQ entry also records whether it is an MLX template, and whether the bundle has an instruction-side fault associated with it.

Backend Instruction Issue

The processor instruction issue out of the queues into the backend pipeline, is in program order, and uses a 4 bundle wide issue window which is populated each cycle by the 4 oldest not yet issued bundles. In the oldest bundle, one or two instructions may have issued last cycle, so in each cycle, there is a marker for the starting syllable which defines the start of the issue window. The 4 bundle wide window can contain up to 12 instructions to issue, but there is an additional constraint on the 4th bundle of the window. Only nops and branches will be allowed to issue from this last bundle. Squashed nops are part of the issue window and issue in program order with the rest of the instructions.

The basic issue algorithm is to look in the issue window from oldest to youngest and find the first reason to stop issue. The reasons are termed architectural stop bits, implicit stops, asymmetric stops, oversubscription stops, and 4th bundle constraint stops. Implicit stops are placed at the end of each non-trivial branch bundle (containing a non-brp, non-nop.b, b-type instruction). Asymmetric stops are on the second M or I type instruction in the issue window if it needs to be on port M0 or I0. Oversubscription stops are on the third M, I, A, or F type instruction in the issue window, because there are only two read ports on each of these instruction queues. Constraint stops are placed on the first non-squashed nop, non-b-type instruction in the 4th bundle.

All of the instructions from the start syllable to the first stop reason will be allowed to issue and the corresponding execution units will receive a notice of instruction validity. Squashed nops will be executed as single bit instructions by the exceptions unit (EPN). There can be up to 12 squashed nops issued in a cycle. For the next cycle, the individual instruction queues will advance their read point by how many of their instructions were issued, and new, un-issued bundles will slide into the issue window while the just issued ones drop out.

Instruction Dispersal Examples

The figures below show how the TQA algorithm described above works on different code examples. Figure 2-6 shows how a 2 bundle wide instruction group would get inserted into the IBQ and issued to the various execution pipelines in the BE. Because the first bundle is a "2I" bundle, the second chooses the "2I Prev" column rule to disperse the MMI template, sending slots 1 and 2 to the AQ.

Figure 2-6. Dispersal Example 1 - 2-bundle group

Bundles Presented	Template	Instruction Subtype	Queue Mapping Rule	Queue Assignment	BE Execution Pipeline	Notes
{.mii	MII		2M/2I			
ld8		M01		mQ	M0	
shl		I01		iQ	I0	
czx		I01		iQ	I1	
}						
{.mmi	MMI		2I Prev			
st4		M01		mQ	M1	
add		MA		aQ	A0	6-wide issue of 2 bundles; 2M, 2I, 2A
sub		IA		aQ	A1	
};;						

Figure 2-7 shows a 3 bundle wide instruction group and how it might disperse. In this case, the first 2 bundles follow the "Default" column rules as software has spread out execution resources evenly. The 3rd bundle is recognized by hardware and labeled as a "2M Prev" and it steers slot0 or the M-slot instruction to the AQ and thus achieve 9 instruction wide issue in the BE.

Figure 2-7. Dispersal Example 2 - 3-bundle group

Bundles Presented	Template	Instruction Subtype	Queue Mapping Rule	Queue Assignment	BE Execution Pipeline	Notes
{.mmi	MMI		Default			
ld8		M01		mQ	M0	
xor		MA		aQ	A0	
czx		I01		iQ	I0	
}						
{.mfi	MFI		Default			
st4		M01		mQ	M1	
fma		FOP		fQ	F0	
sub		IA		iQ	I1	
}						
{.mfb	MFB		2M Prev			
andcm		MA		aQ	A1	
fsub		FOP		fQ	F1	9-wide issue of 3 bundles; 2M, 2I, 2A, 2F, 1B
br		BOP		bQ	B2	
};;						

Backend Instruction Issue Implementation

On the output side of the queues, each cycle, the control queue provides the four oldest bundle entries to the iblvalids block. The iblvalids block controls which instructions are issued, and tells the other queues when to advance to new entries.

The HW implementation has stop reasons that can come into play in addition to those described above. There is an issue width control (iwc) used after replays and for power control (for throttling dispersal) which defines the youngest syllable in the four bundles that is allowed to issue. There are DAF controls for single issue per instruction queue, and there is a DAF-like control for the floating point register file step load control.

2.3.5.2 Issue Group Scheduling

For certain read-after-write hazards that are deemed to be performance critical, the issue logic in IBL will inject bubbles at the IBD stage to avoid these hazards from occurring in the pipeline. This is referred to as Issue Group Scheduling. Two classes of such read-after-write hazards are covered by issue group scheduling: 1) FR-FR hazards or FP register hazards and 2) IntLd-MemAddr hazards, or an integer load instruction target register being consumed as a memory address register.

The FR-FR hazards were deemed to be performance critical due to the FPU pipeline latency increase to 6 cycles on Intel Itanium Processor 9500 Series from 4 cycles for “pre” Intel Itanium Processor 9500 Series. Without issue group scheduling, legacy code scheduled at 4 cycle separation would have triggered EXE replays resulting in a 4 cycle penalty for every such hazard.

All FR writers on the F-pipes are fixed latency at 6 cycles, that is, they can bypass a result in the WB4 or FP6 pipestage to an FR reader in the REG stage. The FR-FR issue group scheduling logic then needs to ensure that minimum 6 cycle separation between an FR writer and FR reader. The separation distance is achieved by a combination of EXE replay and IBD stall. For FR readers separated from FR writers by 1 or 2 cycles, the issue stall logic can be minimized by first letting an EXE replay occur without any additional penalty. This is shown in the first 2 pipeline tables below. For FR readers separated from FR writers by 3, 4 or 5 cycles, the scheduling logic inserts the

appropriate number of IBD stall cycles. FR writers on F-pipes are considered to be all instructions in the class of fp-arith and fp-non-arith as defined in the *Intel® Itanium® Architecture Software Developer's Manual*. FR readers would be fp-arith, fp-non-arith and pr-writers-fp classes on F-pipes and mem-writers-fp, chk.s and getf on M-pipes.

Table 2-34. Case1 - 1 cycle separation for FR read after write

CYCLE	IBD	DEC	REG	EXE (FP1)	DET (FP2)	WRB (FP3)	WB2 (FP4)	WB3 (FP5)	WB4 (FP6)	WB5 (FP7)	WB6 (FP8)	Notes
101	Rd	Wr										
102		Rd	Wr									
103			Rd	Wr								
104				Rd	Wr							EXE Replay
105	Rd					Wr						IBD Stall
106	Rd						Wr					
107		Rd						Wr				
108			Rd						Wr			Bypass from WB4
109				Rd						Wr		
110					Rd						Wr	FRF updated

Table 2-35. Case2 - 2 cycle separation for FR read after write

CYCLE	IBD	DEC	REG	EXE (FP1)	DET (FP2)	WRB (FP3)	WB2 (FP4)	WB3 (FP5)	WB4 (FP6)	WB5 (FP7)	WB6 (FP8)	Notes
101	Rd		Wr									
102		Rd		Wr								
103			Rd		Wr							
104				Rd		Wr						EXE Replay
105	Rd						Wr					
106		Rd						Wr				
107			Rd						Wr			Bypass from WB4
108				Rd						Wr		
109					Rd						Wr	FRF updated
110						Rd						

Table 2-36. Case3 - 3 cycle Separation for FR Read After Write (Sheet 1 of 2)

CYCLE	IBD	DEC	REG	EXE (FP1)	DET (FP2)	WRB (FP3)	WB2 (FP4)	WB3 (FP5)	WB4 (FP6)	WB5 (FP7)	WB6 (FP8)	Notes
101	Rd			Wr								IBD Stall
102	Rd				Wr							IBD Stall
103	Rd					Wr						IBD Stall

Table 2-36. Case3 - 3 cycle Separation for FR Read After Write (Sheet 2 of 2)

CYCLE	IBD	DEC	REG	EXE (FP1)	DET (FP2)	WRB (FP3)	WB2 (FP4)	WB3 (FP5)	WB4 (FP6)	WB5 (FP7)	WB6 (FP8)	Notes
104	Rd						Wr					
105		Rd						Wr				
106			Rd						Wr			Bypass from WB4
107				Rd						Wr		
108					Rd						Wr	FRF updated
109						Rd						
110							Rd					

2.3.5.3 Long-latency FR Hazards

Floating-point load operations are considered to be long latency operations and also exhibit variable latency depending on whether they hit in the MLD, LLC or memory. IBL includes a FR scoreboard to track outstanding updates to the target FR of such operations. Then, whenever a subsequent operation consumes an invalid FR, an EXE replay is triggered followed by an IBD stall if necessary to align the returning data for this FR from the memory hierarchy. A consuming operation in this case can be either a read or a write of the outstanding FR. Subsequent writers must be serialized in this fashion to ensure in-order architectural state updates. The IBL FR scoreboard pending bits are set by mem-readers-fp and setf instruction classes. The consuming operations include mem-readers-fp, mem-writers-fp, setf, getf on M-pipes and fp-arith, fp-non-arith and pr-writers-fp instruction classes on F-pipes.

The pipeline table below shows an example of such a hazard. The table shows a FP load operation (ldf) executing down the main pipeline. The MLDPipe column shows ldf as it progresses in the MLD pipeline. The MLD drives the return regid in the L1M stage and drives the data for the return in the L1C pipe stage. The data is then available for bypass in the L1X pipe stage in the FPU to a consumer operation in the REG stage. The IBL stall logic uses the return regid from the L1M stage to release the IBD stall just-in-time to catch the return data from the L1X pipe stage.

Table 2-37. Long-latency FR Hazards (Sheet 1 of 2)

CYCLE	IBD	DEC	REG	EXE	DET	WRB	WB2	MLD Pipe	Notes
101	ldf								
102		ldf							
103	use		ldf						
104		use		ldf					
105			use		ldf			L1A	
106				use		ldf		L1T	EXE Replay
107	use						ldf	L1H	IBD Stall
108	use							L1M	IBD Stall
109	use							L1D	IBD Stall
110	use							L1C	

Table 2-37. Long-latency FR Hazards (Sheet 2 of 2)

CYCLE	IBD	DEC	REG	EXE	DET	WRB	WB2	MLD Pipe	Notes
111		use						L1W	
112			use					L1X	Bypass MLDRTN
113				use					
114					use				

2.3.5.4 IBL Miscellaneous Replays

IBL is also responsible for satisfying various AR, CR and other serialization pipeline hazards using the EXE replay mechanism. The following table lists such producer-consumer hazards that the IBL logic detects and requests an EXE replay on the appropriate execution pipeline.

Table 2-38. IBL Misc EXE Replays

Producer / Consumer	PsrUm	HpwBus	MvtAny	SyncOut	XpnWr	PFSWr	RSCWr	IdRNAT	RSCWrexe	BSPSWrexe	LCECWr	BRWr	FpPnd	fsetcPnd	FPSRWr
VldOp	All														
RelType		M0 M1													
SrlzOp			M0 B2	M0 B2											
ThashOp			M0												
mvfXpnCr					M0										
allocOp						M0									
mvfBSPS							M0			M0					
mvfBSP										M0					
mvtBSPS							M0								
mvRNAT							M0	M0							
mvfRSC									M0						
mvfLCEC											I0				
brCall											B0 B1 B2				
mvtBR												I1			
mvfFPSR													M0	M0	M0
VldFpOp															F0 F1
VldFpArith														F0 F1	
FpSfOp													F0 F1	F0 F1	

2.4 Intel Itanium Processor 9500 Series Multi-Threading

The processor expands support for Hyper-Threading or HT Technology over previous Itanium processors with key additional hardware to deliver higher utilization of processor core resources and thus higher performance and throughput. As in previous Itanium processors, the processor core duplicates all architectural state and some micro-architectural state to create 2 logical processors in each physical core. The processor introduces the concept of 2 thread domains in the core by exploiting the decoupled pipeline separated by the Instruction Buffer that allows instruction fetch and instruction execution to operate independently. The Front End (FE) thread domain spans the FE of the pipeline, that is, from IPG to IBD while the Back End (BE) thread domain spans the BE of the pipeline, that is, from IBD to WB2. With independent thread domains and a fully duplicated Instruction Buffer, the FE can perform instruction fetch for either thread regardless of which thread the BE is executing. The processor implements Switch-On-Event MT or SOEMT, as in previous Itanium processors, in each thread domain. That is, only 1 thread can be active at any time in the FE pipeline or the BE pipeline but the FE and the BE may be fetching and executing on the same thread or a different thread in any given cycle respectively. The FE thread domain has its own set of thread switch events and a switch controller. The FE thread domain switch controller attempts to maximize instruction fetch bandwidth utilization with the goal of keeping the Instruction Buffer full for both threads. The BE thread domain also has its own set of thread switch events and a separate switch controller. The BE thread domain switch controller attempts to maximize utilization of execution resources by switching between the two thread's Instruction Buffer whenever a thread in the BE incurs a long latency event. The next two sections present more details on the MT implementation for the FE thread domain and the BE thread domain.

2.4.1 Frontend MT Introduction

On the processor, the frontend and backend pipelines are on separate threading domains. The frontend has the choice of filling two instruction buffers, one per thread. This enables greater frontend efficiency by enabling it to hide long latency events, and, more importantly, enables the backend to have a choice of instruction buffers.

The frontend is SoEMT (Switch on Event MT) which means only one thread will be active in the frontend of the pipeline at a time. The frontend will switch threads and begin inserting instructions into the other thread's instruction buffer based on a series of events. There are two types of switch events: implicit and explicit. The implicit switch algorithm utilizes an urgency based urgency system that monitors the relative urgency between the two threads in deciding when to switch. In general, two factors influence the urgency of a thread. The first is determined by whether a thread is believed to have instructions ready to deliver to the instruction buffer, and the second is whether the backend is requesting immediate attention on a particular thread. When one thread has a higher urgency than the other, a thread switch occurs. The purpose of the urgency system is to determine which thread has useful work to do to enable more efficient operation. In addition, the urgency system can be biased to follow the backend's active thread except when there is no useful work to do. Explicit switch events force a thread switch irrespective of the background thread's urgency.

There is no additional user-visible state over what exists on Intel Itanium Processor 9300 Series to enable frontend multithreading. Microarchitecturally, speculative state is kept for both threads and the frontend operates on this speculative state until the backend refreshes the appropriate thread with architected state.

2.4.2 BE Thread Domain

The BE thread domain spans from IBD to WB2 in the main core pipeline. Switching threads in the BE involves 2 primary steps: 1) stopping execution of the current thread and 2) switching the Instruction Buffer to the new thread and launching instruction out of the IBQ into the BE. A thread switch begins with some switch event arriving and being processed through a relative comparison of the two threads run-state. The thread run-state is comprised of a thread priority state and a thread execution state. The thread run-state is comparable to the concept of thread urgency on previous Itanium processors. If the comparison of the thread run-state determines that a thread switch would be useful, a thread switch request is generated and attributed to a switch event. The switch controller in the BE then will request a IBD stage issue stall and wait for the BE pipeline to drain from DEC to WB2 including accounting for any replays and/or flushes occurring on the current thread. Once the main pipeline is idle, the active thread in the BE is changed and after a few cycles valid instructions are launched into the pipeline from the IBQ of the new thread, thus completing a thread switch operation.

2.4.2.1 Thread Priority State

As mentioned above, the concept of urgency of execution for a given thread in “pre” Intel Itanium Processor 9500 Series has been replaced with two separate thread states. One of these is the concept of thread priority and is mostly controlled by software. The hint @priority and hint @pause instructions are two low overhead mechanisms for software to communicate to hardware the urgency of execution for a given thread. Three priority states are defined as follows:

- Nominal
  - This is the default priority state for a thread.
- High
  - This priority state is entered by retiring a hint @priority instruction which can be executed on any M, I or F ports on the processor.
  - Retiring a hint @priority instruction also has the side-effect of reloading the FG thread timeslice with the High Timeout Value (HiTOV) and then it only counts down during "Unstalled" thread execution cycles counting down every cycle in the high state. The expiry of the timeslice leads to termination of High priority. If a High priority thread is switched out for some reason, its remaining HiTOV timeslice is preserved and restored when it is brought to the foreground again.
  - The High priority state of a thread can also be terminated explicitly by software using a hint @pause instruction. Executing a hint @pause at High priority transitions the thread priority down to Nom.
- Low
  - This priority state is entered by retiring a hint @pause instruction at Nom priority and can be executed on any M, I or F ports on the processor.
  - Entering the Low priority state due to hint@pause also has the side-effect of reloading the background (BG) thread timeslice with Low Timeout Value (LoTOV). The expiry of the LoTOV timeslice is then used to terminate Low priority and return to thread priority to Nom. Unlike for high priority, this timeslice counter decrements every cycle.
  - This priority state is also entered when a thread has entered the Low Power state due to the execution of a PAL_HALT_LIGHT call. For this case Low Priority is only exited due to an external interrupt.

This notion of thread priority is provided as a method for software to communicate the future resource requirements to hardware multi-threaded processors to optimize performance. Three specific scenarios under which it is especially important for software to communicate its resource requirements through thread priority are:

- Idle Loops - By placing a hint @pause in idle loops, software communicates to hardware it has no work to perform at present. The hardware in this case acts upon the hint @pause by placing the thread in Low priority and switching away from it until the LoTOV timeslice expiration, interrupt, or other switch event. This allows the other thread to more fully utilize the shared hardware resources for a period of time.
- Wait Loops - By placing a hint @pause and allocating a memory address into the ALAT, software communicates to hardware that it is willing yield resources temporarily until the data at the address is the target of a store on any other processor. The hardware in this case responds to the hint @pause by placing the thread in Low priority and switching away to the other thread. When the allocated ALAT entry is invalidated due to the store, the thread priority is restored to Nom and a thread switch brings back the waiting thread to the FG. If an ALAT invalidation doesn't arrive within a LoTOV timeslice, the timeslice expiration accomplishes the same change in priority and thread switch.
  - — Example code
    - ldxxxx
- Critical Regions - If software is about to enter a critical section, it can communicate to hardware such critical need for resources by preceding the critical section with a hint @priority. The hardware responds by raising the thread priority to High and biases the thread switch behavior to give more resources to this critical thread for a short duration. Software must be aware of a couple of corner cases with respect to using hint @priority instructions:
  - — hint @priority followed by hint @pause in the same issue group: hardware will ignore the hint @priority and the hint @pause will have its normal effects as explained above.
  - — hint @pause followed by hint @priority: the hint @pause will cause a switch out event and when this thread begins execution again the hint @priority will be executed. This works as expected since a hint @pause always causes a WB2 replay on the instruction following the hint @pause to ensure precise thread switch points in the instruction sequence.

  The recommended way to terminate High priority is at the end of the critical region perform the following instructions: hint@priority ;; hint@pause. This minimizes the chance of a timeout while in high priority that would cause the hint@pause to take the thread to low priority instead of nominal priority.

2.4.2.2 Thread Execution State

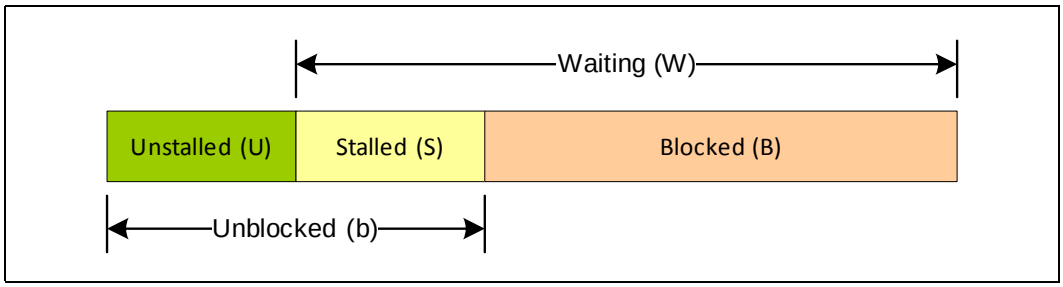
The second piece of thread state that replaces the concept of urgency of execution for a given thread in "pre" Intel Itanium Processor 9500 Series Differences is the thread execution state. Unlike thread priority state, the thread execution is determined and managed completely by hardware. The thread execution state attempts to represent the ability of a thread to make execution progress and utilize shared hardware resources. Thus, there are three main components that determine comprise thread execution state:

- D-cache data unavailable - a use causes an EXE or DET replay followed by a IBD use-stall.
- I-cache fetch unavailable - the FE is unable to supply instructions, represented by an empty IBQ.

- DTLB miss - a memory operation needs a translation that is unavailable in the DTB causing a WB2 replay followed by a IBD hpw-stall.

Each of the above components represent potentially long latency events and hence are candidates for possible thread switch out events (MLDuse, IBQEmpty, HPWmiss respectively). The thread execution status resulting from these events is shown below in the figure:

Figure 2-8. Thread Execution Status



- Unstalled (U): a thread is considered unstalled until one of these long latency events is hit.
- Stalled (S): a thread transitions to this state when one of the long potential latency events is hit.
- Blocked (B): when a thread's stalled state persists to the point where it is likely to remain stalled for a long time, it transitions to this Blocked state for that latency event. It is also upon reaching this Blocked state that a switch out event of type MLDuse, IBQEmpty or HPWmiss is generated. The Blocked state is cancelled at every thread switch for the new FG thread and can be re-engaged if Stalled condition persists and thus generate another switch OUT event. It is also possible that both threads in a core can be in the Blocked state simultaneously.
- Waiting (W): the Waiting state of a thread is simply that the thread is not Unstalled.
- Unblocked (b): Likewise, the Unblocked state is simply that the thread is not Blocked.

2.4.2.3 BE Thread Switch Events

Since, a thread switch operation is no longer a full pipeline flush including a re-fetch of instructions, much finer-grain Switch-On-Event MT (SOEMT) can be implemented in the BE to overlap even more idle cycles of a given thread with execution of the other thread. The BE switch events are shown in the table below.

Table 2-39. BE Thread Switch Events (Sheet 1 of 2)

Event Name	Event Type	Event Description	Switch Type
MLDUse	Blocking	Consumption of a MLD miss; Typically, occurs after an EXE replay to an IBD stall on a use. If the stall lasts for more than typical MLD latency, a MLDUse switch event is generated.	OUT
HPWMiss		Data TLB miss; Typically occurs after a WB2 replay to an IBD stall on a blocking hardware page walk. If missing translation isn't returned within the pre-programmed HPW event delay, a HPWMiss switch event is generated.	OUT
IBQEmpty		Instruction Buffer Empty; Typically occurs after a WRB BruFlush on a mispredicted branch. If the FE doesn't deliver instructions within the pre-programmed IBQevent delay, a IBQEmpty event is generated.	OUT

Table 2-39. BE Thread Switch Events (Sheet 2 of 2)

Event Name	Event Type	Event Description	Switch Type
MLDRtn	Unstall	Data return corresponding to MLDUse	IN
HPWInsrt		Translation from the VHPT is ready for insertion corresponding to HPWMiss	IN
IBQFill		Instruction Buffer Filled or FE is ready to fill corresponding to IBQEmpty	IN
FG TimeSlice	Timeslice	Timeslice for the current High Priority FG thread expires thus returning the FG thread priority to Nom.	N/A
BG TimeSlice		Timeslice for the current BG thread expires thus usually requiring the thread to be switched in.	IN
@Priority	Hint	Elevate FG thread priority to Hi. Does not cause a thread switch event	N/A
@Pause		hint @pause at NOM priority causes the current FG thread to be switched out and moved to Low priority.	OUT
ALATInval	ALAT	ALAT Invalidation; An ALAT entry is invalidated on a hint @paused thread while it is in back-ground.	IN
LPMode	Low Power	LPEnter: A halt.lp called by PAL_HALT_LIGHT retires on the foreground thread and the background thread is not in LP	OUT
		LPExit: A wakeup event (external interrupt) arrives on the background thread causing it to exit LP (	IN
Fairness	Others	Unfairness Meter has crossed into Region3 and a thread switch is required to bring the victim thread into the fore-ground.	IN

The thread-switch overhead when pipeline is busy is about 13 cycles, and when the pipe is empty is about 7 cycles.

MLDUse Event

An EXE or DET replay due to a unavailable register begins the sequence of events leading to a MLDUse switch event. The consuming instruction is replayed to a IBD use-stall and at this point the event delay counter is triggered to count down for a fixed number of cycles. If the use-stall is resolved prior to the event delay counter expiration, there is no thread switch to initiate and the current thread continues execution. If the use-stall is still asserted at the delay counter expiration, a MLDuse switch event is generated and the current thread is now considered to be in the Blocked state for MLD. The IBL block is now also enabled to watch for MLD returns to the regid that was unavailable and thus a MLDRtn can possibly be generated when the current thread is in the BG.

HPWMiss Event

A WB2 replay due to a blocking DTLB miss begins the sequence of events leading to a HPWMiss switch event. The consuming instruction is replayed to a IBD hpw-stall and at this point the event delay counter is triggered to count down for a fixed number of cycles. If the hpw-stall is resolved prior to the event delay counter expiration, there is no thread switch to initiate and the current thread continues execution. If the hpw-stall is still asserted at the delay counter expiration, a HPWMiss switch event is generated and the current thread is now considered to be in the Blocked state for HPW. The HPW logic is now also enabled to watch for MLD returns for the VHPT load operation and thus a HPWInsrt switch event can possibly be generated when the current thread is in the BG.

IBQEmpty Event

When the BE FG thread IBQ is empty, the event delay counter is triggered to count down for a fixed number of cycles. Typically, this will occur when the BE hits a mispredicted branch. A BRU flush of the pipeline occurs in WRB leading to an empty IBQ. If the FE is unable to supply instructions within a fixed number of cycles, a IBQEmpty thread switch event is initiated. At this point the current thread is Blocked for IBQ and when the FE fills the IBQ or indicates that it is ready to fetch for this thread, an IBQFill switch event can be generated when the current thread is in the BG.

Switch IN Events

Switch IN events are those that are generated by the BG thread as a request to become the FG thread in the BE. When a thread switch request generated on behalf of the BG thread is received and the switch controller accepts it, an IBD mt-stall is generated to stop instruction issue on the current thread. Then, it would wait for the pipeline to drain from DEC to WB2 to account for any flushes/replays before initiating the thread switch. Examples for such events are MLDRtn, HPWInsrt, IBQFill, ALATInvalidate, BG timeslice etc.

2.4.2.4 Thread Switch Decision Diagrams

The following tables define how thread priority and thread execution state described above affect the switch controller behavior for each major type of switch event. Each table is organized to show the FG thread priority/execution state combination in columns and the BG thread priority/execution state combination in rows.

Here is a key to reading the tables:

Table 2-40. Thread Switch Transition Key

Symbol	Description
FG	foreground thread
BG	background thread
H	High priority
N	Nominal priority
L	Low priority and non-LP
Lp	Low priority and LP
U	Unstalled execution state, i.e. ~(S || B)
W	Waiting execution state, i.e. (S || B)
S	Stalled execution state i.e. ~(U || B)
B	Blocked execution state i.e. ~(U || S)
b	Not Blocked i.e. (U || S)
a b c	Each table entry cell has three characters. a=new FG priority, b= new BG priority, c=thread switch or not
	For c only, "-" implies no change
	For c only, "T" implies that a thread switch is initiated.
X	Invalid state or don't care due to event assumptions or impossible state combinations

The Blocking event transitions are shown below and apply to MLDUse, HPWMiss and IBQEmpty switch out events.

Table 2-41. Blocking Event Transitions

		FG States		
		HB	NB	LB
BG States	HU	- - T	- - T	X
	HW	- - -	- - -	X
	NU	- - T	- - T	X
	NW	- - -	- - -	X
	LU	- - -	- - -	X
	LW	- - -	- - -	X
	Lp	- - -	- - -	- - -

The Unstall event transitions are shown below and apply to MLDRtn, HPWInsrt and IBQFill switch in events.

Table 2-42. Unstall Event Transitions

		FG States					
		Hb	HB	Nb	NB	Lb	LB
BG States	HU	- - -	- - T	- - T	- - T	X	X
	NU	X	- - T	- - -	- - T	X	X
	LU	- - -	- - -	- - -	- - -	- - -	- - -

Table 2-43. FG Timeslice Expiration

		FG States
		HU
BG States	HU	N - T
	HW	N - -
	NU	N - -
	NW	N - -
	LU	N - -
	LW	N - -
	Lp	N - -

Table 2-44. BG Timeslice Expiration

		FG States			
		HU	HW	NU	NW
BG States	HU	- - -	- - T	- - T	- - T
	HW	- - -	- - T	- - T	- - T
	NU	- - -	- - T	- - T	- - T
	NW	- - -	- - T	- - T	- - T
	LU	- N -	- N T	- N T	- N T
	LW	- N -	- N T	- N T	- N T

Table 2-45. Hint @pause

		FG States	
		HU	NU
BG States	HU	N - T	X
	HW	N - -	L - T
	NU	N - -	L - T
	NW	N - -	L - T
	LU	N - -	L N T
	LW	N - -	L N T
	Lp	N - -	- - -

Table 2-46. Hint @priority

		FG States		
		HU	NU	LU
BG States	HU	- - -	H - -	H - -
	HW	- - -	H - -	H - -
	NU	- - -	H - -	H - -
	NW	- - -	H - -	H - -
	LU	- - -	H - -	H - -
	LW	- - -	H - -	H - -
	Lp	- - -	H - -	H - -

Each hint@priority execution causes a reload of FG timeslice with HiTOV.

Table 2-47. ALAT Invalidation Event In Low Priority

		FG States			
		Hb	HB	Nb	NB
BG States	NU	- - -	- - T	- - T	- - T
	NW	- - -	- - -	- - -	- - -
	LU	- N -	- N T	- N T	- N T
	LW	- N -	- N -	- N -	- N -

BG States NU and NW rows come about if BG was L when the ALATinv occurs, but a switch did not happen.

Table 2-48. LP Halt Event

		FG States	
		HU	NU
BG States	HU	Lp - T	X
	HW	Lp - T	Lp - T
	NU	Lp - T	Lp - T
	NW	Lp - T	Lp - T
	LU	Lp N T	Lp N T
	LW	Lp N T	Lp N T
	Lp	Lp - -	Lp - -

Table 2-49. External Interrupt Event on FG Thread

		FG States						
		Hb	HB	Nb	NB	Lb	LB	Lp
BG States	HU	- - -	X	X	X	X	X	X
	HW	- - -	- - -	- - -	- - -	X	X	X
	NU	- - -	X	- - -	X	X	X	X
	NW	- - -	- - -	- - -	- - -	X	X	X
	LU	- - -	- - -	- - -	- - -	X	X	X
	LW	- - -	- - -	- - -	- - -	X	X	X
	Lp	- - -	- - -	- - -	- - -	- - -	- - -	N - -

Table 2-50. External Interrupt Event on BG Thread

		FG States						
		Hb	HB	Nb	NB	Lb	LB	Lp
BG States	HU	- - -	X	X	X	X	X	X
	HW	- - -	- - -	- - -	- - -	X	X	X
	NU	- - -	X	- - -	X	X	X	X
	NW	- - -	- - -	- - -	- - -	X	X	X
	LU	- - -	- - -	- - -	- - -	X	X	X
	LW	- - -	- - -	- - -	- - -	X	X	X
	Lp	- N T	- N T	- N T	- N T	- N T	- N T	- N T

2.4.2.5 Unfairness Meter

One of the challenges hardware multi-threading implementations have faced is execution fairness between the 2 threads running on the same physical core. The idea of fairness is further complicated by the desire to deliver maximum throughput since a high throughput thread will almost always imply unfairness in execution for its sibling thread. The processor introduces a new concept of an *Unfairness Meter* to track whether a particular thread is being victimized and upon detection of such unfairness to take remedial action.

The Unfairness Meter is designed to measure and track "unfairness" rather than some metric such as IPC that tracks fairness between the two threads in a core. The essential idea is that of a single up/down counter that ticks based on a thread being deemed ready-to-execute but yet unable to execute because either it is in the back-ground in the BE or is unable to execute despite being in the fore-ground in the BE. As shown in the figure below, the unfairness meter defines 4 regions of operation and associated threshold levels for the up/down counter. Region0 or "Green Region" is considered to be normal operation i.e. the meter is allowed to drift between +L1 and –L1 as the 2 threads execute and attempt to deliver maximal throughput without any remedial action being taken. So for example, if T1 is in the back-ground in the BE and is "ready-to-execute", the unfairness meter will start ticking down from 0 toward –L1. A negative meter value indicates that T1 is the "losing" or "victim" thread while a positive meter value indicates that T0 is the "losing" or "victim" thread.

The unfairness meter ticks in the appropriate direction based on which thread is losing its fair share of execution. The conditions under which the meter will tick are:

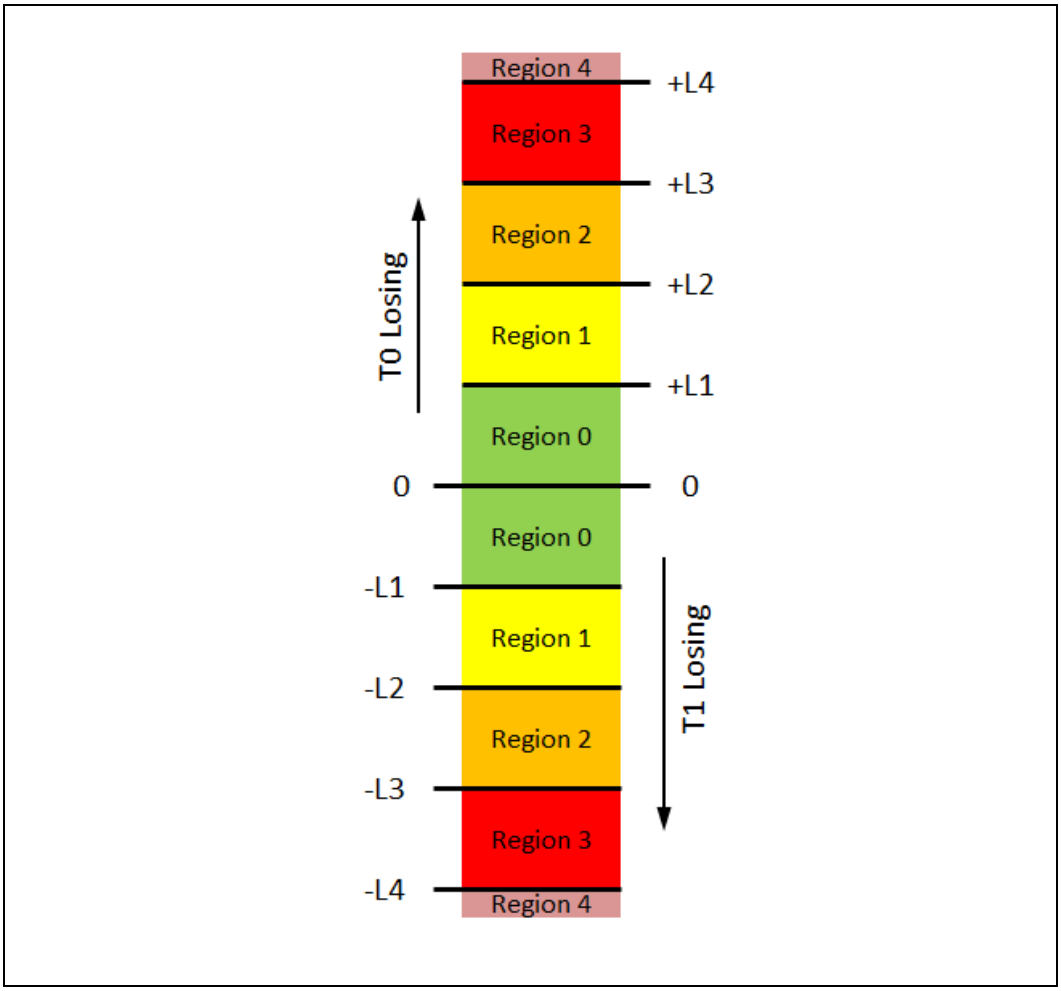
- A thread is in the back-ground in the BE and is not at Low thread priority and is Unstalled.
- A thread is stalled because of the other thread occupying a significant portion of MLD queue resources.

However, a victim thread credits back accumulated unfairness when it is active, except when condition 2 is true.

No unfairness counting happens during a thread switch.

The actual number of ticks the unfairness meter is given is determined by a per-thread pre-scaler value defined in implementation specific registers. This notion of a pre-scaler on the unfairness meter ticks allows for creating a policy of deliberately biased unfairness in favor of one thread by the operating systems. High Priority and Exclusive High Priority settings in PAL_SET_HW_POLICY provide 4:1 pre-scalar ratios, while other performance and fairness personalities provide 1:1 pre-scalar ratio.

Figure 2-9. Unfairness Meter



Remedial actions are associated with each region of the meter and are designed to be progressively worse in terms of maintaining throughput of the aggressor thread. These actions or policies come into effect as the meter crosses over a threshold into a new region and stay in effect until the meter actually returns to 0. A given policy stays in effect starting at the threshold crossover until the meter reaches 0 again. The table below defines each of the meter regions and the policies or actions taken as different regions are entered.

Table 2-51. Unfairness Meter Regions / Actions

Region	Description	Policy / Action
Region0 or Green	Normal operating region. Threads are executing fairly	No action necessary; Maintain default switching behavior.
Region1 or Yellow	Unfairness in play as meter has crossed +L1 or –L1.	Make BG timeslices asymmetric; 1/2x on Victim, 2x on Aggressor as long as thread priority is Nom or High. The FG slice of a High priority thread and the BG slice of a Low priority thread are not scaled to maintain deterministic behavior for hint @pause and @priority from a software view.
Region2 or Orange	Unfairness in play as meter has crossed +L2 or –L2.	Victim thread becomes fore-ground whenever it is Unblocked; All yellow region policy actions also in effect.
Region3 or Red	Unfairness in play as meter has crossed +L3 or –L3.	Bring Victim thread to fore-ground whenever it is not at Low priority and camp on it until fairness restored or victim assumes low priority. The FE thread domain is also locked to the BE thread in this region.
Region 4	Unfairness in play as meter has crossed +L4 or –L4.	Camp on victim until fairness is restored

2.4.2.6 Thread Forward Progress

Thread forward progress responsibility lies with the BE thread domain on the processor. The BE thread forward progress mechanism is essentially the same as in previous Itanium processors. The BE counts and allows a set number of BE thread switches to a given thread with no instruction retirement before requiring forward progress. A thread is considered to be making forward progress if any of the following events are occurring:

- Retirement of an instruction including a predicated-off instruction
- Completion of an RSE fill/spill
- Delivery of an interruption — may be optionally excluded

Once the threshold of the pre-programmed number of thread switches is exceeded, the thread forward progress mode is engaged. The thread forward progress mode is comprised of two primary actions:

- Camp on current thread in the BE until an instruction retires or interruption occurs.
- Request the FE thread switch controller to lock to the BE thread and disable any further FE thread switch events.

An L2/L3/L4 aggressor will not be allowed to BE forward progress camp until fairness is restored.

This BE thread forward progress mechanism, once engaged, is deemed sufficient to ensure that instruction fetch progress in the FE thread domain will occur due to the FE eventually following the BE thread.

Full Time Slice Mode allows a thread that exits forward progress lock to remain in the FG for up to a time-slice after it begins to retire instructions. When this feature is enabled, a thread that exits forward progress mode by retiring an instruction can be switched to the BG only for the following reasons; FG Blocking, FG Time slice expiration, BG Time slice expiration, Hint@pause, Halt.lp, Fairness Meter induced switch events, External interrupt to BG thread.

2.4.2.7 Switch-Delay

Switch-Delay limits the fraction of time spent switching between threads. Thread switch events that occur during a certain duration (or window) of time after the prior thread switch receive special treatment.

- When unfairness is Green/Yellow
  - — Switch-in event (Unstall and ALATinv) induced switches are remembered and acted upon after the switch-wait window expires
  - — Switch-out events are unaffected by the window
- When unfairness is Orange/Red
  - — Blocking induced switches are remembered and acted upon after the switch-wait window expires
  - — Hint@pause is ignored (has no effect) within the switch-wait window
  - — Switch-in events are unaffected by the window

FG and BG timeouts are not affected by the wait window.

2.5 Intel® Virtualization Technology

The Intel Itanium processor 9500 series is the fourth Intel® Itanium® processor to implement Intel® Virtualization Technology, described in the *Intel® Itanium® Architecture Software Developer's Manual*.

The Intel Itanium processor 9500 series provides enhanced Intel® Virtualization Technology (Intel® VT-i3) and Intel® Virtualization Technology for directed I/O (Intel® VT-d).

The Intel Itanium processor 9500 series core supports all of the baseline virtualization support provided in the Intel® Itanium® processors 9300 series. In addition, it implements certain hardware acceleration modes and provides a new Interruption Instruction Bundle control register. The additional virtualization support attempts to further reduce the virtualization overhead.

2.5.1 Intel® VT-i3 Support

Summary of enhancements over Intel VT-i3

- Guest copy of interruption control registers (cr16–17,19–27)
- Guest cover
- Asynchronous VINT delivery, directly to Virtual External Interrupt vector (IVA+0x3400)
- Guest ssm/rsm enhancements
- Guest PSR read (full VPSR)
- Relaxed reserved field checking on mov to psr.l
- Guest tf support (VCPUID4)
- Guest AR.ITC offset (CR.ITO)
- Probe intercepts
- Support illegal op fault on virtual CR access when not in VMAL
- Enhanced bundle capture

- PMU event filtering on psr.vm

Intel VT-i3 is documented in the Intel Itanium Software Developers Manual and supplement.

2.6 IA-32 Execution

IA-32 execution on the Intel Itanium processor 9500 series is enabled supported with PAL-based IA-32 execution and IA-32 Execution Layer (IA-32 EL).

PAL-based IA-32 execution is available after PAL_COPY_PAL is called and provides IA-32 execution support before the OS has booted. PAL-based IA-32 execution is not supported in an OS environment.

IA-32 EL is a software layer that is currently shipping with Intel® Itanium® architecture-based operating systems, which converts IA-32 instructions into Intel® Itanium® processor instructions via dynamic translation. IA-32 EL is OS-based and is only available after an OS has booted.

2.7 Brand Information

The PAL_BRAND_INFO procedure, along with PAL_PROC_GET_FEATURES, allows software to obtain processor branding and feature information. Details on the above functions can be found in the *Intel® Itanium® Architecture Software Developer's Manual*.

Below is the table of the return values for PAL_BRAND_INFO. The Intel Itanium processor 9500 series will implement all three; however, some previous Intel® Itanium® processors are all unable to retrieve the processor frequency, so requests for these fields will return -6, information not available. Also, previous Intel® Itanium® processors cannot return system bus frequency speed. Implementation-specific values are expected to start at value 16 and continue until an invalid argument (-2) is returned.

Table 2-52. PAL_BRAND_INFO Return Values

Value	Definition
19	Stepping - terminated ASCII string corresponding to the processor stepping will be returned in the brand_info return argument.
17	The shared LLC cache size component (in bytes) of the brand identification string will be returned as a binary value in the brand_info return argument.
16	The frequency component (in Hz) of the brand identification string will be returned as a binary value in the brand_info return argument.
0	The ASCII brand identification string will be copied to the address specified in the address input argument. The processor brand identification string is defined to be a maximum of 128 characters long; 127 bytes will contain characters and the 128th byte is defined to be NULL (0). A processor may return less than 127 ASCII characters as long as the string is null terminated. The string length will be placed in the *brand_info* return argument.

There are other processor features that may not be included in the brand name above. To obtain information on if that technology or feature has been implemented, the PAL_PROC_GET_FEATURES procedure should be used. The Intel® Itanium® processor 9500 series features will be available in feature_set (20).

Table 2-53. Intel Itanium Processor 9500 Series Feature Set Return Values

Value	Definition
18	Multi-Threading Technology (MT) – This processor supports Multi-Threading Technology

§

3 Core Performance Monitoring

3.1 Introduction

This section defines the performance monitoring features of the the processor core. The the processor core provides 16 48-bit performance counters per thread, hundreds of monitorable events, and several advanced monitoring capabilities. This chapter outlines the targeted performance monitor usage models and defines the software interface and programming model.

The Itanium architecture incorporates architected mechanisms that allow software to actively and directly manage performance critical processor resources such as branch prediction structures, processor data and instruction caches, virtual memory translation structures, and more. To achieve the highest performance levels, dynamic processor behavior can monitor and be fed back into the code generation process to better encode observed run-time behavior, or to expose higher levels of instruction level parallelism. These measurements are critical for understanding the behavior of compiler optimizations, the use of architectural features such as speculation and predication, or the effectiveness of microarchitectural structures such as the ALAT, the caches, and the TLBs. These measurements provide the data to drive application tuning and future processor, compiler, and operating system designs.

The remainder of the document is split into the following sections:

- Section 3.2 discusses how performance monitors are used, and presents various the processor performance monitoring programming models.
- Section 3.3 defines the the processor specific performance monitoring features, structures and registers.

Refer to Appendix B, “Example Core PMU Event Reports” for examples of PMU core reporting.

3.2 Performance Monitor Programming Models

This section introduces the the processor performance monitoring features from a programming model point of view and describes how the different event monitoring mechanisms can be used effectively. The the processor performance monitor architecture focuses on the following two usage models:

- **Workload Characterization:** The first step in any performance analysis is to understand the performance characteristics of the workload under study. Section 3.2.1 discusses the the processor support for workload characterization.
- **Profiling:** Profiling is used by application developers and profile-guided compilers. Application developers are interested in identifying performance bottlenecks and relating them back to their code. Their primary objective is to understand which program location caused performance degradation at the module, function, and basic block level. For optimization of data placement and the analysis of critical loops, instruction level granularity is desirable. Profile-guided compilers that use advanced Itanium architectural features such as predication and speculation benefit from run-time profile information to optimize instruction schedules. The the processor supports instruction level statistical profiling of branch mispredicts and cache misses. Details of the the processor's profiling support are described in Section 3.2.2.

3.2.1 Workload Characterization

The first step in any performance analysis is to understand the performance characteristics of the workload under study. There are two fundamental measures of interest: event rates and cycle accounting.

- **Event Rate Monitoring:** Event rates of interest include average retired instructions-per-clock (IPC), data and instruction cache miss rates, or branch mispredict rates measured across the entire application. Characterization of operating systems or large commercial workloads (for example, OLTP analysis) requires a system-level view of performance relevant events such as TLB miss rates, VHPT walks/second, interrupts/second, or bus utilization rates. Section 3.2.1.1 discusses event rate monitoring.
- **Cycle Accounting:** The cycle breakdown of a workload attributes a reason to every cycle spent by a program. Apart from a program's inherent execution latency, extra cycles are usually due to pipeline stalls and flushes. Section 3.2.1.4 discusses cycle accounting.

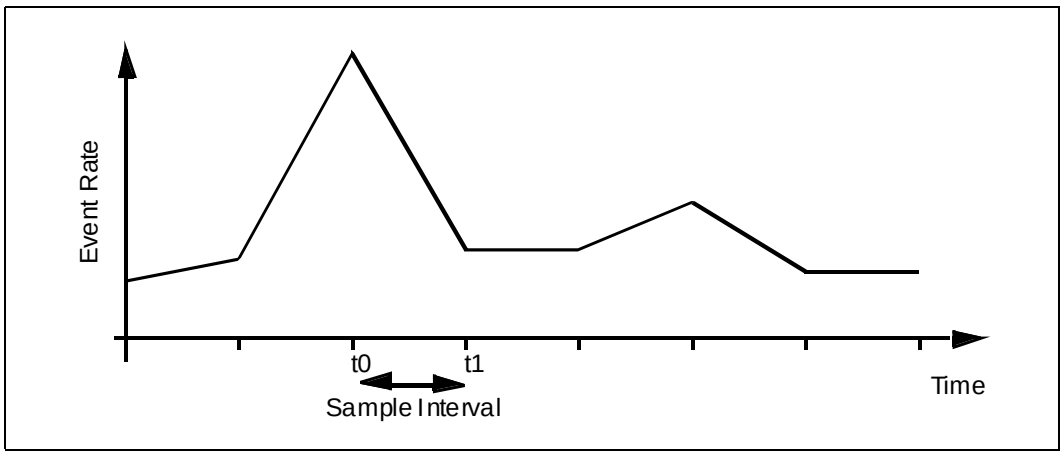
3.2.1.1 Event Rate Monitoring

Event rate monitoring determines event rates by reading processor event occurrence counters before and after the workload is run, and then computing the desired rates. For instance, two basic the processor events that count the number of retired Itanium instructions (IA64_INST_RETIRED) and the number of elapsed clock cycles (CPU_OP_CYCLES) allow a workload's instructions per cycle (IPC) to be computed as follows:

- IPC = ($IA64\_INST\_RETIRED_{t1}$ – $IA64\_INST\_RETIRED_{t0}$) / ($CPU\_OP\_CYCLES_{t1}$ – $CPU\_OP\_CYCLES_{t0}$)

Time-based sampling is the basis for many performance debugging tools [VTune™, gprof, WinNT*]. As shown in Figure 3-1, time-based sampling can be used to plot the event rates over time, and can provide insights into the different phases that the workload moves through.

Figure 3-1. Time-Based Sampling



On the processor, many event types, for example, TLB misses or branch mispredicts are limited to a rate of one per clock cycle. These are referred to as “single occurrence” events. However, in the the processor, multiple events of the same type may occur in the same clock. We refer to such events as “multi-occurrence” events. An example of a multi-occurrence event on the processor is data cache read misses (up to two occur per

clock). Multi-occurrence events, such as the number of entries in the memory request queue, can be used to derive the average number and average latency of memory accesses. The next two sections describe the basic processor mechanisms for monitoring single and multi-occurrence events.

3.2.1.2 Single Occurrence Events and Duration Counts

For all single occurrence events, a counter is incremented by up to one per clock cycle. Duration counters that count the number of clock cycles during which a condition persists are considered “single occurrence” events. Examples of single occurrence events on the processor are TLB misses, branch mispredictions, and cycle-based metrics.

3.2.1.3 Multi-Occurrence Events, Thresholding, and Averaging

Events that, due to hardware parallelism, may occur at rates greater than one per clock cycle are termed “multi-occurrence” events. Examples of such events on the processor are retired instructions or the number of live entries in the memory request queue.

Thresholding capabilities are available in the processor's counters and can be used to plot an event distribution histogram. When a non-zero threshold is specified, the monitor is incremented by one in every cycle in which the observed event count exceeds that programmed threshold. This allows questions such as "For how many cycles did the memory request queue contain more than two entries?" or "During how many cycles did the machine retire more than three instructions?" to be answered. This capability allows microarchitectural buffer sizing experiments to be supported by real measurements. By running a benchmark with different threshold values, a histogram can be drawn up that may help to identify the performance "knee" at a certain buffer size.

For overlapping concurrent events, such as pending memory operations, the average number of concurrently outstanding requests and the average number of cycles that requests were pending are of interest. To calculate the average number or latency of multiple outstanding requests in the memory queue, we need to know the total number of requests (n_{total}) and the number of live requests per cycle (n_{live}/cycle). By summing up the live requests (n_{live}/cycle) using a multi-occurrence counter, Σn_{live} is directly measured by hardware. We can now calculate the average number of requests and the average latency as follows:

- Average outstanding requests/cycle = $\Sigma n_{live} / \Delta t$
- Average latency per request = $\Sigma n_{live} / n_{total}$

An example of this calculation is given in Table 3-1 in which the average outstanding requests/cycle = 15/8 = 1.825, and the average latency per request = 15/5 = 3 cycles.

Table 3-1. Average Latency per Request and Requests per Cycle Calculation Example

Time [Cycles]	1	2	3	4	5	6	7	8
# Requests In	1	1	1	1	1	0	0	0
# Requests Out	0	0	0	1	1	1	1	1
n_{live}	1	2	3	3	3	2	1	0
Σn_{live}	1	3	6	9	12	14	15	15
n_{total}	1	2	3	4	5	5	5	5

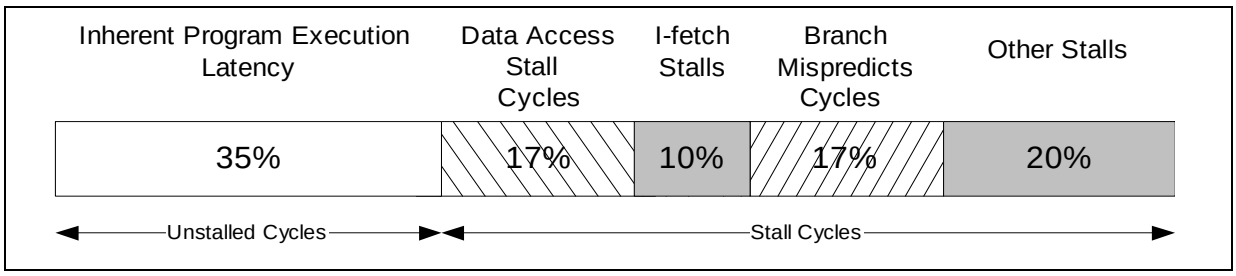
The processor provides the following capabilities to support event rate monitoring:

- Clock cycle counter
- Retired instruction counter
- Event occurrence and duration counters
- Multi-occurrence capable counters
- Counter thresholding

3.2.1.4 Cycle Accounting

While event rate monitoring counts the number of events, it does not tell us whether the observed events are contributing to a performance problem. A commonly used strategy is to plot multiple event rates and correlate them with the measured instructions per cycle (IPC) rate. If a low IPC occurs concurrently with a peak of cache miss activity, chances are that cache misses are causing a performance problem. To eliminate such guess work, the processor provides a set of cycle accounting monitors, that break down the number of cycles that are lost due to various kinds of microarchitectural events. As shown in Figure 3-2, this lets us account for every cycle spent by a program and therefore provides insight into an application's microarchitectural behavior. Note that cycle accounting is different from simple stall or flush duration counting. Cycle accounting is based on the machine's actual stall and flush conditions, and accounts for overlapped pipeline delays, while simple stall or flush duration counters do not. Cycle accounting determines a program's cycle breakdown by stall and flush reasons, while simple duration counters are useful in determining cumulative stall or flush latencies.

Figure 3-2. Itanium® Processor Family Cycle Accounting



The processor cycle accounting monitors account for all major single and multi-cycle stall, replay and flush conditions both for the front-end as well as the back-end thread domain. Overlapping stall, replay and flush conditions are prioritized in reverse pipeline order, that is, delays that occur later in the pipe and that overlap with earlier stage delays are reported as being caused later in the pipeline. The six back-end stall, replay and flush reasons are prioritized in the following order:

1. Exception/Interruption flushes: cycles spent flushing the pipe due to interrupts and exceptions. (event CYC_BE_WB2_FLUSH.XPN Appendix 4.2.4.6)
2. WB2 replays: late pipe events due to FP, exceptions, and blocking hazards (event CYC_BE_WB2_REPLAY.* Appendix 4.2.4.7)
3. Branch Mispredictions: cycles spent flushing the pipe due to branch mispredicts. (event CYC_BE_WB2_FLUSH.BRU Appendix 4.2.4.6)
4. DET replays: mostly due to memory pipe hazards (event CYC_BE_DET_REPLAY.* Appendix 4.2.4.2)
5. EXE replays: mostly register hazards. (event CYC_BE_EXE_REPLAY.* Appendix 4.2.4.3)

6. Issue stall cycles due to load hazards, RSE activities, front end bubbles and others. (event CYC_BE_IBD_STALL.* Appendix 4.2.4.4)

For each of these reasons, a complete set of sub-cause events is available. In addition to the back-end, execution, cycle accounting, analogous monitors are available to diagnose front-end fetch performance.

3.2.2 Profiling

Profiling is used by application developers, profile-guided compilers, optimizing linkers, and run-time systems. Application developers are interested in identifying performance bottlenecks and relating them back to their source code. Based on profile feedback, developers can make changes to the high-level algorithms and data structures of the program. Compilers can use profile feedback to optimize instruction schedules by employing advanced Itanium architectural features such as predication and speculation.

To support profiling, performance monitor counts have to be associated with program locations. The following mechanisms are supported directly by the processor's performance monitors:

- Program Counter Sampling
- Miss Event Address Sampling: The processor event address registers (EARs) provide address and latency information for performance critical events (instruction and data cache accesses, branch mispredicts, and instruction and data TLBs).
- Event Qualification: constrains event monitoring to a specific instruction address range, to certain opcodes or privilege levels.

These profiling features are presented in the next three subsections.

3.2.2.1 Program Counter Sampling

Application tuning tools use time-based or event-based sampling of the program counter and other event counters to identify performance critical functions and basic blocks. As shown in Figure 3-3, the sampled points can be histogrammed by instruction addresses. For application tuning, statistical sampling techniques have been very successful, because the programmer can rapidly identify code hot spots in which the program spends a significant fraction of its time, or where certain event counts are high.

Program counter sampling points the performance analysts at code hot spots, but does not indicate what caused the performance problem. Inspection and manual analysis of the hot-spot region along with a fair amount of guess work are required to identify the root cause of the performance problem. On the processor, the cycle accounting mechanism (described in Section 3.2.1.4) can be used to directly measure an application's microarchitectural behavior.

The Itanium architectural interval timer facilities (ITC and ITM registers) can be used for time-based program counter sampling. Event-based program counter sampling is supported by a dedicated performance monitor overflow interrupt mechanism described in detail in Section 7.2.2 "Performance Monitor Overflow Status Registers (PMC_0...PMC_3)" in Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual*.

Figure 3-3. Event Histogram by Program Counter

To support program counter sampling, the processor provides the following mechanisms:

- Timer interrupt for time-based program counter sampling
- Event count overflow interrupt for event-based program counter sampling
- Hardware-supported cycle accounting

3.2.2.2 Miss Event Address Sampling

Program counter sampling and cycle accounting provide an accurate picture of cumulative microarchitectural behavior, but they do not provide the application developer with pointers to specific program elements (code locations and data structures) that repeatedly cause microarchitectural "miss events". In a cache study of the SPEC92 benchmarks, [Lebeck] used (trace based) cache miss profiling to gain performance improvements of 1.02 to 3.46 on various benchmarks by making simple changes to the source code. This type of analysis requires identification of instruction and data addresses related to microarchitectural "miss events" such as cache misses, branch mispredicts, or TLB misses. Using symbol tables or compiler annotations these addresses can be mapped back to critical source code elements. Like Lebeck, most performance analysts in the past have had to capture hardware traces and resort to trace driven simulation.

Due to the superscalar issue, deep pipelining, and out-of-order instruction completion of today's microarchitectures, the sampled program counter value may not be related to the instruction address that caused a miss event. On a Pentium processor pipeline, the sampled program counter may be off by two dynamic instructions from the instruction that caused the miss event. On a Pentium Pro processor, this distance increases to approximately 32 dynamic instructions. On the processor, it is approximately 48 dynamic instructions. If program counter sampling is used for miss event address identification on the processor, a miss event might be associated with an instruction almost five dynamic basic blocks away from where it actually occurred (assuming that 10% of all instructions are branches). Therefore, it is essential for hardware to precisely identify an event's address.

The processor provides a set of *event address registers* (EARs) that record the instruction and data addresses of data cache accesses, the instruction and data addresses of data TLB misses, and the instruction addresses of instruction TLB and cache misses as well as latency and responder type status associated with the access. A 24 entry deep *execution trace buffer* captures sequences of branch instructions and other instructions and events which causes changes to execution flow. The table below summarizes the capabilities offered by the processor EARs and the execution trace

buffer. Exposing miss event addresses to software allows them to be monitored either by sampling or by code instrumentation. This eliminates the need for trace generation to identify and solve performance problems and enables performance analysis by a much larger audience on unmodified hardware.

Table 3-2. Processor EARs and Branch Trace Buffer

Event Address Register	Triggers On	What is Recorded
Instruction Cache	Instruction fetches that miss the FLI cache (demand fetches only) Instruction fetch missed FLI ITLB (demand fetches only)	Instruction Address Number of cycles fetch was in flight Responder status of the fetch (who serviced the miss) Who serviced L1 ITLB miss: L2 ITLB VHPT or software
Data Cache	Load or store instructions that hit or miss the FLD or MLD data caches Data references to the DTLB ALAT accesses	Instruction Address Data Address Number of cycles load was in flight. Responder status of the load (who serviced the request) Requester op-type (who asked) Current HWPF hint state Who serviced L1 DTLB miss: L2 DTLB, VHPT or software
Execution Trace Buffer	Branch Outcomes rfi, exceptions, failed "chk" instructions which cause a change in execution flow	Source instruction address of the event Target Instruction Address of the event Mispredict status and reason for branches
Retired IP	Retired IPs	Source instruction address of the retired instruction, cycles since last retirement.

The processor EARs enable statistical sampling by configuring a performance counter to count, for instance, the number of data cache miss captures or retired instructions. The performance counter value is set up to interrupt the processor after a predetermined number of events have been observed. The data cache event address register repeatedly captures the instruction and data addresses of actual data cache load misses. Whenever the counter overflows, event address collection is suspended (this prevents software from capturing a miss event that might be caused by the monitoring software itself). When the counter overflows the PMU is frozen, an interrupt is delivered to software, the observed event addresses are collected, and a new observation interval can be set up by rewriting the performance counter registers. For time-based (rather than event-based) sampling methods, the event address registers indicate to software whether or not a qualified event was captured. Statistical sampling can achieve arbitrary event resolution by varying the number of events within an observation interval and by increasing the number of observation intervals.

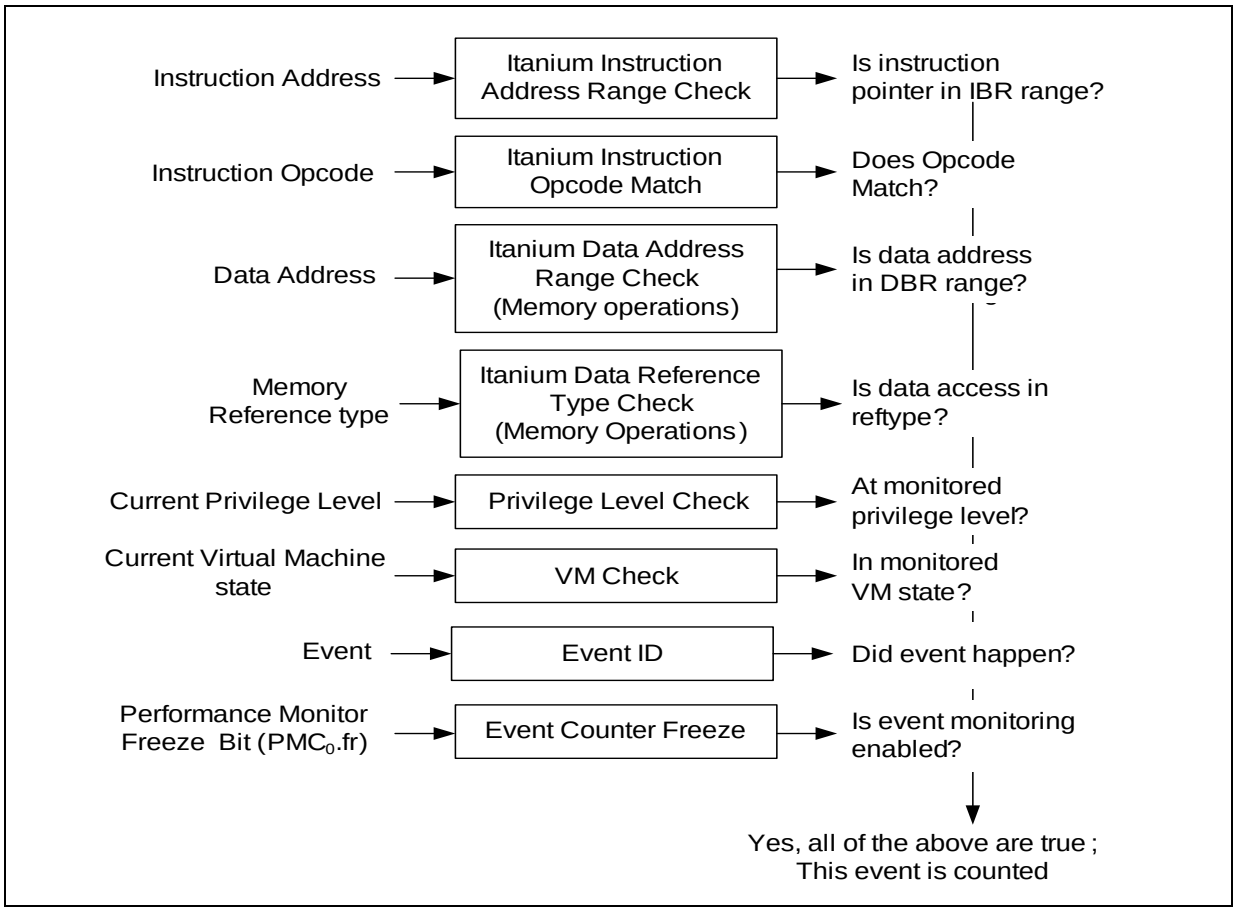
3.2.3 Event Qualification

In the processor, many of the performance monitoring events can be qualified in a number of ways such that only a subset of the events are counted using performance monitoring counters. As shown in the figure below, events can be qualified for monitoring based on instruction address range, instruction opcode, data address range, data reference type, the privilege level and virtual machine state and the status of the performance monitoring freeze bit (PMC_0.fr). The following paragraphs describe these capabilities in more detail.

- Itanium Instruction Address Range Check: The processor allows event monitoring to be constrained to a programmable instruction address range. This enables monitoring of dynamically linked libraries (DLLs), functions, or loops of interest in the context of a large Itanium-based application. The Itanium instruction address

range check is applied at the instruction fetch stage of the pipeline and the resulting qualification is carried by the instruction throughout the pipeline. This enables conditional event counting at a level of granularity smaller than dynamic instruction length of the pipeline (approximately 48 instructions). For details, see Section 3.3.5.

Figure 3-4. Processor Event Qualification



Itanium Instruction Opcode Match: The processor provides two independent Itanium opcode match ranges each of which match the currently issued instruction encodings with a programmable opcode match and mask function. The resulting match events can be selected as an event type for counting by the performance counters. This allows histogramming of instruction types, usage of destination and predicate registers as well as software profiling (through insertion of tagged NOPs). Details are described in Section 3.3.6.

- Itanium Data Address Range Check: The processor allows event collection for memory operations to be constrained to a programmable data address range. This enables selective monitoring of data cache miss behavior of specific data structures. For details, see Section 3.3.7.
- Itanium Data Reference Type Check: The processor allows event collection for memory operations to be constrained to specific memory pipe operations. For details, see Section 3.3.7.
- Privilege Level: The processor supports conditional event counting based on the current privilege level; this allows performance monitoring software to break down

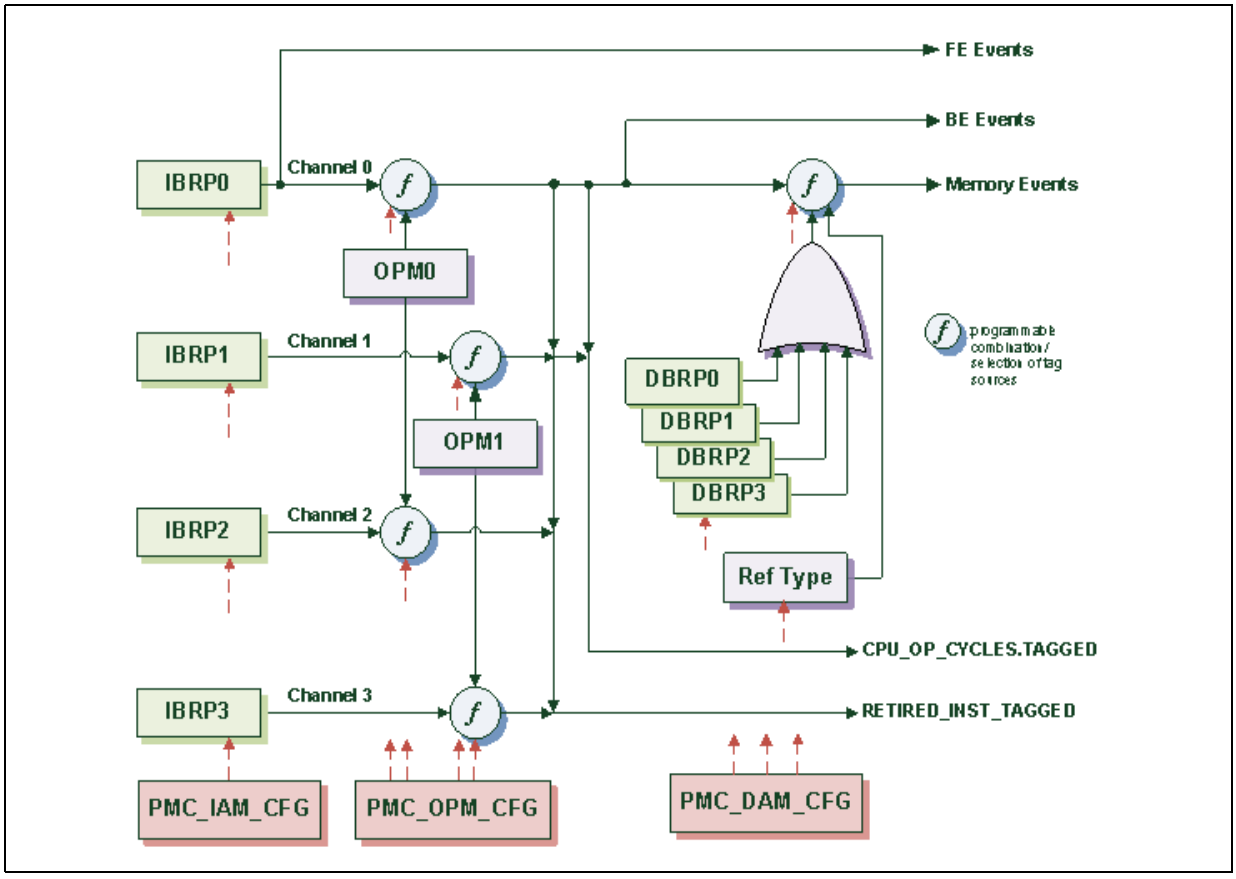
event counts into user and operating system contributions. For details on how to constrain monitoring by privilege level, refer to Section 3.3.1.

- Virtual Machine state: Analogous to privilege level constraints, the processor supports conditional event counting based on the current virtual machine state (PSR.vm). This allows performance monitoring software to further break down event counts into the virtual machine hosts and guest operating system contributions. For details on how to constrain monitoring by virtual machine state level, refer to Section 3.3.1.
- Performance Monitor Freeze: Event counter overflows or software can freeze event monitoring. When frozen, no event monitoring takes place until software clears the monitoring freeze bit (PMC_0.fr). This ensures that the performance monitoring routines themselves, for example, counter overflow interrupt handlers or performance monitoring context switch routines, do not "pollute" the event counts of the system under observation. For details, refer to Section 7.2.4 of Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual.*

3.2.3.1 Combining Instruction Address, Opcode, Data Address and Memory Reference Type Matching

The processor allows various event qualification mechanisms to be combined in a daisy-chained manner by providing the instruction tagging mechanism shown in Figure 3-5.

Figure 3-5. Instruction Tagging Mechanism in the Processor



During Itanium instruction execution, the instruction address range check is applied first. This is applied separately for each IBR pair (IBRP) to generate 4 independent tag bits which flow down the machine in four tag channels. The channel 0 instruction address tag is used to qualify front-end events where applicable. Tags in the four tag channels are then passed to two opcode matchers that combine the instruction address range check with the opcode match and generate the set of four tags passed to the back-end. This is done by combining tag channels 0 and 2 with first opcode match registers and tag channels 1 and 3 with the second opcode match registers as shown in Figure 3-5. Each of the 4 combined tags in the four tag channels can be counted as a retired instruction count event (for details refer to event description "RETIRED_INST_TAGGED"). Additionally tags 0 and 1 are used for the CPU_OP_CYCLES.TAGGED event Appendix 4.2.1.1.

The combined Itanium instruction address range and opcode match in tag channel 0 qualify all downstream pipeline events. Events in the data memory hierarchy (L1 and L2 data cache and data TLB events, can further be qualified using data address range, specified through the DBRs and data reference type constraints.

As summarized in Figure 3-5, data address range checking can be combined with opcode matching and instruction range checking on the processor. Additional event qualifications based on the current privilege level can be applied to all events and are discussed in Section 3.2.3.2.

Table 3-3 below shows register settings for different qualification scenarios. These are the preferred settings, though it may be possible to achieve the same functionality with different settings. Table 3-3's scenarios apply to tag channel 0. Channels 1–3 can be constrained analogously for those events that use/require them. These and other scenarios are left as an exercise to the reader.

Table 3-3. Example Processor Event Qualification Modes (Sheet 1 of 2)

Event Qualification Modes	PMC settings	Constraints Defined By
Unconstrained Monitoring	PMC_IAM_CFG.ig_ibrp = 1 PMC_OPM0_MSK.ig_ad = 1 PMC_OPM_CFG.ch0_ig_opc = 1 PMC_DAM_CFG.cfgdtag0 = 0x18	
Instruction Address Range constraint only	PMC_IAM_CFG.ig_ibrp = 0 PMC_OPM0_MSK.ig_ad = 0 PMC_OPM_CFG.ch0_ig_opc = 1 PMC_DAM_CFG.cfgdtag0 = 0x08	IBRP0
Opcode constraint only	PMC_IAM_CFG.ig_ibrp = 1 PMC_OPM0_MSK.ig_ad = 1 PMC_OPM_CFG.ch0_ig_opc = 0 PMC_DAM_CFG.cfgdtag0 = 0x08	PMC_OPM0_MSK PMC_OPM0_MAT
Instruction Address Range and Opcode constraints	PMC_IAM_CFG.ig_ibrp = 0 PMC_OPM0_MSK.ig_ad = 0 PMC_OPM_CFG.ch0_ig_opc = 0 PMC_DAM_CFG.cfgdtag0 = 0x08	IBRP0 PMC_OPM0_MSK PMC_OPM0_MAT
Data Address Range constraint only	PMC_IAM_CFG.ig_ibrp = 1 PMC_OPM0_MSK.ig_ad = 1 PMC_OPM_CFG.ch0_ig_opc = 1 PMC_DAM_CFG.typemask = 0x0 PMC_DAM_CFG.cfgdtag0 = 0xF0	DBRP0
Data Reference Type constraint only	PMC_IAM_CFG.ig_ibrp = 1 PMC_OPM0_MSK.ig_ad = 1 PMC_OPM_CFG.ch0_ig_opc = 1 PMC_DAM_CFG.cfgdtag0 = 0x18	PMC_DAM_CFG.typemask

Table 3-3. Example Processor Event Qualification Modes (Sheet 2 of 2)

Event Qualification Modes	PMC settings	Constraints Defined By
Instruction and Data Address Range constraints	PMC_IAM_CFG.ig_ibrp = 0 PMC_OPM0_MSK.ig_ad = 0 PMC_OPM_CFG.ch0_ig_opc = 1 PMC_DAM_CFG.typemask = 0x0 PMC_DAM_CFG.cfgdtag0 = 0xE0	IBRP0 DBRP0
Opcode and Data Address Range constraints	PMC_IAM_CFG.ig_ibrp = 1 PMC_OPM0_MSK.ig_ad = 1 PMC_OPM_CFG.ch0_ig_opc = 0 PMC_DAM_CFG.typemask = 0x0 PMC_DAM_CFG.cfgdtag0 = 0xE0	PMC_OPM0_MSK PMC_OPM0_MAT DBRP0
Instruction Address, Opcode, Data Address and Data Reference Type constraints	PMC_IAM_CFG.ig_ibrp = 0 PMC_OPM0_MSK.ig_ad = 0 PMC_OPM_CFG.ch0_ig_opc = 0 PMC_DAM_CFG.cfgdtag0 = 0xE0	IBRP0 PMC_OPM0_MSK PMC_OPM0_MAT DBRP0 PMC_DAM_CFG.typemask

Note: Event address range and optype qualification when PMC.all is set will only produce expected results if both threads' constraints are programmed identically. Per-monitor constraints are applied to the originating thread's state.

Note: Similarly, to correctly constrain floating events (indicated by the "F" threading attribute in the event tables), both threads' constraints must be programmed identically.

3.2.3.2 Privilege Level Constraints

To provide hardware support for various combinations of privilege levels and interrupt handlers, the Itanium architecture specifies three global bits (PSR.up, PSR.pp, DCR.pp) and a per-monitor "privileged monitor" bit (PMC_i.pm). To break down the performance contributions of operating system and user-level application components, each monitor specifies a 4-bit privilege level mask (PMC_i.plm). The mask is compared to the current privilege level in the processor status register (PSR.cpl), and event counting is enabled if PMC_i.plm[PSR.cpl] is one. The processor performance monitors control is discussed in Section 3.3.1.

PMC registers can be configured as user-level monitors (PMC_i.pm is 0) or system-level monitors (PMC_i.pm is 1). A user-level monitor is enabled whenever PSR.up is one. PSR.up can be controlled by an application using the sum/rum instructions. This allows applications to enable/disable performance monitoring for specific code sections. A system-level monitor is enabled whenever PSR.pp is one. PSR.pp can be controlled at privilege level 0 only by way of the ssm/rsm instructions, which allows monitor control without interference from user-level processes. The pp field in the default control register (DCR.pp) is copied into PSR.pp whenever an interruption is delivered. This allows events generated during interruptions to be broken down separately: if DCR.pp is 0, events during interruptions are not counted; if DCR.pp is 1, they are included in the kernel counts.

As shown in Figure 3-6, Figure 3-7, and Figure 3-8, single process, multi-process, and system-level performance monitoring are possible by specifying the appropriate combination of PSR and DCR bits. These bits allow performance monitoring to be controlled entirely from a kernel level device driver, without explicit operating system support. Once the desired monitoring configuration has been set up in a process' processor status register (PSR), "regular" unmodified operating context switch code automatically enables/disables performance monitoring.

With support from the operating system, individual per-process breakdown of event counts can be generated as outlined in the performance monitoring chapter of the *Intel® Itanium® Architecture Software Developer's Manual*.

Figure 3-6. Single Process Monitor

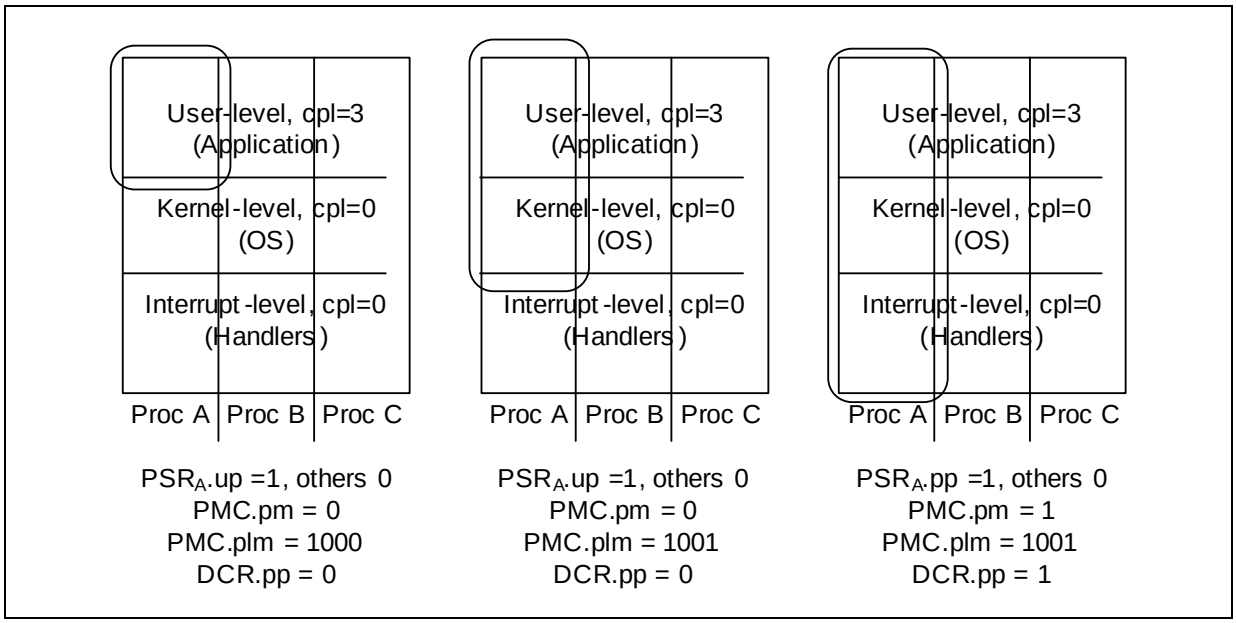


Figure 3-7. Multiple Process Monitor

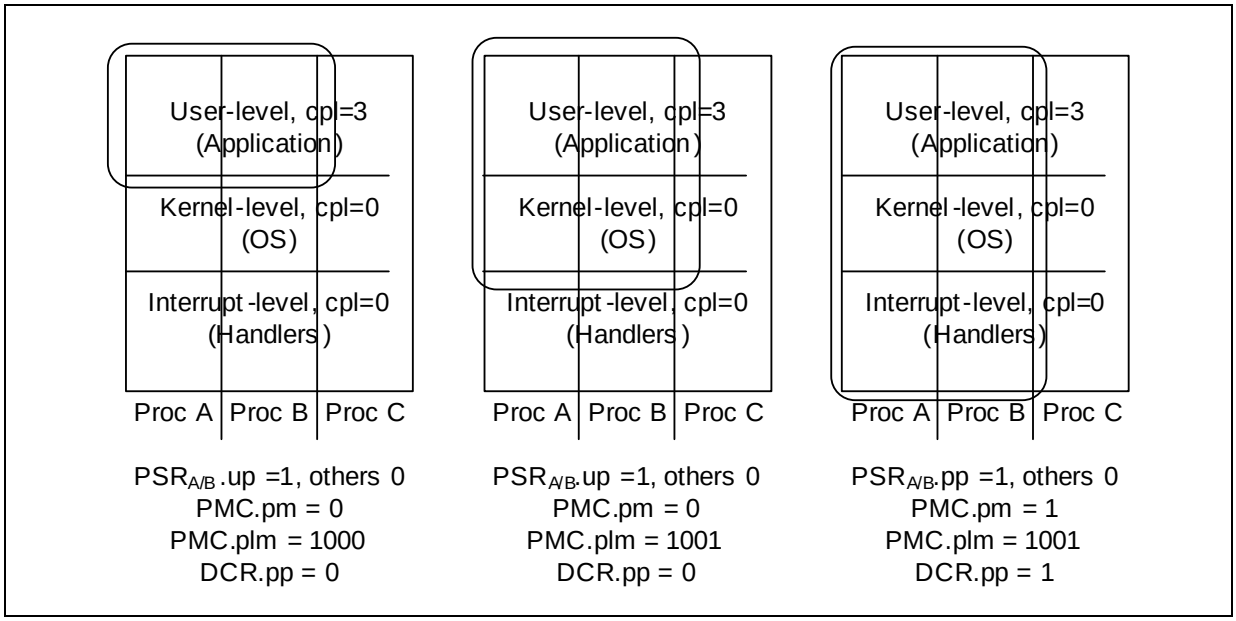
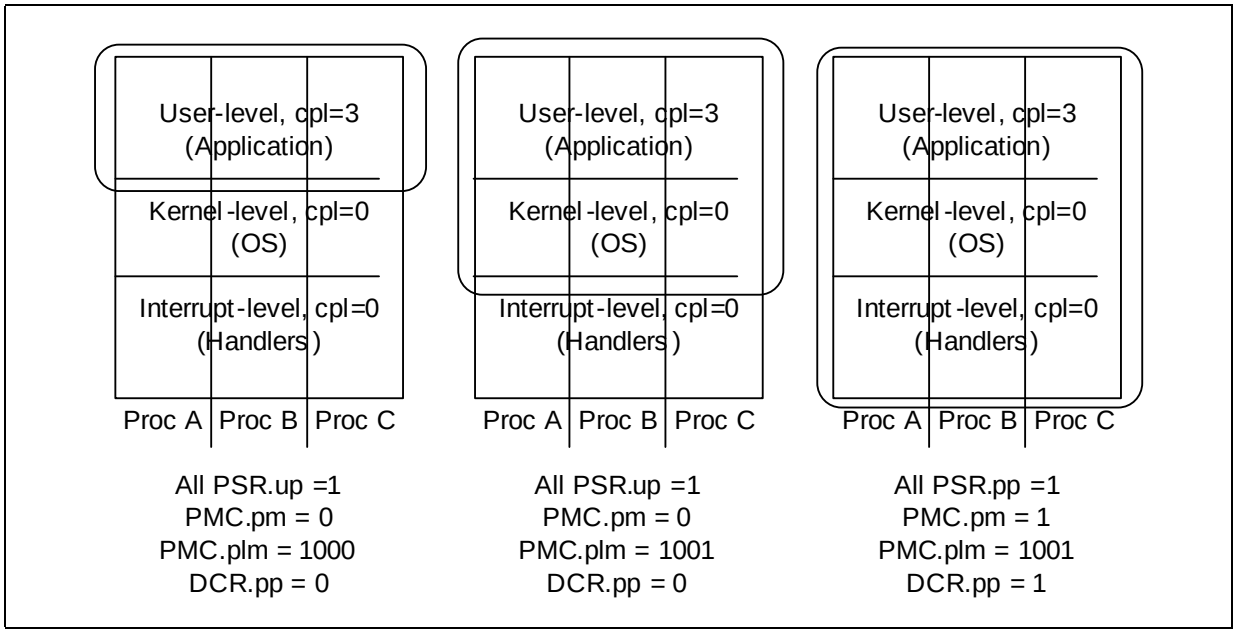


Figure 3-8. System Wide Monitor



3.2.3.3 Virtual Machine State Constraints

To break down the performance contributions of virtual machine operation, each monitor specifies a 2-bit virtual machine state mask (PMC_i.vm). The mask is compared to the current virtual machine state (PSR.vm) and event counting is enabled accordingly. The processor performance monitors control is discussed in Section 3.3.1.

3.2.3.4 Power State Constraints

To allow for halted state accounting, the processor provides a count halted event count qualifier (PMC_i.ch) that enables counting during low-power halted state.

3.2.3.5 Instruction Set Constraints

Instruction set constraint is not supported in the processor.

3.2.4 References

- [gprof] S.L. Graham S.L., P.B. Kessler and M.K. McKusick, "gprof: A Call Graph Execution Profiler", Proceedings SIGPLAN'82 Symposium on Compiler Construction; SIGPLAN Notices; Vol. 17, No. 6, pp. 120–126, June 1982.
- [Lebeck] Alvin R. Lebeck and David A. Wood, "Cache Profiling and the SPEC benchmarks: A Case Study", Tech Report 1164, Computer Science Dept., University of Wisconsin – Madison, July 1993.
- [VTune] Mark Atkins and Ramesh Subramaniam, "PC Software Performance Tuning", IEEE Computer, Vol. 29, No. 8, pp. 47–54, August 1996.
- [WinNT] Russ Blake, "Optimizing Windows NT(tm)", Volume 4 of the Microsoft "Windows NT Resource Kit for Windows NT Version 3.51", Microsoft Press, 1995.

3.3 Performance Monitor State

IA64 Performance Monitoring architecture described in Section 12 and Section 7.2.1 of Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* IA64 defines two sets of performance monitor registers; Performance Monitor Configuration (PMC) registers to configure the monitoring and Performance Monitor Data (PMD) registers to provide data from the monitors. Additionally, the architecture allows for architectural as well as model specific registers. Complying with this architectural definition, the processor provides both kind of PMCs and PMDs. The processor provides 16 48-bit performance counters (PMC/PMD_{4-19} pairs), and a set of model-specific monitoring registers.

The table below defines the PMC/PMD register assignments for each monitoring feature. The interrupt status registers are mapped to PMC_{0-3}. The 16 generic performance counter pairs are assigned to PMC/PMD_{4-19}. The Event Address Registers (EARs) and the Execution Trace Buffer (ETB) are controlled by their respective configuration registers in PMC space. Captured event addresses and cache miss latencies are accessible to software through event address data registers and a execution trace buffer, addressed in PMD space. On the processor, monitoring of some events can additionally be constrained to a programmable instruction address range, by appropriately setting the instruction breakpoint registers (IBR) and the instruction address range check register (PMC_IAM_CFG) and turning on the checking mechanism in the opcode matchers. Two opcode match register sets and an opcode match configuration register allow some events to be further qualified with a programmable opcode mask. For memory operations, events can subsequently be qualified by a programmable data address range by appropriate setting of the data breakpoint registers (DBRs) and the data address range configuration register as well as by data reference type.

Since the processor is capable of running two threads, it provides the illusion of having two processors by providing exactly the same set of performance monitoring features and structures separately for each thread.

Table 3-4. Processor Performance Monitor Register Set (Sheet 1 of 2)

Monitoring Feature	Configuration Registers (PMC)	Data Registers (PMD)	Description
Interrupt Status	$PMC_{0,1,2,3}$	none	See Section 3.3.4, "Performance Monitor Overflow Status Registers (PMC0,1,2,3)"
Event Counters	PMC_{4-19}	PMD_{4-19}	See Section 3.3.2, "Performance Counter Registers"
Instruction Address Match	PMC_IAM_CFG	none	See Section 3.3.5, "Instruction Address Range Matching"
Opcode Matching	PMC_OPM_CFG PMC_OPM0_MSK PMC_OPM0_MAT PMC_OPM1_MSK PMC_OPM1_MAT	none	See Section 3.3.6, "Opcode Match Check",
Data Address Range Match	PMC_DAM_CFG	none	See Section 3.3.7, "Data Address Range Matching (PMC_DAM_CFG)"
Data Reference Type Match	PMC_DAM_CFG	none	See Section 3.3.8, "Data Reference Type Matching (PMC_DAM_CFG)"
Instruction EAR	PMC_IEAR_CFG	PMD_IEAR_STAT PMD_IEAR_IADDR	See Section 3.3.9, "Event Address Registers"
Data EAR	PMC_DEAR_CFG	PMD_DEAR_STAT PMD_DEAR_IADDR PMD_DEAR_DADDR	See Section 3.3.11, "Data Cache EAR"

Table 3-4. Processor Performance Monitor Register Set (Sheet 2 of 2)

Monitoring Feature	Configuration Registers (PMC)	Data Registers (PMD)	Description
Branch Trace Buffer	PMC_ETB_CFG	PMD_ETB_BUFIDX PMD_ETB_{0-23} PMD_ETBEXT_{0-23}	See Section 3.3.12, "Execution Trace Buffer"
Retired IP EAR	PMC_IPEAR_CFG	PMD_ETB_BUFIDX PMD_ETB_{0-23} PMD_ETBEXT_{0-23}	See Section 3.3.12.2, "IP Event Address Capture (IP-EAR)"
Thread Switch EAR	PMC_IPEAR_CFG PMC_BEMT_CTL	PMD_ETB_BUFIDX PMD_ETB_{0-23} PMD_ETBEXT_{0-23}	See Section 3.3.12.3, "IP-EAR User Guide"

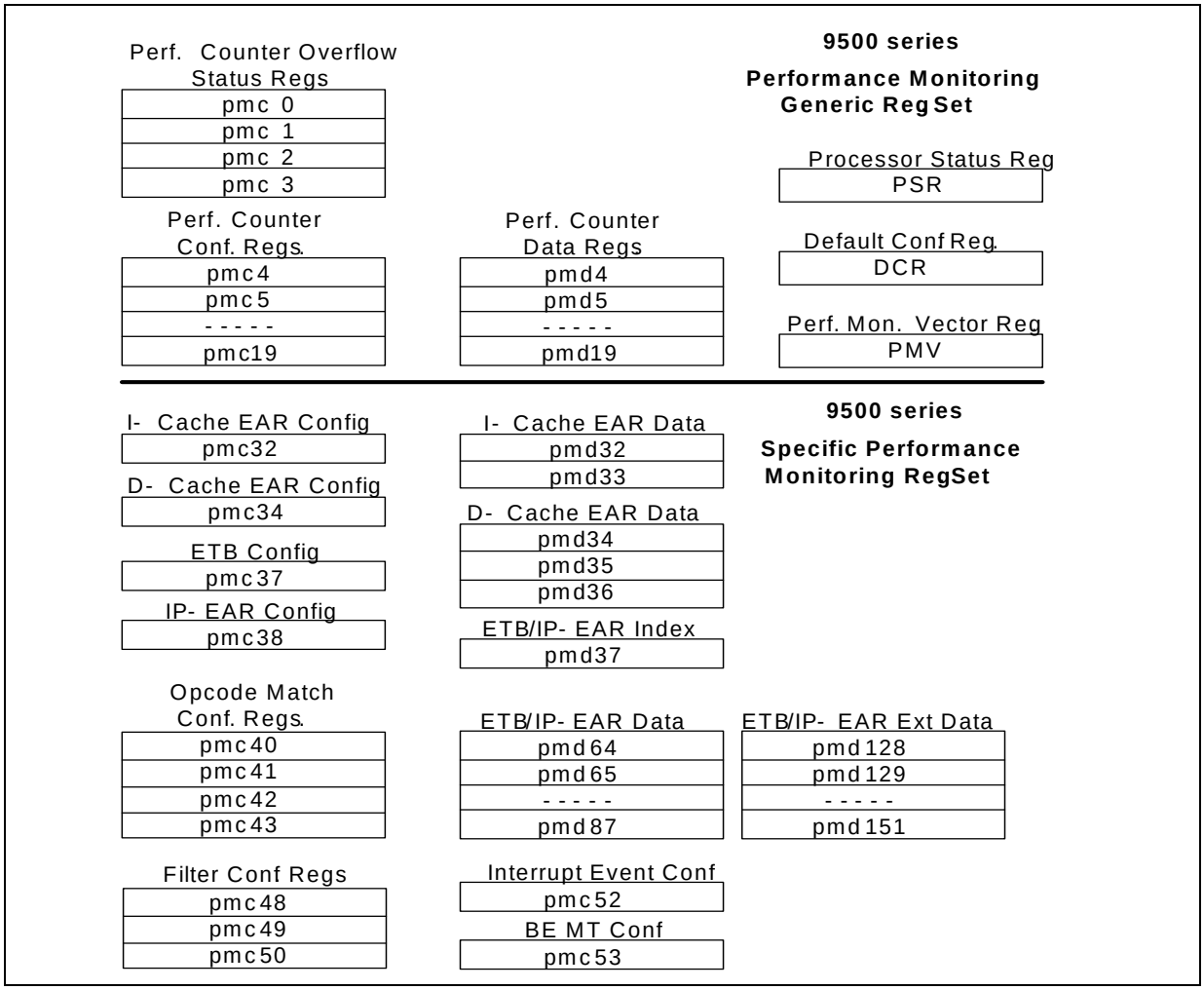
This document refers to the non-generic PMCs and PMDs by their functional names rather than the PMC/PMD numbers; the table below is intended to serve as a cross reference.

Table 3-5. Processor Implementation Specific PMC/PMD Numbering

Feature	Functional Name	PMC/PMD	Notes
Instruction EAR	PMC_IEAR_CFG	PMC_{32}	Instruction Cache EAR Configuration
	PMD_IEAR_STAT	PMD_{32}	Instruction Cache EAR Status
	PMD_IEAR_IADDR	PMD_{33}	Instruction Cache EAR Instruction Address
Data EAR	PMC_DEAR_CFG	PMC_{34}	Data Cache EAR Configuration
	PMD_DEAR_STAT	PMD_{34}	Data Cache EAR Status
	PMD_DEAR_DADDR	PMD_{35}	Data Cache EAR Data Address
	PMD_DEAR_IADDR	PMD_{36}	Data Cache EAR Instruction Address
Execution Trace Buffer	PMC_ETB_CFG	PMC_{37}	ETB Configuration
Retired IP-EAR	PMC_IPEAR_CFG	PMC_{38}	IP-EAR Configuration
	PMC_ETB_BUFIDX	PMD_{37}	ETB/IP-EAR Buffer Index
	PMD_ETB_{0-23}	PMD_{64-87}	ETB/IP-EAR Buffer
	PMD_ETBEXT_{0-23}	$PMD_{128-151}$	ETB/IP-EAR Buffer Extension
Opcode Matching	PMC_OPM0_MSK	PMC_{40}	Opcode Matcher 0 Mask Register
	PMC_OPM0_MAT	PMC_{41}	Opcode Matcher 0 Match Register
	PMC_OPM1_MSK	PMC_{42}	Opcode Matcher 1 Mask Register
	PMC_OPM1_MAT	PMC_{43}	Opcode Matcher 1 Match Register
Instruction Address Matching	PMC_IAM_CFG	PMC_{48}	Instruction Address Matcher Configuration
Opcode Matching	PMC_OPM_CFG	PMC_{49}	Opcode Matcher Configuration
Data Address Matching	PMC_DAM_CFG	PMC_{50}	Data Address Matcher Configuration
Interrupt Event Mask	PMC_IVAEV_MSK	PMC_{52}	Interrupt Event Configuration
Thread State Event Control	PMC_BEMT_CTL	PMC_{53}	Thread State Event Configuration

The figure below gives an overview of the processor Performance Monitor register map:

Figure 3-9. Processor Performance Monitor Register Mode



3.3.1 Performance Monitor Control and Accessibility

As in other Intel Itanium processors, in the processor event collection is controlled by the Performance Monitor Configuration (PMC) registers and the processor status register (PSR). Five PSR fields (PSR.up, PSR.pp, PSR.cpl, PSR.vm and PSR.sp) and the performance monitor freeze bit (PMC_0.fr) affect the behavior of all performance monitor registers.

Per-monitor control is provided by three PMC register fields (PMC_i.plm, PMC_i.vm, and PMC_i.pm). Event collection for a monitor is enabled under the following constraints on the processor:

```
vmmatch = PMCi.vm[PSR.vm] OR PMCi.vm == 0

Monitor Enablei = (not PMC0.fr) and
    PMCi.plm[PSR.cpl] and
    vmmmatch] and
    ((not PMCi.pm and PSR.up) or
    ( PMCi.pm and PSR.pp)) and
```

```
(not halted or PMCi.ch)
```

Figure 3-10 defines the PSR control fields that affect performance monitoring. For a detailed definition of how the PSR bits affect event monitoring and control accessibility of PMD registers, please refer to Section 3.3.2 and Section 7.2.1 of Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual.*

Figure 3-10. Processor Status Register (PSR) Fields for Performance Monitoring

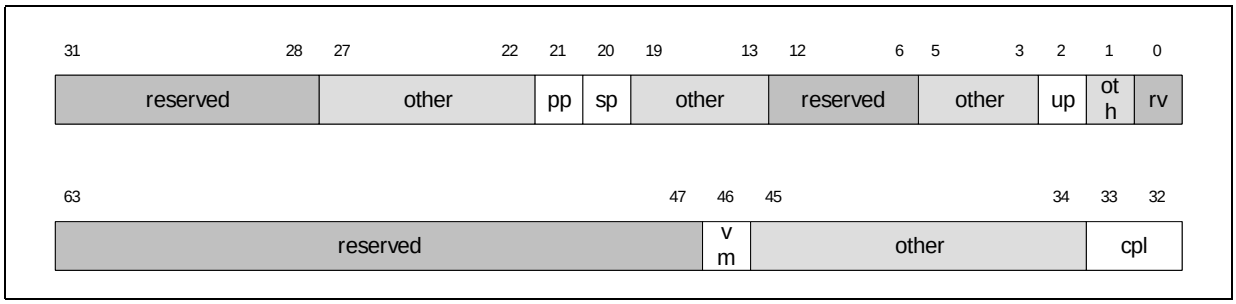


Table 3-6 defines per monitor controls that apply to $PMC_{4-19,32,34,37,38}$. As defined in Table 3-4, “Processor Performance Monitor Register Set", each of these PMC registers controls the behavior of its associated performance monitor data registers (PMD). The processor model-specific PMD registers associated with instruction/data EARs and the branch trace buffer should be read only when event monitoring is frozen (PMC_0.fr is one).

Any PMC fields labeled "ignore" or "ig" are ignored on writes and should be treated as don’t care on reads.

Table 3-6. Performance Monitor PMC Register Control Fields (PMC_{4-19})

Field	Bits	Description
vm	34:33	Virtual machine mask — Controls performance monitor operation relative to the virtual machine state. Each of the two bits corresponds to one of the states of PSR.vm. A mask bit value of 1 indicates that the monitor is enabled at that vm state. 01 — enable when psr.vm is off 10 — enable when psr.vm is set 11/00 — enable regardless of psr.vm
pm	6	Privileged monitor — When 0, the performance monitor is configured as a user monitor and enabled by PSR.up. When PMC.pm is 1, the performance monitor is configured as a privileged monitor, enabled by PSR.pp, and PMD can only be read by privileged software. Any read of the PMD by non-privileged software in this case will return 0.
plm	3:0	Privilege Level Mask — Controls performance monitor operation for a specific privilege level. Each bit corresponds to one of the 4 privilege levels, with bit 0 corresponding to privilege level 0, bit 1 with privilege level 1, etc. A bit value of 1 indicates that the monitor is enabled at that privilege level. Writing zeros to all plm bits effectively disables the monitor. NOTE: With plm set to 0, the processor may not preserve the value of the corresponding PMD register(s).

3.3.1.1 Notes on the Execution Trace Buffer

3.3.2 Performance Counter Registers

Performance Monitors are not shared between hardware threads. Each hardware thread has its own set of 16 generic performance counter (PMC/PMD_{4-19}) pairs.

PMC/PMD pairs are not entirely symmetrical in their ability to count events. Please refer to Section 3.3.3 for more information.

The figure and table below define the layout of the processor Performance Counter Configuration Registers (PMC_{4-19}). The main task of these configuration registers is to select the events to be monitored by the respective performance monitor data counters. The event selection (es) and unit mask (umask) in the PMC registers perform the selection of the event. The rest of the fields in PMCs specify under what conditions the counting should be done (plm, pm, vm, ch, all), how the counter should be incremented (thresh), and what needs to be done if the counter overflows (oi).

Figure 3-11. Processor Generic PMC Registers (PMC_{4-19})

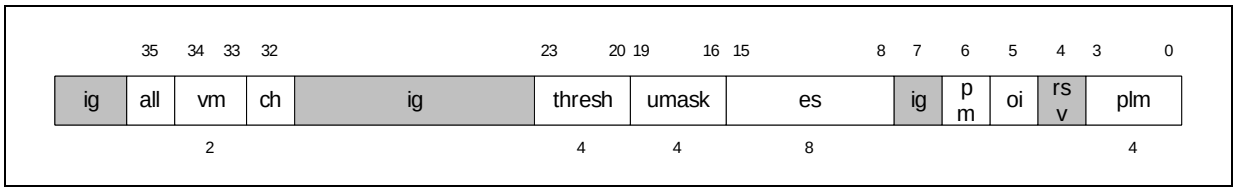


Table 3-7. Processor Generic PMC Register Fields (PMC_{4-19})

Field	Bits	Description
all	35	All threads; This bit selects whether or not to monitor just the self thread or both threads. If 1, events from both threads are monitored; If 0, only self thread is monitored.
vm	34:33	Virtual Machine Mask — This field control counting based on the psr.vm filed value.
ch	32	Count Halted state — if this bit is set, counting during low-power halted state is enabled
thresh	23:20	Threshold — Enables thresholding for "multi-occurrence" events. When threshold is zero, the counter sums up all observed event values. When the threshold is non-zero, the counter increments by one in every cycle in which the observed event value exceeds the threshold.
umask	19:16	Unit Mask — Combined with .es selects the performance event to be monitored.
es	15:8	Event select — Combined with .umask selects the performance event to be monitored.
pm	6	Privileged Monitor. See Table 3-6.
oi	5	Overflow interrupt — When 1, a Performance Monitor Interrupt is raised and the performance monitor freeze bit (PMC_0.fr) is set when the monitor overflows. When 0, no interrupt is raised and the performance monitor freeze bit (PMC_0.fr) remains unchanged. Counter overflows generate only one interrupt. Setting the corresponding PMC_0 bit on an overflow will be independent of this bit.
plm	3:0	Privilege Level Mask. See Table 3-6.

The figure and table below the layout of the processor Performance Counter Data Registers (PMD_{4-19}). A counter overflow occurs when the counter wraps (i.e., a carry out from bit 46 is detected). Software can force an external interruption or external notification after N events by preloading the monitor with a count value of 2^{47} – N.

When accessible, software can continuously read the performance counter registers PMD_{4-19} without disabling event collection. Any read of the PMD from software without the appropriate privilege level will return 0 (See "plm" in table above). The processor ensures that software will see monotonically increasing counter values.

Figure 3-12. Processor Generic PMD Registers (PMD_{4-19})

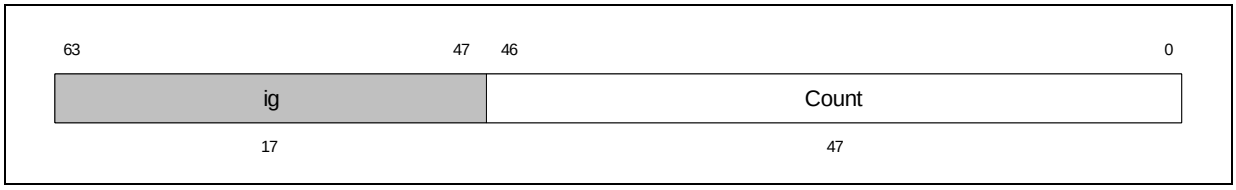


Table 3-8. Processor Generic PMD Register Fields

Field	Bits	Description
count	46:0	Event Count. The counter is defined to overflow when the count field wraps (carry out from bit 46).

3.3.3 Performance Monitor Event Counting Restrictions Overview

Similar to other Itanium processors, not all performance monitoring events can be monitored using any generic performance monitor counters (PMD_{4-19}). The following need to be noted when determining which counter to be used to monitor events.

The processor supports a large number of performance events that are collected in a decentralized manner. Many constraints posed by previous Intel Itanium generations were eliminated. In turn, the processor poses some restriction in terms of counter associativity.

Each events counter affinity is indicated by the counter affinity field in the event description table. The field is a hexadecimal number, where a set bit *i* indicates that PMC/PMD pair *i* can count the event. For example, a value of 0x05550 means $PMD_{14, 12, 10, 8, 6 \text{ and } 4}$ can count this particular event.

The associativity was designed to allow significant flexibility.

More details can be found in the Performance Monitor Event Section (Appendix 4.1.5).

3.3.4 Performance Monitor Overflow Status Registers ($PMC_{0,1,2,3}$)

As previously mentioned, the processor supports 16 performance monitoring counters per thread. The overflow status of these 16 counters is indicated in register PMC_0. As shown in the figure and table below, only PMC_0[19:4,0] bits are populated. All other overflow bits are ignored, i.e., they read as zero and ignore writes.

Figure 3-13. Processor Performance Monitor Overflow Status Registers ($PMC_{0,1,2,3}$)

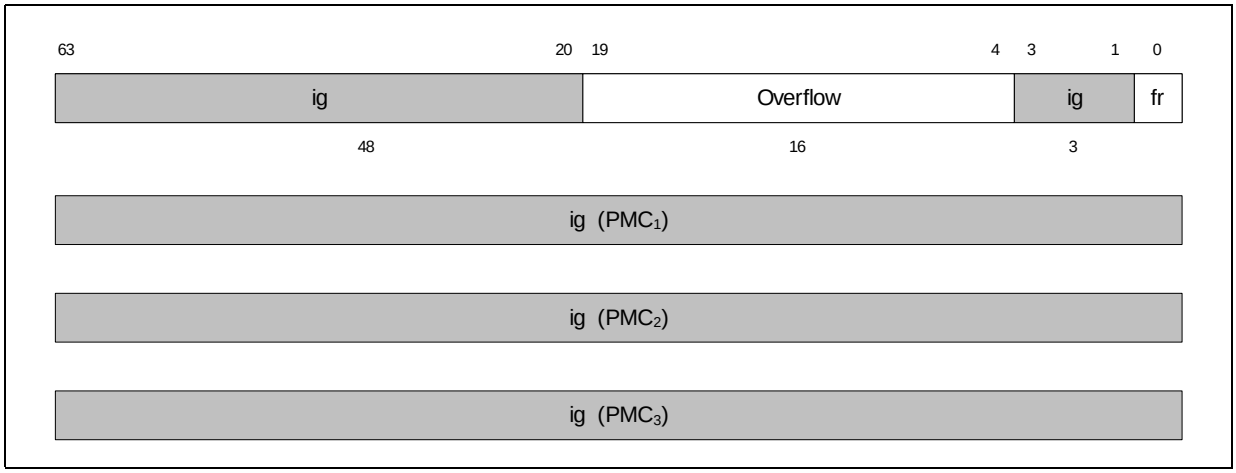


Table 3-9. Processor Performance Monitor Overflow Register Fields ($PMC_{0,1,2,3}$)

Register	Field	Bits	Description
$PMC_{1,2,3}$	ig	63:0	Read zero, Writes ignored.
PMC_0	ig	63:20	Read zero, Writes ignored.
PMC_0	overflow	19:4	Event Counter Overflow — When bit n is one, indicate that the PMD_n overflowed. This is a bit vector indicating which performance monitor overflowed. These overflow bits are set on their corresponding counters overflow regardless of the state of the PMC.oi bit. Software may also set these bits. These bits are sticky and multiple bits may be set.
PMC_0	ig	3:1	Read zero, Writes ignored.
PMC_0	fr	0	Performance Monitor "freeze" bit — When 1, event monitoring is disabled. When 0, event monitoring is enabled. This bit is set by hardware whenever a performance monitor overflow occurs and its corresponding overflow interrupt bit (PMC.oi) is set to one. SW is responsible for clearing it. When the PMC.oi bit is not set, then counter overflows do not set this bit.

3.3.5 Instruction Address Range Matching

The processor allows event monitoring to be constrained to a range of instruction addresses. Once programmed with these constraints, only the events generated by instructions with their addresses within this range are counted using PMD_{4-19} and tracked by the instruction cache, data cache and IP EAR as well as the ETB. The four architectural Instruction Breakpoint Register Pairs $IBRP_{0-3}$ (IBR_{0-7}) are used to specify the desired address ranges. Using these IBR pairs it is possible to define up to 4 different address ranges (or 2 address ranges in "fine mode") that can be used to qualify event monitoring.

Once programmed, each of these 4 address restrictions can be applied separately to all events that are identified to do so. The event, IA64_INST_RETIRED, is the only event that can be constrained using any of the four address ranges. Events described as instruction prefetch events can only be constrained using the address range 2 (IBRP1). All other events can only use the first address range (IBRP0) and this range will be considered as the default for this section.

In addition to constraining events based on instruction addresses, the processor allows event qualification based on the opcode of the instruction, and, if applicable, the data reference address and data reference type. These are done by applying these constraints to the same 4 instruction address ranges described in this section. These features are explained in Section 3.3.6, “Opcode Match Check”, Section 3.3.7, “Data Address Range Matching (PMC_DAM_CFG)”, and Section 3.3.8, “Data Reference Type Matching (PMC_DAM_CFG)”. See also Section 4.2.7.2, “Asynchronous Data References and Event Matching Constraints” for important information regarding opcode matching and data reference events.

3.3.5.1 PMC_IAM_CFG

The Performance Monitoring Instruction Address Match Configuration register is the main control register for Instruction Address Range matching feature. In addition to this register, PMC_OPM0_MSK also controls certain aspects of this feature as explained below.

Figure 3-14 and Table 3-10 describe the fields of PMC_IAM_CFG.

Instruction address range checking is controlled by the "ignore address range check" bit (PMC_OPM0_MSK.ig_ad and PMC_IAM_CFG.ig_ibrp0). When PMC_OPM0_MSK.ig_ad is one (or PMC_IAM_CFG.ig_ibrp0 is one), all instructions are

included (i.e. un-constrained) regardless of IBR settings. When both PMC_OPM_CFG.ig_ad and PMC_IAM_CFG.ig_ibrp0 are zero, the instruction address range check based on the IBRP0 settings is applied to all Itanium code fetches.

The processor compares every Itanium instruction fetch address IP{63:0} against the address range programmed into the architectural instruction breakpoint register pair $IBRP_0$. Regardless of the value of the instruction breakpoint fault enable (IBR x-bit), the following expression is evaluated for the processor's $IBRP_0$:

```
match(IP, PSR.cpl, IBR0.addr, IBR1.mask, IBR1.plm) =
    (IP[63:56] == IBR0.addr[63:56]) &&
    ((IP[55:4] & IBR1.mask[55:4]) == (IBR0.addr[55:4] & IBR1.mask[55:4])) &&
    (IBR1.plm[PSR.cpl] == 1)
```

For further details refer to chapter 7 in the 2.3 *Intel® Itanium® Architecture Software Developer's Manual*.

The events which occur before the instruction dispersal stage will fire only if this qualified match (IBRmatch) is true. This qualified match will be combined with the result of Opcode Matcher 0 PMC_OPM0_MSK, PMC_OPM0_MAT before being passed down the pipeline. The events which occur after instruction dispersal stage will use this new qualified match (IBRP0-OpCode0 match).

Figure 3-14. Instruction Address Range Configuration Register (PMC_IAM_CFG)

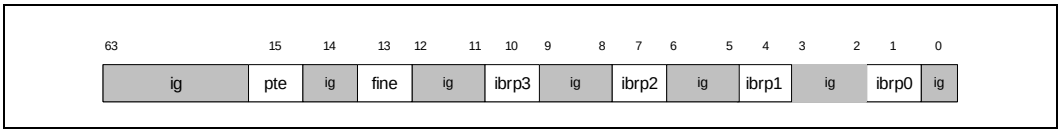


Table 3-10. Instruction Address Range Configuration Register Fields

Field	Bits	Description
pte	15	Prefetch Tag Enable — Enable RIL tagging of instruction prefetches. If set, this bit causes the ring interface logic to tag all instruction prefetches sourced by the MLI.
fine	13	Enable fine-mode address range checking (non power of 2) 1: $IBRP_{0,2}$ and $IBRP_{1,3}$ are paired to define two address ranges 0: Normal mode If set to 1, IBRP0 and IBRP2 define the lower and upper limits for address range 0 respectively; Similarly, IBRP1 and IBRP3 define the lower and upper limits for address range 1. Bits [63:16] of upper and lower limits need to be exactly the same but could have any value. Bits[15:4] of upper limit needs to be greater than bits[15:4] of lower limit. If an address falls in the range defined by and including the upper and lower limits, then a match will be signaled only in address ranges 0 or 1. Any event qualification based on address ranges 2 and 3 are not defined. NOTE: The mask bits programmed in IBRs 1,3,5,7 for bits [15:4] have no effect in this mode. When using fine mode address range 0, it is necessary to program both MC_IAM_CFG.ig_ibrp0,ig_brp2 to 0. Similarly, when using address range 1, it is necessary to set both PMC_IAM_CFG.ig_ibrp1,ig_ibrp3 to 0.
ig_ibrp3	10	1: No constraint 0: Address range 3 based on IBRP3 is enabled
ig_ibrp2	7	1: No constraint 0: Address range 2 based on IBRP2 is enabled
ig_ibrp1	4	1: No constraint 0: Address range 1 based on IBRP1 is enabled
ig_ibrp0	1	1: No constraint 0: Address range 0 based on IBRP0 is enabled

The $IBRP_0$ match is generated in the following fashion. Note that unless fine mode is used, arbitrary range checking cannot be performed since the mask bits are in powers of 2. In fine mode, two IBR pairs are used to specify the upper and lower limits of a range within a 64kB region (the upper bits of lower and upper limits must be exactly the same).

```
if PMC_IAM_CFG.Fine=0,
   IBRmatch0 = match(IP(63:0), PSR.cpl, IBR0(63:0), IBR1(55:0), IBR1.plm)
 else,
   IBRmatch0 = match (IP(63:16), PSR.cpl, IBR0(63:16), IBR1(55:16), IBR1.plm)
               and  (IP(15:4) >= IBR0(15:4))
               and  (IP(15:0) <= IBR4(15:4))

     ibrp0match =  (PMC_OPM0_MSK.ig_ad and
                   PMC_IAM_CFG.ig_ibrp0) or
                   IBRmatch0
```

The instruction range checking considers the address range specified by IBRPi only if PMC_OPM0_MSK.ig_ad(for i=0) and PMC_IAM_CFG.ig_ibrpi are 0.

3.3.5.2 Use of IBRP0 for Instruction Address Range Check — Exception 1

The address range constraint for instruction prefetch events is on the target address of these events rather than the address of the prefetch instruction. Therefore, $IBRP_1$ must be used for constraining these events. Calculation of $IBRP_1$ match is the same as that of $IBRP_0$ match with the exception that we use $IBR_{2,3,6}$ instead of $IBR_{0,1,4}$.

3.3.5.3 Use of IBRP0 for Instruction Address Range Check — Exception 2

The Address Range Constraint for RETIRED_INST_TAGGED event uses all four IBR pairs with a distinct subevent for each pair. Calculation of $IBRP_2$ match is the same as that of $IBRP_0$ match with the exception that $IBR_{4,5}$ (in non-fine mode) are used instead of $IBR_{0,1}$. Calculation of $IBRP_3$ match is the same as that of $IBRP_1$ match with the exception that we use $IBR_{6,7}$ (in non-fine mode) instead of $IBR_{2,3}$.

The instruction range check tag is computed early in the processor pipeline and therefore includes speculative, wrong-path as well as predicated off instructions.

As described in Section 3.2.3.1, “Combining Instruction Address, Opcode, Data Address and Memory Reference Type Matching”, the instruction range check result may be combined with the results of the IA-64 opcode match registers described in the next section.

3.3.5.4 Fine Mode Address Range Check

In addition to providing coarse address range checking described above, the processor can be programmed to perform address range checks in fine mode. The processor provides the use of two address ranges for fine mode. The first range is defined using IBRP0 and IBRP2 while the second is defined using IBRP1 and IBRP3. When properly programmed to use address range 0, all performance monitoring events that have been indicated to be able to qualify with IBRP0 would now qualify with this new address range (defined collectively by IBRP0 and IBRP2). Similarly, when using the address range 1, all events that could be qualified with IBRP1, now get qualified with this new address range.

A user can configure the processor PMU to use fine mode address range 0 by following these steps:

- Program IBRP0 and IBRP2 to define the instruction address range.
- Program PMC_OPM0_MSK[ig_ad,inv] = '00 to turn off default tags injected into tag channel 0
- Program PMC_IAM_CFG[ig_ibrp0,ig_ibrp2] = '00 to turn on address tagging based on IBRP0 and IBRP2.
- Program PMC_IAM_CFG.fine = 1

Similarly, a user can configure the processor PMU to use fine mode address range by following the same steps as above but this time with IBRP1 and 3. The only exception is that PMC_OPM1_MSK.[ig_ad,inv] need not be programmed.

3.3.6 Opcode Match Check

As shown in Figure 3-5, event monitoring can be constrained based on the Itanium encoding of an instruction. Registers PMC_OPM{0,1}_{MSK,MAT} and PMC_OPM_CFG allow configuring this feature. In the processor, registers PMC_OPM0_MSK, PMC_OPM0_MAT and PMC_OPM1_MSK, PMC_OPM1_MAT define 2 opcode matchers (Opcode matcher 0 (OpM0) and Opcode Matcher 1 (OpM1)). Register PMC_OPM_CFG controls how to apply opcode range checking to the four instruction address ranges defined by using IBRPs.

(See Section 4.2.7.2, “Asynchronous Data References and Event Matching Constraints” for important information regarding opcode matching and data reference events.)

3.3.6.1 PMC_OPM{0,1}_{MSK,MAT}

Figure 3-15, Figure 3-16 and Table 3-11, Table 3-12 describe the fields of the opcode matcher mask and match registers. Figure 3-17 and Table 3-13 describes the opcode matcher configuration register.

All combinations of setting for PMC_OPMx_MSK.{m,i,f,b} are supported. To match a A-slot instruction, it is necessary to set both PMC_OPMx_MSK.{m,i} to 1. To match all instruction types, all of PMC_OPMx_MSK.{m,i,f,b} should be set to 1. To ensure that all events are counted independent of the opcode matcher, all mifb and all mask bits of PMC_OPMx_MSK should be set to 1 (all opcodes match) while keeping the inv bit cleared.

Once the opcode matcher constraints are generated, they are ANDed with the address range constraints available on 4 IBRP channels to form 4 combined address range and opcode match ranges as described. The constraints defined by OpM0 are ANDed with address constraints defined by IBRP0 and IBRP2 to form combined constraints for channels 0 and 2. Similarly, the constraints defined by OpM1 are ANDed with address constraints defined by $IBRP_1$ and $IBRP_3$ to form combined constraints for channels 1 and 3.

Figure 3-15. Opcode Matcher Mask Registers

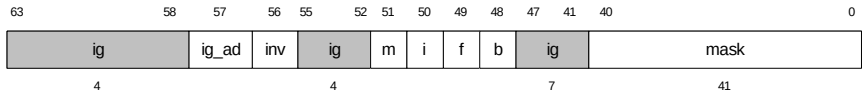


Table 3-11. Opcode Matcher Mask Register Fields

Field	Bits	Description
ig_ad	57	Ignore Instruction Address Range Checking for tag channel0 If set to 1, all instruction addresses are considered for events. If 0, IBRs 0–1 will be used for address constraints. NOTE: This bit is not implemented in PMC_OPM1_MSK, but behaves as if set to 0.
inv	56	Invert Range Check. for tag channel 0 If set to 1, the address ranged specified by IBRP0 is inverted. Effective only when ig_ad bit is set to 0. NOTE: This bit is not implemented in PMC_OPM1_MSK, but behaves as if set to 0.
m	51	If 1: match if opcode is an M-slot
i	50	If 1: match if opcode is an I-slot
f	49	If 1: match if opcode is an F-slot
b	48	If 1: match if opcode is an B-slot
mask	40:0	Bits that mask Itanium® instruction encoding bits. Any of the 41 syllable bits can be selectively masked. If mask bit is set to 1, the corresponding opcode bit is not used for opcode matching.

Figure 3-16. Opcode Matcher Match Registers

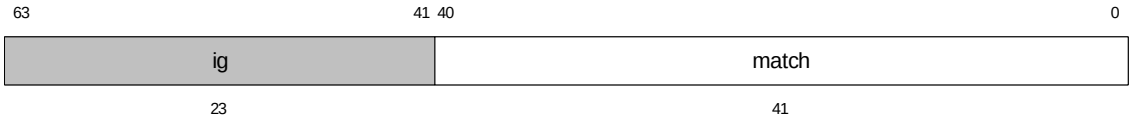


Table 3-12. Opcode Matcher Match Register Field

Field	Bits	Description
match	40:0	Opcode bits against which to match the Itanium instruction encoding Each opcode bit has a corresponding bit position here.

3.3.6.2 PMC_OPM_CFG

The Performance Monitoring Configuration register PMC_OPM_CFG controls whether or not to apply opcode matching in event qualification. As mentioned earlier, opcode matching is applied to the four instruction address ranges defined by using IBRPs.

Figure 3-17. Opcode Match Configuration Register

Table 3-13. Opcode Match Configuration Register Fields

Field	Bits	Description
ig	63:4	Ignored bits
Ch3_ig_OpC	3	1: Tag channel3 events (RETIRED_INST_TAGGED.IAM3_OPM1) won't be constrained by opcode 0: Tag channel2 events will be opcode constrained by OpM1
Ch2_ig_OpC	2	1: Tag channel2 events (RETIRED_INST_TAGGED.IAM2_OPM0) won't be constrained by opcode 0: Tag channel2 events will be opcode constrained by OpM0
Ch1_ig_OPC	1	1: tag channel1 events (RETIRED_INST_TAGGED.IAM1_OPM1) won't be constrained by opcode 0: tag channel1 events will be opcode constrained by OpM1
Ch0_ig_OPC	0	1: Tag channel0 PMU events will not be constrained by opcode 0: Tag channel0 PMU events (including RETIRED_INST_TAGGED.IAM0_OPM0) will be opcode constrained by OpM0

For opcode matching purposes, an Itanium instruction is defined by two items: the instruction type "itype" (one of M, I, F or B) and the 41-bit encoding "enco{40:0}" defined the *Intel® Itanium® Architecture Software Developer's Manual*. Each instruction is evaluated against each opcode match register (OpCM0 and OpCM1) as follows:

```
Match(OpCM[i]) = imatch(itype, OpCM[i].mifb) AND
                 ematch(enco,OpM[i].match,OpM[i].mask))
```

Where:

```
imatch(itype,OpCMi.mifb) = (itype=M AND PMC_OPMi_MSK.m) OR
                           (itype=I AND PMC_OPMi_MSK.i) OR
                           (itype=F AND PMC_OPMi_MSK.f) OR
                           (itype=B AND PMC_OPMi_MSK.b)

 ematch(enco,match,mask) = AND b=40..0 ((enco{b}=match{b}) OR mask{b})
```

The IBRP matches are advanced with the instruction pointer to the point where opcodes are being dispersed. The matches from opcode matchers are ANDed with the IBRP matches at this point.

This produces two opcode match events that are combined with the instruction range check tag (IBRRangeTag, see Section 3.3.5, “Instruction Address Range Matching”) as follows:

```
Tag(IBRChnl0) = Match(OpCM0) and IBRRangeTag0
 Tag(IBRChnl1) = Match(OpCM1) and IBRRangeTag1
 Tag(IBRChnl2) = Match(OpCM0) and IBRRangeTag2
 Tag(IBRChnl3) = Match(OpCM1) and IBRRangeTag3
```

As shown in Figure 3-5, the 4 tags, Tag(IBRChnli; i=0–3) are staged down the processor pipeline until instruction retirement and can be selected as a retired instruction count event (see event description "RETIRED_INST_TAGGED"). In this way, a performance counter ($PMC/PMD_{4–19}$) can be used to count the number of retired instructions within the programmed range that match the specified opcode(s).

3.3.7 Data Address Range Matching (PMC_DAM_CFG)

For instructions that reference memory, the processor allows event counting to be constrained by data address ranges. The 4 architectural Data Breakpoint Registers (DBRs) can be used to specify the desired data address range. The actual range is defined as the OR combination of the DBRPs used. For further qualifications selected via this register, refer to Section 3.3.8, "Data Reference Type Matching (PMC_DAM_CFG)". See also Section 4.2.7.2, "Asynchronous Data References and Event Matching Constraints" for important information regarding data address range matching and data reference events.

Figure 3-18 and Table 3-14 describe the fields of register PMC_DAM_CFG.

Figure 3-18. Data Address Match Configuration Register

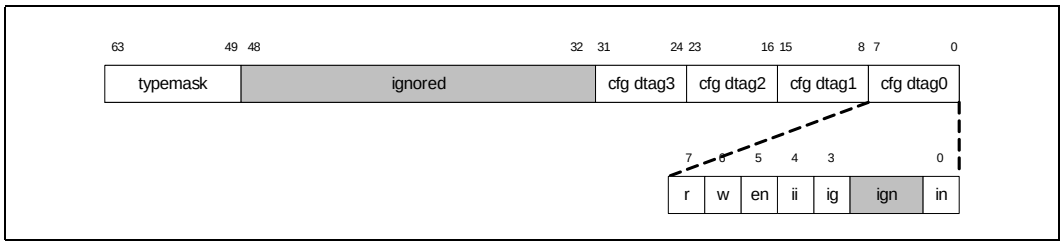


Table 3-14. Memory Pipeline Event Constraints Fields (PMC_DAM_CFG) (Sheet 1 of 2)

Field	Bits	Description
typemask	63:49	Memory Optype Mask — this is a bit-vector to constrain the filtering for PMU, data EAR, and data debug triggers to specific memory pipe operations. Any number or combination of these bits may be set, permitting any arbitrary group of operations to cause triggers. A value of all 0s for this field indicates that data optype constraints are disabled. However, a value of all 1s will filter out some operations as the typemask does not represent all memory operations. Note that this type mask applies to all four DBRs. Matches on specific DBRs are further constrained by the r and w bits described below. 63 — misc DTB ops (thash, ttag, tak, tpa, probe) 62 — DTB TLB transfers 61 — PTCs and PTRs 60 — snoops 59 — LOAD_HPW 58 — HW_PREF 57 — fc 56 — SEMAPHORE 55 — LFETCH 54 — STORE_RSE 53 — STORE_FP 52 — STORE_INT 51 — LOAD_RSE 50 — LOAD_FP 49 — LOAD_INT
cfgdtag3	31:24	These bits determine whether and how $DBRP_3$ should be used for constraining memory pipeline events, for PMU tagging, data EAR tagging, and/or data debug triggers. The bits match those defined below for $DBRP_0$; simply add 24 to the bit number in cfgdtag0 to get the corresponding bit position in cfgdtag3.
cfgdtag2	23:16	These bits determine whether and how $DBRP_2$ should be used for constraining memory pipeline events, for PMU tagging, data EAR tagging, and/or data debug triggers. The bits match those defined below for $DBRP_0$; simply add 16 to the bit number in cfgdtag0 to get the corresponding bit position in cfgdtag2.
cfgdtag1	15:8	These bits determine whether and how $DBRP_1$ should be used for constraining memory pipeline events, for PMU tagging, data EAR tagging, and/or data debug triggers. The bits match those defined below for $DBRP_0$; simply add 8 to the bit number in cfgdtag0 to get the corresponding bit position in cfgdtag1.

Table 3-14. Memory Pipeline Event Constraints Fields (PMC_DAM_CFG) (Sheet 2 of 2)

Field	Bits	Description
cfgdtag0	7:0	These bits determine whether and how $DBRP_0$ should be used for constraining memory pipeline events, for PMU tagging, data EAR tagging, and/or data debug triggers. The individual bits are as follows: 0 — in: Invert the sense of the address match (i.e. matching addresses will *not* trigger a DBR match; all other addresses *will* cause a DBR match) 3 — id: Ignore the generated data match (data address, privilege, and read/write) 4 — ii: Ignore the inbound syllable tag (instruction address & opcode match) 5 — en: Enable DBRP0 - operations that match $DBR_0/R_0/W_0$/typemask get tagged for PMU and/or data EAR 6 — w0:Write bit for PMU, data EAR (enable write type operations to be tagged, analogous to the DBR.w bit) 7 — r0: Read bit for PMU, data EAR (enables read type operations to be tagged, analogous to the DBR.r bit

Note that the DBRs are overloaded to detect address matches for several different features. Firmware must ensure that any particular DBR is not configured for two conflicting uses at the same time.

In particular, in order to allow simultaneous use of some DBRs for Performance Monitoring and the others for debugging (the architected purpose of these registers), separate mechanisms are provided for enabling DBRs. For example, the DBR r/w bits must be cleared to 0 for DBRs which are going to be used for the PMU. PSR.db bit has no effect when DBRs are used for this purpose.

The various features are listed below, together with the bits which configure a DBR to be used for that particular feature, and the bits which configure matches for that feature:

- Architected data debug breakpoint faults:
  - — enabled by the architected DBR_x.r and DBR_x.w bits (depending on instruction type);
  - — matching addresses are selected via the architected DBR_x.addr and DBR_x.mask fields;
  - — matching privilege levels are selected via the architected DBR_x.plm field.
- PMU matches:
  x is 0..3, depending on the applicable DBR
  - — enabled by the r*x* and w*x* bits in this register (depending on instruction type);
  - — matching addresses and privilege levels are selected as described above for data debug faults, with the addition that bit 0 of the cfgdtag*x* field in this register allows the sense of the address match to be inverted;
  - — matching operations must also be one of the types selected by the typemask field in this register;
  - — *HOWEVER*, if the ignore data match bit (cfgdtag*x*.id) in this register is set, all of the above enabling and matching logic is disabled, and a PMU match will be generated based solely on the inbound syllable tag and master PMU enable bit (below);
  - — matching operations must also have a syllable tag, unless the ignore inbound syllable tag bit (cfgdtag*x*.ii) is set to ignore syllable tags;
  - — and the master PMU data match enable bit (cfgdtag*x*.en) in this register must be set;
  - — *HOWEVER*, if all four cfgdtag*x*.en bits are zero (which would normally disable all PMU tagging), *all* enabling and matching logic is disabled, and every single operation will match.

- D-EAR matches:
  - — All enabling and matching is identical to that described above for PMU matches, including all of the cfgdtag[x].en qualification; plus
  - — the operation must also meet the additional qualifications specified in Section 3.3.11, “Data Cache EAR” for the specific type of data events that are desired to be captured by the D-EAR.

3.3.8 Data Reference Type Matching (PMC_DAM_CFG)

For instructions that reference memory, the processor allows event counting to be constrained by data reference type. The following is a list of primary data reference types:

- LOAD_INT: integer loads (see DATA_REF.LOAD_INT)
- LOAD_FP: floating point loads (see DATA_REF.LOAD_FP)
- LOAD_RSE: RSE loads (see DATA_REF.LOAD_RSE)
- LOAD_HPW: hardware page walker loads (see DATA_REF.LOAD_HPW)
- STORE_INT: integer stores (see DATA_REF.STORE_INT)
- STORE_FP: floating point stores (see DATA_REF.STORE_FP)
- STORE_RSE: RSE stores (see DATA_REF.STORE_RSE)
- SEMAPHORE: semaphores (see DATA_REF.SEMAPHORE)
- LFETCH: software prefetches (see DATA_REF.LFETCH)
- HW_PREF: hardware prefetches (see DATA_REF.HW_PREF and RIL_REQ_REF_DATA.WB_MLD_BUDDY)

The primary data reference types (see Section 4.2.7.1, “Primary Data Reference Types”) are more widely supported by a variety of PMU events than the following data reference types are:

- Cache flushes (fc)
- TLB purges (ptc, ptr)
- Data TLB transfers
- Other Data TLB ops (thash, ttag, tak, tpa, probe)
- Snoops

The data references types to be matched are encoded in PMC_DAM_CFG.typemask as shown in Table 3-14.

The data reference type match operates independently of, but can be combined with other types of PMU event counting constraints. See Section 4.2.7.2, “Asynchronous Data References and Event Matching Constraints” for important information regarding data reference type matching and data reference events.

3.3.9 Event Address Registers

This section defines the register layout for the processor instruction cache and data cache event address registers (EARs). Sampling of the following events is supported on the processor:

- Instruction demand fetch misses
- Instruction TLB misses

- Data cache accesses
- Data TLB accesses
- ALAT operations

The EARs are configured through two PMC registers (PMC_DEAR_CFG and PMC_IEAR_CFG). The EAR unit masks allow software to specify event collection latency thresholds to hardware. Instruction and data addresses, operation latencies, request and response type and other captured event parameters are provided in five PMD registers (PMD_IEAR_STAT, PMD_IEAR_IADDR, PMD_DEAR_STAT, PMD_DEAR_IADDR, PMD_DEAR_DADDR). The instruction and data cache EARs report the latency of captured cache events and allow latency thresholding to qualify event capture. Event address data registers contain consistent data only when event collection is frozen (PMC_0.fr is one). Reads of EAR PMDs while event collection is enabled return undefined values.

3.3.10 Instruction Cache EAR

The instruction event address configuration register (PMC_IEAR_CFG) can be programmed to monitor either L1 instruction cache or instruction TLB miss events. Figure 3-19 and Table 3-15 detail the register layout of PMC_IEAR_CFG. Table 3-21 describes the associated event address data registers PMD_IEAR_{STAT,IADDR}.

Figure 3-19. Instruction Event Address Configuration Register (PMC_IEAR_CFG)

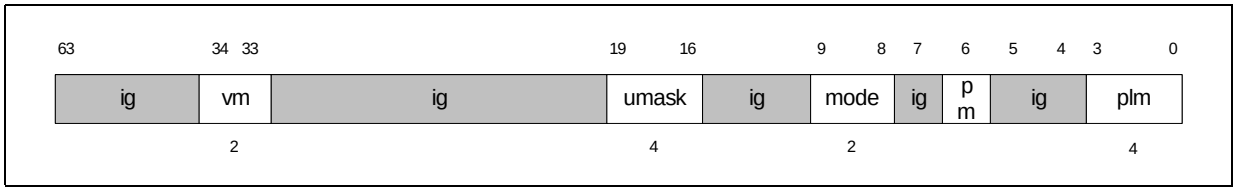


Table 3-15. Instruction Event Address Configuration Register Fields (PMC_IEAR_CFG)

Field	Bits	Description
vm	34:33	See Table 3-6
umask	19:16	Instruction EAR unit mask mode 01: instruction cache unit mask (definition see Table 3-16) mode 10: instruction TLB unit mask (definition see Table 3-17)
mode	9:8	Instruction EAR mode selector: 00:Not active 01: Instruction cache fetch misses 10: TLB accesses
pm	6	See Table 3-6.
plm	3:0	See Table 3-6.

Table 3-16. PMC_IEAR_CFG.Umask Field in Instruction Cache Mode (Sheet 1 of 2)

Umask Bits 19:16	Latency Threshold [CPU cycles]	Umask Bits 19:16	Latency Threshold [CPU cycles]
0000	(Any latency)	1000	≥ 384
0001	≥ 16	1001	≥ 512
0010	≥ 24	1010	≥ 1024
0011	≥ 32	1011	≥ 1536

Table 3-16. PMC_IEAR_CFG.Umask Field in Instruction Cache Mode (Sheet 2 of 2)

Umask Bits 19:16	Latency Threshold [CPU cycles]	Umask Bits 19:16	Latency Threshold [CPU cycles]
0100	≥ 64	1100	≥ 2048
0101	≥ 96	1101	≥ 3072
0110	≥ 128	1110	≥ 4096
0111	≥ 256	1111	≥ 5120

Table 3-17. PMC_IEAR_CFG.Umask Field in Instruction TLB Mode

DTLB Miss Type	PMC.umask[19:16]	Description
None	x000	Disabled; nothing will be counted
All	x111	All — any combination valid
FL TLB miss	xxx1	L1 ITLB misses
ML TLB miss	xx1x	L2 ITLB misses
HPW miss	x1xx	HPW misses

When the PMC_IEAR_CFG.mode is set to 01, instruction cache accesses are monitored. When it is set to 10, instruction TLB misses are monitored. The interpretation of the umask field depend on the setting of this field and is described in Table 3-16 and Table 3-17 respectively. The interpretation of the performance monitor data registers PMD_IEAR_STAT and PMD_IEAR_IADDR is described in Table 3-18 below and the field validity in the different modes in Table 3-19.

When PMC_IEAR_CFG.mode is '01, the instruction event address register captures instruction addresses, access latency, RAB status and responder type for instruction demand fetch misses. Only accesses whose latency meets or exceed the threshold in PMC_IEAR_CFG.umask will be captured.

Figure 3-20. Instruction Cache EAR Data Registers Format

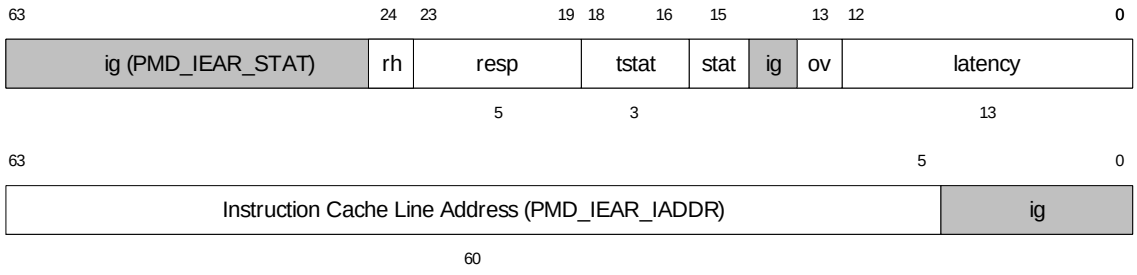


Table 3-18. Instruction Cache EAR Data Registers Field Descriptions

Register	Fields	Bit Range	Description
PMD_IEAR_STAT	rh	24	RAB hit — when set, this field indicates the access hit in the RAB. See for more on the RAB.
	resp	23:19	Responder Type: 0x00: Off core — no information due to “no information” 0x02: MLI hit 0x10: Off core — signaled no information 0x11: non-DRAM socket local system address 0x12: non-DRAM socket remote system address 0x13: non DRAM system address — no details 0x14: LLC hit, minimum latency 0x15: LLC hit, local core snoop required, no forwarding 0x16:LLC hit, local core snoop required, core cache data forwarded 0x17: LLC hit — no details 0x18: local DRAM, no remote snoops 0x19: local DRAM, remote snoops required, no forwarding 0x1A:local DRAM, remote snoops required, remote cache data forwarded 0x1B: local DRAM — no further details 0x1C: remote DRAM, no additional snoops required 0x1D: remote DRAM, additional snoops required, no forwarding 0x1E: remote DRAM, additional snoops required, remote cache data forwarded 0x1F: remote DRAM — no further details
	tstat	19:16	TLB Status: 000: no data 001: FL Instruction TLB hit 010: ML Instruction TLB hit 011: HPW hit 100: other — implied page fault
	stat	15	Status: 0: No valid information in PMD_IEAR_IADDR and rest of PMD_IEAR_STAT 1: Valid information in PMD_IEAR_* NOTE: This bit should be cleared before the EAR is reused.
	overflow	13	Overflow — If 1, latency counter has overflowed one or more times before data was returned
	latency	12:0	Latency in CPU clocks
PMD_IEAR_IADDR	Instruction Cache Line Address	63:5	Virtual address of the cache line access

Table 3-19. Instruction EAR Status Register Field Validity in Different Modes

Mode	Resp	rh	tstat	stat	ov	latency
00 (inactive)	undef	undef	undef	undef	undef	undef
01 (I cache load miss mode)	valid	valid	valid	valid	valid	valid
10 (TLB mode)	undef	undef	valid	valid	undef	undef

As defined in Table 3-18, the address of the instruction cache line missed the instruction cache is provided in PMD_IEAR_IADDR. If no qualified event was captured, it is indicated in PMD_IEAR_STAT.stat. The latency of the captured instruction cache miss in CPU clock cycles is provided in PMD_IEAR_STAT.latency.

When PMC_IEAR_CFG.mode is 10, the instruction event address register captures addresses of instruction TLB misses. The unit mask allows event address collection to capture specific subsets of instruction TLB misses. Table 3-17 summarizes the instruction TLB umask settings. All combinations of the mask bits are supported.

3.3.11 Data Cache EAR

The data event address register can be programmed to monitor either data loads, data stores, data TLB misses, or ALAT misses. Figure 3-21 and Table 3-20 detail the register layout of PMC_DEAR_CFG. Figure 3-22 describes the associated event address data registers PMD_DEAR_{STAT,IADDR,DADDR}. The mode bits in PMC_DEAR_CFG select data cache, data TLB, or ALAT monitoring. The interpretation of the umask field and registers PMD_DEAR_* depends on the setting of the mode bits and is described in Section 3.3.11.1, “Data Cache Monitoring” for data cache load monitoring, Section 3.3.11.2, “Data TLB Monitoring” for data TLB monitoring, and Section 3.3.11.3, “ALAT Monitoring” for ALAT monitoring.

Figure 3-21. Data Event Address Configuration Register (PMC_DEAR_CFG)

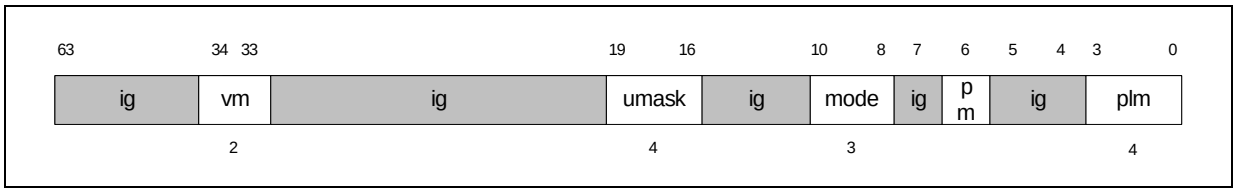


Table 3-20. Data Event Address Configuration Register Fields (PMC_DEAR_CFG)

Field	Bits	Description
vm	34:33	See Table 3-6
umask	19:16	Data EAR unit mask mode 01x: data cache unit mask (definition see Table 3-21) mode 100: data TLB unit mask (definition see Table 3-22) mode 11x: the field is ignored.
mode	10:8	Data EAR mode selector: 000:Not active 010: Data cache loads 011: Data cache stores 100: TLB accesses 110: ALAT misses 111: ALAT store hits
pm	6	See Table 3-6.
plm	3:0	See Table 3-6.

Table 3-21. PMC_DEAR_CFG.Umask Field in Data Cache Load Mode (010) (Sheet 1 of 2)

umask Bits 19:16	Latency Threshold [CPU cycles]	umask Bits 19:16	Latency Threshold [CPU cycles]
0000	(Any latency)	1000	≥ 384
0001	≥ 4	1001	≥ 512
0010	≥ 24	1010	≥ 1024
0011	≥ 32	1011	≥ 1536
0100	≥ 64	1100	≥ 2048

Table 3-21. PMC_DEAR_CFG.Umask Field in Data Cache Load Mode (010) (Sheet 2 of 2)

umask Bits 19:16	Latency Threshold [CPU cycles]	umask Bits 19:16	Latency Threshold [CPU cycles]
0101	≥ 96	1101	≥ 3072
0110	≥ 128	1110	≥ 4096
0111	≥ 256	1111	≥ 5120

Table 3-22. PMC_DEAR_CFG.Umask Field in Data TLB Mode

TLB Access Type	PMC.umask[19:16]	Description
None	0000	Disabled; nothing will be counted
FL TLB hit	xxx1	First level data TLB hits will be counted
ML TLB hit	xx1x	Mid level data TLB hits (with first level TLB miss) will be counted
HPW hit	x1xx	Hardware page walker hits (with TLB misses) will be counted
HPW miss	1xxx	Hardware page walker misses will be counted
All	1111	All - any combination valid

For snoops captured in TLB mode, the umask should be set to all ones (0xF), as the results (for the snoop-related capture) are undefined otherwise and should be ignored.

In ALAT mode (PMC_DEAR_CFG.mode=11x), the umask field is ignored.

Figure 3-22. Data Cache EAR Data Registers Format

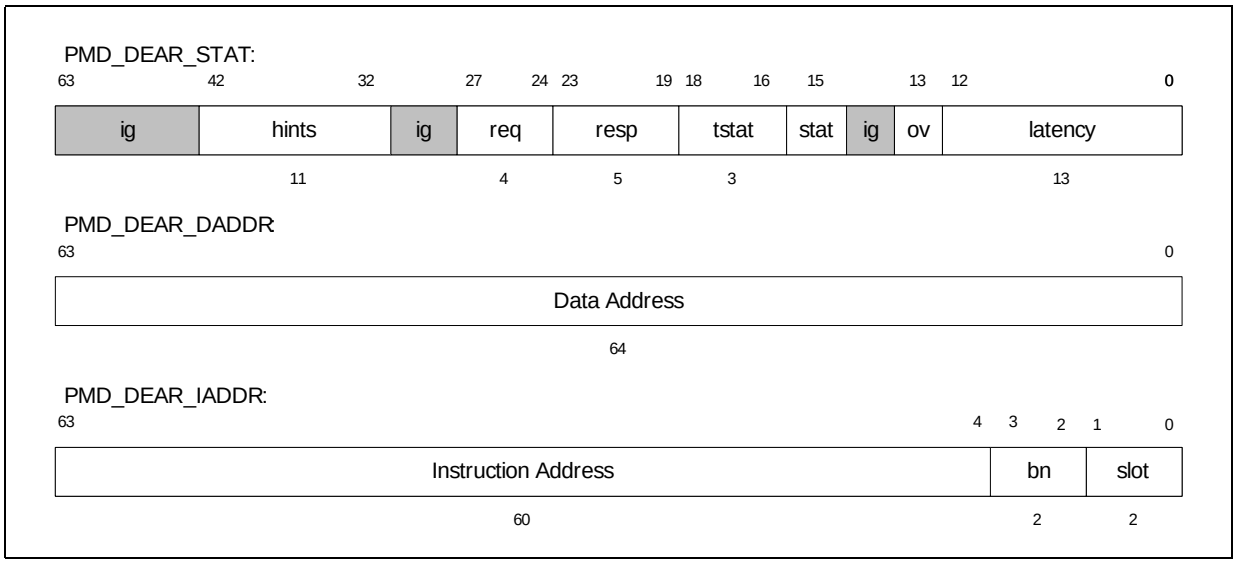


Table 3-23. Data Cache EAR Data Field Descriptions (Sheet 1 of 2)

Register	Fields	Bit Range	Description
PMD_DEAR_STAT	hints	42:32	DAHR Hints read out and applied to the captured operation (see the DPF chapter for a detailed description of these hint fields as they are defined for the processor, and a definition of their various values): 42 — BIAS 41 — NON_BLOCKING_SPEC_DISALLOWED 40:39 — PF_DROP 38:37 — HWPF 36 — LLC_LOCALITY 35:34 — MLD_LOCALITY
	req	27:24	Requestor Type: 0x00: No information 0x01: Int load 0x02: FP load 0x03: RSE load 0x04: Int store 0x05: FP store 0x06: RSE store 0x07: lfetch 0x08: Semaphore 0x09: fc 0x0A: Hardware prefetch 0x0B: HPW load 0x0C: Snoop 0x0D: PTC or PTR 0x0E: DTB TLB transfer 0x0F: misc DTB op (thash, ttag, tak, tpa, probe) ***Note:*** Table 3-25 below indicates which requestor types are valid in whichmode.
	resp	23:19	Responder Type: 0x00: Deferral: DTB miss/DTB NaT, latency up to HPW 0x01: FLD hit, implied FLD TLB hit 0x02: MLD hit, implied MLD TLB hit 0x04: SMQ hit, implied MLD TLB hit 0x08: Deferral: MLD NaT, FLD address NaT 0x10: Off core — no information 0x11: non-DRAM socket local system address 0x12: non-DRAM socket remote system address 0x13: non DRAM system address — no details 0x14: LLC hit, minimum latency 0x15: LLC hit, local core snoop required, no forwarding 0x16:LLC hit, local core snoop required, core cache data forwarded 0x17: LLC hit — no details 0x18: local DRAM, no remote snoops 0x19: local DRAM, remote snoops required, no forwarding 0x1A:local DRAM, remote snoops required, remote cache data forwarded 0x1B: local DRAM — no further details 0x1C: remote DRAM, no additional snoops required 0x1D: remote DRAM, additional snoops required, no forwarding 0x1E: remote DRAM, additional snoops required, remote cache data forwarded 0x1F: remote DRAM — no further data
	tstat	19:16	TLB Status: 000: No information 001: FL Data TLB hit 010: ML Data TLB hit 011: HPW hit 100: other — implied page fault

Table 3-23. Data Cache EAR Data Field Descriptions (Sheet 2 of 2)

Register	Fields	Bit Range	Description
	stat	15	Status: 0: No valid information in PMD_DEAR_*ADDR and rest of PMD_DEAR_STAT 1: Valid information in PMD_DEAR_STAT, PMD_DEAR_DADDR and in PMD_DEAR_IADDR as indicated by slot field. NOTE: This bits should be cleared before the EAR is reused.
	overflow	13	Overflow — If 1, latency counter has overflowed one or more times before data was returned
	latency	12:0	Latency in CPU clocks.
PMD_DEAR_DADDR	Data Address	63:0	64-bit virtual address of data item that caused miss. Exception: for snoops, this is the physical address.
PMD_DEAR_IADDR	Instruction Address	63:4	Virtual address of the first bundle in the dispersal window which was being executed at the time of the miss. If ".bn" is n, then n * 16 should be added to the address to arrive at the correct bundle address. Exception: for snoops, an address is recorded, it is however invalid.
	bn	3:2	Bundle field, indicates which of the executed bundles is associated with the captured miss
	slot	1:0	Slot field; if ≠ 3,indicates the Instruction bundle slot of memory instruction. A value of 3 indicates the captured address is not valid.

Table 3-24. PMD_DEAR_STAT Field validity in different modes

Mode	hints	req	resp	tstat	stat	ov	latency
000 (inactive)	undef	undef	undef	undef	undef	undef	undef
010 (D cache load mode)	valid	valid	valid	valid	valid	valid	valid
011 (D cache store mode)	valid	valid	=0x00 or 0x01	valid	valid	undef	undef
100 (TLB mode)	valid	valid	undef	valid	valid	undef	undef
110 (ALAT miss)	valid	valid	undef	undef	valid	undef	undef
111 (ALAT st hit)	valid	valid	undef	undef	valid	undef	undef

Table 3-25. Valid PMD_DEAR_STAT.req values (Sheet 1 of 2)

Request type	D cache load	D cache store	Data TLB	ALAT miss	ALAT hit
int load	valid		valid	ldc that misses	
fp load	valid		valid	ldc that misses	valid
RSE load	valid		valid		valid
HPW load			valid		
int store		valid	valid		valid
fp store		valid	valid		valid
RSE store		valid	valid		
lfetch			valid		
semaphore	valid	valid	valid		valid
HW prefetch			valid		
snoop			valid		
fc					

Table 3-25. Valid PMD_DEAR_STAT.req values (Sheet 2 of 2)

Request type	D cache load	D cache store	Data TLB	ALAT miss	ALAT hit
ptc			valid		
DTB TLB transfer			valid		
misc DTB ops			valid		

3.3.11.1 Data Cache Monitoring

If the Data EAR is configured to monitor the data cache, the umask is used as a load latency threshold as defined by Table 3-21.

As defined in Table 3-22, the instruction and data addresses as well as the load latency of a captured data cache events as well as the responder type for load accesses are presented to software in three registers PMD_DEAR_{STAT,DADDR,IADDR}. In addition the TLB status associated with the data access is reported. Note that in Data Cache Monitoring mode, the TLB umask is implicitly '1111 (Refer to Table 3-22). If no qualified event was captured, the valid bit in PMD_DEAR_STAT is zero.

Only the types of operations enabled in PMC_DAM_CFG.typemask will be sampled. Other types of instructions (such as `setf` and reads from ccv) cannot be monitored, even though the memory subsystem may be involved in their execution.

The detection of data cache load misses requires a load instruction to be tracked during multiple clock cycles from instruction issue to cache miss occurrence. Since multiple loads may be outstanding at any point in time and the processor data cache miss event address register can only track a single event at a time, not all data cache load misses may be captured. When the processor hardware captures the address of a load (called the monitored load), it ignores all other overlapped concurrent loads for at least 8 cycles, or until the status of the monitored load is determined (whichever comes later). If the monitored load turns out to be a something that is eligible to be captured, its parameters are then latched into PMD_DEAR_*. The processor randomizes the choice of which load instructions are tracked to prevent the same data cache load miss from always being captured (in a regular sequence of overlapped data cache load misses). This mechanism will sub-sample data cache events by a factor of 8 to remove capture bias in loops, making its accuracy sufficient to be used by statistical sampling or code instrumentation.

3.3.11.2 Data TLB Monitoring

If the Data EAR is configured to monitor data TLB accesses, the umask defined in Table 3-22 determines which data TLB misses are captured by the Data EAR. For TLB monitoring, all combinations of the mask bits are supported.

3.3.11.3 ALAT Monitoring

The Data EAR provides two ALAT-related modes:

- ALAT miss mode

  This mode captures ld.c and chk.a instructions that miss the ALAT, indicating an instance of unsuccessful speculation.

- ALAT store hit mode

  This mode captures store operations that hit the ALAT and clear an entry.

  Store operations do not include snoops, but do include xchg, semaphore and fetchadd operations.

Additionally, this mode only captures ALAT invalidations for the currently active back-end thread. While a store may invalidate an ALAT entry for the inactive thread, these invalidations will not be captured.

3.3.12 Execution Trace Buffer

The execution trace buffer provides information about the most recent Itanium control flow changes. The execution trace buffer configuration register (PMC_ETB_CFG) defines the conditions under which instructions which cause the changes to the execution flow are captured, and allows the trace buffer to capture specific subsets of these events.

In addition to the branches, the processor's ETB captures rfis, exceptions (excluding asynchronous interrupts) and and silently resteered (non-faulting failing) chk events. Passing chk instructions are not captured under any programming conditions (except when there is another capturable event).

In every cycle in which a qualified change to the execution flow happens, its source bundle address and slot number are written to the execution trace buffer. This event's target address is written to the next buffer location. If the target instruction bundle itself contains a qualified execution flow change, the execution trace buffer records that target instruction as a branch source instead. As a result, the branch trace buffer may contain multiple source address entries in sequence, which implies that the second (and subsequent) source entry implies a target entry with same bundle group address as the source.

The setting of PMC_IPEAR_CFG can override the setting of PMC_ETB_CFG. PMC_IPEAR_CFG is used to configure the Execution Trace Buffer's alternate modes: the IP-EAR. Please refer to the IP_EAR mode Section 3.3.12.2 for more information about this mode. PMC_IPEAR_CFG.mode must be set to 0 to enable normal execution trace capture in PMD_ETB_{0-23} and PMD_ETBEXT_{0-23} as described below. If PMC_IPEAR_CFG.mode is set to values other than 0, PMC_ETB_CFG's contents will be ignored.

- The Execution Trace Buffer fails to record the target address of an rfi. The buffer will show back-to-back branch entries instead of the usual branch-target-branch-target sequences. Historically back-to-back branch entries implies the target of the first branch entry is syllable 0 of the address of the issue group indicated by the 2nd branch entry. Failing to record the target of the rfi results in analysis tools assuming the rfi's target is at the address of the issue group of the next taken branch.

  It can only occur if the rfi is in an MBB or BBB bundle. There needs to be a WB2 replay in the rfi issue group (and squashed nops don't replay). So if there is a stop bit prior to the MBB or BBB bundle, and if the M or Bs are just nops, then there won't be an issue.

- When an external interrupt occurs in the shadow of a replay (where there are unexecuted IP's in the pipeline) the Execution Trace Buffer records one of the unexecuted IP's in the pipeline as the source IP of the interruption. In the normal (non-replay) case, the source IP of an interrupt event is one of the IP's of the last retired bundle group.

3.3.12.1 Execution Trace Capture

The subsequent subsections describe the operation of the Execution Trace Buffer when configured to capture an execution trace (or "enhanced" branch trace).

PMC_ETB_CFG defines the conditions under which execution flow changes are to be captured. These conditions are given in Figure 3-23 and Table 3-26, which refer to conditions associated with branch prediction and execution. These conditions are:

- Whether the target of the branch should be captured
- The path of the branch (not taken/taken), and
- Whether or not the branch path was mispredicted
- Whether or not the target of the branch was mispredicted
- What type of branch should be captured

All instructions eligible for capture are subject to filtering by the "plm" and "vm" fields but only branches are affected by PMC_ETB_CFG's other filters (tm, ptm, ppm, brt, rtg and cir) as well as the Instruction Addr Range and Opcode Match filters. The chk, rfi, and interruption, when selected, always log both source and target IP. To not log branches set one of the fields ppm, ptm, or tm to zero.

Figure 3-23. Execution Trace Buffer Configuration Register fields

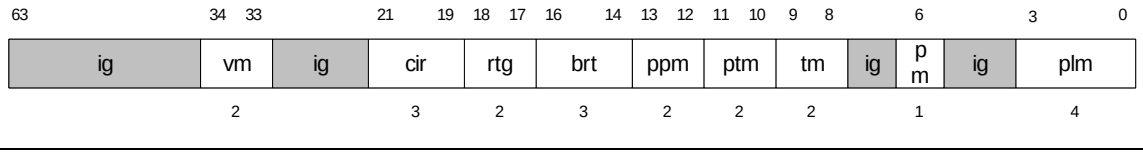


Table 3-26. Execution Trace Buffer Configuration Register field description (Sheet 1 of 2)

Field	Bits	Description
vm	34:33	VM mask
cir	21:19	Exception mask 1xx: capture chk instructions x1x: capture interrupts xx1: capture rfi instructions
rtg	18:17	Record Target 11: log target for all taken branches 10: log target for ip-relative branches only 01: log target for indirect taken branches only 00: do not log target
brt	16:14	Branch Type Mask: 111: all indirect branches captured 110: only calls and return branches will be captured 101: only IP-relative loopy branches will be captured (cloop, ctop, cexit, wtop, wexit) 011: only non-return indirect branches captured 010: only return branches will be captured 001: only IP-relative branches will be captured 000: all branches are captured
ppm	13:12	Predicted Predicate Mask: 11: capture branch regardless of predicate prediction outcome 10: branch predicted branch path (taken/not taken) correctly 01: branch mispredicted branch path (taken/not taken) 00: No branch is captured
ptm	11:10	Predicted Target Address Mask: 11: capture branch regardless of target prediction outcome 10: branch target address predicted correctly 01: branch target address mispredicted 00: No branch is captured

Table 3-26. Execution Trace Buffer Configuration Register field description (Sheet 2 of 2)

Field	Bits	Description
tm	9:8	Taken Mask: 11: all branches 10: Taken branches only 01: Not Taken branches only 00: No branch is captured
pm	6	See Table 3-6. Note: This bit is applied at the time the event's source address is captured. Once the source IP is captured, the target IP of this event is always captured even if the ETB is disabled.
plm	3:0	See Table 3-6. Note: This mask is applied at the time the event's source address is captured. Once the source IP is captured, the target IP of this event is always captured even if the ETB is disabled.

To capture all correctly predicted branches, the branch trace buffer configuration settings in PMC_ETB_CFG should be: tm=11, ptm=10, ppm=10, brt=000, rtg=11, cir=000

Either branches whose path was mispredicted can be captured (tm=11, ptm=11, ppm=01, brt=000) or branches with a target misprediction (tm=11, ptm=01, ppm=11, brt=000) can be captured but not both. A setting of tm=11, ptm=01, ppm=01, brt=000 will result in an empty buffer. If a branch's path is mispredicted, no target prediction is recorded.

Instruction Address Range Matching (Section 3.3.5) and Opcode Matching (Section 3.3.6) may also be used to constrain what is captured in the Branch Trace Buffer.

These twenty-four execution trace buffer registers PMD_ETB_{0-23} and their extension PMD_ETBEXT_{0-23} provide information about the outcome of a captured event sequence. Every ETB capture records 80 bits of data, which is accessible across a pair of PMDs, PMD_ETB_i and PMD_ETBEXT_i.

Figure 3-24. Execution Trace Buffer Entry Format

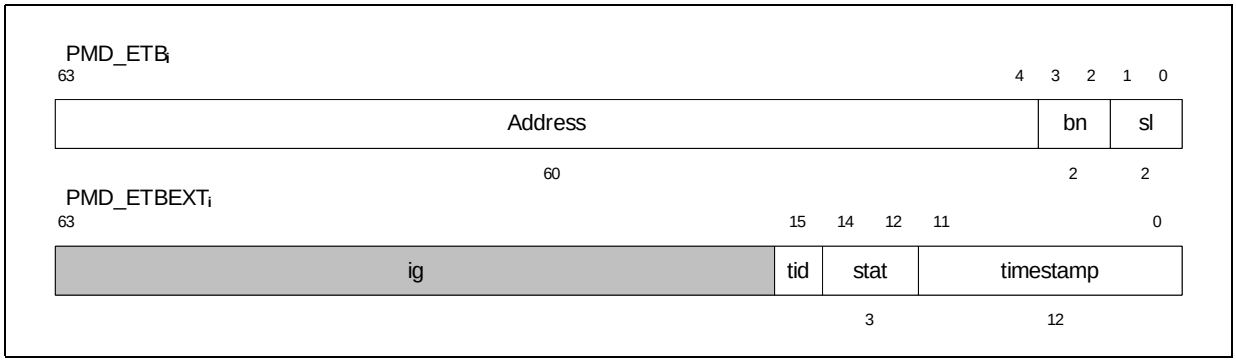


Table 3-27. Execution Trace Buffer Entry Fields

Field	Bit Range	Description
Address	63:4	60-bit bundle address of bundle 0 of the issue group of the branch instruction or branch target
bn	1:0	Bundle of the taken event
slot	3:2	Slot index of taken event in bundle 00: Slot 0 source/target 01: Slot 1 source/target 10: Slot 2 source/target 11: this was a not taken event
tid	15 (79)	Thread ID - will always have the same value for a given thread.
stat	14:12 (78:76)	ETB entry status: '000: the content of the entry is invalid '010: the content of the entry is a target address '100: the content of the entry is an rfi source '110: the content of the entry is an exception source 'xx1: the content of the entry is a branch source Note that a "11" in the slot field indicates that no taken branch was within that bundle. '001: branch, correctly predicted '011: branch mispredicted '111: branch mispredicted due to target or path misprediction (implied BRU flush)
timestamp	11:0 (75:64)	Timestamp — value of the free-running core-clock timestamp-counter

The branch trace buffer registers contain valid data only when event collection is frozen (PMC_0.fr is one). While event collection is enabled, reads of PMD_ETB* return undefined values. The registers follow the layout defined in Figure 3-24, and Table 3-27 contain the address of either a captured branch instruction or a branch target. For branch instructions, the stat field contains more information about branch mispredictions. An execution trace register with a zero stat field indicates an invalid buffer entry. SW needs to initialize these field. The processor never writes 0 to stat. The slot field captures the slot number of the first taken Itanium branch instruction in the captured instruction bundle. A slot number of 3 indicates a not-taken branch.

In every cycle in which a qualified Itanium branch retires[1], its source bundle address and bundle and slot number are written to the branch trace buffer. If within the next clock, the target instruction bundle contains a branch that retires and meets the same conditions, the address of the second branch is stored. Otherwise, either the branches' target address or details of the branch prediction are written to the next buffer location. As a result, the branch trace buffer may contain a mixed sequence of sources and targets.

The ETB contains a feature of tracking exceptions, controlled by the PMC_ETB_CFG.cir field. It will record exceptions and their targets. For exception targets, the slot field in the buffer shall be ignored regardless of value.

Note: The following behaviors of the processor ETB need to be taken into account when reconstructing an execution flow:

- If a branch is immediately followed by a control flow change at the target (branch or exception), another source entry will be recorded with the target's IP rather than a source/target pair.

1. 1n some cases, the processor execution trace buffer will capture the source (but not the target) address of an excepting branch instruction. This occurs on trapping branch instructions as well as faulting br.ia, break.b and multi-way branches.

- A branch taking a trap will be recorded as a branch with as its target the trap handler IP. The subsequent rfi will then point to the branch's original target.
- A correctly predicted branch which takes an exception may create extraneous (identical) exception source records.
- Exception target addresses may record an incorrect slot field. Exception targets will always have a slot of 0 and the recorded value can be discarded.
- Exceptions due to an external interrupt may record an incorrect source IP which should be ignored. The subsequent rfi will then point to the correct interrupted instruction.
- A branch target record may be misrecorded as an exception source. These can be identified and eliminated as follows: Any exception source record follows a branch source that isn't followed by a target record is spurious and should be changed to a target record.
- RFIs may fail to record their target in certain cases where the rfi is in an MBB or BBB template bundle. Similarly, synthesized RFIs will not be captured.

The branch trace buffer is a circular buffer containing the last twelve to twenty-four qualified Itanium branches. The Branch Trace Buffer Index Register (PMD_ETB_BUFIDX) defined in Figure 3-25 and Table 3-28 identify the most recently recorded branch or target. In every cycle in which a qualified branch or target is recorded, the execution buffer index (ebi) is post-incremented. After 24 entries have been recorded, the branch index wraps around, and the next qualified branch will overwrite the first trace buffer entry. The wrap condition itself is recorded in PMD_ETB_BUFIDX.f. The ebi field of PMD_ETB_BUFIDX.ebi defines the next branch buffer index that is about to be written. The following equation computes the last written branch trace buffer PMD index from PMD_ETB_BUFIDX:

```
last-written-PMD-index = (PMC_ETB_BUFIDX.ebi - 1) % 24
```

If both the full bit and the ebi field of PMD_ETB_BUFIDX are zero, no qualified branch has been captured by the branch trace buffer. The full bit gets set every time the branch trace buffer wraps. Once set, the full bit remains set until explicitly cleared by software, i.e. it is a sticky bit. Software can reset the ebi index and the full bit by writing to PMD_ETB_BUFIDX.

Figure 3-25. Execution Trace Buffer Index Register Format

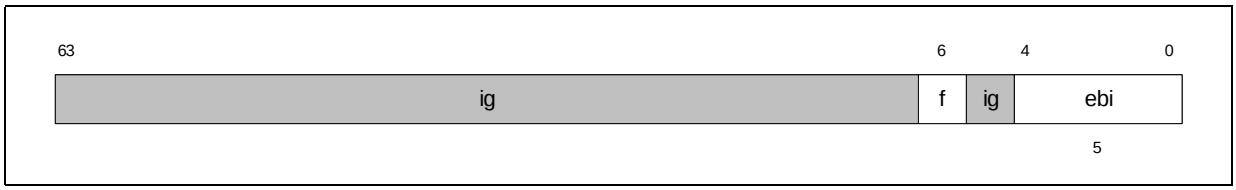


Table 3-28. Execution Trace Buffer Index Register Fields

Field	Bit Range	Description
full	6	Full Bit (sticky) if full=1: branch trace buffer has wrapped if full=0: branch trace buffer has not wrapped
ebi	4:0	Execution Buffer Index [Range 0..23 — Index 0 indicates PMD_{64}] Pointer to the next execution trace buffer entry to be written if full=1: points to the oldest recorded branch/target if full=0: points to the next location to be written

Notes on the Execution Trace Buffer

Although the processor ETB does not capture asynchronous interrupts as events, the address of these handlers can be captured as target addresses. This could happen if, at the target of a captured event (for example, taken branch), an asynchronous event is taken before executing any instruction at the target.

3.3.12.2 IP Event Address Capture (IP-EAR)

The processor has a feature called Instruction Pointer Event Address Capture (or IP-EAR). This feature is intended to facilitate the correlation of performance monitoring events to IP values. To do this, the processor's Execution Trace buffer (ETB) can be configured to capture IPs of retired instructions. When a performance monitoring event is used to trigger an IP-EAR freeze, if the IP which caused the event gets to retirement there is a good chance that IP would be captured in the ETB. The IP-EAR freezes after a programmable number of cycles following a PMU freeze as described below, in order to allow IPs for early pipe events to proceed to retirement.

PMC_IPEAR_CFG is used to configure this feature and the ETB registers (PMD_ETB*) are used to capture the data. PMD_ETB_BUFIDX holds the index and overflow bits for the IP Buffer much as it does for the ETB.

Setting PMC_IPEAR_CFG.mode to 4 will override the setting of PMC_ETB_CFG (the configuration register for the normal ETB mode).

Figure 3-26. IP-EAR Configuration Register

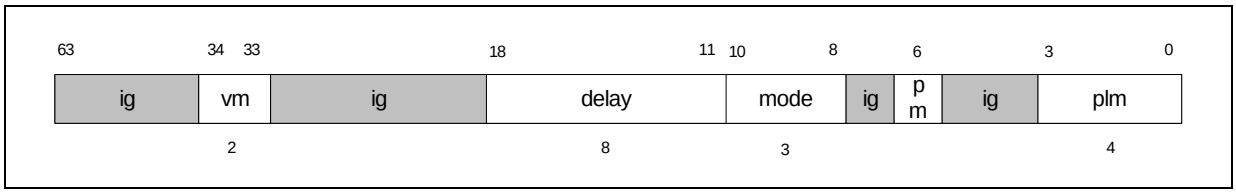


Table 3-29. IP-EAR Configuration Register Field Description

Field	Bits	Description
vm	34:33	See Table 3-6.
delay	18:11	Programmable delay before freezing
mode	10:8	IP EAR mode: 000: ETB Mode (IP-EAR not functional; ETB is functional) 100: IP-EAR Mode (IP-EAR functional; ETB not functional)
pm	6	See Table 3-6.
plm	3:0	See Table 3-6.

The IP_EAR functions by continuously capturing retired IPs in PMD_ETB*$_{0-23}$ as long as it is enabled. It captures retired IPs and elapsed time between retirements. Up to 24 entries can be captured.

The IP-EAR has a different freezing model than the rest of the Performance Monitors. It is capable of delaying its freeze for a number of cycles past the point of PMU freeze. The user can program an 8-bit number to determine the number of cycles the freeze will be delayed.

Figure 3-27 represent the layout of an execution trace buffer entry in IP-EAR mode across the PMD_ETB and PMD_ETBEXT register pairs.

Figure 3-27. IP-EAR Entry Format

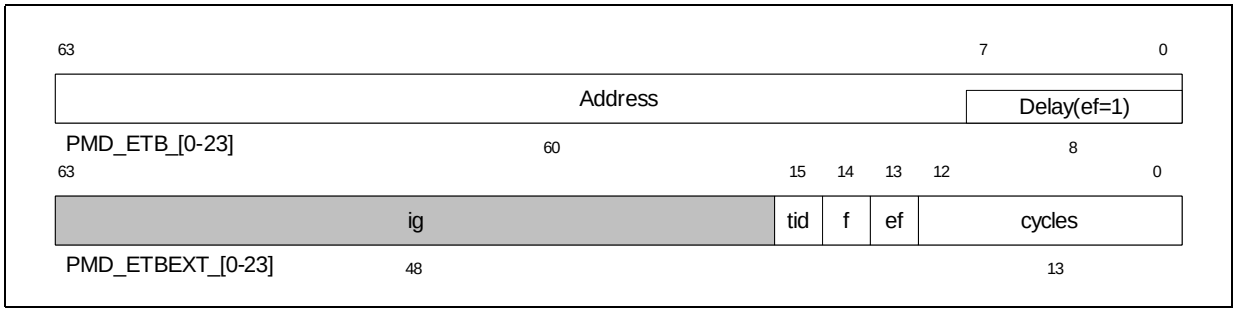


Table 3-30. IP-EAR Entry Fields

Field	Bits	Description
Address	63:0	Retired IP — bits 63:0 — bits 3:2 indicate the bundle, bits 1:0 the instruction slot
delay	7:0	Aborted delay count — if .ef is set, indicates the remainder of the aborted delay count, otherwise contains the rest of the retired IP value
tid	15 (79)	Thread ID - will always have the same value for a given thread.
f	14 (78)	Flush — Indicates whether there has been a pipe flush since the last entry
ef	13 (77)	Early freeze — When set, indicates the current entry is an early freeze case, due to one of the following: a change in PSR bits caused the IP-EAR to become disabled or a thread switch occurred
cycles	12:0 (76:64)	Elapsed cycle count from the previous retired IP. This is a saturating counter and will stay at all 1s when counted up to the maximum value.

Figure 3-28. IP Trace Buffer Index Register Format (PMD_ETB_BUFIDX)

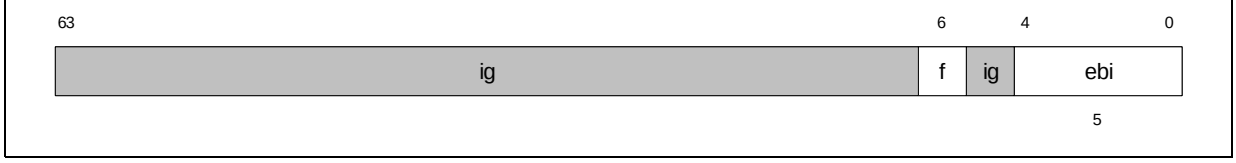


Table 3-31. IP Trace Buffer Index Register Fields Description

Field	Bit Range	Description
full	6	Full Bit (sticky) if full=1: IP trace buffer has wrapped if full=0: IP trace buffer has not wrapped
ebi	4:0	IP Trace Buffer Index [Range 0.. 23 — Index 0 indicates PMD_{64}] Pointer to the next IP trace buffer entry to be written if full=1: points to the oldest recorded IP entry if full=0: points to the next location to be written

Notes on the IP-EAR

When the IP-EAR freezes due to its normal freeze mechanism (that is, PMU freeze + delay), it captures one last entry with "ef"=0. The IP value in this entry could be incorrect since it is not assured that the CPU would be retiring an IP at this particular time. Since this is always the youngest entry captured in IP_EAR buffer, it should be easier to identify this event.

3.3.12.3 IP-EAR User Guide

The following section contains a user guide to programming the IP-EAR and interpreting the captured results.

1. Chose a core PMU event to guide the IP-EAR. Note that only events which have the IP-EAR parameter entry can be used with the IP-EAR.
   For each event we need the following two values:
   a. L: the event count latency. This is the time from the event occurrence to when it increments the PMD counter.
   b. P: the event pipeline latency. This is the number of pipeline stages from the event occurrence to when the associated IP retirement gets recorded in the ETB.
2. Program a PMC/PMD counter pair to record this event. The PMD is pre-loaded with the desired sample count (maximum possible count +1 - sample count). Typically, the PMC is configured to interrupt on overflow.
3. The IP-EAR (PMC_IPEAR_CFG) is programmed as follows:
   a. PMC_IPEAR_CFG.{pm, plm, vm} are configured as desired
   b. PMC_IPEAR_CFG.mode is set to 'b100 to set IP-EAR mode
   c. PMC_IPEAR_CFG.delay, called D below, is set according to the estimated core cycles it takes the instructions associated with the event to retire (there will be some trial and error here).
4. The PMU is then run on the workload of choice. It will overflow and cause an interrupt.
5. The ETB data is collected and analyzed off-line:
   a. Reorder the buffer in ascending chronological order (PMC_ETB_BUFIDX points to the oldest entry).
   b. Starting at the newest record (chronologically), walk back and forward to locate the IP that is associated with the event:
      - Walk the buffer backwards through time until the sum of the timestamp fields in the ETB entries >= L+D
      - Next walk the buffer forward P entries. This entry should have the IP associated with the event

c. If during the walk back we go past the beginning of the buffer, then the delay value D is too large.

d. If during the walk forward we go past the end of the buffer, the delay value D is too small.

3.3.13 Thread-State Event Configuration

The processor allows PMU users to count background thread state cycles (MT_BE_BGND_CYCLES_IN_STATE.*). To allow capture of thread state combinations with a limited number of events, PMC_BEMT_CTL allows to select which FG/BG thread state combinations are captured.

The respective MT_BE_BGND_CYC_IN_STATE event will only fire when the foreground thread is in the thread state(s) indicated by PMC_BEMT_CTL.

Figure 3-29. Thread State Event Control Register Format

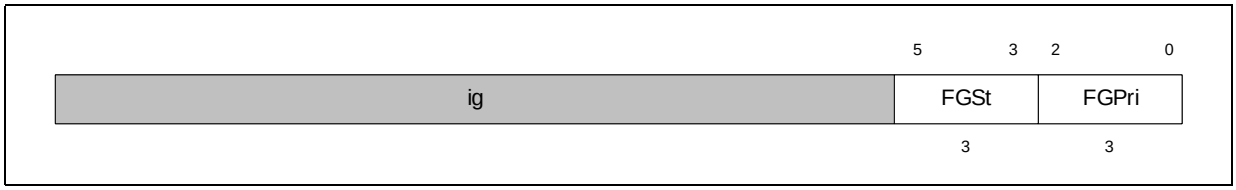


Table 3-32. Thread State Event Control Register Field Description

Field	Bit Range	Description
FGSt	5:3	Exec-state selects for foreground thread 5: signal background thread state when foreground thread is unstalled 4: signal background thread state when foreground thread is blocked 3: signal background thread state when foreground thread is stalled
FGPri	2:0	Priority selects for foreground thread 2: signal background thread state when foreground thread is high priority 1: signal background thread state when foreground thread is nominal priority 0: signal background thread state when foreground thread is low priority

3.3.14 Interrupt Counting

The processor allows PMU users to count IVA based interrupts based on the IVA offset. The interrupts to be counted are programmed by programming the register PMC_IVAEV_CFG. which supports an 11bit IVA offset and 11bit mask to select the interrupt. In addition there is a unmasked version of this event.

The IVA_mask and IVA_offs apply to address bits [14:4].

Figure 3-30. Interrupt Counting Event Configuration Register Format

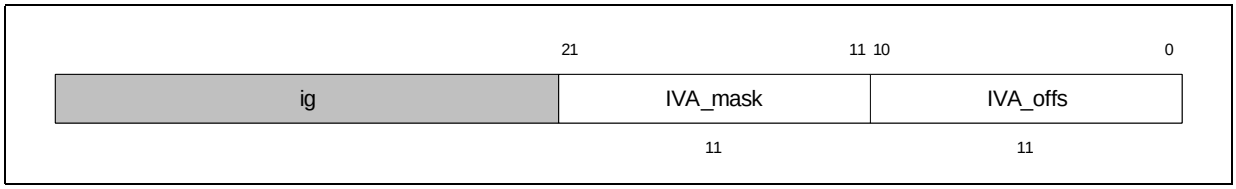


Table 3-33. Interrupt Counting Event Configuration Register Field Description

Field	Bit Range	Description
IVA_mask	21:11	Mask for IVA offset bits [14:4] programmed into IVA_offs field If 1: The corresponding bit in IVA_offs is already considered as matched if 0: The bit corresponding bit in the IVA_offs needs to match the real offset to increment the event INTERRUPT_EVENT.
IVA_offs	10:0	IVA offset; In order for the event INTERRUPT_EVENT to increment it is necessary to match the IVA offset bits [14:4] of the interrupt to the value defined by IVA_offs and IVA_mask defined in this register.

3.3.15 PerfMon Interrupts

Each one of registers PMD_{4-19} will cause an interrupt if the following conditions are all true:

- PMC_i.oi=1 (that is, overflow interrupt is enabled for PMD_i) and PMD_i overflows. Note that there is only one interrupt line that will be raised regardless of which PMC/PMD set meets this condition.

This interrupt is an "External Interrupt" with Vector= 0x3000 and will be recognized only if the following conditions are true:

- PMV.m=0 and PMV.vector is set up correctly; i.e. Performance Monitor interrupts are not masked and a proper vector is programmed for this interrupt by executing a "`mov cr73=r2`".
- PSR.i =1 and PSR.ic=1; i.e. interruptions are unmasked and interruption collection is enabled in the Processor Status Register by executing either the "`ssm imm`" or "`mov psr.i=r2`" instruction.
  TPR.mmi=0 (i.e. all external interrupts are not masked) and TPR.mic is a value that the priority class that Performance Monitor Interrupt belongs to are not masked. For example, if we assign vector 0xD2 to the Performance Monitor Interrupt, according to Table 5–7 "Interrupt Priorities, Enabling, and Masking" in Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual*, it will be priority class 13. So any value less than 13 for TPR.mic is okay for recognizing this interrupt. A "`mov cr66=r1`" will write to this register.
- There are no higher priority faults, traps, or external interrupts pending.

The interrupt service routine needs to read IVR register "`mov r1=cr65`" in order to figure out the highest priority external interrupt which needs to be serviced.

Before returning from the interrupt service routine, the Performance Monitor needs to be initialized such that the interrupt will be cleared. This could be done by clearing the PMC.oi and/or re-initializing the PMD which caused the interrupt (you will know this by reading PMC_0). In addition to this, all bits of PMC_0 need to be cleared if further monitoring needs to be done.

§

4 Core Performance Monitor Events

4.1 Introduction

This section enumerates and describes the performance monitoring events available on the Intel® Itanium® processor 9500 series core.

4.1.1 Categorization of Events

Performance related events are grouped into the following categories:

- Basic Events: clock cycles, retired instructions (4.2.1)
- Instruction Dispersal Events: instruction decode and issue (4.2.2)
- Instruction Execution Events: instruction execution, data and control speculation, and memory operations (4.2.3)
- Cycle accounting: stall and replay cycle break-down for both front- and back-end (4.2.4, 4.2.5)
- Branch Events: branch prediction (4.2.6)
- Memory Hierarchy: instruction and data caches (4.2.8, 4.2.9, 4.2.12, 4.2.13)
- TLB Events: instruction and data TLBs (4.2.10, 4.2.11, 4.2.14, 4.2.15)
- Data Prefetch Events (4.2.16)
- RIL Events: Off-core request and responses (4.2.17)
- LLC Events: Last-level cache events (4.2.19)
- RSE Events: Register Stack Engine (4.2.18)
- System Events: operating system monitors (4.2.20)
- Multi-Threading Events (4.2.21)

4.1.2 Multi-Threading and Event Types

The processor implements a type of hardware based multithreading that effectively allows two threads to coexist within a processor core although only one thread is "active" within the core's front and back-end pipeline respectively at any moment in time. This affects how events are generated. Certain events may occur while the thread they belong to is inactive. This also affects how events are assigned to the threads occupying the same core. Certain events do not have the concept of a "home" thread.

To help decipher these effects, events have been classified by the following types:

- **Active** - This event can only occur when the thread that generated it is "active" (currently executing in the processor core's pipeline) and is considered to be generated by the active thread. Example(s): IA64_INST_RETIRED.
- **Causal** - This event does not belong to a thread. It is assigned to the active thread. Example(s): CPU_OP_CYCLES.
- **Floating** - This event belongs to a thread, but could have been generated when its thread was inactive (or "in the background"). Example(s): MLD_REFERENCES.

If a monitor's PMC[i].all is not set, only events associated with the monitoring thread will be captured. If a monitor's PMC[i].all is set, events associated with both threads will be captured.

4.1.3 Performance Event Associativity

The processor supports a large number of performance events that are collected in a decentralized manner. Many constraints posed by previous Intel Itanium generations were eliminated. In turn the processor poses some restriction in terms of counter associativity.

Each events counter affinity is indicated by the counter affinity field in the event description table. The field is a hexadecimal number, where a set bit *i* indicates that PMC/PMD pair *i* can count the event. For example, a value of 0x05550 means $PMD_{14, 12, 10, 8, 6 \text{ and } 4}$ can count this particular event.

The associativity was designed to allow significant flexibility and be easy to capture algorithmically. The counters that can be associated with (i.e. count) an event can in most cases be derived from the event ID (in addition, some fundamental events may support a superset of this).

```
case (EventID[11:10]) // PMC.es[7:6]
   00 : candidate_counters = PMCs 4..19
      if (EventID[0] == 0) // PMC.umask[0]
         possible_counters = even_pmcs(candidate_counters)
      else
         possible_counters = odd_pmcs(candidate_counters)
   10 : candidate_counters = PMCs 4..11, 16..19
      if (EventID[0] == 0) // PMC.umask[0]
         possible_counters = even_pmcs(candidate_counters)
      else
         possible_counters = odd_pmcs(candidate_counters)
   11 :candidate _counters = PMCs 8..19
      possible_counters = pmcs(counter_affinity)
endcase
```

In English: most even-numbered events are associated with even-numbered counters, odd-numbered events are associated with odd-numbered counters, and the counters available are affected by the top-level collection station, but some RIL collected events have mod3 and mod4 affinity.

The exact values are indicated by the counter affinity field in the event tables.

4.1.4 Event Description Tables Field Definition

This section elaborates on the fields used in the PMU event tables in the following chapter.

Description	General description of the monitor in question
Max Inc/Cyc	Maximum increment of this monitor per core clock cycle
MT Capture Type	MT capture type (see Appendix 4.1.2 for more details)
Subevents:	
SUBEVENT	First subevent name (This document uses the convention: EVENT.SUBEVENT when referring to subevents)
Counter Affinity	Indicates which counters are capable of monitoring this event (see Appendix 4.1.3 for more details)

IAR/OPC/DAR/ DREF	Filtering capability for this monitor, taking a form of "I/O/D/R": I - the event can be constrained by instruction address matching O - the event can be constrained by op-code matching D - the event can be constrained by data address matching R - the event can be constrained by data reference type matching - - the event cannot be constrained by the particular constraint type Lower case i/o/d/r letters indicate that there may be components to an event that are not subject to a particular type of constraint.
Definition	More detailed definition of the event/subevent
Note	Particular properties/issues/caveats associated with the event

4.1.5 Performance Monitor Events Ordered by Event Code

Table 4-1 presents all of the performance monitors provided in the processor ordered by their event code. The event code is comprised of PMC.es and PMC.umask concatenated, PMC.umask forming the lesser significant portion.

Events which include MLD sourced HW prefetching do not respect the Instruction Address Range filter for those prefetching operations. For these events the prefetches are always counted. MLD_REF__ANY and LLC_REF_HIT__ANY are two of the events affected by this.

Table 4-1. All Performance Monitors Ordered by Code (Sheet 1 of 20)

Event Code	Symbol Name	Area	Section
0x001	CPU_REF_CYCLES	Basic	4.2.1.2
0x002	CPU_OP_CYCLES	Basic	4.2.1.1
0x003	CPU_OP_CYCLES.HALTED	Basic	4.2.1.1
0x004	CPU_OP_CYCLES.TAGGED	Basic	4.2.1.1
0x005	RETIRED_INST_TAGGED.IAM0_OPM0	Execution	4.2.3.31
0x005	IA64_INST_RETIRED	Basic	4.2.1.4
0x006	RETIRED_INST_TAGGED.IAM1_OPM1	Execution	4.2.3.31
0x007	RETIRED_INST_TAGGED.IAM2_OPM0	Execution	4.2.3.31
0x008	RETIRED_INST_TAGGED.IAM3_OPM1	Execution	4.2.3.31
0x009	RETIRED_PREDICATE_SQUASHED	Execution	4.2.3.32
0x00A	RETIRED_INST_NOP	Execution	4.2.3.25
0x00B	RETIRED_INST_FP	Execution	4.2.3.21
0x00C	FP_FLOP	Execution	4.2.3.15
0x00D	RETIRED_INST_M.ANY	Execution	4.2.3.24
0x00E	RETIRED_INST_M.MOVTODAHR	Execution	4.2.3.24
0x00F	DSPEC_CHKA_LDC.ANY	Execution	4.2.3.6
0x010	DSPEC_LDC.HIT	Execution	4.2.3.8
0x011	DSPEC_CHKA_LDC_FAIL.ANY	Execution	4.2.3.7
0x012	DSPEC_CHKA_LDC_FAIL.INT	Execution	4.2.3.7
0x013	DSPEC_CHKA_LDC_FAIL.FP	Execution	4.2.3.7
0x014	CSPEC_CHKS.ANY	Execution	4.2.3.3
0x015	CSPEC_CHKS_FAIL.ANY	Execution	4.2.3.4
0x016	CSPEC_CHKS_FAIL.INT	Execution	4.2.3.4
0x017	CSPEC_CHKS_FAIL.FP	Execution	4.2.3.4

Table 4-1. All Performance Monitors Ordered by Code (Sheet 2 of 20)

Event Code	Symbol Name	Area	Section
0x018	EAR_EVENT_ETB	Execution	4.2.3.10
0x019	EAR_EVENT_DATA	Execution	4.2.3.9
0x01A	CPU_CPL_CHANGE.ANY	System	4.2.20.1
0x01B	CPU_CPL_CHANGE.FROM0	System	4.2.20.1
0x01C	CPU_CPL_CHANGE.FROM1	System	4.2.20.1
0x01D	CPU_CPL_CHANGE.FROM2	System	4.2.20.1
0x01E	CPU_CPL_CHANGE.FROM3	System	4.2.20.1
0x01F	INTERRUPT_EVENT.MASKED	System	4.2.20.2
0x020	INTERRUPT_EVENT.UNMASKED	System	4.2.20.2
0x021	SERIALIZATION_EVENT	System	4.2.20.3
0x023	CYC_BE_NO_BUBBLE	Stall	4.2.4.5
0x024	CYC_BE_BUBBLE.ANY	Stall	4.2.4.1
0x025	CYC_BE_IBD_STALL.ANY	Stall	4.2.4.4
0x026	CYC_BE_IBD_STALL.RSE_ANY	Stall	4.2.4.4
0x027	CYC_BE_IBD_STALL.RSE_CFLE	Stall	4.2.4.4
0x028	CYC_BE_IBD_STALL.RSE_ST	Stall	4.2.4.4
0x029	CYC_BE_IBD_STALL.RSE_LOAD	Stall	4.2.4.4
0x02A	CYC_BE_IBD_STALL.RSE_WAIT	Stall	4.2.4.4
0x02B	CYC_BE_IBD_STALL.THRSW	Stall	4.2.4.4
0x02C	CYC_BE_IBD_STALL.HPW	Stall	4.2.4.4
0x02D	CYC_BE_IBD_STALL.OZQFULL	Stall	4.2.4.4
0x02E	CYC_BE_IBD_STALL.ACQ	Stall	4.2.4.4
0x02F	CYC_BE_IBD_STALL.GR_LOAD	Stall	4.2.4.4
0x030	CYC_BE_IBD_STALL.FR_LOAD	Stall	4.2.4.4
0x031	CYC_BE_IBD_STALL.SRLZ	Stall	4.2.4.4
0x032	CYC_BE_IBD_STALL.REL	Stall	4.2.4.4
0x033	CYC_BE_IBD_STALL.MTOM	Stall	4.2.4.4
0x034	CYC_BE_IBD_STALL.FTOF	Stall	4.2.4.4
0x035	CYC_BE_IBD_STALL.FLD_DMND	Stall	4.2.4.4
0x036	CYC_BE_IBD_STALL.WB2_TRAP	Stall	4.2.4.4
0x037	CYC_BE_IBD_STALL.QFULL	Stall	4.2.4.4
0x038	CYC_BE_IBD_STALL.FEBUB	Stall	4.2.4.4
0x039	CYC_BE_IBD_STALL.DEBUG	Stall	4.2.4.4
0x03A	CYC_BE_EXE_REPLAY.ANY	Stall	4.2.4.3
0x03B	CYC_BE_EXE_REPLAY.GR_LOAD_RAW	Stall	4.2.4.3
0x03C	CYC_BE_EXE_REPLAY.FR_LOAD_RAW	Stall	4.2.4.3
0x03D	CYC_BE_EXE_REPLAY.GR_LOAD_WAW	Stall	4.2.4.3
0x03E	CYC_BE_EXE_REPLAY.FR_LOAD_WAW	Stall	4.2.4.3
0x03F	CYC_BE_EXE_REPLAY.GR_GR	Stall	4.2.4.3
0x040	CYC_BE_EXE_REPLAY.FR_FR	Stall	4.2.4.3
0x041	CYC_BE_EXE_REPLAY.MT1_HIGH	Stall	4.2.4.3

Table 4-1. All Performance Monitors Ordered by Code (Sheet 3 of 20)

Event Code	Symbol Name	Area	Section
0x042	CYC_BE_EXE_REPLAY.FCMP	Stall	4.2.4.3
0x043	CYC_BE_EXE_REPLAY.PRED	Stall	4.2.4.3
0x044	CYC_BE_EXE_REPLAY.NOTN	Stall	4.2.4.3
0x045	CYC_BE_EXE_REPLAY.FPSR	Stall	4.2.4.3
0x046	CYC_BE_EXE_REPLAY.SRLZ	Stall	4.2.4.3
0x047	CYC_BE_EXE_REPLAY.REL	Stall	4.2.4.3
0x048	CYC_BE_EXE_REPLAY.ARCR	Stall	4.2.4.3
0x049	CYC_BE_EXE_REPLAY.MT1_LOW	Stall	4.2.4.3
0x04A	CYC_BE_DET_REPLAY.ANY	Stall	4.2.4.2
0x04B	CYC_BE_DET_REPLAY.GR_LOAD	Stall	4.2.4.2
0x04C	CYC_BE_DET_REPLAY.DCS_HZRD	Stall	4.2.4.2
0x04D	CYC_BE_DET_REPLAY.STORE_VS_STORE	Stall	4.2.4.2
0x04E	CYC_BE_DET_REPLAY.LOAD_AFTER_WRITE	Stall	4.2.4.2
0x04F	CYC_BE_DET_REPLAY.LOAD_ACQ	Stall	4.2.4.2
0x050	CYC_BE_DET_REPLAY.FLUSH_STORE	Stall	4.2.4.2
0x051	CYC_BE_DET_REPLAY.HPW_HZRD	Stall	4.2.4.2
0x052	CYC_BE_DET_REPLAY.WRITE_HIT_VS_FILL	Stall	4.2.4.2
0x053	CYC_BE_DET_REPLAY.WRITE_MISS_VS_FILL	Stall	4.2.4.2
0x054	CYC_BE_DET_REPLAY.MT1	Stall	4.2.4.2
0x055	CYC_BE_WB2_REPLAY.ANY	Stall	4.2.4.7
0x056	CYC_BE_WB2_REPLAY.LDC	Stall	4.2.4.7
0x057	CYC_BE_WB2_REPLAY.PAUSE	Stall	4.2.4.7
0x058	CYC_BE_WB2_REPLAY.ALLOC_PEC	Stall	4.2.4.7
0x059	CYC_BE_WB2_REPLAY.MOV_PSR_UM	Stall	4.2.4.7
0x05A	CYC_BE_WB2_REPLAY.VIRT_INT	Stall	4.2.4.7
0x05B	CYC_BE_WB2_REPLAY.FP_DEN	Stall	4.2.4.7
0x05C	CYC_BE_WB2_REPLAY.FP_SIR	Stall	4.2.4.7
0x05D	CYC_BE_WB2_REPLAY.BLK_HPW	Stall	4.2.4.7
0x05E	CYC_BE_WB2_REPLAY.OZQ_FULL	Stall	4.2.4.7
0x05F	CYC_BE_WB2_REPLAY.STORE_ALIAS	Stall	4.2.4.7
0x060	CYC_BE_WB2_REPLAY.NAT_HZRD	Stall	4.2.4.7
0x061	CYC_BE_WB2_REPLAY.DAHR_HZRD	Stall	4.2.4.7
0x062	CYC_BE_WB2_REPLAY.LOAD_ACQ	Stall	4.2.4.7
0x063	CYC_BE_WB2_REPLAY.MT1	Stall	4.2.4.7
0x064	CYC_BE_WB2_REPLAY.SER	Stall	4.2.4.7
0x065	CYC_BE_WB2_FLUSH.ANY	Stall	4.2.4.6
0x066	CYC_BE_WB2_FLUSH.XPN	Stall	4.2.4.6
0x067	CYC_BE_WB2_FLUSH.BRU	Stall	4.2.4.6
0x068	IBL_ISSUE.ANY	Dispersal	4.2.2.2
0x069	IBL_ISSUE.M_PIPE	Dispersal	4.2.2.2
0x06A	IBL_ISSUE_STOP.NONE	Dispersal	4.2.2.3

Table 4-1. All Performance Monitors Ordered by Code (Sheet 4 of 20)

Event Code	Symbol Name	Area	Section
0x06B	IBL_ISSUE_STOP.REPLAY	Dispersal	4.2.2.3
0x06C	IBL_ISSUE_STOP.EXPLICIT	Dispersal	4.2.2.3
0x06D	IBL_ISSUE_STOP.POWER	Dispersal	4.2.2.3
0x06E	IBL_ISSUE_STOP.DROOP	Dispersal	4.2.2.3
0x06F	IBL_ISSUE_STOP.ASYM_I	Dispersal	4.2.2.3
0x070	IBL_ISSUE_STOP.ASYM_M	Dispersal	4.2.2.3
0x071	IBL_ISSUE_STOP.FLD_DMND_M0	Dispersal	4.2.2.3
0x072	IBL_ISSUE_STOP.FLD_DMND_M1	Dispersal	4.2.2.3
0x073	IBL_ISSUE_STOP.OVRSUB_M	Dispersal	4.2.2.3
0x074	IBL_ISSUE_STOP.OVRSUB_I	Dispersal	4.2.2.3
0x075	IBL_ISSUE_STOP.OVRSUB_A	Dispersal	4.2.2.3
0x076	IBL_ISSUE_STOP.OVRSUB_F	Dispersal	4.2.2.3
0x077	IBL_ISSUE_STOP.STRUCT	Dispersal	4.2.2.3
0x078	IBL_ISSUE_STOP.BUNDLE	Dispersal	4.2.2.3
0x079	IBL_ISSUE_STOP.9PLUS3	Dispersal	4.2.2.3
0x07A	IBL_ISSUE_LOST_BW.ANY	Dispersal	4.2.2.1
0x07B	IBL_ISSUE_LOST_BW.POWER	Dispersal	4.2.2.1
0x07C	IBL_ISSUE_LOST_BW.DROOP	Dispersal	4.2.2.1
0x07D	IBL_ISSUE_LOST_BW.ASYM_I	Dispersal	4.2.2.1
0x07E	IBL_ISSUE_LOST_BW.ASYM_M	Dispersal	4.2.2.1
0x07F	IBL_ISSUE_LOST_BW.FLD_DMND_M0	Dispersal	4.2.2.1
0x080	IBL_ISSUE_LOST_BW.FLD_DMND_M1	Dispersal	4.2.2.1
0x081	IBL_ISSUE_LOST_BW.OVRSUB_A	Dispersal	4.2.2.1
0x082	IBL_ISSUE_LOST_BW.OVRSUB_F	Dispersal	4.2.2.1
0x083	IBL_ISSUE_LOST_BW.OVRSUB_I	Dispersal	4.2.2.1
0x084	IBL_ISSUE_LOST_BW.OVRSUB_M	Dispersal	4.2.2.1
0x085	IBL_ISSUE_LOST_BW.STRUCT	Dispersal	4.2.2.1
0x086	IBL_ISSUE_LOST_BW.9PLUS3	Dispersal	4.2.2.1
0x087	DPFQ_ENQ.ANY	DPF	4.2.16.6
0x088	DPFQ_ENQ.INST_ANY	DPF	4.2.16.6
0x089	DPFQ_ENQ.LFETCH	DPF	4.2.16.6
0x08A	DPFQ_ENQ.LFETCH_COUNT	DPF	4.2.16.6
0x08B	DPFQ_ENQ.MOV_BSPST	DPF	4.2.16.6
0x08C	DPFQ_ENQ.RSE_ANY	DPF	4.2.16.6
0x08D	DPFQ_ENQ.RSE_LOAD	DPF	4.2.16.6
0x08E	DPFQ_ENQ.RSE_STORE	DPF	4.2.16.6
0x08F	DPFQ_ENQ.FLD_ANY	DPF	4.2.16.6
0x090	DPFQ_ENQ.FLD_TARGET	DPF	4.2.16.6
0x091	DPFQ_ENQ.FLD_FWD	DPF	4.2.16.6
0x092	DPFQ_ENQ.FLD_BWD	DPF	4.2.16.6
0x093	DPFQ_ENQ.FLD_BIDI	DPF	4.2.16.6

Table 4-1. All Performance Monitors Ordered by Code (Sheet 5 of 20)

Event Code	Symbol Name	Area	Section
0x094	DPFQ_ENQ.MLD	DPF	4.2.16.6
0x095	DPFQ_ENQ_OVERFLOW.ANY	DPF	4.2.16.7
0x096	DPFQ_ENQ_OVERFLOW.INST_ANY	DPF	4.2.16.7
0x097	DPFQ_ENQ_OVERFLOW.LFETCH	DPF	4.2.16.7
0x098	DPFQ_ENQ_OVERFLOW.LFETCH_COUNT	DPF	4.2.16.7
0x099	DPFQ_ENQ_OVERFLOW.MOV_BSPST	DPF	4.2.16.7
0x09A	DPFQ_ENQ_OVERFLOW.RSE_ANY	DPF	4.2.16.7
0x09B	DPFQ_ENQ_OVERFLOW.RSE_LOAD	DPF	4.2.16.7
0x09C	DPFQ_ENQ_OVERFLOW.RSE_STORE	DPF	4.2.16.7
0x09D	DPFQ_ENQ_OVERFLOW.FLD_ANY	DPF	4.2.16.7
0x09E	DPFQ_ENQ_OVERFLOW.FLD_TARGET	DPF	4.2.16.7
0x09F	DPFQ_ENQ_OVERFLOW.FLD_FWD	DPF	4.2.16.7
0x0A0	DPFQ_ENQ_OVERFLOW.FLD_BWD	DPF	4.2.16.7
0x0A1	DPFQ_ENQ_OVERFLOW.FLD_BIDI	DPF	4.2.16.7
0x0A2	DPFQ_ENQ_OVERFLOW.MLD	DPF	4.2.16.7
0x0A3	DPFQ_DEQ.ANY	DPF	4.2.16.2
0x0A4	DPFQ_DEQ.INST_ANY	DPF	4.2.16.2
0x0A5	DPFQ_DEQ.LFETCH	DPF	4.2.16.2
0x0A6	DPFQ_DEQ.LFETCH_COUNT	DPF	4.2.16.2
0x0A7	DPFQ_DEQ.MOV_BSPST	DPF	4.2.16.2
0x0A8	DPFQ_DEQ.RSE_ANY	DPF	4.2.16.2
0x0A9	DPFQ_DEQ.RSE_LOAD	DPF	4.2.16.2
0x0AA	DPFQ_DEQ.RSE_STORE	DPF	4.2.16.2
0x0AB	DPFQ_DEQ.FLD_ANY	DPF	4.2.16.2
0x0AC	DPFQ_DEQ.FLD_TARGET	DPF	4.2.16.2
0x0AD	DPFQ_DEQ.FLD_FWD	DPF	4.2.16.2
0x0AE	DPFQ_DEQ.FLD_BWD	DPF	4.2.16.2
0x0AF	DPFQ_DEQ.FLD_BIDI	DPF	4.2.16.2
0x0B0	DPFQ_DEQ.MLD	DPF	4.2.16.2
0x0B1	DPFQ_DEQ_PREEMPT.ANY	DPF	4.2.16.3
0x0B2	DPFQ_DEQ_PREEMPT.INST_ANY	DPF	4.2.16.3
0x0B3	DPFQ_DEQ_PREEMPT.LFETCH	DPF	4.2.16.3
0x0B4	DPFQ_DEQ_PREEMPT.LFETCH_COUNT	DPF	4.2.16.3
0x0B5	DPFQ_DEQ_PREEMPT.MOV_BSPST	DPF	4.2.16.3
0x0B6	DPFQ_DEQ_PREEMPT.TIMEOUT	DPF	4.2.16.3
0x0B7	DPFQ_DEQ_REJECT.ANY	DPF	4.2.16.5
0x0B8	DPFQ_DEQ_REJECT.INST_ANY	DPF	4.2.16.5
0x0B9	DPFQ_DEQ_REJECT.LFETCH	DPF	4.2.16.5
0x0BA	DPFQ_DEQ_REJECT.LFETCH_COUNT	DPF	4.2.16.5
0x0BB	DPFQ_DEQ_REJECT.MOV_BSPST	DPF	4.2.16.5
0x0BC	DPFQ_DEQ_REJECT.RSE_ANY	DPF	4.2.16.5

Table 4-1. All Performance Monitors Ordered by Code (Sheet 6 of 20)

Event Code	Symbol Name	Area	Section
0x0BD	DPFQ_DEQ_REJECT.RSE_LOAD	DPF	4.2.16.5
0x0BE	DPFQ_DEQ_REJECT.RSE_STORE	DPF	4.2.16.5
0x0BF	DPFQ_DEQ_REJECT.FLD_ANY	DPF	4.2.16.5
0x0C0	DPFQ_DEQ_REJECT.FLD_TARGET	DPF	4.2.16.5
0x0C1	DPFQ_DEQ_REJECT.FLD_FWD	DPF	4.2.16.5
0x0C2	DPFQ_DEQ_REJECT.FLD_BWD	DPF	4.2.16.5
0x0C3	DPFQ_DEQ_REJECT.FLD_BIDI	DPF	4.2.16.5
0x0C4	DPFQ_DEQ_REJECT.MLD	DPF	4.2.16.5
0x0C5	DPFQ_DEQ_PREEMPT_REJECT.ANY	DPF	4.2.16.4
0x0C6	DPFQ_DEQ_PREEMPT_REJECT.LFETCH	DPF	4.2.16.4
0x0C7	DPFQ_DEQ_PREEMPT_REJECT.LFETCH_COUNT	DPF	4.2.16.4
0x0C8	DPFQ_DEQ_PREEMPT_REJECT.MOV_BSPST	DPF	4.2.16.4
0x0C9	DAHS_UNDERFLOW	DPF	4.2.16.1
0x0CA	RETIRED_INST_M.MOVTOBSPST	Execution	4.2.3.24
0x0CB	FLD_HWPREF_INS.ANY	DPF	4.2.16.8
0x0CC	FLD_HWPREF_INS.CANCEL_FILL	DPF	4.2.16.8
0x0CD	FLD_HWPREF_INS.DTLB_MISS	DPF	4.2.16.8
0x0CE	FLD_HWPREF_INS.FLDTLB_MISS	DPF	4.2.16.8
0x0CF	FLD_HWPREF_INS.NEIGHBOR	DPF	4.2.16.8
0x0D0	FLD_HWPREF_INS.STORE_ALIAS	DPF	4.2.16.8
0x0D1	FLD_HWPREF_INS.OZQ_FULL	DPF	4.2.16.8
0x0D2	FLD_HWPREF_INS.FLUSH_STORE	DPF	4.2.16.8
0x0D3	FLD_HWPREF_INS.ACQ_PEND	DPF	4.2.16.8
0x0D4	FLD_HWPREF_INS.REL_OP	DPF	4.2.16.8
0x0D5	FLD_HWPREF_INS.DTLB_MISS_LFETCH	DPF	4.2.16.8
0x0D6	FLD_HWPREF_INS.FLDTLB_MISS_LFETCH	DPF	4.2.16.8
0x0D7	FLD_HWPREF_INS.OZQ_FULL_LFETCH	DPF	4.2.16.8
0x0D8	FLD_HINT_NO_MULIT_HWPREF	FLD	4.2.12.6
0x0D9	PREF_DROP.FLDTLB_MISS	DPF	4.2.16.9
0x0DA	PREF_DROP.DTLB_MISS	DPF	4.2.16.9
0x0DB	PREF_DROP.FLD_HIT	DPF	4.2.16.9
0x0DC	PREF_DROP.FLD_SECONDARY_MISS	DPF	4.2.16.9
0x0DD	DATA_REF.ANY	FLD	4.2.12.1
0x0DE	DATA_REF.LOAD_INT	FLD	4.2.12.1
0x0DE	RETIRED_INST_LD_INT	Execution	4.2.3.23
0x0DF	DATA_REF.LOAD_FP	FLD	4.2.12.1
0x0DF	RETIRED_INST_LD_FP	Execution	4.2.3.22
0x0E0	DATA_REF.LOAD_RSE	FLD	4.2.12.1
0x0E1	DATA_REF.STORE_INT	FLD	4.2.12.1
0x0E1	RETIRED_INST_ST_INT	Execution	4.2.3.30
0x0E2	DATA_REF.STORE_FP	FLD	4.2.12.1

Table 4-1. All Performance Monitors Ordered by Code (Sheet 7 of 20)

Event Code	Symbol Name	Area	Section
0x0E2	RETIRED_INST_ST_FP	Execution	4.2.3.29
0x0E3	DATA_REF.STORE_RSE	FLD	4.2.12.1
0x0E4	DATA_REF.LFETCH	FLD	4.2.12.1
0x0E5	DATA_REF.SEMAPHORE	FLD	4.2.12.1
0x0E5	RETIRED_INST_SEMAPHORE	Execution	4.2.3.28
0x0E6	DATA_REF.HW_PREF	FLD	4.2.12.1
0x0E7	DATA_REF.LOAD_HPW	FLD	4.2.12.1
0x0E8	DATA_REF.LOAD_ANY	FLD	4.2.12.1
0x0E9	DATA_REF.STORE_ANY	FLD	4.2.12.1
0x0EA	FLD_LOAD.ANY	FLD	4.2.12.9
0x0EB	FLD_LOAD.INT	FLD	4.2.12.9
0x0EC	FLD_LOAD_MISS.ANY	FLD	4.2.12.10
0x0ED	FLD_LOAD_MISS.INT	FLD	4.2.12.10
0x0EE	FLD_LOAD_MISS.RSE	FLD	4.2.12.10
0x0EF	FLD_HIT.ANY	FLD	4.2.12.7
0x0F0	FLD_SPEC_INVAL.ANY	FLD	4.2.12.11
0x0F1	FLD_SPEC_INVAL.INST	FLD	4.2.12.11
0x0F2	FLD_SPEC_INVAL.FLUSH_STORE	FLD	4.2.12.11
0x0F3	FLD_SPEC_INVAL.SNOOP	FLD	4.2.12.11
0x0F4	RETIRED_INST_M.ACQ	Execution	4.2.3.24
0x0F5	RETIRED_INST_M.REL	Execution	4.2.3.24
0x0F6	FLD_HINT_NOALLOC	FLD	4.2.12.5
0x0F7	FLD_LINE_DEMOTE	FLD	4.2.12.8
0x0F8	FLD_FILL_REQ.ANY	FLD	4.2.12.4
0x0F9	FLD_FILL_REQ.LOAD_INT	FLD	4.2.12.4
0x0FA	FLD_FILL_REQ.LOAD_RSE	FLD	4.2.12.4
0x0FB	FLD_FILL_REQ.LFETCH	FLD	4.2.12.4
0x0FC	FLD_FILL_REQ.HW_PREF	FLD	4.2.12.4
0x0FD	FLD_FILL_CANCEL.ANY	FLD	4.2.12.2
0x0FE	FLD_FILL_CANCEL.MLD	FLD	4.2.12.2
0x0FF	FLD_FILL_CANCEL.INFAB	FLD	4.2.12.2
0x100	FLD_FILL_CANCEL.POSTFAB	FLD	4.2.12.2
0x101	FLD_FILL	FLD	4.2.12.1
0x102	FLD_FILL_LRU	FLD	4.2.12.3
0x103	FLDTLB_LOAD_MISS.ANY	FLDTLB	4.2.14.2
0x104	FLDTLB_LOAD_MISS.INT	FLDTLB	4.2.14.2
0x105	FLDTLB_LOAD_MISS.RSE	FLDTLB	4.2.14.2
0x106	FLDTLB_INS_REQ.COMPLETE	FLDTLB	4.2.14.1
0x107	FLDTLB_INS_REQ.RETIRED	FLDTLB	4.2.14.1
0x108	FLDTLB_INS_REQ.NON_RETIRED	FLDTLB	4.2.14.1
0x109	FLDTLB_INS_REQ.CANCEL	FLDTLB	4.2.14.1

Table 4-1. All Performance Monitors Ordered by Code (Sheet 8 of 20)

Event Code	Symbol Name	Area	Section
0x10A	M_ASYNC_OP_ISSUE.ANY	Dispersal	4.2.2.4
0x10B	M_ASYNC_OP_ISSUE.NONE	Dispersal	4.2.2.4
0x10C	M_ASYNC_OP_ISSUE.SNOOP_PALKUP	Dispersal	4.2.2.4
0x10D	M_ASYNC_OP_ISSUE.SNOOP	Dispersal	4.2.2.4
0x10E	M_ASYNC_OP_ISSUE.SNOOP_S	Dispersal	4.2.2.4
0x10F	M_ASYNC_OP_ISSUE.HPW_LOAD	Dispersal	4.2.2.4
0x110	M_ASYNC_OP_ISSUE.CRAB_RET	Dispersal	4.2.2.4
0x111	M_ASYNC_OP_ISSUE.HW_PREF	Dispersal	4.2.2.4
0x112	M_ASYNC_OP_ISSUE.PAPURGE	Dispersal	4.2.2.4
0x113	M_ASYNC_OP_ISSUE.VAMERR_VAPURGE	Dispersal	4.2.2.4
0x114	M_ASYNC_OP_ISSUE.DTLBTRNSFR_TLBINSERT	Dispersal	4.2.2.4
0x115	M_ASYNC_OP_ISSUE.FLUSH_ST_INVAL	Dispersal	4.2.2.4
0x116	M_ASYNC_OP_ISSUE.PAMERR_PAPURGE	Dispersal	4.2.2.4
0x117	M_ASYNC_OP_ISSUE.VRNRIDVPN_PURGE	Dispersal	4.2.2.4
0x118	M_ASYNC_OP_ISSUE.RIDVPN_PURGE	Dispersal	4.2.2.4
0x119	M_ASYNC_OP_ISSUE.TLB_TSWITCH	Dispersal	4.2.2.4
0x11A	M_ASYNC_OP_ISSUE.TSWITCH	Dispersal	4.2.2.4
0x11B	M_ASYNC_OP_ISSUE.HPW_TLBINSERT	Dispersal	4.2.2.4
0x11C	M_ASYNC_OP_ISSUE.HPW_FAULT	Dispersal	4.2.2.4
0x11D	M_ASYNC_OP_ISSUE.ITC_D	Dispersal	4.2.2.4
0x11E	M_ASYNC_OP_ISSUE.ITR_D	Dispersal	4.2.2.4
0x11F	M_ASYNC_OP_ISSUE.PTR_D	Dispersal	4.2.2.4
0x120	M_ASYNC_OP_ISSUE.MOVTORR	Dispersal	4.2.2.4
0x121	M_ASYNC_OP_ISSUE.MOVTOPKR	Dispersal	4.2.2.4
0x122	M_ASYNC_OP_ISSUE.PTC_L	Dispersal	4.2.2.4
0x123	M_ASYNC_OP_ISSUE.PTC_E	Dispersal	4.2.2.4
0x124	M_ASYNC_OP_ISSUE.PTC_G	Dispersal	4.2.2.4
0x125	M_ASYNC_OP_ISSUE.PTC_GA	Dispersal	4.2.2.4
0x126	M_ASYNC_OP_ISSUE.SHOOTDOWN_G	Dispersal	4.2.2.4
0x127	M_ASYNC_OP_ISSUE.SHOOTDOWN_GA	Dispersal	4.2.2.4
0x128	M_ASYNC_OP_ISSUE.RSE_STORE	Dispersal	4.2.2.4
0x129	M_ASYNC_OP_ISSUE.RSE_LOAD	Dispersal	4.2.2.4
0x12A	MT_BE_THRSW_ACTUAL_OUT.ANY	Multithreading	4.2.21.6
0x12B	MT_BE_THRSW_ACTUAL_OUT.MLD_USE	Multithreading	4.2.21.6
0x12C	MT_BE_THRSW_ACTUAL_OUT.HPW_MISS	Multithreading	4.2.21.6
0x12D	MT_BE_THRSW_ACTUAL_OUT.IBQ_EMPTY	Multithreading	4.2.21.6
0x12E	MT_BE_THRSW_ACTUAL_OUT.ATPAUSE	Multithreading	4.2.21.6
0x12F	MT_BE_THRSW_ACTUAL_OUT.LP_ENTER	Multithreading	4.2.21.6
0x130	MT_BE_THRSW_ACTUAL_OUT.RFIX	Multithreading	4.2.21.6
0x131	MT_BE_THRSW_ACTUAL_OUT.INJ_DBG	Multithreading	4.2.21.6
0x132	MT_BE_THRSW_ACTUAL_IN.TIMEOUT	Multithreading	4.2.21.5

Table 4-1. All Performance Monitors Ordered by Code (Sheet 9 of 20)

Event Code	Symbol Name	Area	Section
0x133	MT_BE_THRSW_ACTUAL_IN.MLDRTN	Multithreading	4.2.21.5
0x134	MT_BE_THRSW_ACTUAL_IN.HPWINS	Multithreading	4.2.21.5
0x135	MT_BE_THRSW_ACTUAL_IN.IBQ_NOTEMPTY	Multithreading	4.2.21.5
0x136	MT_BE_THRSW_ACTUAL_IN.ALAT_INVAL	Multithreading	4.2.21.5
0x137	MT_BE_THRSW_ACTUAL_IN.LP_EXIT	Multithreading	4.2.21.5
0x138	MT_BE_THRSW_ACTUAL_IN.FAIR	Multithreading	4.2.21.5
0x139	MT_BE_THRSW_DROP	Multithreading	4.2.21.8
0x13A	MT_BE_THRSW_DISABLE.EXPL	Multithreading	4.2.21.7
0x13B	MT_BE_THRSW_DISABLE.IMPL	Multithreading	4.2.21.7
0x13C	MT_BE_THRSW_HOLD	Multithreading	4.2.21.9
0x13D	MT_BE_THRSW_STALL.ANY	Multithreading	4.2.21.10
0x13E	MT_BE_THRSW_STALL.SWITCH	Multithreading	4.2.21.10
0x13F	MT_BE_THRSW_STALL.PIPE	Multithreading	4.2.21.10
0x140	MT_BE_THRSW_STALL.RSE	Multithreading	4.2.21.10
0x141	MT_BE_THRSW_STALL.CRAB	Multithreading	4.2.21.10
0x142	MT_BE_THRSW_STALL.FLD	Multithreading	4.2.21.10
0x143	MT_BE_BGND_CYC_IN_STATE.HU	Multithreading	4.2.21.2
0x144	MT_BE_BGND_CYC_IN_STATE.HW	Multithreading	4.2.21.2
0x145	MT_BE_BGND_CYC_IN_STATE.NU	Multithreading	4.2.21.2
0x146	MT_BE_BGND_CYC_IN_STATE.NW	Multithreading	4.2.21.2
0x147	MT_BE_BGND_CYC_IN_STATE.LU	Multithreading	4.2.21.2
0x148	MT_BE_BGND_CYC_IN_STATE.LW	Multithreading	4.2.21.2
0x149	MT_BE_FAIR_STATE.GREEN	Multithreading	4.2.21.3
0x14A	MT_BE_FAIR_STATE.YELLOW	Multithreading	4.2.21.3
0x14B	MT_BE_FAIR_STATE.ORANGE	Multithreading	4.2.21.3
0x14C	MT_BE_FAIR_STATE.RED	Multithreading	4.2.21.3
0x14D	MT_BE_FAIR_TRANSITION.GRN2YLW	Multithreading	4.2.21.4
0x14E	MT_BE_FAIR_TRANSITION.YLW2ORN	Multithreading	4.2.21.4
0x14F	MT_BE_FAIR_TRANSITION.ORN2RED	Multithreading	4.2.21.4
0x150	MT_BE_FAIR_TRANSITION.GRN0	Multithreading	4.2.21.4
0x151	ALAT_STORE_HIT	Execution	4.2.3.2
0x152	ALAT_ENTRY_REPLACED	Execution	4.2.3.1
0x153	RSE_CURRENT_REG.MSB	RSE	4.2.18.1
0x154	RSE_CURRENT_REG.LSB	RSE	4.2.18.1
0x155	RSE_DIRTY_REG.MSB	RSE	4.2.18.2
0x156	RSE_DIRTY_REG.LSB	RSE	4.2.18.2
0x157	RETIRED_INST_RSE	Execution	4.2.3.27
0x158	RSE_REF_RETIRED.ANY	RSE	4.2.18.3
0x159	RSE_REF_RETIRED.LOAD	RSE	4.2.18.3
0x15A	RSE_REF_RETIRED.STORE	RSE	4.2.18.3
0x801	BR_BE_PRED_DETAIL.STG	Branch	4.2.6.1

Table 4-1. All Performance Monitors Ordered by Code (Sheet 10 of 20)

Event Code	Symbol Name	Area	Section
0x802	BR_BE_PRED_DETAIL.ROT	Branch	4.2.6.1
0x803	BR_BE_PRED_DETAIL.PFS	Branch	4.2.6.1
0x804	BR_BE_PRED_DETAIL.OTHER	Branch	4.2.6.1
0x805	BR_BE_PRED_DETAIL.ANY_RETIRED	Branch	4.2.6.1
0x806	BR_BE_PRED_DETAIL.UNRETIRED	Branch	4.2.6.1
0x807	BR_BE_PRED_DETAIL.ANY	Branch	4.2.6.1
0x808	BR_PATH_PRED.ANY_MISPRED_NOT_TAKEN	Branch	4.2.6.3
0x809	BR_PATH_PRED.ANY_MISPRED_TAKEN	Branch	4.2.6.3
0x80A	BR_PATH_PRED.ANY_OKPRED_NOT_TAKEN	Branch	4.2.6.3
0x80B	BR_PATH_PRED.ANY_OKPRED_TAKEN	Branch	4.2.6.3
0x80C	BR_PATH_PRED.IPREL_MISPRED_NOT_TAKEN	Branch	4.2.6.3
0x80D	BR_PATH_PRED.IPREL_MISPRED_TAKEN	Branch	4.2.6.3
0x80E	BR_PATH_PRED.IPREL_OKPRED_NOT_TAKEN	Branch	4.2.6.3
0x80F	BR_PATH_PRED.IPREL_OKPRED_TAKEN	Branch	4.2.6.3
0x810	BR_PATH_PRED.RETURN_MISPRED_NOT_TAKEN	Branch	4.2.6.3
0x811	BR_PATH_PRED.RETURN_MISPRED_TAKEN	Branch	4.2.6.3
0x812	BR_PATH_PRED.RETURN_OKPRED_NOT_TAKEN	Branch	4.2.6.3
0x813	BR_PATH_PRED.RETURN_OKPRED_TAKEN	Branch	4.2.6.3
0x814	BR_PATH_PRED.NON_RETIND_MISPRED_NOT_TAKEN	Branch	4.2.6.3
0x815	BR_PATH_PRED.NON_RETIND_MISPRED_TAKEN	Branch	4.2.6.3
0x816	BR_PATH_PRED.NON_RETIND_OKPRED_NOT_TAKEN	Branch	4.2.6.3
0x817	BR_PATH_PRED.NON_RETIND_OKPRED_TAKEN	Branch	4.2.6.3
0x818	BR_PRED_UNKNOWN.ANY	Branch	4.2.6.5
0x819	BR_PRED_UNKNOWN.ANY_TAKEN	Branch	4.2.6.5
0x81A	BR_PRED_UNKNOWN.IPREL	Branch	4.2.6.5
0x81B	BR_PRED_UNKNOWN.IPREL_TAKEN	Branch	4.2.6.5
0x81C	BR_PRED_UNKNOWN.RETURN	Branch	4.2.6.5
0x81D	BR_PRED_UNKNOWN.RETURN_TAKEN	Branch	4.2.6.5
0x81E	BR_PRED_UNKNOWN.NON_RETIND	Branch	4.2.6.5
0x81F	BR_PRED_UNKNOWN.NON_RETIND_TAKEN	Branch	4.2.6.5
0x820	BR_PRED_DETAIL.ANY_ANY_PRED	Branch	4.2.6.4
0x820	RETIRED_INST_BR	Execution	4.2.3.18
0x821	BR_PRED_DETAIL.ANY_CORR_PRED	Branch	4.2.6.4
0x822	BR_PRED_DETAIL.ANY_WRONG_PATH	Branch	4.2.6.4
0x823	BR_PRED_DETAIL.ANY_WRONG_TARGET	Branch	4.2.6.4
0x824	BR_PRED_DETAIL.IPREL_ANY_PRED	Branch	4.2.6.4
0x825	BR_PRED_DETAIL.IPREL_CORR_PRED	Branch	4.2.6.4
0x826	BR_PRED_DETAIL.IPREL_WRONG_PATH	Branch	4.2.6.4
0x827	BR_PRED_DETAIL.IPREL_WRONG_TARGET	Branch	4.2.6.4
0x828	BR_PRED_DETAIL.RETURN_ANY_PRED	Branch	4.2.6.4
0x829	BR_PRED_DETAIL.RETURN_CORR_PRED	Branch	4.2.6.4

Table 4-1. All Performance Monitors Ordered by Code (Sheet 11 of 20)

Event Code	Symbol Name	Area	Section
0x82A	BR_PRED_DETAIL.RETURN_WRONG_PATH	Branch	4.2.6.4
0x82B	BR_PRED_DETAIL.RETURN_WRONG_TARGET	Branch	4.2.6.4
0x82C	BR_PRED_DETAIL.NON_RETIND_ANY_PRED	Branch	4.2.6.4
0x82D	BR_PRED_DETAIL.NON_RETIND_CORR_PRED	Branch	4.2.6.4
0x82E	BR_PRED_DETAIL.NON_RETIND_WRONG_PATH	Branch	4.2.6.4
0x82F	BR_PRED_DETAIL.NON_RETIND_WRONG_TARGET	Branch	4.2.6.4
0x830	BR_ENC_PRED_DETAIL.ENC_ANY_PRED	Branch	4.2.6.2
0x831	BR_ENC_PRED_DETAIL.ENC_CORR_PRED	Branch	4.2.6.2
0x832	BR_ENC_PRED_DETAIL.ENC_WRONG_PATH	Branch	4.2.6.2
0x833	BR_ENC_PRED_DETAIL.ENC_WRONG_TARGET	Branch	4.2.6.2
0x834	BR_ENC_PRED_DETAIL.OVRSUB_ANY_PRED	Branch	4.2.6.2
0x835	BR_ENC_PRED_DETAIL.OVRSUB_CORR_PRED	Branch	4.2.6.2
0x836	BR_ENC_PRED_DETAIL.OVRSUB_WRONG_PATH	Branch	4.2.6.2
0x837	BR_ENC_PRED_DETAIL.OVRSUB_WRONG_TARGET	Branch	4.2.6.2
0x838	BR_ENC_PRED_DETAIL.ENC_OVRSUB_ANY_PRED	Branch	4.2.6.2
0x839	BR_ENC_PRED_DETAIL.ENC_OVRSUB_CORR_PRED	Branch	4.2.6.2
0x83A	BR_ENC_PRED_DETAIL.ENC_OVRSUB_WRONG_PATH	Branch	4.2.6.2
0x83B	BR_ENC_PRED_DETAIL.ENC_OVRSUB_WRONG_TARGET	Branch	4.2.6.2
0x83C	MLI_READ.ANY_ANY	MLI	4.2.9.2
0x83D	MLI_READ.ANY_DMND	MLI	4.2.9.2
0x83E	MLI_READ.ANY_PREF	MLI	4.2.9.2
0x83F	MLI_READ.HIT_ANY	MLI	4.2.9.2
0x840	MLI_READ.HIT_DMND_NOLRU	MLI	4.2.9.2
0x841	MLI_READ.HIT_PREF_NOLRU	MLI	4.2.9.2
0x842	MLI_READ.HIT_DMND_LRU	MLI	4.2.9.2
0x843	MLI_READ.HIT_PREF_LRU	MLI	4.2.9.2
0x844	MLI_READ.MISS_ANY	MLI	4.2.9.2
0x845	MLI_READ.MISS_DMND	MLI	4.2.9.2
0x846	MLI_READ.MISS_PREF	MLI	4.2.9.2
0x847	MLI_READ_UC.ANY	MLI	4.2.9.3
0x848	MLI_READ_UC.DMND	MLI	4.2.9.3
0x849	MLI_READ_UC.PREF	MLI	4.2.9.3
0x84A	MLI_RECIRCULATE.ANY	MLI	4.2.9.4
0x84B	MLI_RECIRCULATE.DMND	MLI	4.2.9.4
0x84C	MLI_RECIRCULATE.PREF	MLI	4.2.9.4
0x84D	MLI_SNOOP_INVAL_BLK_LOOKUP	MLI	4.2.9.6
0x84E	MLI_HIT_CONFLICT.ANY	MLI	4.2.9.1
0x84F	MLI_HIT_CONFLICT.DMND	MLI	4.2.9.1
0x850	MLI_HIT_CONFLICT.PREF	MLI	4.2.9.1
0x851	MLI_SPEC_ABORT	MLI	4.2.9.7
0x852	MLI_SNOOP_HIT	MLI	4.2.9.5

Table 4-1. All Performance Monitors Ordered by Code (Sheet 12 of 20)

Event Code	Symbol Name	Area	Section
0x853	FP_FCHKF_FAIL	Execution	4.2.3.14
0x854	FP_FALSE_SIR	Execution	4.2.3.13
0x855	FP_TRUE_SIR	Execution	4.2.3.17
0x856	FP_DENORMAL	Execution	4.2.3.12
0x857	FP_FLUSH_TO_ZERO.FTZ_REAL	Execution	4.2.3.16
0x858	FP_FLUSH_TO_ZERO.FTZ_POSS	Execution	4.2.3.16
0x859	FLI_READ.ANY	FLI	4.2.8.10
0x85A	FLI_READ_MISS.ANY	FLI	4.2.8.11
0x85B	FLI_READ.DMND	FLI	4.2.8.10
0x85C	FLI_FILL	FLI	4.2.8.3
0x85D	FLI_READ_MISS.DMND	FLI	4.2.8.11
0x85E	FLI_READ.PREF	FLI	4.2.8.10
0x85F	FLI_READ_MISS.PREF	FLI	4.2.8.11
0x860	MLI_RETURN_LINE	FLI	4.2.8.14
0x861	FLI_PREF_STALL.ANY	FLI	4.2.8.5
0x862	FLI_PREF_STALL.FLOW	FLI	4.2.8.5
0x863	FLI_READ.SNOOP	FLI	4.2.8.10
0x864	FLI_READ.SNOOP_HIT	FLI	4.2.8.10
0x865	FLI_PURGE	FLI	4.2.8.6
0x866	FLI_STREAM_PREF	FLI	4.2.8.13
0x867	FLI_RAB_FULL	FLI	4.2.8.9
0x868	FLI_RAB_ALMOST_FULL	FLI	4.2.8.8
0x869	FLI_FETCH_RAB_HIT.DMND	FLI	4.2.8.2
0x86A	FLI_FETCH_RAB_HIT.PREF	FLI	4.2.8.2
0x86B	FLI_FETCH_JIT_HIT	FLI	4.2.8.1
0x86C	FLI_PVAB_OVERFLOW	FLI	4.2.8.7
0x86D	FLITLB_MISS	FLITLB	4.2.10.2
0x86E	MLITLB_MISS	MLITLB	4.2.11.2
0x86F	FLITLB_INSERT_HPW	FLITLB	4.2.10.1
0x870	MLITLB_HPW_ABORTS	MLITLB	4.2.11.1
0x871	EAR_EVENT_INST	Execution	4.2.3.11
0x872	FLI_INST_INSERT_RAB	FLI	4.2.8.4
0x873		FLI	4.2.8.12
0x874	MT_FE_BE_IN_SAME_THREAD	Multithreading	4.2.21.11
0x875	FE_OP_CYCLES	Basic	4.2.1.3
0x876	MT_FE_THRSW_ACTUAL_OUT.ANY	Multithreading	4.2.21.14
0x877	MT_FE_THRSW_ACTUAL_OUT.TIMEOUT	Multithreading	4.2.21.14
0x878	MT_FE_THRSW_ACTUAL_OUT.IBQ_FULL	Multithreading	4.2.21.14
0x879	MT_FE_THRSW_ACTUAL_IN.IBQ_NOTFULL	Multithreading	4.2.21.13
0x87A	MT_FE_THRSW_ACTUAL_IN.IBQ_EMPTY	Multithreading	4.2.21.13
0x87B	MT_FE_THRSW_ACTUAL_OUT.IBQ_NOTEMPTY	Multithreading	4.2.21.14

Table 4-1. All Performance Monitors Ordered by Code (Sheet 13 of 20)

Event Code	Symbol Name	Area	Section
0x87C	MT_FE_THRSW_ACTUAL_OUT.MLI_WBMISS	Multithreading	4.2.21.14
0x87D	MT_FE_THRSW_ACTUAL_IN.MLI_WBRTN	Multithreading	4.2.21.13
0x87E	MT_FE_THRSW_ACTUAL_OUT.MLI_UCMISS	Multithreading	4.2.21.14
0x87F	MT_FE_THRSW_ACTUAL_IN.MLI_UCRTN	Multithreading	4.2.21.13
0x880	MT_FE_THRSW_ACTUAL_OUT.BRQ_BLK	Multithreading	4.2.21.14
0x881	MT_FE_THRSW_ACTUAL_IN.BRQ_NON_BLK	Multithreading	4.2.21.13
0x882	MT_FE_THRSW_ACTUAL_OUT.HINT_BSWT	Multithreading	4.2.21.14
0x883	MT_FE_THRSW_ACTUAL_OUT.BE_FOLLOW	Multithreading	4.2.21.14
0x884	MT_FE_THRSW_ACTUAL_OUT.LOCKED	Multithreading	4.2.21.14
0x885	MT_FE_THRSW_MISS_OUT.ANY	Multithreading	4.2.21.16
0x886	MT_FE_THRSW_MISS_IN.ANY	Multithreading	4.2.21.15
0x887	MT_FE_THRSW_MISS_OUT.TIMEOUT	Multithreading	4.2.21.16
0x888	MT_FE_THRSW_MISS_OUT.IBQ_FULL	Multithreading	4.2.21.16
0x889	MT_FE_THRSW_MISS_IN.IBQ_NOTFULL	Multithreading	4.2.21.15
0x88A	MT_FE_THRSW_MISS_IN.IBQ_EMPTY	Multithreading	4.2.21.15
0x88B	MT_FE_THRSW_MISS_OUT.IBQ_NOTEMPTY	Multithreading	4.2.21.16
0x88C	MT_FE_THRSW_MISS_OUT.MLI_WBMISS	Multithreading	4.2.21.16
0x88D	MT_FE_THRSW_MISS_IN.MLI_WBRTN	Multithreading	4.2.21.15
0x88E	MT_FE_THRSW_MISS_OUT.MLI_UCMISS	Multithreading	4.2.21.16
0x88F	MT_FE_THRSW_MISS_IN.MLI_UCRTN	Multithreading	4.2.21.15
0x890	MT_FE_THRSW_MISS_OUT.BRQ_BLK	Multithreading	4.2.21.16
0x891	MT_FE_THRSW_MISS_IN.BRQ_NON_BLK	Multithreading	4.2.21.15
0x892	MT_FE_THRSW_MISS_OUT.HINT_BSWT	Multithreading	4.2.21.16
0x893	MT_FE_THRSW_MISS_OUT.BE_FOLLOW	Multithreading	4.2.21.16
0x894	MT_FE_THRSW_MISS_OUT.LOCKED	Multithreading	4.2.21.16
0x895	MT_FE_THRSW_STALL.ANY	Multithreading	4.2.21.17
0x896	MT_FE_THRSW_STALL.MTLCK	Multithreading	4.2.21.17
0x897	MT_FE_THRSW_STALL.EXPL	Multithreading	4.2.21.17
0x898	MT_FE_THRSW_STALL.IMPL	Multithreading	4.2.21.17
0x899	MT_FE_THRSW_STALL.BLK_ANY	Multithreading	4.2.21.17
0x89A	MT_FE_THRSW_STALL.BLK_IPC_MISS	Multithreading	4.2.21.17
0x89B	MT_FE_THRSW_STALL.BLK_IN_PROG	Multithreading	4.2.21.17
0x89C	MT_FE_THRSW_STALL.	Multithreading	4.2.21.17
0x89D	MT_FE_THRSW_STALL.BLK_THRESH	Multithreading	4.2.21.17
0x89E	MT_FE_BGND_CYC_IN_STATE.LOW	Multithreading	4.2.21.12
0x89F	MT_FE_BGND_CYC_IN_STATE.NOMINAL	Multithreading	4.2.21.12
0x8A0	MT_FE_BGND_CYC_IN_STATE.HIGH	Multithreading	4.2.21.12
0x8A1	CYC_FE_FWPROG	Multithreading	4.2.21.1
0x8A2	CYC_FE_NO_BUBBLE	Stall	4.2.5.4
0x8A3	CYC_FE_BUBBLE.ANY	Stall	4.2.5.1
0x8A4	CYC_FE_RESTEER.ANY	Stall	4.2.5.5

Table 4-1. All Performance Monitors Ordered by Code (Sheet 14 of 20)

Event Code	Symbol Name	Area	Section
0x8A5	CYC_FE_RESTEER.IPREL	Stall	4.2.5.5
0x8A6	CYC_FE_RESTEER.BR_RETIND	Stall	4.2.5.5
0x8A7	CYC_FE_RESTEER.NON_RETIND	Stall	4.2.5.5
0x8A8	CYC_FE_RESTEER.SEQ_BR	Stall	4.2.5.5
0x8A9	CYC_FE_RESTEER.TSWITCH	Stall	4.2.5.5
0x8AA	CYC_FE_RESTEER.BE_FLUSH	Stall	4.2.5.5
0x8AB	CYC_FE_FET_REPLAY.ANY	Stall	4.2.5.2
0x8AC	CYC_FE_FET_REPLAY.BR_INIT	Stall	4.2.5.2
0x8AD	CYC_FE_FET_REPLAY.BRQ_FULL	Stall	4.2.5.2
0x8AE	CYC_FE_FET_REPLAY.BRQ_WAIT	Stall	4.2.5.2
0x8AF	CYC_FE_FET_REPLAY.BR_INTRLCK	Stall	4.2.5.2
0x8B0	CYC_FE_FET_REPLAY.RAB_FULL	Stall	4.2.5.2
0x8B1	CYC_FE_FET_REPLAY.MT1	Stall	4.2.5.2
0x8B2	CYC_FE_FET_STALL.ANY	Stall	4.2.5.3
0x8B3	CYC_FE_FET_STALL.IBQ_FULL	Stall	4.2.5.3
0x8B4	CYC_FE_FET_STALL.FLI_MISS	Stall	4.2.5.3
0x8B5	CYC_FE_FET_STALL.FLITLB_MISS	Stall	4.2.5.3
0x8B6	CYC_FE_FET_STALL.MT1	Stall	4.2.5.3
0x8B7	FE_RESTEER.ANY	Stall	4.2.5.6
0x8B8	FE_RESTEER.FET_REPLAY	Stall	4.2.5.6
0x8B9	FE_RESTEER.0B_IPREL	Stall	4.2.5.6
0x8BA	FE_RESTEER.1B_IPREL	Stall	4.2.5.6
0x8BB	FE_RESTEER.3B_IPREL	Stall	4.2.5.6
0x8BC	FE_RESTEER.1B_BR_RETIND	Stall	4.2.5.6
0x8BD	FE_RESTEER.3B_BR_RETIND	Stall	4.2.5.6
0x8BE	FE_RESTEER.3B_NON_RETIND	Stall	4.2.5.6
0x8BF	FE_RESTEER.1B_SEQ_BR	Stall	4.2.5.6
0x8C0	FE_RESTEER.3B_SEQ_BR	Stall	4.2.5.6
0x8C1	FE_RESTEER.3B_MT	Stall	4.2.5.6
0x8C2	FE_RESTEER.4B_MT	Stall	4.2.5.6
0x8C3	FE_RESTEER.BE_FLUSH	Stall	4.2.5.6
0xC01	RIL_SNOOP_REQ.ANY	RIL	4.2.17.28
0xC02	RIL_SNOOP_REQ.CODE_ANY	RIL	4.2.17.28
0xC03	RIL_SNOOP_REQ.CODE_SELF	RIL	4.2.17.28
0xC04	RIL_SNOOP_REQ.CODE_SIBLING	RIL	4.2.17.28
0xC05	RIL_SNOOP_REQ.DATA_ANY	RIL	4.2.17.28
0xC06	RIL_SNOOP_REQ.DATA_SELF	RIL	4.2.17.28
0xC07	RIL_SNOOP_REQ.DATA_SIBLING	RIL	4.2.17.28
0xC08	RIL_SNOOP_REQ.INVAL_ANY	RIL	4.2.17.28
0xC09	RIL_SNOOP_REQ.INVAL_SELF	RIL	4.2.17.28
0xC0A	RIL_SNOOP_REQ.INVAL_SIBLING	RIL	4.2.17.28

Table 4-1. All Performance Monitors Ordered by Code (Sheet 15 of 20)

Event Code	Symbol Name	Area	Section
0xC0B	RIL_SNOOP_REQ.INVAL_LLC_EVICT	RIL	4.2.17.28
0xC0C	RIL_SNOOP_RESP.MLD_MISS	RIL	4.2.17.29
0xC0D	RIL_SNOOP_RESP.WRQ_HIT_M	RIL	4.2.17.29
0xC0E	RIL_SNOOP_RESP.MLD_HIT_S	RIL	4.2.17.29
0xC0F	RIL_SNOOP_RESP.MLD_HIT_E	RIL	4.2.17.29
0xC10	RIL_SNOOP_RESP.MLD_HIT_M	RIL	4.2.17.29
0xC11	RIL_SNOOP_RESP.MLD_DEFER	RIL	4.2.17.29
0xC12	RIL_REQ.ANY	RIL	4.2.17.17
0xC13	RIL_REQ_REF.ANY	RIL	4.2.17.20
0xC14	RIL_REQ_REF_INST.ANY	RIL	4.2.17.22
0xC15	RIL_REQ_REF_INST.NC	RIL	4.2.17.22
0xC16	RIL_REQ_REF_INST.WB_ANY	RIL	4.2.17.22
0xC17	RIL_REQ_REF_INST.WB_DMND	RIL	4.2.17.22
0xC18	RIL_REQ_REF_DATA.ANY	RIL	4.2.17.21
0xC19	RIL_REQ_REF_DATA.WB_ANY	RIL	4.2.17.21
0xC1A	RIL_REQ_REF_DATA.WB_MLD_ANY	RIL	4.2.17.21
0xC1B	RIL_REQ_REF_DATA.WB_MLD_BUDDY	RIL	4.2.17.21
0xC1C	RIL_REQ_REF_DATA.WB_CRD	RIL	4.2.17.21
0xC1D	RIL_REQ_REF_DATA.WB_DRD	RIL	4.2.17.21
0xC1E	RIL_REQ_REF_DATA.WB_RFO	RIL	4.2.17.21
0xC1F	RIL_REQ_REF_DATA.WB_SELF_SNOOP	RIL	4.2.17.21
0xC20	RIL_REQ_REF_DATA.NC_ANY	RIL	4.2.17.21
0xC21	RIL_REQ_REF_DATA.NC_READ_ANY	RIL	4.2.17.21
0xC22	RIL_REQ_REF_DATA.NC_READ_UC	RIL	4.2.17.21
0xC23	RIL_REQ_REF_DATA.NC_WRITE_ANY	RIL	4.2.17.21
0xC24	RIL_REQ_REF_DATA.NC_WRITE_WC_ANY	RIL	4.2.17.21
0xC25	RIL_REQ_REF_DATA.NC_WRITE_WC_FULL	RIL	4.2.17.21
0xC26	RIL_REQ_REF_DATA.NC_WRITE_UC	RIL	4.2.17.21
0xC27	RIL_REQ_REF_DATA.NC_WRITE_WC_MLD	RIL	4.2.17.21
0xC28	RIL_REQ_REF_DATA.DRQ_ANY	RIL	4.2.17.21
0xC29	RIL_REQ_REF_DATA.WRQ_ANY	RIL	4.2.17.21
0xC2A	RIL_REQ_OTHER.WRTBCK_WRQ	RIL	4.2.17.19
0xC2B	RIL_REQ_OTHER.WRQ_FC_FCI	RIL	4.2.17.19
0xC2C	RIL_REQ_OTHER.WRTBCK_WRQ_SKIP	RIL	4.2.17.19
0xC2D	RIL_REQ_OTHER.WRQ_SKIP_LRUHINT	RIL	4.2.17.19
0xC2E	RIL_REQ_OTHER.WRTBCK_MLD_EVICT	RIL	4.2.17.19
0xC2F	RIL_REQ_OTHER.WRTBCK_MLD_FC	RIL	4.2.17.19
0xC30	RIL_REQ_OTHER.FC	RIL	4.2.17.19
0xC30	RETIRED_INST_FC	Execution	4.2.3.19
0xC31	RIL_REQ_OTHER.FCI	RIL	4.2.17.19
0xC31	RETIRED_INST_FCI	Execution	4.2.3.20

Table 4-1. All Performance Monitors Ordered by Code (Sheet 16 of 20)

Event Code	Symbol Name	Area	Section
0xC32	RIL_REQ_OTHER.CC	RIL	4.2.17.19
0xC33	RIL_REQ_OTHER.DRQ_ANY	RIL	4.2.17.19
0xC34	RIL_REQ_OTHER.PTCG	RIL	4.2.17.19
0xC34	RETIRED_INST_PTCG	Execution	4.2.3.26
0xC35	RIL_REQ_OTHER.PTCG_PEND	RIL	4.2.17.19
0xC36	RIL_REQ_OTHER.LRUHINT_FROM_MLD	RIL	4.2.17.19
0xC37	RIL_REQ_OTHER.LRUHINT_ANY	RIL	4.2.17.19
0xC38	RIL_REQ_OTHER.LRUHINT_MLD	RIL	4.2.17.19
0xC39	RIL_REQ_OTHER.LRUHINT_MISS_ANY	RIL	4.2.17.19
0xC3A	RIL_REQ_OTHER.LRUHINT_MISS_MLD	RIL	4.2.17.19
0xC3B	RIL_REQ_HINT_NRU	RIL	4.2.17.18
0xC3C	LLC_REF_HIT.ANY	LLC	4.2.19.1
0xC3D	LLC_REF_HIT.NO_SNOOP	LLC	4.2.19.1
0xC3E	LLC_REF_HIT.SNOOP	LLC	4.2.19.1
0xC3F	LLC_REF_HIT.SNOOP_FWD	LLC	4.2.19.1
0xC40	LLC_REF_SYS_ANY	LLC	4.2.19.5
0xC41	LLC_REF_MISS.ANY	LLC	4.2.19.2
0xC42	LLC_REF_MISS.MEM_LCL_ANY	LLC	4.2.19.2
0xC43	LLC_REF_MISS.MEM_RMT_ANY	LLC	4.2.19.2
0xC44	LLC_REF_MISS.MEM_LCL_NO_SNOOP	LLC	4.2.19.2
0xC45	LLC_REF_MISS.MEM_LCL_SNOOP	LLC	4.2.19.2
0xC46	LLC_REF_MISS.MEM_LCL_SNOOP_FWD	LLC	4.2.19.2
0xC47	LLC_REF_MISS.MEM_RMT_NO_SNOOP	LLC	4.2.19.2
0xC48	LLC_REF_MISS.MEM_RMT_SNOOP	LLC	4.2.19.2
0xC49	LLC_REF_MISS.MEM_RMT_SNOOP_FWD	LLC	4.2.19.2
0xC4A	LLC_REF_UNKNOWN	LLC	4.2.19.6
0xC4B	LLC_REF_MISS_DATA.ANY	LLC	4.2.19.3
0xC4C	LLC_REF_MISS_DATA.READ	LLC	4.2.19.3
0xC4D	LLC_REF_MISS_INST.ANY	LLC	4.2.19.4
0xC4E	LLC_REF_MISS_INST.PRIMARY	LLC	4.2.19.4
0xC4F	RIL_SHOOTDOWN	RIL	4.2.17.26
0xC50	RIL_SHOOTDOWN_PEND_CYC	RIL	4.2.17.27
0xC51	RIL_INTERRUPT	RIL	4.2.17.14
0xC52	RIL_CBQ_EVICT.WCB_FLUSH	RIL	4.2.17.3
0xC53	RIL_CBQ_EVICT.FULL	RIL	4.2.17.3
0xC54	RIL_DATA_RETURN.PRI_ANY	RIL	4.2.17.8
0xC55	RIL_DATA_RETURN.PRI_MLD	RIL	4.2.17.8
0xC56	RIL_DATA_RETURN.MLD_ANY	RIL	4.2.17.8
0xC57	RIL_DATA_RETURN.MLD_CRIT	RIL	4.2.17.8
0xC58	RIL_DATA_RETURN.EARLY_FILL_EM	RIL	4.2.17.8
0xC59	RIL_DATA_RETURN.EARLY_FILL_S	RIL	4.2.17.8

Table 4-1. All Performance Monitors Ordered by Code (Sheet 17 of 20)

Event Code	Symbol Name	Area	Section
0xC5A	RIL_RESP.GO	RIL	4.2.17.23
0xC5B	RIL_RESP.WRITEPULL	RIL	4.2.17.23
0xC5C	RIL_BL_WRITE.ANY	RIL	4.2.17.2
0xC5D	RIL_BL_WRITE.WLB	RIL	4.2.17.2
0xC5E	RIL_BL_WRITE.WLB_BOGUS	RIL	4.2.17.2
0xC5F	RIL_BL_WRITE.SLB	RIL	4.2.17.2
0xC60	RIL_ARB_PRI_LOST.AD	RIL	4.2.17.1
0xC61	RIL_ARB_PRI_LOST.AD_FWD_PROG	RIL	4.2.17.1
0xC62	RIL_ARB_PRI_LOST.BL	RIL	4.2.17.1
0xC63	RIL_ARB_PRI_LOST.BL_FWD_PROG	RIL	4.2.17.1
0xC64	RIL_CRDT_PRI_BLK.AD_ALL	RIL	4.2.17.6
0xC65	RIL_CRDT_PRI_BLK.AD_FRQ	RIL	4.2.17.6
0xC66	RIL_CRDT_PRI_BLK.AD_DRQ	RIL	4.2.17.6
0xC67	RIL_CRDT_PRI_BLK.AD_WRQ	RIL	4.2.17.6
0xC68	RIL_CRDT_PRI_BLK.AD_CBQ	RIL	4.2.17.6
0xC69	RIL_CRDT_PRI_BLK.AK_ALL	RIL	4.2.17.6
0xC6A	RIL_CRDT_PRI_BLK.BL_ALL	RIL	4.2.17.6
0xC6B	RIL_CRDT_PRI_BLK.BL_SNQ	RIL	4.2.17.6
0xC6C	RIL_CRDT_PRI_BLK.BL_WRQ	RIL	4.2.17.6
0xC6D	RIL_CRDT_PRI_BLK.BL_CBQ	RIL	4.2.17.6
0xC6E	RIL_CRDT_MLD_FDB_FULL	RIL	4.2.17.4
0xC6F	RIL_CRDT_MLD_FDB_FULL_BLK	RIL	4.2.17.5
0xC70	RIL_CRDT_SNQ_BLK.ANY	RIL	4.2.17.7
0xC71	RIL_CRDT_SNQ_BLK.HALT	RIL	4.2.17.7
0xC72	RIL_CRDT_SNQ_BLK.SRLZ	RIL	4.2.17.7
0xC73	RIL_CRDT_SNQ_BLK.MLI_FWD_PROG	RIL	4.2.17.7
0xC74	RIL_CRDT_SNQ_BLK.MLD_FWD_PROG	RIL	4.2.17.7
0xC75	RIL_CRDT_SNQ_BLK.MLI_OR_MLD_FWD_PROG	RIL	4.2.17.7
0xC76	RIL_CRDT_SNQ_BLK.MLI_FULL	RIL	4.2.17.7
0xC77	RIL_CRDT_SNQ_BLK.MLD_FULL	RIL	4.2.17.7
0xC78	RIL_CRDT_SNQ_BLK.MLI_OR_MLD_FULL	RIL	4.2.17.7
0xC79	RIL_CRDT_SNQ_BLK.DFRQ	RIL	4.2.17.7
0xC7A	RIL_CRDT_SNQ_BLK.RSPQ	RIL	4.2.17.7
0xC7B	RIL_CRDT_SNQ_BLK.SLB_DQ	RIL	4.2.17.7
0xC7C	RIL_CRDT_SNQ_BLK.WLB_DQ	RIL	4.2.17.7
0xC7D	RIL_CRDT_SNQ_BLK.ANY_Q_FULL	RIL	4.2.17.7
0xC7E	RIL_FRQ.EMPTY	RIL	4.2.17.12
0xC7F	RIL_FRQ.LIMIT_HIT	RIL	4.2.17.12
0xC80	RIL_DRQ.EMPTY	RIL	4.2.17.9
0xC81	RIL_DRQ.LIMIT_HIT	RIL	4.2.17.9
0xC82	RIL_WRQ.EMPTY	RIL	4.2.17.32

Table 4-1. All Performance Monitors Ordered by Code (Sheet 18 of 20)

Event Code	Symbol Name	Area	Section
0xC83	RIL_WRQ.LIMIT_HIT	RIL	4.2.17.32
0xC84	RIL_RRQ.LIMIT_HIT	RIL	4.2.17.24
0xC85	RIL_SNQ.EMPTY	RIL	4.2.17.30
0xC86	RIL_SNQ.LIMIT_HIT	RIL	4.2.17.30
0xC87	RIL_FRQ_VALID.MSB	RIL	4.2.17.13
0xC88	RIL_FRQ_VALID.LSB	RIL	4.2.17.13
0xC89	RIL_DRQ_VALID.MSB	RIL	4.2.17.11
0xC8A	RIL_DRQ_VALID.LSB	RIL	4.2.17.11
0xC8B	RIL_WRQ_VALID.MSB	RIL	4.2.17.33
0xC8C	RIL_WRQ_VALID.LSB	RIL	4.2.17.33
0xC8D	RIL_SNQ_VALID.MSB	RIL	4.2.17.31
0xC8E	RIL_SNQ_VALID.LSB	RIL	4.2.17.31
0xC8F	RIL_DRQ_PACE_BUBBLE	RIL	4.2.17.10
0xC90	RIL_PRI_THROTTLE_ASSERTED	RIL	4.2.17.15
0xC91	RIL_PRI_THROTTLE_RECOV	RIL	4.2.17.16
0xC92	RIL_SEB.PTC_QUIESCE_PEND	RIL	4.2.17.25
0xC93	RIL_SEB.LDST_QUIESCE_PEND	RIL	4.2.17.25
0xC94	RIL_SEB.BGF_QUIESCE_ACTIVE	RIL	4.2.17.25
0xC96	RIL_CRDT_SNQ_BLK.USEMANY_ANY	RIL	4.2.17.7
0xC97	RIL_CRDT_SNQ_BLK.USEMANY_BYP	RIL	4.2.17.7
0xC9B	UNCORE_FREEZE	System	4.2.20.4
0xC9C	DTLB_HPWREQ_BLK_MISS.SUCCEED	MLDTLB	4.2.15.2
0xC9D	DTLB_REF.NONSPEC	MLDTLB	4.2.15.4
0xC9E	DTLB_HPWREQ_SPEC_MISS	MLDTLB	4.2.15.3
0xC9F	DTLB_HPWREQ_BLK_MISS.COAL	MLDTLB	4.2.15.2
0xCA0	DTLB_HPWREQ_BLK_MISS.FAIL	MLDTLB	4.2.15.2
0xCA1	CSPEC_LOAD.ANY	Execution	4.2.3.5
0xCA2	CSPEC_LOAD.NAT	Execution	4.2.3.5
0xCA3	DTLB_HPWHINT_BLK	MLDTLB	4.2.15.1
0xCA4	DATA_REF.LOAD_UC	FLD	4.2.12.1
0xCA5	DATA_REF.STORE_UC	FLD	4.2.12.1
0xCA6	DTLB_REF.ANY	MLDTLB	4.2.15.4
0xCA7	MLD_ISSUE_SRC.ANY	MLD	4.2.13.14
0xCA8	MLD_ISSUE_SRC.BYPASS	MLD	4.2.13.14
0xCA9	MLD_ISSUE_SRC.OZQ	MLD	4.2.13.14
0xCAA	MLD_ISSUE_SRC.SMQ	MLD	4.2.13.14
0xCAB	MLD_ISSUE_SRC.FAB	MLD	4.2.13.14
0xCAC	MLD_ISSUE_SRC.SNOOP	MLD	4.2.13.14
0xCAD	MLD_REF.ANY	MLD	4.2.13.23
0xCAE	MLD_REF.HIT	MLD	4.2.13.23
0xCAF	MLD_REF.MISS	MLD	4.2.13.23

Table 4-1. All Performance Monitors Ordered by Code (Sheet 19 of 20)

Event Code	Symbol Name	Area	Section
0xCB0	MLD_REF.PRIMARY	MLD	4.2.13.23
0xCB1	MLD_REF.SECONDARY	MLD	4.2.13.23
0xCB2	MLD_REF.SECONDARY_DROP	MLD	4.2.13.23
0xCB3	MLD_REF.UC_WC_STORE	MLD	4.2.13.23
0xCB4	MLD_LOAD.ANY	MLD	4.2.13.15
0xCB5	MLD_LOAD.HIT	MLD	4.2.13.15
0xCB6	MLD_LOAD.MISS	MLD	4.2.13.15
0xCB7	MLD_LOAD.PRIMARY	MLD	4.2.13.15
0xCB8	MLD_LOAD.SECONDARY	MLD	4.2.13.15
0xCB9	MLD_SMQ_REF.ANY	MLD	4.2.13.26
0xCBA	MLD_SMQ_REF.HIT	MLD	4.2.13.26
0xCBB	MLD_SMQ_REF.MISS	MLD	4.2.13.26
0xCBC	MLD_SMQ_REF.PRIMARY	MLD	4.2.13.26
0xCBD	MLD_SMQ_REF.SECONDARY	MLD	4.2.13.26
0xCBE	MLD_SMQ_REF.SECONDARY_DROP	MLD	4.2.13.26
0xCBF	MLD_FILL_MESI_STATE_PRIMARY.ANY	MLD	4.2.13.8
0xCC0	MLD_FILL_MESI_STATE_PRIMARY.M	MLD	4.2.13.8
0xCC1	MLD_FILL_MESI_STATE_PRIMARY.E	MLD	4.2.13.8
0xCC2	MLD_FILL_MESI_STATE_PRIMARY.S	MLD	4.2.13.8
0xCC3	MLD_FILL_MESI_STATE_PRIMARY.I	MLD	4.2.13.8
0xCC4	MLD_FILL_MESI_STATE_BUDDY.ANY	MLD	4.2.13.7
0xCC5	MLD_FILL_MESI_STATE_BUDDY.E	MLD	4.2.13.7
0xCC6	MLD_FILL_MESI_STATE_BUDDY.S	MLD	4.2.13.7
0xCC7	MLD_FILL_MESI_STATE_BUDDY.I	MLD	4.2.13.7
0xCC8	MLD_SNOOP_DEFER	MLD	4.2.13.27
0xCC9	MLD_NOALLOC_FILL	MLD	4.2.13.18
0xCCA	MLD_NOALLOC_CASTOUT	MLD	4.2.13.17
0xCCB	MLD_HINT_NOALLOC	MLD	4.2.13.10
0xCCC	MLD_HINT_NRU	MLD	4.2.13.13
0xCCD	MLD_HINT_NO_BUDDY	MLD	4.2.13.11
0xCCE	MLD_HINT_NO_MULTI_HWPREF	MLD	
0xCCF	MLD_HINT_PREF_DROP	MLD	4.2.13.12
0xCD0	MLD_HINT_DEFER	MLD	4.2.13.9
0xCD1	MLD_FAB_OVERFLOW	MLD	4.2.13.6
0xCD2	MLD_OZQ_INSERT	MLD	4.2.13.21
0xCD3	MLD_BYPASS_ATTEMPT	MLD	4.2.13.3
0xCD4	MLD_BYPASS	MLD	4.2.13.2
0xCD5	MLD_OZQ_PREEMPTED	MLD	4.2.13.22
0xCD6	MLD_BWMODE_CYC	MLD	4.2.13.1
0xCD7	MLD_SMQ_PRIORITY	MLD	4.2.13.25
0xCD8	MLD_CYC_STALL.ANY	MLD	4.2.13.4

Table 4-1. All Performance Monitors Ordered by Code (Sheet 20 of 20)

Event Code	Symbol Name	Area	Section
0xCD9	MLD_CYC_STALL.RW_BANK	MLD	4.2.13.4
0xCDA	MLD_CYC_STALL.RAW	MLD	4.2.13.4
0xCDB	MLD_CYC_STALL.SEMAPHORE	MLD	4.2.13.4
0xCDC	MLD_CYC_STALL.FILL_W	MLD	4.2.13.4
0xCDD	MLD_CYC_STALL.CRIT_BYP	MLD	4.2.13.4
0xCDE	MLD_CYC_STALL.WB_FIFO	MLD	4.2.13.4
0xCDF	MLD_CYC_STALL.HPW	MLD	4.2.13.4
0xCE0	MLD_CYC_STALL.TAG_ERR	MLD	4.2.13.4
0xCE1	MLD_CYC_STALL.SPLIT_WW_BANK	MLD	4.2.13.4
0xCE2	MLD_CYC_STALL.SPLIT_RR_BANK	MLD	4.2.13.4
0xCE3	MLD_CYC_STALL.SPLIT_RW_BANK	MLD	4.2.13.4
0xCE4	MLD_CYC_STALL.SPLIT_RAW	MLD	4.2.13.4
0xCE5	MLD_CYC_STALL.SPLIT_OVERSUB	MLD	4.2.13.4
0xCE6	MLD_CYC_STALL.SPLIT_CRIT_BYP	MLD	4.2.13.4
0xCE7	MLD_LOST_BW.ANY	MLD	4.2.13.16
0xCE8	MLD_LOST_BW.OZQ_NOP	MLD	4.2.13.16
0xCE9	MLD_LOST_BW.OZQ_NOP_ACQ	MLD	4.2.13.16
0xCEA	MLD_LOST_BW.OZQ_FAB_FULL	MLD	4.2.13.16
0xCEB	MLD_LOST_BW.OZQ_SMQ_FULL	MLD	4.2.13.16
0xCEC	MLD_LOST_BW.OZQ_FAB_PREEMPT	MLD	4.2.13.16
0xCED	MLD_LOST_BW.OZQ_SMQ_PREEMPT	MLD	4.2.13.16
0xCEE	MLD_LOST_BW.OZQ_SNOOP_PREEMPT	MLD	4.2.13.16
0xCEF	MLD_LOST_BW.FAB_NOP	MLD	4.2.13.16
0xCF0	MLD_LOST_BW.SMQ_NOP	MLD	4.2.13.16
0xCF1	MLD_LOST_BW.SPLIT_BUBBLE	MLD	4.2.13.16
0xCF2	MLD_LOST_BW.NOP_STALL	MLD	4.2.13.16
0xCF3	MLD_LOST_BW.STALL	MLD	4.2.13.16
0xCF4	MLD_LOST_BW.NOP	MLD	4.2.13.16
0xCF5	MLD_OZQ_COUNT.LSB	MLD	4.2.13.20
0xCF6	MLD_OZQ_COUNT.MSB	MLD	4.2.13.20
0xCF7	MLD_FAB_COUNT.LSB	MLD	4.2.13.5
0xCF8	MLD_FAB_COUNT.MSB	MLD	4.2.13.5
0xCF9	MLD_SMQ_COUNT.LSB	MLD	4.2.13.24
0xCFA	MLD_SMQ_COUNT.MSB	MLD	4.2.13.24
0xCFB	MLD_WLB_COUNT.LSB	MLD	4.2.13.29
0xCFC	MLD_WLB_COUNT.MSB	MLD	4.2.13.29
0xCFD	MLD_WCB_CREDIT	MLD	4.2.13.28
0xCFE	MLD_OZDATA_COUNT.LSB	MLD	4.2.13.19
0xCFF	MLD_OZDATA_COUNT.MSB	MLD	4.2.13.19

4.1.6 Performance Monitor Events Ordered by Event Name

Table 4-2 presents all of the performance monitors provided in the processor ordered by their event name.

Table 4-2. All Performance Monitors Ordered by Name (Sheet 1 of 20)

Symbol Name	Event Code	Area	Section
ALAT_ENTRY_REPLACED	0x152	Execution	4.2.3.1
ALAT_STORE_HIT	0x151	Execution	4.2.3.2
BR_BE_PRED_DETAIL.ANY	0x807	Branch	4.2.6.1
BR_BE_PRED_DETAIL.ANY_RETIRED	0x805	Branch	4.2.6.1
BR_BE_PRED_DETAIL.OTHER	0x804	Branch	4.2.6.1
BR_BE_PRED_DETAIL.PFS	0x803	Branch	4.2.6.1
BR_BE_PRED_DETAIL.ROT	0x802	Branch	4.2.6.1
BR_BE_PRED_DETAIL.STG	0x801	Branch	4.2.6.1
BR_BE_PRED_DETAIL.UNRETIRED	0x806	Branch	4.2.6.1
BR_ENC_PRED_DETAIL.ENC_ANY_PRED	0x830	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_CORR_PRED	0x831	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_OVRSUB_ANY_PRED	0x838	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_OVRSUB_CORR_PRED	0x839	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_OVRSUB_WRONG_PATH	0x83A	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_OVRSUB_WRONG_TARGET	0x83B	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_WRONG_PATH	0x832	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.ENC_WRONG_TARGET	0x833	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.OVRSUB_ANY_PRED	0x834	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.OVRSUB_CORR_PRED	0x835	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.OVRSUB_WRONG_PATH	0x836	Branch	4.2.6.2
BR_ENC_PRED_DETAIL.OVRSUB_WRONG_TARGET	0x837	Branch	4.2.6.2
BR_PATH_PRED.ANY_MISPRED_NOT_TAKEN	0x808	Branch	4.2.6.3
BR_PATH_PRED.ANY_MISPRED_TAKEN	0x809	Branch	4.2.6.3
BR_PATH_PRED.ANY_OKPRED_NOT_TAKEN	0x80A	Branch	4.2.6.3
BR_PATH_PRED.ANY_OKPRED_TAKEN	0x80B	Branch	4.2.6.3
BR_PATH_PRED.IPREL_MISPRED_NOT_TAKEN	0x80C	Branch	4.2.6.3
BR_PATH_PRED.IPREL_MISPRED_TAKEN	0x80D	Branch	4.2.6.3
BR_PATH_PRED.IPREL_OKPRED_NOT_TAKEN	0x80E	Branch	4.2.6.3
BR_PATH_PRED.IPREL_OKPRED_TAKEN	0x80F	Branch	4.2.6.3
BR_PATH_PRED.NON_RETIND_MISPRED_NOT_TAKEN	0x814	Branch	4.2.6.3
BR_PATH_PRED.NON_RETIND_MISPRED_TAKEN	0x815	Branch	4.2.6.3
BR_PATH_PRED.NON_RETIND_OKPRED_NOT_TAKEN	0x816	Branch	4.2.6.3
BR_PATH_PRED.NON_RETIND_OKPRED_TAKEN	0x817	Branch	4.2.6.3
BR_PATH_PRED.RETURN_MISPRED_NOT_TAKEN	0x810	Branch	4.2.6.3
BR_PATH_PRED.RETURN_MISPRED_TAKEN	0x811	Branch	4.2.6.3
BR_PATH_PRED.RETURN_OKPRED_NOT_TAKEN	0x812	Branch	4.2.6.3
BR_PATH_PRED.RETURN_OKPRED_TAKEN	0x813	Branch	4.2.6.3

Table 4-2. All Performance Monitors Ordered by Name (Sheet 2 of 20)

Symbol Name	Event Code	Area	Section
BR_PRED_DETAIL.ANY_ANY_PRED	0x820	Branch	4.2.6.4
BR_PRED_DETAIL.ANY_CORR_PRED	0x821	Branch	4.2.6.4
BR_PRED_DETAIL.ANY_WRONG_PATH	0x822	Branch	4.2.6.4
BR_PRED_DETAIL.ANY_WRONG_TARGET	0x823	Branch	4.2.6.4
BR_PRED_DETAIL.IPREL_ANY_PRED	0x824	Branch	4.2.6.4
BR_PRED_DETAIL.IPREL_CORR_PRED	0x825	Branch	4.2.6.4
BR_PRED_DETAIL.IPREL_WRONG_PATH	0x826	Branch	4.2.6.4
BR_PRED_DETAIL.IPREL_WRONG_TARGET	0x827	Branch	4.2.6.4
BR_PRED_DETAIL.NON_RETIND_ANY_PRED	0x82C	Branch	4.2.6.4
BR_PRED_DETAIL.NON_RETIND_CORR_PRED	0x82D	Branch	4.2.6.4
BR_PRED_DETAIL.NON_RETIND_WRONG_PATH	0x82E	Branch	4.2.6.4
BR_PRED_DETAIL.NON_RETIND_WRONG_TARGET	0x82F	Branch	4.2.6.4
BR_PRED_DETAIL.RETURN_ANY_PRED	0x828	Branch	4.2.6.4
BR_PRED_DETAIL.RETURN_CORR_PRED	0x829	Branch	4.2.6.4
BR_PRED_DETAIL.RETURN_WRONG_PATH	0x82A	Branch	4.2.6.4
BR_PRED_DETAIL.RETURN_WRONG_TARGET	0x82B	Branch	4.2.6.4
BR_PRED_UNKNOWN.ANY	0x818	Branch	4.2.6.5
BR_PRED_UNKNOWN.ANY_TAKEN	0x819	Branch	4.2.6.5
BR_PRED_UNKNOWN.IPREL	0x81A	Branch	4.2.6.5
BR_PRED_UNKNOWN.IPREL_TAKEN	0x81B	Branch	4.2.6.5
BR_PRED_UNKNOWN.NON_RETIND	0x81E	Branch	4.2.6.5
BR_PRED_UNKNOWN.NON_RETIND_TAKEN	0x81F	Branch	4.2.6.5
BR_PRED_UNKNOWN.RETURN	0x81C	Branch	4.2.6.5
BR_PRED_UNKNOWN.RETURN_TAKEN	0x81D	Branch	4.2.6.5
CPU_CPL_CHANGE.ANY	0x01A	System	4.2.20.1
CPU_CPL_CHANGE.FROM0	0x01B	System	4.2.20.1
CPU_CPL_CHANGE.FROM1	0x01C	System	4.2.20.1
CPU_CPL_CHANGE.FROM2	0x01D	System	4.2.20.1
CPU_CPL_CHANGE.FROM3	0x01E	System	4.2.20.1
CPU_OP_CYCLES	0x002	Basic	4.2.1.1
CPU_OP_CYCLES.HALTED	0x003	Basic	4.2.1.1
CPU_OP_CYCLES.TAGGED	0x004	Basic	4.2.1.1
CPU_REF_CYCLES	0x001	Basic	4.2.1.2
CSPEC_CHKS_FAIL.ANY	0x015	Execution	4.2.3.4
CSPEC_CHKS_FAIL.FP	0x017	Execution	4.2.3.4
CSPEC_CHKS_FAIL.INT	0x016	Execution	4.2.3.4
CSPEC_CHKS.ANY	0x014	Execution	4.2.3.3
CSPEC_LOAD.ANY	0xCA1	Execution	4.2.3.5
CSPEC_LOAD.NAT	0xCA2	Execution	4.2.3.5
CYC_BE_BUBBLE.ANY	0x024	Stall	4.2.4.1
CYC_BE_DET_REPLAY.ANY	0x04A	Stall	4.2.4.2

Table 4-2. All Performance Monitors Ordered by Name (Sheet 3 of 20)

Symbol Name	Event Code	Area	Section
CYC_BE_DET_REPLAY.DCS_HZRD	0x04C	Stall	4.2.4.2
CYC_BE_DET_REPLAY.FLUSH_STORE	0x050	Stall	4.2.4.2
CYC_BE_DET_REPLAY.GR_LOAD	0x04B	Stall	4.2.4.2
CYC_BE_DET_REPLAY.HPW_HZRD	0x051	Stall	4.2.4.2
CYC_BE_DET_REPLAY.LOAD_ACQ	0x04F	Stall	4.2.4.2
CYC_BE_DET_REPLAY.LOAD_AFTER_WRITE	0x04E	Stall	4.2.4.2
CYC_BE_DET_REPLAY.MT1	0x054	Stall	4.2.4.2
CYC_BE_DET_REPLAY.STORE_VS_STORE	0x04D	Stall	4.2.4.2
CYC_BE_DET_REPLAY.WRITE_HIT_VS_FILL	0x052	Stall	4.2.4.2
CYC_BE_DET_REPLAY.WRITE_MISS_VS_FILL	0x053	Stall	4.2.4.2
CYC_BE_EXE_REPLAY.ANY	0x03A	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.ARCR	0x048	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.FCMP	0x042	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.FPSR	0x045	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.FR_FR	0x040	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.FR_LOAD_RAW	0x03C	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.FR_LOAD_WAW	0x03E	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.GR_GR	0x03F	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.GR_LOAD_RAW	0x03B	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.GR_LOAD_WAW	0x03D	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.MT1_HIGH	0x041	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.MT1_LOW	0x049	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.NOTN	0x044	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.PRED	0x043	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.REL	0x047	Stall	4.2.4.3
CYC_BE_EXE_REPLAY.SRLZ	0x046	Stall	4.2.4.3
CYC_BE_IBD_STALL.ACQ	0x02E	Stall	4.2.4.4
CYC_BE_IBD_STALL.ANY	0x025	Stall	4.2.4.4
CYC_BE_IBD_STALL.DEBUG	0x039	Stall	4.2.4.4
CYC_BE_IBD_STALL.FEBUB	0x038	Stall	4.2.4.4
CYC_BE_IBD_STALL.FLD_DMND	0x035	Stall	4.2.4.4
CYC_BE_IBD_STALL.FR_LOAD	0x030	Stall	4.2.4.4
CYC_BE_IBD_STALL.FTOF	0x034	Stall	4.2.4.4
CYC_BE_IBD_STALL.GR_LOAD	0x02F	Stall	4.2.4.4
CYC_BE_IBD_STALL.HPW	0x02C	Stall	4.2.4.4
CYC_BE_IBD_STALL.MTOM	0x033	Stall	4.2.4.4
CYC_BE_IBD_STALL.OZQFULL	0x02D	Stall	4.2.4.4
CYC_BE_IBD_STALL.QFULL	0x037	Stall	4.2.4.4
CYC_BE_IBD_STALL.REL	0x032	Stall	4.2.4.4
CYC_BE_IBD_STALL.RSE_ANY	0x026	Stall	4.2.4.4
CYC_BE_IBD_STALL.RSE_CFLE	0x027	Stall	4.2.4.4

Table 4-2. All Performance Monitors Ordered by Name (Sheet 4 of 20)

Symbol Name	Event Code	Area	Section
CYC_BE_IBD_STALL.RSE_LOAD	0x029	Stall	4.2.4.4
CYC_BE_IBD_STALL.RSE_ST	0x028	Stall	4.2.4.4
CYC_BE_IBD_STALL.RSE_WAIT	0x02A	Stall	4.2.4.4
CYC_BE_IBD_STALL.SRLZ	0x031	Stall	4.2.4.4
CYC_BE_IBD_STALL.THRSW	0x02B	Stall	4.2.4.4
CYC_BE_IBD_STALL.WB2_TRAP	0x036	Stall	4.2.4.4
CYC_BE_NO_BUBBLE	0x023	Stall	4.2.4.5
CYC_BE_WB2_FLUSH.ANY	0x065	Stall	4.2.4.6
CYC_BE_WB2_FLUSH.BRU	0x067	Stall	4.2.4.6
CYC_BE_WB2_FLUSH.XPN	0x066	Stall	4.2.4.6
CYC_BE_WB2_REPLAY.ALLOC_PEC	0x058	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.ANY	0x055	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.BLK_HPW	0x05D	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.DAHR_HZRD	0x061	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.FP_DEN	0x05B	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.FP_SIR	0x05C	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.LDC	0x056	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.LOAD_ACQ	0x062	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.MOV_PSR_UM	0x059	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.MT1	0x063	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.NAT_HZRD	0x060	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.OZQ_FULL	0x05E	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.PAUSE	0x057	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.SER	0x064	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.STORE_ALIAS	0x05F	Stall	4.2.4.7
CYC_BE_WB2_REPLAY.VIRT_INT	0x05A	Stall	4.2.4.7
CYC_FE_BUBBLE.ANY	0x8A3	Stall	4.2.5.1
CYC_FE_FET_REPLAY.ANY	0x8AB	Stall	4.2.5.2
CYC_FE_FET_REPLAY.BRQ_FULL	0x8AD	Stall	4.2.5.2
CYC_FE_FET_REPLAY.BRQ_WAIT	0x8AE	Stall	4.2.5.2
CYC_FE_FET_REPLAY.BR_INIT	0x8AC	Stall	4.2.5.2
CYC_FE_FET_REPLAY.BR_INTRLCK	0x8AF	Stall	4.2.5.2
CYC_FE_FET_REPLAY.MT1	0x8B1	Stall	4.2.5.2
CYC_FE_FET_REPLAY.RAB_FULL	0x8B0	Stall	4.2.5.2
CYC_FE_FET_STALL.ANY	0x8B2	Stall	4.2.5.3
CYC_FE_FET_STALL.FLITLB_MISS	0x8B5	Stall	4.2.5.3
CYC_FE_FET_STALL.FLI_MISS	0x8B4	Stall	4.2.5.3
CYC_FE_FET_STALL.IBQ_FULL	0x8B3	Stall	4.2.5.3
CYC_FE_FET_STALL.MT1	0x8B6	Stall	4.2.5.3
CYC_FE_FWPROG	0x8A1	Multithreading	4.2.21.1
CYC_FE_NO_BUBBLE	0x8A2	Stall	4.2.5.4

Table 4-2. All Performance Monitors Ordered by Name (Sheet 5 of 20)

Symbol Name	Event Code	Area	Section
CYC_FE_RESTEER.ANY	0x8A4	Stall	4.2.5.5
CYC_FE_RESTEER.BE_FLUSH	0x8AA	Stall	4.2.5.5
CYC_FE_RESTEER.BR_RETIND	0x8A6	Stall	4.2.5.5
CYC_FE_RESTEER.IPREL	0x8A5	Stall	4.2.5.5
CYC_FE_RESTEER.NON_RETIND	0x8A7	Stall	4.2.5.5
CYC_FE_RESTEER.SEQ_BR	0x8A8	Stall	4.2.5.5
CYC_FE_RESTEER.TSWITCH	0x8A9	Stall	4.2.5.5
DAHS_UNDERFLOW	0x0C9	DPF	4.2.16.1
DATA_REF.ANY	0x0DD	FLD	4.2.12.1
DATA_REF.HW_PREF	0x0E6	FLD	4.2.12.1
DATA_REF.LFETCH	0x0E4	FLD	4.2.12.1
DATA_REF.LOAD_ANY	0x0E8	FLD	4.2.12.1
DATA_REF.LOAD_FP	0x0DF	FLD	4.2.12.1
DATA_REF.LOAD_HPW	0x0E7	FLD	4.2.12.1
DATA_REF.LOAD_INT	0x0DE	FLD	4.2.12.1
DATA_REF.LOAD_RSE	0x0E0	FLD	4.2.12.1
DATA_REF.LOAD_UC	0xCA4	FLD	4.2.12.1
DATA_REF.SEMAPHORE	0x0E5	FLD	4.2.12.1
DATA_REF.STORE_ANY	0x0E9	FLD	4.2.12.1
DATA_REF.STORE_FP	0x0E2	FLD	4.2.12.1
DATA_REF.STORE_INT	0x0E1	FLD	4.2.12.1
DATA_REF.STORE_RSE	0x0E3	FLD	4.2.12.1
DATA_REF.STORE_UC	0xCA5	FLD	4.2.12.1
DPFQ_DEQ_PREEMPT_REJECT.ANY	0x0C5	DPF	4.2.16.4
DPFQ_DEQ_PREEMPT_REJECT.LFETCH	0x0C6	DPF	4.2.16.4
DPFQ_DEQ_PREEMPT_REJECT.LFETCH_COUNT	0x0C7	DPF	4.2.16.4
DPFQ_DEQ_PREEMPT_REJECT.MOV_BSPST	0x0C8	DPF	4.2.16.4
DPFQ_DEQ_PREEMPT.ANY	0x0B1	DPF	4.2.16.3
DPFQ_DEQ_PREEMPT.INST_ANY	0x0B2	DPF	4.2.16.3
DPFQ_DEQ_PREEMPT.LFETCH	0x0B3	DPF	4.2.16.3
DPFQ_DEQ_PREEMPT.LFETCH_COUNT	0x0B4	DPF	4.2.16.3
DPFQ_DEQ_PREEMPT.MOV_BSPST	0x0B5	DPF	4.2.16.3
DPFQ_DEQ_PREEMPT.TIMEOUT	0x0B6	DPF	4.2.16.3
DPFQ_DEQ_REJECT.ANY	0x0B7	DPF	4.2.16.5
DPFQ_DEQ_REJECT.FLD_ANY	0x0BF	DPF	4.2.16.5
DPFQ_DEQ_REJECT.FLD_BIDI	0x0C3	DPF	4.2.16.5
DPFQ_DEQ_REJECT.FLD_BWD	0x0C2	DPF	4.2.16.5
DPFQ_DEQ_REJECT.FLD_FWD	0x0C1	DPF	4.2.16.5
DPFQ_DEQ_REJECT.FLD_TARGET	0x0C0	DPF	4.2.16.5
DPFQ_DEQ_REJECT.INST_ANY	0x0B8	DPF	4.2.16.5
DPFQ_DEQ_REJECT.LFETCH	0x0B9	DPF	4.2.16.5

Table 4-2. All Performance Monitors Ordered by Name (Sheet 6 of 20)

Symbol Name	Event Code	Area	Section
DPFQ_DEQ_REJECT.LFETCH_COUNT	0x0BA	DPF	4.2.16.5
DPFQ_DEQ_REJECT.MLD	0x0C4	DPF	4.2.16.5
DPFQ_DEQ_REJECT.MOV_BSPST	0x0BB	DPF	4.2.16.5
DPFQ_DEQ_REJECT.RSE_ANY	0x0BC	DPF	4.2.16.5
DPFQ_DEQ_REJECT.RSE_LOAD	0x0BD	DPF	4.2.16.5
DPFQ_DEQ_REJECT.RSE_STORE	0x0BE	DPF	4.2.16.5
DPFQ_DEQ.ANY	0x0A3	DPF	4.2.16.2
DPFQ_DEQ.FLD_ANY	0x0AB	DPF	4.2.16.2
DPFQ_DEQ.FLD_BIDI	0x0AF	DPF	4.2.16.2
DPFQ_DEQ.FLD_BWD	0x0AE	DPF	4.2.16.2
DPFQ_DEQ.FLD_FWD	0x0AD	DPF	4.2.16.2
DPFQ_DEQ.FLD_TARGET	0x0AC	DPF	4.2.16.2
DPFQ_DEQ.INST_ANY	0x0A4	DPF	4.2.16.2
DPFQ_DEQ.LFETCH	0x0A5	DPF	4.2.16.2
DPFQ_DEQ.LFETCH_COUNT	0x0A6	DPF	4.2.16.2
DPFQ_DEQ.MLD	0x0B0	DPF	4.2.16.2
DPFQ_DEQ.MOV_BSPST	0x0A7	DPF	4.2.16.2
DPFQ_DEQ.RSE_ANY	0x0A8	DPF	4.2.16.2
DPFQ_DEQ.RSE_LOAD	0x0A9	DPF	4.2.16.2
DPFQ_DEQ.RSE_STORE	0x0AA	DPF	4.2.16.2
DPFQ_ENQ_OVERFLOW.ANY	0x095	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.FLD_ANY	0x09D	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.FLD_BIDI	0x0A1	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.FLD_BWD	0x0A0	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.FLD_FWD	0x09F	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.FLD_TARGET	0x09E	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.INST_ANY	0x096	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.LFETCH	0x097	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.LFETCH_COUNT	0x098	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.MLD	0x0A2	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.MOV_BSPST	0x099	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.RSE_ANY	0x09A	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.RSE_LOAD	0x09B	DPF	4.2.16.7
DPFQ_ENQ_OVERFLOW.RSE_STORE	0x09C	DPF	4.2.16.7
DPFQ_ENQ.ANY	0x087	DPF	4.2.16.6
DPFQ_ENQ.FLD_ANY	0x08F	DPF	4.2.16.6
DPFQ_ENQ.FLD_BIDI	0x093	DPF	4.2.16.6
DPFQ_ENQ.FLD_BWD	0x092	DPF	4.2.16.6
DPFQ_ENQ.FLD_FWD	0x091	DPF	4.2.16.6
DPFQ_ENQ.FLD_TARGET	0x090	DPF	4.2.16.6
DPFQ_ENQ.INST_ANY	0x088	DPF	4.2.16.6

Table 4-2. All Performance Monitors Ordered by Name (Sheet 7 of 20)

Symbol Name	Event Code	Area	Section
DPFQ_ENQ.LFETCH	0x089	DPF	4.2.16.6
DPFQ_ENQ.LFETCH_COUNT	0x08A	DPF	4.2.16.6
DPFQ_ENQ.MLD	0x094	DPF	4.2.16.6
DPFQ_ENQ.MOV_BSPST	0x08B	DPF	4.2.16.6
DPFQ_ENQ.RSE_ANY	0x08C	DPF	4.2.16.6
DPFQ_ENQ.RSE_LOAD	0x08D	DPF	4.2.16.6
DPFQ_ENQ.RSE_STORE	0x08E	DPF	4.2.16.6
DSPEC_CHKA_LDC_FAIL.ANY	0x011	Execution	4.2.3.7
DSPEC_CHKA_LDC_FAIL.FP	0x013	Execution	4.2.3.7
DSPEC_CHKA_LDC_FAIL.INT	0x012	Execution	4.2.3.7
DSPEC_CHKA_LDC.ANY	0x00F	Execution	4.2.3.6
DSPEC_LDC.HIT	0x010	Execution	4.2.3.8
DTLB_HPWHINT_BLK	0xCA3	MLDTLB	4.2.15.1
DTLB_HPWREQ_BLK_MISS.SUCCEED	0xC9C	MLDTLB	4.2.15.2
DTLB_HPWREQ_BLK_MISS.COAL	0xC9F	MLDTLB	4.2.15.2
DTLB_HPWREQ_BLK_MISS.FAIL	0xCA0	MLDTLB	4.2.15.2
DTLB_HPWREQ_SPEC_MISS	0xC9E	MLDTLB	4.2.15.3
DTLB_REF.ANY	0xCA6	MLDTLB	4.2.15.4
DTLB_REF.NONSPEC	0xC9D	MLDTLB	4.2.15.4
EAR_EVENT_DATA	0x019	Execution	4.2.3.9
EAR_EVENT_ETB	0x018	Execution	4.2.3.10
EAR_EVENT_INST	0x871	Execution	4.2.3.11
FE_OP_CYCLES	0x875	Basic	4.2.1.3
FE_RESTEER.0B_IPREL	0x8B9	Stall	4.2.5.6
FE_RESTEER.1B_BR_RETIND	0x8BC	Stall	4.2.5.6
FE_RESTEER.1B_IPREL	0x8BA	Stall	4.2.5.6
FE_RESTEER.1B_SEQ_BR	0x8BF	Stall	4.2.5.6
FE_RESTEER.3B_BR_RETIND	0x8BD	Stall	4.2.5.6
FE_RESTEER.3B_IPREL	0x8BB	Stall	4.2.5.6
FE_RESTEER.3B_MT	0x8C1	Stall	4.2.5.6
FE_RESTEER.3B_NON_RETIND	0x8BE	Stall	4.2.5.6
FE_RESTEER.3B_SEQ_BR	0x8C0	Stall	4.2.5.6
FE_RESTEER.4B_MT	0x8C2	Stall	4.2.5.6
FE_RESTEER.ANY	0x8B7	Stall	4.2.5.6
FE_RESTEER.BE_FLUSH	0x8C3	Stall	4.2.5.6
FE_RESTEER.FET_REPLAY	0x8B8	Stall	4.2.5.6
FLDTLB_INS_REQ.CANCEL	0x109	FLDTLB	4.2.14.1
FLDTLB_INS_REQ.COMPLETE	0x106	FLDTLB	4.2.14.1
FLDTLB_INS_REQ.NON_RETIRED	0x108	FLDTLB	4.2.14.1
FLDTLB_INS_REQ.RETIRED	0x107	FLDTLB	4.2.14.1
FLDTLB_LOAD_MISS.ANY	0x103	FLDTLB	4.2.14.2

Table 4-2. All Performance Monitors Ordered by Name (Sheet 8 of 20)

Symbol Name	Event Code	Area	Section
FLDTLB_LOAD_MISS.INT	0x104	FLDTLB	4.2.14.2
FLDTLB_LOAD_MISS.RSE	0x105	FLDTLB	4.2.14.2
FLD_FILL	0x101	FLD	4.2.12.1
FLD_FILL_CANCEL.ANY	0x0FD	FLD	4.2.12.2
FLD_FILL_CANCEL.INFAB	0x0FF	FLD	4.2.12.2
FLD_FILL_CANCEL.MLD	0x0FE	FLD	4.2.12.2
FLD_FILL_CANCEL.POSTFAB	0x100	FLD	4.2.12.2
FLD_FILL_LRU	0x102	FLD	4.2.12.3
FLD_FILL_REQ.ANY	0x0F8	FLD	4.2.12.4
FLD_FILL_REQ.HW_PREF	0x0FC	FLD	4.2.12.4
FLD_FILL_REQ.LFETCH	0x0FB	FLD	4.2.12.4
FLD_FILL_REQ.LOAD_INT	0x0F9	FLD	4.2.12.4
FLD_FILL_REQ.LOAD_RSE	0x0FA	FLD	4.2.12.4
FLD_HINT_NOALLOC	0x0F6	FLD	4.2.12.5
FLD_HINT_NO_MULTI_HWPREF	0x0D8	FLD	4.2.12.6
FLD_HIT.ANY	0x0EF	FLD	4.2.12.7
FLD_HWPREF_INS.ACQ_PEND	0x0D3	DPF	4.2.16.8
FLD_HWPREF_INS.ANY	0x0CB	DPF	4.2.16.8
FLD_HWPREF_INS.CANCEL_FILL	0x0CC	DPF	4.2.16.8
FLD_HWPREF_INS.DTLB_MISS	0x0CD	DPF	4.2.16.8
FLD_HWPREF_INS.DTLB_MISS_LFETCH	0x0D5	DPF	4.2.16.8
FLD_HWPREF_INS.FLDTLB_MISS	0x0CE	DPF	4.2.16.8
FLD_HWPREF_INS.FLDTLB_MISS_LFETCH	0x0D6	DPF	4.2.16.8
FLD_HWPREF_INS.FLUSH_STORE	0x0D2	DPF	4.2.16.8
FLD_HWPREF_INS.NEIGHBOR	0x0CF	DPF	4.2.16.8
FLD_HWPREF_INS.OZQ_FULL	0x0D1	DPF	4.2.16.8
FLD_HWPREF_INS.OZQ_FULL_LFETCH	0x0D7	DPF	4.2.16.8
FLD_HWPREF_INS.REL_OP	0x0D4	DPF	4.2.16.8
FLD_HWPREF_INS.STORE_ALIAS	0x0D0	DPF	4.2.16.8
FLD_LINE_DEMOTE	0x0F7	FLD	4.2.12.8
FLD_LOAD_MISS.ANY	0x0EC	FLD	4.2.12.10
FLD_LOAD_MISS.INT	0x0ED	FLD	4.2.12.10
FLD_LOAD_MISS.RSE	0x0EE	FLD	4.2.12.10
FLD_LOAD.ANY	0x0EA	FLD	4.2.12.9
FLD_LOAD.INT	0x0EB	FLD	4.2.12.9
FLD_SPEC_INVAL.ANY	0x0F0	FLD	4.2.12.11
FLD_SPEC_INVAL.FLUSH_STORE	0x0F2	FLD	4.2.12.11
FLD_SPEC_INVAL.INST	0x0F1	FLD	4.2.12.11
FLD_SPEC_INVAL.SNOOP	0x0F3	FLD	4.2.12.11
FLITLB_INSERT_HPW	0x86F	FLITLB	4.2.10.1
FLITLB_MISS	0x86D	FLITLB	4.2.10.2

Table 4-2. All Performance Monitors Ordered by Name (Sheet 9 of 20)

Symbol Name	Event Code	Area	Section
FLI_FETCH_JIT_HIT	0x86B	FLI	4.2.8.1
FLI_FETCH_RAB_HIT.DMND	0x869	FLI	4.2.8.2
FLI_FETCH_RAB_HIT.PREF	0x86A	FLI	4.2.8.2
FLI_FILL	0x85C	FLI	4.2.8.3
FLI_INST_INSERT_RAB	0x872	FLI	4.2.8.4
FLI_PREF_STALL.ANY	0x861	FLI	4.2.8.5
FLI_PREF_STALL.FLOW	0x862	FLI	4.2.8.5
FLI_PURGE	0x865	FLI	4.2.8.6
FLI_PVAB_OVERFLOW	0x86C	FLI	4.2.8.7
FLI_RAB_ALMOST_FULL	0x868	FLI	4.2.8.8
FLI_RAB_FULL	0x867	FLI	4.2.8.9
FLI_READ_MISS.ANY	0x85A	FLI	4.2.8.11
FLI_READ_MISS.DMND	0x85D	FLI	4.2.8.11
FLI_READ_MISS.PREF	0x85F	FLI	4.2.8.11
FLI_READ.ANY	0x859	FLI	4.2.8.10
FLI_READ.DMND	0x85B	FLI	4.2.8.10
FLI_READ.PREF	0x85E	FLI	4.2.8.10
FLI_READ.SNOOP	0x863	FLI	4.2.8.10
FLI_READ.SNOOP_HIT	0x864	FLI	4.2.8.10
FLI_STEPPING	0x873	FLI	4.2.8.12
FLI_STREAM_PREF	0x866	FLI	4.2.8.13
FP_DENORMAL	0x856	Execution	4.2.8.12
FP_FALSE_SIR	0x854	Execution	4.2.8.13
FP_FCHKF_FAIL	0x853	Execution	4.2.3.14
FP_FLOP	0x00C	Execution	4.2.3.15
FP_FLUSH_TO_ZERO.FTZ_POSS	0x858	Execution	4.2.3.16
FP_FLUSH_TO_ZERO.FTZ_REAL	0x857	Execution	4.2.3.16
FP_TRUE_SIR	0x855	Execution	4.2.3.17
IA64_INST_RETIRED	0x005	Basic	4.2.1.4
IBL_ISSUE_LOST_BW.9PLUS3	0x086	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.ANY	0x07A	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.ASYM_I	0x07D	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.ASYM_M	0x07E	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.DROOP	0x07C	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.FLD_DMND_M0	0x07F	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.FLD_DMND_M1	0x080	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.OVRSUB_A	0x081	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.OVRSUB_F	0x082	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.OVRSUB_I	0x083	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.OVRSUB_M	0x084	Dispersal	4.2.2.1
IBL_ISSUE_LOST_BW.POWER	0x07B	Dispersal	4.2.2.1

Table 4-2. All Performance Monitors Ordered by Name (Sheet 10 of 20)

Symbol Name	Event Code	Area	Section
IBL_ISSUE_LOST_BW.STRUCT	0x085	Dispersal	4.2.2.1
IBL_ISSUE_STOP.9PLUS3	0x079	Dispersal	4.2.2.3
IBL_ISSUE_STOP.ASYM_I	0x06F	Dispersal	4.2.2.3
IBL_ISSUE_STOP.ASYM_M	0x070	Dispersal	4.2.2.3
IBL_ISSUE_STOP.BUNDLE	0x078	Dispersal	4.2.2.3
IBL_ISSUE_STOP.DROOP	0x06E	Dispersal	4.2.2.3
IBL_ISSUE_STOP.EXPLICIT	0x06C	Dispersal	4.2.2.3
IBL_ISSUE_STOP.FLD_DMND_M0	0x071	Dispersal	4.2.2.3
IBL_ISSUE_STOP.FLD_DMND_M1	0x072	Dispersal	4.2.2.3
IBL_ISSUE_STOP.NONE	0x06A	Dispersal	4.2.2.3
IBL_ISSUE_STOP.OVRSUB_A	0x075	Dispersal	4.2.2.3
IBL_ISSUE_STOP.OVRSUB_F	0x076	Dispersal	4.2.2.3
IBL_ISSUE_STOP.OVRSUB_I	0x074	Dispersal	4.2.2.3
IBL_ISSUE_STOP.OVRSUB_M	0x073	Dispersal	4.2.2.3
IBL_ISSUE_STOP.POWER	0x06D	Dispersal	4.2.2.3
IBL_ISSUE_STOP.REPLAY	0x06B	Dispersal	4.2.2.3
IBL_ISSUE_STOP.STRUCT	0x077	Dispersal	4.2.2.3
IBL_ISSUE.ANY	0x068	Dispersal	4.2.2.2
IBL_ISSUE.M_PIPE	0x069	Dispersal	4.2.2.2
INTERRUPT_EVENT.MASKED	0x01F	System	4.2.20.2
INTERRUPT_EVENT.UNMASKED	0x020	System	4.2.20.2
LLC_REF_HIT.ANY	0xC3C	LLC	4.2.19.1
LLC_REF_HIT.NO_SNOOP	0xC3D	LLC	4.2.19.1
LLC_REF_HIT.SNOOP	0xC3E	LLC	4.2.19.1
LLC_REF_HIT.SNOOP_FWD	0xC3F	LLC	4.2.19.1
LLC_REF_MISS_DATA.ANY	0xC4B	LLC	4.2.19.3
LLC_REF_MISS_DATA.READ	0xC4C	LLC	4.2.19.3
LLC_REF_MISS_INST.ANY	0xC4D	LLC	4.2.19.4
LLC_REF_MISS_INST.PRIMARY	0xC4E	LLC	4.2.19.4
LLC_REF_MISS.ANY	0xC41	LLC	4.2.19.2
LLC_REF_MISS.MEM_LCL_ANY	0xC42	LLC	4.2.19.2
LLC_REF_MISS.MEM_LCL_NO_SNOOP	0xC44	LLC	4.2.19.2
LLC_REF_MISS.MEM_LCL_SNOOP	0xC45	LLC	4.2.19.2
LLC_REF_MISS.MEM_LCL_SNOOP_FWD	0xC46	LLC	4.2.19.2
LLC_REF_MISS.MEM_RMT_ANY	0xC43	LLC	4.2.19.2
LLC_REF_MISS.MEM_RMT_NO_SNOOP	0xC47	LLC	4.2.19.2
LLC_REF_MISS.MEM_RMT_SNOOP	0xC48	LLC	4.2.19.2
LLC_REF_MISS.MEM_RMT_SNOOP_FWD	0xC49	LLC	4.2.19.2
LLC_REF_SYS_ANY	0xC40	LLC	4.2.19.5
LLC_REF_UNKNOWN	0xC4A	LLC	4.2.19.6
MLD_BWMODE_CYC	0xCD6	MLD	4.2.13.1

Table 4-2. All Performance Monitors Ordered by Name (Sheet 11 of 20)

Symbol Name	Event Code	Area	Section
MLD_BYPASS	0xCD4	MLD	4.2.13.2
MLD_BYPASS_ATTEMPT	0xCD3	MLD	4.2.13.3
MLD_CYC_STALL.ANY	0xCD8	MLD	4.2.13.4
MLD_CYC_STALL.CRIT_BYP	0xCDD	MLD	4.2.13.4
MLD_CYC_STALL.FILL_W	0xCDC	MLD	4.2.13.4
MLD_CYC_STALL.HPW	0xCDF	MLD	4.2.13.4
MLD_CYC_STALL.RAW	0xCDA	MLD	4.2.13.4
MLD_CYC_STALL.RW_BANK	0xCD9	MLD	4.2.13.4
MLD_CYC_STALL.SEMAPHORE	0xCDB	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_CRIT_BYP	0xCE6	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_OVERSUB	0xCE5	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_RAW	0xCE4	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_RR_BANK	0xCE2	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_RW_BANK	0xCE3	MLD	4.2.13.4
MLD_CYC_STALL.SPLIT_WW_BANK	0xCE1	MLD	4.2.13.4
MLD_CYC_STALL.TAG_ERR	0xCE0	MLD	4.2.13.4
MLD_CYC_STALL.WB_FIFO	0xCDE	MLD	4.2.13.4
MLD_FAB_COUNT.LSB	0xCF7	MLD	4.2.13.5
MLD_FAB_COUNT.MSB	0xCF8	MLD	4.2.13.5
MLD_FAB_OVERFLOW	0xCD1	MLD	4.2.13.6
MLD_FILL_MESI_STATE_BUDDY.ANY	0xCC4	MLD	4.2.13.7
MLD_FILL_MESI_STATE_BUDDY.E	0xCC5	MLD	4.2.13.7
MLD_FILL_MESI_STATE_BUDDY.I	0xCC7	MLD	4.2.13.7
MLD_FILL_MESI_STATE_BUDDY.S	0xCC6	MLD	4.2.13.7
MLD_FILL_MESI_STATE_PRIMARY.ANY	0xCBF	MLD	4.2.13.8
MLD_FILL_MESI_STATE_PRIMARY.E	0xCC1	MLD	4.2.13.8
MLD_FILL_MESI_STATE_PRIMARY.I	0xCC3	MLD	4.2.13.8
MLD_FILL_MESI_STATE_PRIMARY.M	0xCC0	MLD	4.2.13.8
MLD_FILL_MESI_STATE_PRIMARY.S	0xCC2	MLD	4.2.13.8
MLD_HINT_DEFER	0xCD0	MLD	4.2.13.9
MLD_HINT_NOALLOC	0xCCB	MLD	4.2.13.10
MLD_HINT_NO_BUDDY	0xCCD	MLD	4.2.13.11
MLD_HINT_NO_MULTI_HWPREF	0xCCE	MLD	
MLD_HINT_NRU	0xCCC	MLD	4.2.13.13
MLD_HINT_PREF_DROP	0xCCF	MLD	4.2.13.12
MLD_ISSUE_SRC.ANY	0xCA7	MLD	4.2.13.14
MLD_ISSUE_SRC.BYPASS	0xCA8	MLD	4.2.13.14
MLD_ISSUE_SRC.FAB	0xCAB	MLD	4.2.13.14
MLD_ISSUE_SRC.OZQ	0xCA9	MLD	4.2.13.14
MLD_ISSUE_SRC.SMQ	0xCAA	MLD	4.2.13.14
MLD_ISSUE_SRC.SNOOP	0xCAC	MLD	4.2.13.14

Table 4-2. All Performance Monitors Ordered by Name (Sheet 12 of 20)

Symbol Name	Event Code	Area	Section
MLD_LOAD.ANY	0xCB4	MLD	4.2.13.15
MLD_LOAD.HIT	0xCB5	MLD	4.2.13.15
MLD_LOAD.MISS	0xCB6	MLD	4.2.13.15
MLD_LOAD.PRIMARY	0xCB7	MLD	4.2.13.15
MLD_LOAD.SECONDARY	0xCB8	MLD	4.2.13.15
MLD_LOST_BW.ANY	0xCE7	MLD	4.2.13.16
MLD_LOST_BW.FAB_NOP	0xCEF	MLD	4.2.13.16
MLD_LOST_BW.NOP	0xCF4	MLD	4.2.13.16
MLD_LOST_BW.NOP_STALL	0xCF2	MLD	4.2.13.16
MLD_LOST_BW.OZQ_FAB_FULL	0xCEA	MLD	4.2.13.16
MLD_LOST_BW.OZQ_FAB_PREEMPT	0xCEC	MLD	4.2.13.16
MLD_LOST_BW.OZQ_NOP	0xCE8	MLD	4.2.13.16
MLD_LOST_BW.OZQ_NOP_ACQ	0xCE9	MLD	4.2.13.16
MLD_LOST_BW.OZQ_SMQ_FULL	0xCEB	MLD	4.2.13.16
MLD_LOST_BW.OZQ_SMQ_PREEMPT	0xCED	MLD	4.2.13.16
MLD_LOST_BW.OZQ_SNOOP_PREEMPT	0xCEE	MLD	4.2.13.16
MLD_LOST_BW.SMQ_NOP	0xCF0	MLD	4.2.13.16
MLD_LOST_BW.SPLIT_BUBBLE	0xCF1	MLD	4.2.13.16
MLD_LOST_BW.STALL	0xCF3	MLD	4.2.13.16
MLD_NOALLOC_CASTOUT	0xCCA	MLD	4.2.13.17
MLD_NOALLOC_FILL	0xCC9	MLD	4.2.13.18
MLD_OZDATA_COUNT.LSB	0xCFE	MLD	4.2.13.19
MLD_OZDATA_COUNT.MSB	0xCFF	MLD	4.2.13.19
MLD_OZQ_COUNT.LSB	0xCF5	MLD	4.2.13.20
MLD_OZQ_COUNT.MSB	0xCF6	MLD	4.2.13.20
MLD_OZQ_INSERT	0xCD2	MLD	4.2.13.21
MLD_OZQ_PREEMPTED	0xCD5	MLD	4.2.13.22
MLD_REF.ANY	0xCAD	MLD	4.2.13.23
MLD_REF.HIT	0xCAE	MLD	4.2.13.23
MLD_REF.MISS	0xCAF	MLD	4.2.13.23
MLD_REF.PRIMARY	0xCB0	MLD	4.2.13.23
MLD_REF.SECONDARY	0xCB1	MLD	4.2.13.23
MLD_REF.SECONDARY_DROP	0xCB2	MLD	4.2.13.23
MLD_REF.UC_WC_STORE	0xCB3	MLD	4.2.13.23
MLD_SMQ_COUNT.LSB	0xCF9	MLD	4.2.13.24
MLD_SMQ_COUNT.MSB	0xCFA	MLD	4.2.13.24
MLD_SMQ_PRIORITY	0xCD7	MLD	4.2.13.25
MLD_SMQ_REF.ANY	0xCB9	MLD	4.2.13.26
MLD_SMQ_REF.HIT	0xCBA	MLD	4.2.13.26
MLD_SMQ_REF.MISS	0xCBB	MLD	4.2.13.26
MLD_SMQ_REF.PRIMARY	0xCBC	MLD	4.2.13.26

Table 4-2. All Performance Monitors Ordered by Name (Sheet 13 of 20)

Symbol Name	Event Code	Area	Section
MLD_SMQ_REF.SECONDARY	0xCBD	MLD	4.2.13.26
MLD_SMQ_REF.SECONDARY_DROP	0xCBE	MLD	4.2.13.26
MLD_SNOOP_DEFER	0xCC8	MLD	4.2.13.27
MLD_WCB_CREDIT	0xCFD	MLD	4.2.13.28
MLD_WLB_COUNT.LSB	0xCFB	MLD	4.2.13.29
MLD_WLB_COUNT.MSB	0xCFC	MLD	4.2.13.29
MLITLB_HPW_ABORTS	0x870	MLITLB	4.2.11.1
MLITLB_MISS	0x86E	MLITLB	4.2.11.2
MLI_HIT_CONFLICT.ANY	0x84E	MLI	4.2.9.1
MLI_HIT_CONFLICT.DMND	0x84F	MLI	4.2.9.1
MLI_HIT_CONFLICT.PREF	0x850	MLI	4.2.9.1
MLI_READ_UC.ANY	0x847	MLI	4.2.9.3
MLI_READ_UC.DMND	0x848	MLI	4.2.9.3
MLI_READ_UC.PREF	0x849	MLI	4.2.9.3
MLI_READ.ANY_ANY	0x83C	MLI	4.2.9.2
MLI_READ.ANY_DMND	0x83D	MLI	4.2.9.2
MLI_READ.ANY_PREF	0x83E	MLI	4.2.9.2
MLI_READ.HIT_ANY	0x83F	MLI	4.2.9.2
MLI_READ.HIT_DMND_LRU	0x842	MLI	4.2.9.2
MLI_READ.HIT_DMND_NOLRU	0x840	MLI	4.2.9.2
MLI_READ.HIT_PREF_LRU	0x843	MLI	4.2.9.2
MLI_READ.HIT_PREF_NOLRU	0x841	MLI	4.2.9.2
MLI_READ.MISS_ANY	0x844	MLI	4.2.9.2
MLI_READ.MISS_DMND	0x845	MLI	4.2.9.2
MLI_READ.MISS_PREF	0x846	MLI	4.2.9.2
MLI_RECIRCULATE.ANY	0x84A	MLI	4.2.9.4
MLI_RECIRCULATE.DMND	0x84B	MLI	4.2.9.4
MLI_RECIRCULATE.PREF	0x84C	MLI	4.2.9.4
MLI_RETURN_LINE	0x860	FLI	4.2.8.14
MLI_SNOOP_HIT	0x852	MLI	4.2.9.5
MLI_SNOOP_INVAL_BLK_LOOKUP	0x84D	MLI	4.2.9.6
MLI_SPEC_ABORT	0x851	MLI	4.2.9.7
MT_BE_BGND_CYC_IN_STATE.HU	0x143	Multithreading	4.2.21.2
MT_BE_BGND_CYC_IN_STATE.HW	0x144	Multithreading	4.2.21.2
MT_BE_BGND_CYC_IN_STATE.LU	0x147	Multithreading	4.2.21.2
MT_BE_BGND_CYC_IN_STATE.LW	0x148	Multithreading	4.2.21.2
MT_BE_BGND_CYC_IN_STATE.NU	0x145	Multithreading	4.2.21.2
MT_BE_BGND_CYC_IN_STATE.NW	0x146	Multithreading	4.2.21.2
MT_BE_FAIR_STATE.GREEN	0x149	Multithreading	4.2.21.3
MT_BE_FAIR_STATE.ORANGE	0x14B	Multithreading	4.2.21.3
MT_BE_FAIR_STATE.RED	0x14C	Multithreading	4.2.21.3

Table 4-2. All Performance Monitors Ordered by Name (Sheet 14 of 20)

Symbol Name	Event Code	Area	Section
MT_BE_FAIR_STATE.YELLOW	0x14A	Multithreading	4.2.21.3
MT_BE_FAIR_TRANSITION.GRN0	0x150	Multithreading	4.2.21.4
MT_BE_FAIR_TRANSITION.GRN2YLW	0x14D	Multithreading	4.2.21.4
MT_BE_FAIR_TRANSITION.ORN2RED	0x14F	Multithreading	4.2.21.4
MT_BE_FAIR_TRANSITION.YLW2ORN	0x14E	Multithreading	4.2.21.4
MT_BE_THRSW_ACTUAL_IN.ALAT_INVAL	0x136	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.HPWINS	0x134	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.IBQ_NOTEMPTY	0x135	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.LP_EXIT	0x137	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.MLDRTN	0x133	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.TIMEOUT	0x132	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_IN.FAIR	0x138	Multithreading	4.2.21.5
MT_BE_THRSW_ACTUAL_OUT.ANY	0x12A	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.ATPAUSE	0x12E	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.HPW_MISS	0x12C	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.IBQ_EMPTY	0x12D	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.INJ_DBG	0x131	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.LP_ENTER	0x12F	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.MLD_USE	0x12B	Multithreading	4.2.21.6
MT_BE_THRSW_ACTUAL_OUT.RFIX	0x130	Multithreading	4.2.21.6
MT_BE_THRSW_DISABLE.EXPL	0x13A	Multithreading	4.2.21.7
MT_BE_THRSW_DISABLE.IMPL	0x13B	Multithreading	4.2.21.7
MT_BE_THRSW_DROP	0x139	Multithreading	4.2.21.8
MT_BE_THRSW_HOLD	0x13C	Multithreading	4.2.21.9
MT_BE_THRSW_STALL.ANY	0x13D	Multithreading	4.2.21.10
MT_BE_THRSW_STALL.CRAB	0x141	Multithreading	4.2.21.10
MT_BE_THRSW_STALL.FLD	0x142	Multithreading	4.2.21.10
MT_BE_THRSW_STALL.PIPE	0x13F	Multithreading	4.2.21.10
MT_BE_THRSW_STALL.RSE	0x140	Multithreading	4.2.21.10
MT_BE_THRSW_STALL.SWITCH	0x13E	Multithreading	4.2.21.10
MT_FE_BE_IN_SAME_THREAD	0x874	Multithreading	4.2.21.11
MT_FE_BGND_CYC_IN_STATE.HIGH	0x8A0	Multithreading	4.2.21.12
MT_FE_BGND_CYC_IN_STATE.LOW	0x89E	Multithreading	4.2.21.12
MT_FE_BGND_CYC_IN_STATE.NOMINAL	0x89F	Multithreading	4.2.21.12
MT_FE_THRSW_ACTUAL_IN.BRQ_NON_BLK	0x881	Multithreading	4.2.21.13
MT_FE_THRSW_ACTUAL_IN.IBQ_EMPTY	0x87A	Multithreading	4.2.21.13
MT_FE_THRSW_ACTUAL_IN.IBQ_NOTFULL	0x879	Multithreading	4.2.21.13
MT_FE_THRSW_ACTUAL_IN.MLI_UCRTN	0x87F	Multithreading	4.2.21.13
MT_FE_THRSW_ACTUAL_IN.MLI_WBRTN	0x87D	Multithreading	4.2.21.13
MT_FE_THRSW_ACTUAL_OUT.ANY	0x876	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.BE_FOLLOW	0x883	Multithreading	4.2.21.14

Table 4-2. All Performance Monitors Ordered by Name (Sheet 15 of 20)

Symbol Name	Event Code	Area	Section
MT_FE_THRSW_ACTUAL_OUT.BRQ_BLK	0x880	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.HINT_BSWT	0x882	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.IBQ_FULL	0x878	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.IBQ_NOTEMPTY	0x87B	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.LOCKED	0x884	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.MLI_UCMISS	0x87E	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.MLI_WBMISS	0x87C	Multithreading	4.2.21.14
MT_FE_THRSW_ACTUAL_OUT.TIMEOUT	0x877	Multithreading	4.2.21.14
MT_FE_THRSW_MISS_IN.ANY	0x886	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_IN.BRQ_NON_BLK	0x891	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_IN.IBQ_EMPTY	0x88A	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_IN.IBQ_NOTFULL	0x889	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_IN.MLI_UCRTN	0x88F	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_IN.MLI_WBRTN	0x88D	Multithreading	4.2.21.15
MT_FE_THRSW_MISS_OUT.ANY	0x885	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.BE_FOLLOW	0x893	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.BRQ_BLK	0x890	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.HINT_BSWT	0x892	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.IBQ_FULL	0x888	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.IBQ_NOTEMPTY	0x88B	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.LOCKED	0x894	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.MLI_UCMISS	0x88E	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.MLI_WBMISS	0x88C	Multithreading	4.2.21.16
MT_FE_THRSW_MISS_OUT.TIMEOUT	0x887	Multithreading	4.2.21.16
MT_FE_THRSW_STALL.ANY	0x895	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.BLK_ANY	0x899	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.BLK_FW_PROG	0x89C	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.BLK_IN_PROG	0x89B	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.BLK_IPC_MISS	0x89A	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.BLK_THRESH	0x89D	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.EXPL	0x897	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.IMPL	0x898	Multithreading	4.2.21.17
MT_FE_THRSW_STALL.MTLCK	0x896	Multithreading	4.2.21.17
M_ASYNC_OP_ISSUE.ANY	0x10A	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.CRAB_RET	0x110	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.DTLBTRNSFR_TLBINSERT	0x114	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.FLUSH_ST_INVAL	0x115	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.HPW_FAULT	0x11C	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.HPW_LOAD	0x10F	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.HPW_TLBINSERT	0x11B	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.HW_PREF	0x111	Dispersal	4.2.2.4

Table 4-2. All Performance Monitors Ordered by Name (Sheet 16 of 20)

Symbol Name	Event Code	Area	Section
M_ASYNC_OP_ISSUE.ITC_D	0x11D	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.ITR_D	0x11E	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.MOVTOPKR	0x121	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.MOVTORR	0x120	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.NONE	0x10B	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PAMERR_PAPURGE	0x116	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PAPURGE	0x112	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PTC_E	0x123	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PTC_G	0x124	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PTC_GA	0x125	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PTC_L	0x122	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.PTR_D	0x11F	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.RIDVPN_PURGE	0x118	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.RSE_LOAD	0x129	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.RSE_STORE	0x128	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.SHOOTDOWN_G	0x126	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.SHOOTDOWN_GA	0x127	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.SNOOP	0x10D	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.SNOOP_PALKUP	0x10C	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.SNOOP_S	0x10E	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.TLB_TSWITCH	0x119	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.TSWITCH	0x11A	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.VAMERR_VAPURGE	0x113	Dispersal	4.2.2.4
M_ASYNC_OP_ISSUE.VRNRIDVPN_PURGE	0x117	Dispersal	4.2.2.4
PREF_DROP.DTLB_MISS	0x0DA	DPF	4.2.16.9
PREF_DROP.FLDTLB_MISS	0x0D9	DPF	4.2.16.9
PREF_DROP.FLD_HIT	0x0DB	DPF	4.2.16.9
PREF_DROP.FLD_SECONDARY_MISS	0x0DC	DPF	4.2.16.9
RETIRED_INST_BR	0x820	Execution	4.2.3.18
RETIRED_INST_FC	0xC30	Execution	4.2.3.19
RETIRED_INST_FCI	0xC31	Execution	4.2.3.20
RETIRED_INST_FP	0x00B	Execution	4.2.3.21
RETIRED_INST_LD_FP	0x0DF	Execution	4.2.3.22
RETIRED_INST_LD_INT	0x0DE	Execution	4.2.3.23
RETIRED_INST_M.ACQ	0x0F4	Execution	4.2.3.24
RETIRED_INST_M.ANY	0x00D	Execution	4.2.3.24
RETIRED_INST_M.MOVTOBSPST	0x0CA	Execution	4.2.3.24
RETIRED_INST_M.MOVTODAHR	0x00E	Execution	4.2.3.24
RETIRED_INST_M.REL	0x0F5	Execution	4.2.3.24
RETIRED_INST_NOP	0x00A	Execution	4.2.3.25
RETIRED_INST_PTCG	0xC34	Execution	4.2.3.26

Table 4-2. All Performance Monitors Ordered by Name (Sheet 17 of 20)

Symbol Name	Event Code	Area	Section
RETIRED_INST_RSE	0x157	Execution	4.2.3.27
RETIRED_INST_SEMAPHORE	0x0E5	Execution	4.2.3.28
RETIRED_INST_ST_FP	0x0E2	Execution	4.2.3.29
RETIRED_INST_ST_INT	0x0E1	Execution	4.2.3.30
RETIRED_INST_TAGGED.IAM0_OPM0	0x005	Execution	4.2.3.31
RETIRED_INST_TAGGED.IAM1_OPM1	0x006	Execution	4.2.3.31
RETIRED_INST_TAGGED.IAM2_OPM0	0x007	Execution	4.2.3.31
RETIRED_INST_TAGGED.IAM3_OPM1	0x008	Execution	4.2.3.31
RETIRED_PREDICATE_SQUASHED	0x009	Execution	4.2.3.32
RIL_ARB_PRI_LOST.AD	0xC60	RIL	4.2.17.1
RIL_ARB_PRI_LOST.AD_FWD_PROG	0xC61	RIL	4.2.17.1
RIL_ARB_PRI_LOST.BL	0xC62	RIL	4.2.17.1
RIL_ARB_PRI_LOST.BL_FWD_PROG	0xC63	RIL	4.2.17.1
RIL_BL_WRITE.ANY	0xC5C	RIL	4.2.17.2
RIL_BL_WRITE.SLB	0xC5F	RIL	4.2.17.2
RIL_BL_WRITE.WLB	0xC5D	RIL	4.2.17.2
RIL_BL_WRITE.WLB_BOGUS	0xC5E	RIL	4.2.17.2
RIL_CBQ_EVICT.FULL	0xC53	RIL	4.2.17.3
RIL_CBQ_EVICT.WCB_FLUSH	0xC52	RIL	4.2.17.3
RIL_CRDT_MLD_FDB_FULL	0xC6E	RIL	4.2.17.4
RIL_CRDT_MLD_FDB_FULL_BLK	0xC6F	RIL	4.2.17.5
RIL_CRDT_PRI_BLK.AD_ALL	0xC64	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.AD_CBQ	0xC68	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.AD_DRQ	0xC66	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.AD_FRQ	0xC65	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.AD_WRQ	0xC67	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.AK_ALL	0xC69	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.BL_ALL	0xC6A	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.BL_CBQ	0xC6D	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.BL_SNQ	0xC6B	RIL	4.2.17.6
RIL_CRDT_PRI_BLK.BL_WRQ	0xC6C	RIL	4.2.17.6
RIL_CRDT_SNQ_BLK.ANY	0xC70	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.ANY_Q_FULL	0xC7D	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.DFRQ	0xC79	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.HALT	0xC71	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLD_FULL	0xC77	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLD_FWD_PROG	0xC74	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLI_FULL	0xC76	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLI_FWD_PROG	0xC73	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLI_OR_MLD_FULL	0xC78	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.MLI_OR_MLD_FWD_PROG	0xC75	RIL	4.2.17.7

Table 4-2. All Performance Monitors Ordered by Name (Sheet 18 of 20)

Symbol Name	Event Code	Area	Section
RIL_CRDT_SNQ_BLK.RSPQ	0xC7A	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.SLB_DQ	0xC7B	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.SRLZ	0xC72	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.USEMANY_ANY	0xC96	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.USEMANY_BYP	0xC97	RIL	4.2.17.7
RIL_CRDT_SNQ_BLK.WLB_DQ	0xC7C	RIL	4.2.17.7
RIL_DATA_RETURN.EARLY_FILL_EM	0xC58	RIL	4.2.17.8
RIL_DATA_RETURN.EARLY_FILL_S	0xC59	RIL	4.2.17.8
RIL_DATA_RETURN.MLD_ANY	0xC56	RIL	4.2.17.8
RIL_DATA_RETURN.MLD_CRIT	0xC57	RIL	4.2.17.8
RIL_DATA_RETURN.PRI_ANY	0xC54	RIL	4.2.17.8
RIL_DATA_RETURN.PRI_MLD	0xC55	RIL	4.2.17.8
RIL_DRQ_PACE_BUBBLE	0xC8F	RIL	4.2.17.10
RIL_DRQ_VALID.LSB	0xC8A	RIL	4.2.17.11
RIL_DRQ_VALID.MSB	0xC89	RIL	4.2.17.11
RIL_DRQ.EMPTY	0xC80	RIL	4.2.17.9
RIL_DRQ.LIMIT_HIT	0xC81	RIL	4.2.17.9
RIL_FRQ_VALID.LSB	0xC88	RIL	4.2.17.13
RIL_FRQ_VALID.MSB	0xC87	RIL	4.2.17.13
RIL_FRQ.EMPTY	0xC7E	RIL	4.2.17.12
RIL_FRQ.LIMIT_HIT	0xC7F	RIL	4.2.17.12
RIL_INTERRUPT	0xC51	RIL	4.2.17.14
RIL_PRI_THROTTLE_ASSERTED	0xC90	RIL	4.2.17.15
RIL_PRI_THROTTLE_RECOV	0xC91	RIL	4.2.17.16
RIL_REQ_HINT_NRU	0xC3B	RIL	4.2.17.18
RIL_REQ_OTHER.CC	0xC32	RIL	4.2.17.19
RIL_REQ_OTHER.DRQ_ANY	0xC33	RIL	4.2.17.19
RIL_REQ_OTHER.FC	0xC30	RIL	4.2.17.19
RIL_REQ_OTHER.FCI	0xC31	RIL	4.2.17.19
RIL_REQ_OTHER.LRUHINT_ANY	0xC37	RIL	4.2.17.19
RIL_REQ_OTHER.LRUHINT_FROM_MLD	0xC36	RIL	4.2.17.19
RIL_REQ_OTHER.LRUHINT_MISS_ANY	0xC39	RIL	4.2.17.19
RIL_REQ_OTHER.LRUHINT_MISS_MLD	0xC3A	RIL	4.2.17.19
RIL_REQ_OTHER.LRUHINT_MLD	0xC38	RIL	4.2.17.19
RIL_REQ_OTHER.PTCG	0xC34	RIL	4.2.17.19
RIL_REQ_OTHER.PTCG_PEND	0xC35	RIL	4.2.17.19
RIL_REQ_OTHER.WRQ_FC_FCI	0xC2B	RIL	4.2.17.19
RIL_REQ_OTHER.WRQ_SKIP_LRUHINT	0xC2D	RIL	4.2.17.19
RIL_REQ_OTHER.WRTBCK_MLD_EVICT	0xC2E	RIL	4.2.17.19
RIL_REQ_OTHER.WRTBCK_MLD_FC	0xC2F	RIL	4.2.17.19
RIL_REQ_OTHER.WRTBCK_WRQ	0xC2A	RIL	4.2.17.19

Table 4-2. All Performance Monitors Ordered by Name (Sheet 19 of 20)

Symbol Name	Event Code	Area	Section
RIL_REQ_OTHER.WRTBCK_WRQ_SKIP	0xC2C	RIL	4.2.17.19
RIL_REQ_REF_DATA.ANY	0xC18	RIL	4.2.17.21
RIL_REQ_REF_DATA.DRQ_ANY	0xC28	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_ANY	0xC20	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_READ_ANY	0xC21	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_READ_UC	0xC22	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_WRITE_ANY	0xC23	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_WRITE_UC	0xC26	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_WRITE_WC_ANY	0xC24	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_WRITE_WC_FULL	0xC25	RIL	4.2.17.21
RIL_REQ_REF_DATA.NC_WRITE_WC_MLD	0xC27	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_ANY	0xC19	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_CRD	0xC1C	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_DRD	0xC1D	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_MLD_ANY	0xC1A	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_MLD_BUDDY	0xC1B	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_RFO	0xC1E	RIL	4.2.17.21
RIL_REQ_REF_DATA.WB_SELF_SNOOP	0xC1F	RIL	4.2.17.21
RIL_REQ_REF_DATA.WRQ_ANY	0xC29	RIL	4.2.17.21
RIL_REQ_REF_INST.ANY	0xC14	RIL	4.2.17.22
RIL_REQ_REF_INST.NC	0xC15	RIL	4.2.17.22
RIL_REQ_REF_INST.WB_ANY	0xC16	RIL	4.2.17.22
RIL_REQ_REF_INST.WB_DMND	0xC17	RIL	4.2.17.22
RIL_REQ_REF.ANY	0xC13	RIL	4.2.17.20
RIL_REQ.ANY	0xC12	RIL	4.2.17.17
RIL_RESP.GO	0xC5A	RIL	4.2.17.23
RIL_RESP.WRITEPULL	0xC5B	RIL	4.2.17.23
RIL_RRQ.LIMIT_HIT	0xC84	RIL	4.2.17.24
RIL_SEB.BGF_QUIESCE_ACTIVE	0xC94	RIL	4.2.17.25
RIL_SEB.LDST_QUIESCE_PEND	0xC93	RIL	4.2.17.25
RIL_SEB.PTC_QUIESCE_PEND	0xC92	RIL	4.2.17.25
RIL_SHOOTDOWN	0xC4F	RIL	4.2.17.26
RIL_SHOOTDOWN_PEND_CYC	0xC50	RIL	4.2.17.27
RIL_SNOOP_REQ.ANY	0xC01	RIL	4.2.17.28
RIL_SNOOP_REQ.CODE_ANY	0xC02	RIL	4.2.17.28
RIL_SNOOP_REQ.CODE_SELF	0xC03	RIL	4.2.17.28
RIL_SNOOP_REQ.CODE_SIBLING	0xC04	RIL	4.2.17.28
RIL_SNOOP_REQ.DATA_ANY	0xC05	RIL	4.2.17.28
RIL_SNOOP_REQ.DATA_SELF	0xC06	RIL	4.2.17.28
RIL_SNOOP_REQ.DATA_SIBLING	0xC07	RIL	4.2.17.28
RIL_SNOOP_REQ.INVAL_ANY	0xC08	RIL	4.2.17.28

Table 4-2. All Performance Monitors Ordered by Name (Sheet 20 of 20)

Symbol Name	Event Code	Area	Section
RIL_SNOOP_REQ.INVAL_LLC_EVICT	0xC0B	RIL	4.2.17.28
RIL_SNOOP_REQ.INVAL_SELF	0xC09	RIL	4.2.17.28
RIL_SNOOP_REQ.INVAL_SIBLING	0xC0A	RIL	4.2.17.28
RIL_SNOOP_RESP.MLD_DEFER	0xC11	RIL	4.2.17.29
RIL_SNOOP_RESP.MLD_HIT_E	0xC0F	RIL	4.2.17.29
RIL_SNOOP_RESP.MLD_HIT_M	0xC10	RIL	4.2.17.29
RIL_SNOOP_RESP.MLD_HIT_S	0xC0E	RIL	4.2.17.29
RIL_SNOOP_RESP.MLD_MISS	0xC0C	RIL	4.2.17.29
RIL_SNOOP_RESP.WRQ_HIT_M	0xC0D	RIL	4.2.17.29
RIL_SNQ_VALID.LSB	0xC8E	RIL	4.2.17.31
RIL_SNQ_VALID.MSB	0xC8D	RIL	4.2.17.31
RIL_SNQ.EMPTY	0xC85	RIL	4.2.17.30
RIL_SNQ.LIMIT_HIT	0xC86	RIL	4.2.17.30
RIL_WRQ_VALID.LSB	0xC8C	RIL	4.2.17.33
RIL_WRQ_VALID.MSB	0xC8B	RIL	4.2.17.33
RIL_WRQ.EMPTY	0xC82	RIL	4.2.17.32
RIL_WRQ.LIMIT_HIT	0xC83	RIL	4.2.17.32
RSE_CURRENT_REG.LSB	0x154	RSE	4.2.18.1
RSE_CURRENT_REG.MSB	0x153	RSE	4.2.18.1
RSE_DIRTY_REG.LSB	0x156	RSE	4.2.18.2
RSE_DIRTY_REG.MSB	0x155	RSE	4.2.18.2
RSE_REF_RETIRED.ANY	0x158	RSE	4.2.18.3
RSE_REF_RETIRED.LOAD	0x159	RSE	4.2.18.3
RSE_REF_RETIRED.STORE	0x15A	RSE	4.2.18.3
SERIALIZATION_EVENT	0x021	System	4.2.20.3
UNCORE_FREEZE	0xC9B	System	4.2.20.4

4.2 Performance Monitor Events by Area

4.2.1 Basic Events

This section enumerates Basic performance monitoring events.

4.2.1.1 CPU_OP_CYCLES

Description	CPU back-end pipeline execution cycle count
Max Inc/Cyc	1
MT Capture Type	C
Subevents:	
(ANY)	0x002
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	-/-/-/-

Definition	CPU cycle count
HALTED	0x003
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Halted state cycle count. Cycles spent in low-power halted state. Note: In order to count correctly, this event requires PMC.ch to be set as well. It will count halted cycles spent both in the foreground as well as background thread. To count foreground halted state cycles: CPU_OP_CYCLES.me(ch=1) - CPU_OP_CYCLES.me(ch=0)
TAGGED	0x004
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
Definition	Tagged CPU cycle count. To use this monitor, program tag channel 0 to the (Instruction Address Range, Opcode) regions of interest, channel 1 to unconstrained tagging. Counting will be enabled whenever a valid channel 0 tag is seen and continue until a valid channel 1 tag without a channel 0 tag is seen.

4.2.1.2 CPU_REF_CYCLES

Description	Back-end execution reference cycle count
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x001
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	CPU reference cycle count. This monitor counts the number of ITC timebase reference cycles spent in the back-end.

4.2.1.3 FE_OP_CYCLES

Description	CPU front-end pipeline execution cycle count
Max Inc/Cyc	1
MT Capture Type	F
Event Code	0x875
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	This event counts the cycles the front end spends in its thread, it's the front-end's version of CPU_OP_CYCLES. Due to the processor's separately threaded front-end, this monitor must be used to normalize front-end thread specific events.
NOTE	-

4.2.1.4 IA64_INST_RETIRED

Description	Instructions retired
Max Inc/Cyc	12
MT Capture Type	A
Event Code	0x005
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	I/O/-/-

IP-EAR	L = 5, P = 1
Definition	Retired instructions count
NOTE	This is an alias for IA64TAGGED_INST_RETIRED.IAM0OPM0

4.2.2 Dispersal Events

This section enumerates Dispersal performance monitoring events.

4.2.2.1 IBL_ISSUE_LOST_BW

The ISSUE_LOST and ISSUE_STOP events work together. See the issue_stop explanations for more information about the issue_lost event. Both of these events use the Channel 0 address tag from IBR address matching.

Description	Issue syllables lost estimate
Max Inc/Cyc	11
MT Capture Type	A
Definition	Counts the number of syllables to the next stop bit that were not issued in this issue cycle
NOTE	This monitor counts syllables as bundle slots and not instructions, i.e., for MLX, the LX counts as two
Subevents:	
9PLUS3	0x086
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to bundle restrictions
ANY	0x07a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost for any of the possible reasons
ASYM_I	0x07d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to I pipe asymmetry
ASYM_M	0x07e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to M pipe asymmetry
DROOP	0x07c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to 3rd droop power management

FLD_DMND_M0	0x07f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to FLD request for M0
FLD_DMND_M1	0x080
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to FLD request for M1
OVRSUB_A	0x081
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to A pipe oversubscription
OVRSUB_F	0x082
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to F pipe oversubscription
OVRSUB_I	0x083
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to I pipe oversubscription
OVRSUB_M	0x084
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to M pipe oversubscription
POWER	0x07b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to TDP power management
STRUCT	0x085
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue syllables lost due to structural limitation

4.2.2.2 IBL_ISSUE

Description	Syllables dispersed at IBD
Max Inc/Cyc	12
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
ANY	0x068
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Syllables dispersed at IBD
M_PIPE	0x069
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	M pipe syllables dispersed at IBD

4.2.2.3 IBL_ISSUE_STOP

Description	The reason for the size of the instruction issue group (in the backend pipeline)
Max Inc/Cyc	1
MT Capture Type	A
Definition	Within a 4 bundle wide issue window, what caused the first effective stop bit in an issue cycle.
NOTE	Whenever an issue stall is required, or we have no instructions to issue, this event signals NONE. Only one of these reasons will be signalled per cycle. They are calculated in a priority order such that effects of the compiler scheduled program instruction stream are reported first, before asynchronous microarchitecture events. So, explicit stop bit and over-subscription are reported before FLD demand or power throttling.
Subevents:	
9PLUS3	0x079
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to 4th bundle limitations; we encountered an M, I, A, or F in the 4th bundle
ASYM_I	0x06f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to I pipe asymmetry; an instruction was ready to issue on I1, but needed to be on I0
ASYM_M	0x070
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2

Definition	Issue stop due to M pipe asymmetry; an instruction was ready to issue on M1, but needed to be on M0
BUNDLE	0x078
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to lack of bundles; we have less than 4 bundles in the window, issued all the bundles that we had, and didn't see some other reason to stop issue.
DROOP	0x06e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to 3rd droop power management
EXPLICIT	0x06c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to explicit stop bit or the end of a non-brp, non-nop.b branch bundle.
FLD_DMND_M0	0x071
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to FLD demand for M0; there was an instruction that was ready to issue on M0, but an asynchronous cache operation required M0; so, we stopped issue at that M0 instruction
FLD_DMND_M1	0x072
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to FLD demand for M1; there was an instruction that was ready to issue on M1, but an asynchronous cache operation required M1; so, we stopped issue at that M1 instruction
NONE	0x06a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to issue stall or instruction buffer empty, that is, we did not issue, because we couldn't
OVRSUB_A	0x075
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to A pipe oversubscription; we reached the third A instruction in the window
OVRSUB_F	0x076
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to F pipe oversubscription, we reached the third F instruction in the window

OVRSUB_I	0x074
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to I pipe oversubscription; we reached the third I instruction in the window
OVRSUB_M	0x073
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to M pipe oversubscription; we reached the third M instruction in the window
POWER	0x06d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to TDP power management
REPLAY	0x06b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to a replay event; after a replay we always re-issue as we did the first time.
STRUCT	0x077
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/-/-/-
IP-EAR	L = 5, P = 2
Definition	Issue stop due to a structural limitation or power throttling of instruction types; the limitation includes the end of the 4 bundle wide issue group.

4.2.2.4 M_ASYNC_OP_ISSUE

Description	FLD asynchronous operation
Max Inc/Cyc	2
MT Capture Type	A
Definition	Asynchronous memory operations injected into the backend execution pipeline
NOTE	WARNING: Any kind of PMU event filtering (instruction address, opcode matching, data address, data reference type) can cause these events to be assigned to the wrong thread.
Subevents:	
ANY	0x10a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD asynchronous operation
CRAB_RET	0x110
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation CRAB return
DTLBTRNSFR_TLBINSERT	0x114

Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation DTB transfer TLB insert
FLUSH_ST_INVAL	0x115
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation flushed store invalidate
HPW_FAULT	0x11c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation HPW fault
HPW_LOAD	0x10f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation HPW load
HPW_TLBINSERT	0x11b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation HPW TLB insert
HW_PREF	0x111
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation HW prefetch
ITC_D	0x11d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation ITC.D
ITR_D	0x11e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation ITR.D
MOVTOPKR	0x121
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation move to PKR
MOVTORR	0x120
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation move to RR
NONE	0x10b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation none issued
PAMERR_PAPURGE	0x116

Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation FLDTLBPAM error pa purge
PAPURGE	0x112
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PA purge
PTC_E	0x123
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PTC.E
PTC_G	0x124
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PTC.G
PTC_GA	0x125
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PTC.GA
PTC_L	0x122
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PTC.L
PTR_D	0x11f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation PTR.D
RIDVPN_PURGE	0x118
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation RID VPN Purge
RSE_LOAD	0x129
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation RSE load
RSE_STORE	0x128
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation RSE store
SHOOTDOWN_G	0x126
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation shootdown G
SHOOTDOWN_GA	0x127

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation shootdown GA
SNOOP	0x10d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation snoop
SNOOP_PALKUP	0x10c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation snoop PA lookup
SNOOP_S	0x10e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation snoop shared
TLB_TSWITCH	0x119
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation TLB thread switch
TSWITCH	0x11a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation thread switch
VAMERR_VAPURGE	0x113
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation VA purge
VRNRIDVPN_PURGE	0x117
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLD asynchronous operation VRN RID VPN purge

4.2.3 Execution Events

This section enumerates Execution performance monitoring events.

4.2.3.1 ALAT_ENTRY_REPLACED

Description	ALAT entry replaced
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x152
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	I/O/D/-
IP-EAR	L = 6, P = 1
Definition	An advanced load which was not already in the ALAT replaced a different ALAT entry because the ALAT was full. Since M0 ALAT instructions typically use ALAT locations 0 to 15, and M1 16 to 31 this signal may fire if only one half of the ALAT is full. Both the M0 and M1 signals may be asserted on the same clock.

4.2.3.2 ALAT_STORE_HIT

Description	ALAT store hit
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x151
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/-
Definition	This is a per-port and per-thread signal (i.e. 4 of them) which is asserted when a store on M0 or M1 clears an ALAT entry on thread 0 or thread 1. Remember that a store on either thread can clear ALAT entries from the other thread.
NOTE	For counting this event the PMU OR's together the M0 and M1 bits of this signal together to produce a 2 bit per-thread signal.

4.2.3.3 CSPEC_CHKS

Description	Retired CHK.S instructions
Max Inc/Cyc	6
MT Capture Type	A
Event Code	0x014
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	Count of 'chk.s.[mi] *r2*, *target25*' and 'chk.s *f2*, *target25*' instructions that are predicated-on and retire (inclusive of resteering chk.s).
NOTE	-

4.2.3.4 CSPEC_CHKS_FAIL

Description	Failed CHK.S instructions
Max Inc/Cyc	1
MT Capture Type	A
Definition	Count of resteering 'chk.s.[mi] *r2*, *target25*' and 'chk.s *f2*, *target25*' instructions (implies predicated-on).
NOTE	-
Subevents:	
ANY	0x015
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1

Definition	Failed integer or floating point CHK.S instructions
FP	0x017
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	Failed floating point CHK.S instructions
INT	0x016
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	Failed integer CHK.S instructions

4.2.3.5 CSPEC_LOAD

Description	ld.s inst commits
Max Inc/Cyc	2
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
ANY	0xca1
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 1
Definition	ld.s inst commits
NAT	0xca2
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 1
Definition	ld.s inst commits, fails (i.e. NAT bit is set)

4.2.3.6 DSPEC_CHKA_LDC

Description	ALAT chka ldc
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x00f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	This is a per Mport event which is asserted when a ld.c.* or chk.a (integer or floating point) instruction retires.

4.2.3.7 DSPEC_CHKA_LDC_FAIL

Description	ALAT failed chka ldc
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
Subevents:	
ANY	0x011
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 5, P = 1
Definition	Any ALAT failed chka ldc
FP	0x013
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 5, P = 1
Definition	ALAT failed floating point chka ldc
INT	0x012
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 5, P = 1
Definition	ALAT failed integer chka ldc

4.2.3.8 DSPEC_LDC

Description	ld.c hitting the ALAT
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x010
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	This is a per Mport event which is asserted when a ld.c.* instruction hits in the ALAT.

Correct speculation event ratios depend on speculation resteers being enabled, faulting mis-speculations are not counted and will distort the computation.

4.2.3.9 EAR_EVENT_DATA

Description	Data cache EAR capture
Max Inc/Cyc	2
MT Capture Type	F
Event Code	0x019
Counter Affinity	0xaaaa0

IAR/OPC/DAR/DREF	I/O/D/R
Definition	This monitor counts the number of captures by the data cache EAR
NOTE	-

4.2.3.10 EAR_EVENT_ETB

Description	ETB-EAR capture
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x018
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	This monitor counts the number of captures by the ETB or IP-EAR respectively
NOTE	When in ETB mode, exception sources are not counted, only branch source records

4.2.3.11 EAR_EVENT_INST

Description	FLI EAR event captured
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x871
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	This monitor counts the number of captures by the instruction cache EAR
NOTE	-

4.2.3.12 FP_DENORMAL

Description	FPU DENORM causing REPLAY
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x856
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 1
Definition	This counts 1 for every op which has at least 1 denormal input and causes a normalization replay to occur.
NOTE	FPU will only do denormal replay if f2 is not zero which includes Fmult and Fnorm.

4.2.3.13 FP_FALSE_SIR

Description	FPU FALSE SIR
Max Inc/Cyc	1
MT Capture Type	A

Event Code	0x854
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 1
Definition	This counts 1 for every SIR replay which does not result in a fault after replay.

4.2.3.14 FP_FCHKF_FAIL

Description	FPU failed FCHKF
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x853
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 1
Definition	This counts 1 for every fchkf which fails causing a branch or fault to occur.

4.2.3.15 FP_FLOP

Description	Floating point weighed flop count
Max Inc/Cyc	4
MT Capture Type	A
Event Code	0x00c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 0
Definition	This counts FPU flops retired. Flops are defined to be 1 for Adds - 1 for Mults - 2 for FMAs (also FMS,FNMA). Also counts 1 for Fcvtfx, F[a]max, F[a]min, fcmp, frcpa and frsqrta.

4.2.3.16 FP_FLUSH_TO_ZERO

Description	FPU flush to zero RES
Max Inc/Cyc	2
MT Capture Type	A
Subevents:	
FTZ_POSS	0x858
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = -2
Definition	This counts 1 for every op that would be flushed to zero if FTZ was enabled.
FTZ_REAL	0x857
Counter Affinity	0xa0aa0

IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = -2
Definition	This counts 1 for every op that was flushed to zero because it was too small to represent as a normal number.

4.2.3.17 FP_TRUE_SIR

Description	FPU TRUE SIR
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x855
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 1
Definition	This counts 1 for every SIR replay which required a fault after the op was replayed.

4.2.3.18 RETIRED_INST_BR

Description	ALIAS for BR_PRED_DETAIL.ANY_ANY_PRED
Max Inc/Cyc	3
MT Capture Type	A
Event Code	0x820
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	Retired branch instructions

4.2.3.19 RETIRED_INST_FC

Description	ALIAS for RIL_REQ_OTHER.FC
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc30
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	-
NOTE	-

4.2.3.20 RETIRED_INST_FCI

Description	ALIAS for RIL_REQ_OTHER.FCI
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc31
Counter Affinity	0x49200

IAR/OPC/DAR/DREF	I/O/D/R
Definition	-
NOTE	-

4.2.3.21 RETIRED_INST_FP

Description	Floating point ops count
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x00b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	This counts 1 for every FPU op retired.

4.2.3.22 RETIRED_INST_LD_FP

Description	ALIAS for DATA_REF.LD_FP
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0df
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.3.23 RETIRED_INST_LD_INT

Description	ALIAS for DATA_REF.LD_INT
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0de
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.3.24 RETIRED_INST_M

Description	M ops retired
Max Inc/Cyc	2
MT Capture Type	A

Subevents:	
ACQ	0x0f4
Counter Affinity	0xaaaa0
IP-EAR	L = 6, P = 0
Definition	Includes all predicated on, retired instructions with acquire semantics. (Memory fence semantics are considered to include acquire semantics.)
ANY	0x00d
Counter Affinity	0xaaaa0
IP-EAR	L = 6, P = 0
Definition	Any M ops retired
MOVTOBSPST	0x0ca
Counter Affinity	0xaaaa0
IP-EAR	L = 6, P = 0
Definition	Data prefetch ASB invalidated by BSP store
MOVTODAHR	0x00e
Counter Affinity	0xaaaa0
IP-EAR	L = 6, P = 0
Definition	M ops retired with DAHR update
REL	0x0f5
Counter Affinity	0xaaaa0
IP-EAR	L = 6, P = 0
Definition	Includes all predicated on, retired instructions with release semantics. Also includes shootdowns. (Memory fence semantics are considered to include release semantics.)

4.2.3.25 RETIRED_INST_NOP

Description	Nops retired
Max Inc/Cyc	12
MT Capture Type	A
Event Code	0x00a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 0
Definition	Counts the number of no-ops retired
NOTE	When Nop squashing is enabled, this count will include brp

4.2.3.26 RETIRED_INST_PTCG

Description	ALIAS for RIL_REQ_OTHER.PTCG
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc34
Counter Affinity	0x49200

IAR/OPC/DAR/DREF	I/O/D/R
Definition	-
NOTE	-

4.2.3.27 RETIRED_INST_RSE

Description	RSE event retired
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x157
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	An RSE affecting instruction is retiring and committing architectural state: alloc, flushrs, loadrs, call, cover, return, or rfi with IFS.v set.
NOTE	>Per-instruction PMU tags (address and opcode matching) are not used for this event.

4.2.3.28 RETIRED_INST_SEMAPHORE

Description	ALIAS for DATA_REF.SEMAPHORE
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0e5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.3.29 RETIRED_INST_ST_FP

Description	ALIAS for DATA_REF.ST_FP
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0e2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.3.30 RETIRED_INST_ST_INT

Description	ALIAS for DATA_REF.ST_INT
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0e1
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.3.31 RETIRED_INST_TAGGED

Description	Instructions retired
Max Inc/Cyc	12
MT Capture Type	A
Subevents:	
IAM0_OPM0	0x005
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	Instructions retired, tag channel 0 qualified
IAM1_OPM1	0x006
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	Instructions retired, tag channel 1 qualified
IAM2_OPM0	0x007
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	Instructions retired, tag channel 2 qualified
IAM3_OPM1	0x008
Counter Affinity	0xffff0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	Instructions retired, tag channel 3 qualified

4.2.3.32 RETIRED_PREDICATE_SQUASHED

Description	Instructions retired with predicate off
Max Inc/Cyc	12
MT Capture Type	A

Event Code	0x009
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 5, P = 1
Definition	Counts the number of retired instructions with their predicate disabled
NOTE	When nop squashing is enabled, this count will exclude predicated off squashed nops

4.2.4 Back-End Cycle Accounting

This section enumerates the processor back-end cycle-accounting, also known as stall, performance monitoring events. The events are ordered by pipe-stage, from earliest to latest.

The processor's cycle accounting events are structured in a way that are non-overlapping down to the sub-event level. This is done by appropriate prioritization where possible, or by the introduction of "more-than-one" subevents, where multiple sub-events may assert simultaneously.

At the replay/stall/flush level, the replays are prioritized by the oldest, i.e. latest pipe-stage involved within the issue group. I.e. WB2 replays trump DET replays.

The non-overlapping property allows for complete sub-level accounting.

4.2.4.1 CYC_BE_BUBBLE

Description	Backend cycles stalled
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x024
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	This event captures any cycles lost to replays, flushes or bubbles, including partial replay cycles.

4.2.4.2 CYC_BE_DET_REPLAY

Description	Backend DET replay cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	A number of structural and data hazards associated with data memory references and a few register hazards are resolved through DET replay. The pipeline stages emptied by the replay are counted as "cycles" or bubbles by this event.
NOTE	The number of occurrences of DET replays is tightly lower bound by DET_REPLAY.* / 5
Subevents:	
ANY	0x04a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0

Definition	Any backend DET replay cycles
DCS_HZRD	0x04c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to DCS having a full tracking buffer, a data bug conflict between synchronous and asynchronous returns, or ITC/RUC reads issued too closely together.
FLUSHED_STORE	0x050
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to flushed store. Details: In the rare event that an FLDST updates the FLD, fails to commit, and then fails to invalidate the corresponding FLD cache line before an FLD hitting FLDLD makes it down the pipeline because there is a snoop in the pipeline, that FLDLD is replayed and these cycles are counted by this event.
GR_LOAD	0x04b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles roughly due to single cycle GR load hazards for loads that do not hit in FLD. (More precisely, DET replay cycles for single-cycle "memory op producer" to use bypasses that do not either hit FLD or DCS. Memory op producer: (M1+M2+M3).ldc_op# + M16 + M17 + M19 + M31.mov_from_urnat# + M33 + M34 + M36 + M38 + M39 + M43 + M46 + M1002). Includes replay cycles due to speculative predicate replays.
HPW_HZRD	0x051
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to HPW conflict. Details: Mov-to-rr instructions, mov-to-pkr instructions, and instructions that can cause DTB or IPC purges each take 1 of a particular resource in HPW. Instructions that can cause DTB or IPC inserts take 2 of the same resource. There are only 2 of these resources in HPW. The resources are freed up after the associated operations have completed. Any instruction that would require the simultaneous existence of more than 2 resources will be DET replayed.
LOAD_ACQ	0x04f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to Load Acquire. Details: An FLDLD/FLDST hit, a ld.c, or a chk.a will replay as long as an operation with acquire semantics from the same thread is outstanding. Previous processors forced misses on these operations. Replaying instead reduces the number of operations that can go into the MLD OZQ behind a ld.acq. The other thread may then be able to make better use of the MLD OZQ. Ld.acq replays are broken into two chunks. DET replays cover the case when an outstanding operation with acquire semantics has reached WB6 by the time the operation to be replayed reaches DET. In other words, DET replays cover the case where the outstanding operation with acquire semantics is at least 6 cycles ahead of the replayed op. In cases where the outstanding acquire op is less than 6 cycles ahead of the replayed op, a WB2 replay is used. The DET replays replay to an IBD issue stall. WB2 replays do not.
LOAD_AFTER_WRITE	0x04e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0

Definition	Backend DET replay cycles due to Load after Write. Details: A hitting FLDLD following an overlapping hitting FLDWR operation either within the same cycle or one cycle later is replayed. If PSR.ac is set (alignment checking on), "overlapping" means true overlap. This is better than previous processors. If PSR.ac is not set, "overlapping" means within the same 8-byte chunk.
MT1	0x054
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to more than 1 reason.
STORE_VS_STORE	0x04d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to Store vs Store hazard. Details: A structural hazard between two simultaneous hitting FLDSTs with same VA[7:5] and different VA[11:8] causes the second FLDST to replay. This is similar to a hazard existing in previous processors, but how the specific address bits conflict has changed.
WRITE_HIT_VS_FILL	0x052
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to Store Hit vs Fill. Details: A hitting FLDST is replayed if it occurs simultaneously with a fill with same VA[7:6] and different VA[11:8] due to a structural hazard in the FLD data array. A hitting FLDWR is replayed if it occurs at the same time as or one cycle following a fill with the same VA[11:6] so that a write targeting a line being replaced will not incorrectly write to the line that is replacing it.
WRITE_MISS_VS_FILL	0x053
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend DET replay cycles due to Write Miss vs Fill. Details: An FLDST that misses the FLD and has the same VA[13:6] as a fill that occurs at the same time as or one cycle following the FLDST is DET replayed. An FLDWR that misses the FLD and has the same VA[13:6] as a fill that occurs one cycle before the FLDWR is DET replayed. These replays happen because there is not time for the FLDWR to cancel the fill in the case that the FLDWR and fill are associated with the same VA[63:6].

4.2.4.3 CYC_BE_EXE_REPLAY

Description	Backend EXE replay cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	Register hazards and miscellaneous other hazards are resolved through EXE replay. The pipeline stages emptied by the replay are counted as "cycles" or bubbles by this event.
NOTE	Sometimes the replay ends with an issue stall until a hazard has resolved. The issue stall itself is counted by another event. The number of occurrences of EXE replays is tightly lower bound by EXE_REPLAY__* / 4
Subevents:	
ANY	0x03a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0

Definition	Backend EXE replay cycles due to any reason
ARCR	0x048
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to AR-CR hazard
FCMP	0x042
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to fcmp; a predicate consumer needs a predicate from an fp producer
FPSR	0x045
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to fpsr hazards
FR_FR	0x040
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to FR-FR hazard; a new instruction has a source register targeted by a prior fp arithmetic operation; the HW uses a combination of replays and issue stalls to perfectly schedule floating point instructions to the needed 6 cycles of separation between producer and consumer. No extra wasted cycles are generated.
FR_LOAD_RAW	0x03c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to FR load RAW; a new instruction has a source register targeted by an outstanding load, or getf instruction.
FR_LOAD_WAW	0x03e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to FR load WAW; a new instruction has a destination register targeted by an outstanding load, or getf instruction.
GR_GR	0x03f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to GR-GR hazard; a new instruction has a source register targeted by a prior non-load type instruction; or there is a UNAT register conflict.
GR_LOAD_RAW	0x03b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0

Definition	Backend EXE replay cycles due to GR load RAW; a new instruction has a source register targeted by an outstanding load, or outstanding long latency move, or TLB related operation.
GR_LOAD_WAW	0x03d
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to GR load WAW; a new instruction has a destination register targeted by an outstanding load, or outstanding long latency move, or TLB related operation.
MT1_HIGH	0x041
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to more than one (MT1) of GR_LOAD_*, FR_LOAD_*, GR_GR, FR_FR.
MT1_LOW	0x049
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to more than one (MT1) of FPSR, SRLZ, REL, ARCR
NOTN	0x044
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to a not-needed replay; predicate prediction turned out wrong
PRED	0x043
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to miscellaneous predicate hazards;
REL	0x047
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to operations with release semantics
SRLZ	0x046
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend EXE replay cycles due to serialization operations

There are as many as 11 issued instructions in a cycle that can each need to signal an EXE replay, (squashed nops will not signal EXE replay) and each of these instructions can need an EXE replay for more than one reason. For replay reporting, there is a priority per port, a priority per port pair (e.g., M0 and M1), and a priority across ports. Per port (highest to lowest): PDU(FCMP or PRED), NOTN, GRLD_RAW(FRLD_RAW), GRLD_WAW(FRLD_WAW), GRGR(FRFRHERS; Per port pair: For GR replays, PDU on P0 wipes out reason on P1; Across ports: the order of the above table, top to bottom.

Only one of the non-any replay reasons is recorded per replay event. If a more-then-one replay reason is given, the individual replay reasons are not available for that event.

4.2.4.4 CYC_BE_IBD_STALL

Description	Backend IBD stall cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	Records the reason for each cycle that no instructions are issued from the instruction buffers
NOTE	Otherwise known as issue "bubbles"
Subevents:	
ACQ	0x02e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to an acquire op outstanding; starts after a DET replay.
ANY	0x025
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles for any reason
DEBUG	0x039
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to debug; a defeaturing mode is active which creates unnecessary stalls
FEBUB	0x038
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to FE bubble; the instruction fetch engine hasn't provided anything to issue
FLD_DMND	0x035
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to FLD memory port demands; No instruction can issue because FLD is using the memory pipeline for a purge, snoop, HW prefetch, etc., and (1) the first instruction trying to issue is an M-op, or (2) we just replayed and are trying to reissue the same group of instructions, or (3) any RSE load or store wants to issue.
FR_LOAD	0x030
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to FR load RAW or WAW condition; starts after an EXE replay

FTOF	0x034
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to F to F hazard; floating point scheduling stall
GR_LOAD	0x02f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to GR load RAW or WAW condition; starts after an EXE replay or DET replay
HPW	0x02c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to HPW; an HW page walk is in progress and a memory op is waiting for the result; starts after a WB2 replay.
MTOM	0x033
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to M to M hazard; this is the one cycle load to load address scheduling stall; creates one bubble between a first load and a second load where the second load uses the first load's target register as an address register.
OZQFULL	0x02d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to OZQ full or OZ data full; a memory op is denied entry into OZ (MLD cache control) and is waiting on availability; starts after a WB2 replay.
QFULL	0x037
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to speculative IBQ full bubble. IBL signaled queue full to FE while FE was trying to hand IBL some instructions, and later that bubble was exposed to the issue logic.
REL	0x032
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to release semantics; a memory instruction with .rel behavior is waiting for prior TLB operations to complete; starts after an EXE replay.
RSE_ANY	0x026
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to any RSE subreason

RSE_CFLE	0x027
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to injection of RSE fill into the pipeline caused by br.ret or rfi
RSE_LOAD	0x029
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to injection of RSE fill into the pipeline caused by loadrs
RSE_ST	0x028
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to injection of RSE store into the pipeline caused by alloc or flushrs
RSE_WAIT	0x02a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to RSE waiting to inject another load or store; see below for more info
SRLZ	0x031
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to serialization; an rfi, srlz.i, or srlz.d instruction is waiting for prior resource updates to complete, or sync.i instruction to finish; starts after an EXE replay
THRSW	0x02b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to thread switch controller waiting to switch or in the act of switching
WB2_TRAP	0x036
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend IBD bubbles due to WB2 replay trap; some WB2 replays leave nothing to re-issue for one cycle after the replay. For example, a ld.c that misses the ALAT is handled as a WB2 replay-trap, where the load retires but the following instructions replay. If the ld.c is at the end of the issue group, there will be nothing to re-issue in it's group.

Only one of the non-any stall reasons is recorded per cycle. The priority order for the non-any events is, top to bottom, THRSW down to ARCR, RSE*, MTOM, FTOF, DEBUG, QFULL, FEBUB, FLDDMND, WB2TRAP. Some of these stall events are mutually exclusive, so the priority order doesn't always affect the reporting.

The RSE_WAIT item can be due to a variety of reasons as described in the table below.

RSE_WAIT can be due to:	
1	initial 2 bubble wait before first RSE load or store can be issued
2	initial wait on any AR dependency (like mov to AR[BSPS])
3	initial wait on arch RSE state updates prior to loadrs execution
4	during load/store in pipeline, wait for rnat bit dependencies to clear
5	after an rse rnat load, wait for completion before next load (could miss to memory)
6	unneeded 3-cycle stall due to frontend RSE mis-speculation on RSE.ndirty

4.2.4.5 CYC_BE_NO_BUBBLE

Description	Backend cycles not stalled
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x023
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	This event counts the number of cycles an entire issue group was able to retire.
NOTE	This event differs from its counterpart in a stall-based pipeline in that it assesses a penalty for issue groups that retire partially. The "any instruction retired" flavor can be obtained by measuring IA64_INSTS_RETIRED with PMC.thres set to 1.

4.2.4.6 CYC_BE_WB2_FLUSH

Description	Backend WB2 flush cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
ANY	0x065
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 flush cycles
BRU	0x067
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 flush cycles due to BRU
XPN	0x066
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 flush cycles due to XPN

4.2.4.7 CYC_BE_WB2_REPLAY

Description	Backend WB2 replay cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
ALLOC_PEC	0x058
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to alloc pec hazard
ANY	0x055
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles
BLK_HPW	0x05d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to blocking HPW. Details: An operation that is going to do a blocking hardware page walk is WB2 replayed to an IBL issue stall.
DAHR_HZRD	0x061
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to DAHR RAW hazard. Details: If the DPFQ back pressure mechanism is turned on, that is likely the largest component in these replay cycles. DPFQ back pressure can occur whenever the MLD OZQ is full and the DPFQ is filling up with lfetches from the A-ports or when the DPFQ is filling up with lfetches from the A-ports and its issuing is blocked. As the event name suggests, this event also includes Data Access Hint Register (DAHR) hazards typically involving multiple DAHS modifiers simultaneously in flight in the pipeline.
FP_DEN	0x05b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to FPU denormal
FP_SIR	0x05c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-

IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to FPU SIR
LDC	0x056
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to a ld.c.
LOAD_ACQ	0x062
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to load acquire. Details: See the LOAD_ACQ DET replay. This WB2 replay covers the case where the outstanding acquire op is less than 6 cycles ahead of the op to be replayed.
MOV_PSR_UM	0x059
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to move to psr.um
MT1	0x063
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to more than 1
NAT_HZRD	0x060
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to NaT hazard. Details: If the DTB defers an operation that hits in the FLD (i.e. any deferral except NAT address consumption or a spontaneous deferral on MLD miss), then all subsequent operations are WB2 replayed so that any potential consumers of the NAT will see it. Also, if a ld.a or ld.c.nc in WRB fails to allocate in the ALAT because of some DTB translation issue, any chk.a or ld.c with same register identifier in DET will be WB2 replayed.
OZQ_FULL	0x05e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to OZQ full. When the MLD OZQ proper is full or the MLD OZ Data Queue is full, operations that need to go into the MLD OZQ or Data Queue, respectively, are WB2 replayed to an IBL issue stall.
PAUSE	0x057
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to hint \\@pause
SER	0x064
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to parity and similar errors
STORE_ALIAS	0x05f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to store alias. Details: This replay is to handle a data hazard between an FLDWR miss (or physical mode FLDWR instrs) followed by an FLDLD hit with the same PA. The FLDTLB does not allow virtually aliased (VA1->PAx, VA2->PAx) translations (including cross thread) to coexist in it. Therefore, this case can only arise for a store that misses the FLDTLB (and thus the FLD). Such stores check the PA portion of the FLDTLB to see if a virtually aliased translation exists. If such a translation exists, it is purged before any FLDLD hit can use it. This is accomplished by watching for and replaying any FLDLD hits after an alias has been detected but before the purge of that translation. FLDWR misses (or physical mode FLDWR instrs) in such a window are also replayed in order to avoid oversubscribing the purge pipeline. Also, for two such operations in the same cycle, instead of checking the PA portion of the FLDTLB, the operations are just checked to see if they potentially overlap by just comparing size and VA[11:0].
VIRT_INT	0x05a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Backend WB2 replay cycles due to virtual interrupt

4.2.4.8 Derived back-end events

At times, cause-centric measurements are desired, i.e. aggregate GR load penalties. For these, several monitors need to be added up. Using GR loads as example, the following needs to be done:

GR_LOAD_CYCLES_ALL = IBD_STALL.GR_LOAD + EXE_REPLAY.GR_LOAD_RAW + EXE_REPLAY.GR_LOAD_WAW + DET_REPLAY.GR_LOAD

4.2.5 Front-End Cycle Accounting

This section enumerates the processor front-end cycle-accounting performance monitoring events. The events are ordered by pipe-stage, from earliest to latest.

As with the back end, the processor's cycle accounting events are structured in a way that are non-overlapping down to the sub-event level.

The non-overlapping property allows for complete sub-level accounting.

4.2.5.1 CYC_FE_BUBBLE.ANY

Description	Front end stalled cycles
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x8a3
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Cycles in which no valid bundles were delivered to IBD. Includes any reason for a stall, including flushes, replays, and bubbles.

4.2.5.2 CYC_FE_FET_REPLAY

Description	Front end bubbles due to FET replay
Max Inc/Cyc	1
MT Capture Type	A
Definition	Front end bubbles due to FET replay
NOTE	CYC_FE_FET_REPLAY.ANY is more accurate than the sum of its sub-events.
Subevents:	
ANY	0x8ab
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to FET replay
BRQ_FULL	0x8ad
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to BRQ full FET replay
BRQ_WAIT	0x8ae
Counter Affinity	0x505a5
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to BRQ wait FET replay
BR_INIT	0x8ac
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to Br init FET replay
BR_INTRLCK	0x8af
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to Br interlock FET replay
MT1	0x8b1
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to more than 1 FET replay
RAB_FULL	0x8b0
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to RAB full FET replay

4.2.5.3 CYC_FE_FET_STALL

Description	Front end FET stall cycles
Max Inc/Cyc	1
MT Capture Type	A
Subevents:	
ANY	0x8b2

Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end FET stall cycles
FLITLB_MISS	0x8b5
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end FET stall cycles due to FLI TLB miss
FLI_MISS	0x8b4
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end FET stall cycles due to FLI miss
IBQ_FULL	0x8b3
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end FET stall cycles due to IBQ full
MT1	0x8b6
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end FET stall cycles due to more than 1

4.2.5.4 CYC_FE_NO_BUBBLE

Description	Cycles with valid bundles delivered to IBD
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x8a2
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
IAR/OPC/DAR/DREF	-/-/-/-
Definition	-
NOTE	-

4.2.5.5 CYC_FE_RESTEER

Description	Front end bubble cycles
Max Inc/Cyc	1
MT Capture Type	A
Subevents:	
ANY	0x8a4
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubble cycles
BE_FLUSH	0x8aa
Counter Affinity	0x50550

IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to Back-end resteer
BR_RETIND	0x8a6
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to Br return ind
IPREL	0x8a5
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to IP relative branch
NON_RETIND	0x8a7
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to Br not return ind
SEQ_BR	0x8a8
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to sequential Br
TSWITCH	0x8a9
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end bubbles due to thread switch

4.2.5.6 FE_RESTEER

This event provides occurrence counts for the resteer events

Description	Front end resteer
Max Inc/Cyc	1
MT Capture Type	A
Subevents:	
0B_IPREL	0x8b9
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 0-bubble IP relative
1B_BR_RETIND	0x8bc
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 1-bubble BR return indirect
1B_IPREL	0x8ba
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 1-bubble IP relative
1B_SEQ_BR	0x8bf
Counter Affinity	0xa0aa0

IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 1-bubble sequential BR
3B_BR_RETIND	0x8bd
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 3-bubble BR return indirect
3B_IPREL	0x8bb
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 3-bubble IP relative
3B_MT	0x8c1
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 3-bubble thread switch
3B_NON_RETIND	0x8be
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 3-bubble BR non-return indirect
3B_SEQ_BR	0x8c0
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 3-bubble sequential BR
4B_MT	0x8c2
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, 4-bubble thread switch
ANY	0x8b7
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer
BE_FLUSH	0x8c3
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, back-end resteer
FET_REPLAY	0x8b8
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end resteer, FET replay

4.2.6 Branch Events

This section enumerates branch performance monitoring events. Most of these events record the results of "reached" and "committed" branches. (Exceptions are the EAR_EVENT_ETB_IP_MT event, and portions of the BR_BE_PRED_DETAIL event). Itanium architecture allows up to three branches in a bundle, and on current microprocessors all three branches execute at the same time as a large instruction

word. Reached refers to no prior taken branches in the instruction group or bundle, and committed refers to no prior faults or traps in the instruction group or bundle, and no fault on the referenced branch. Thus, the branches counted by these events are "retired" branches.

For the branch events that count aspects of correct predictions and mispredictions, only the first (oldest) misprediction in a bundle is reported as a misprediction. As a consequence of this, any retiring branch that is after a predicted taken branch in a bundle, is logged as a correctly predicted branch whether taken or not taken. This is a result of the large instruction word (LIW) behavior of all three branches being executed at the same time, with only one misprediction being reported per bundle. There is at most one misprediction penalty per bundle of branches.

The reached and committed limitation on reporting these events creates an oddity. For example, a predicted taken branch that is not taken and not reached, is not itself counted as a misprediction. Though the prior branch that is taken is counted as a misprediction.

A similar oddity is caused by the one-hot per-bundle branch path prediction vector provided by the instruction fetch engine in the frontend. After the first predicted taken branch in a bundle, the younger branch slots are indicated as predicted not-taken, even if some part of the HW could have predicted them taken. These younger branch slots' prediction status is referred to as "unknown." Therefore, branches with a prediction status of unknown, that are reached and committed, are classified as correctly predicted.

4.2.6.1 BR_BE_PRED_DETAIL

Description	backend branch misprediction detail
Max Inc/Cyc	1
MT Capture Type	A
Definition	Backend pipeline caused branch mispredictions
NOTE	This event records branch related pipeline flushes and some of their causes. These are mostly due to activity outside of the instruction fetch engine's target address and taken/not-taken path prediction machinery. The subevents, STG, ROT, PFS, OTHER, and ANY_RETIRED only record data when no path or target misprediction occurred on the same bundle of branches. On Intel Itanium Processor 9300 Series, these events were never complete. On the processor, there are a few additional sources of backend caused mispredictions such as deconfiguration bit settings, or, for example, a ld.c missing the ALAT with a dependent cmp instruction targeting the qp of a branch.
Subevents:	
ANY	0x807
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	A b-type instruction was present and a BRU-generated pipeline flush occurred. This subevent is the sum: ANY_RETIRED + UNRETIRED + BR_PRED_DETAIL.ANY_WRONG_PATH + BR_PRED_DETAIL.ANY_WRONG_TARGET
ANY_RETIRED	0x805
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	For convenience, Any_retired is the sum: STG + ROT + PFS + OTHER.

OTHER	0x804
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	There are several OTHER reasons for backend mispredictions, such as an epc in slot 0 or 1 of a bundle, or privilege level mispredictions on a branch return, or a forced flush due to a prior replayed unretired flush, etc. OTHER will be recorded if any branch retired during the flush.
PFS	0x803
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	The rename unit (RNM) used the wrong PFS.pfm value to update CFM. A check of this speculative value with the architectural PFS shows a mismatch. This check is done on predicted taken branch returns.
ROT	0x802
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	The frontend rename unit (RNM) mispredicted the register rotation. These predictions are based on a speculative AR[EC] value coupled with FE path prediction.
STG	0x801
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	The backend BRU told the predicate delivery unit (PDU) the wrong stage predicate update in the REG pipeline stage, and must correct that with a pipeline flush. This is effectively AR[LC] misprediction, and happens when a move-to-LC is still in flight at the time a rotating branch gets to the REG pipeline stage.
UNRETIRED	0x806
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	No branch instruction retired, but some b-type instruction was present, and a branch flush occurred. This is generally due to a WB2 replay taking precedence over a flush, but could be due to an epc in slot 0 or 1 with no branch in the bundle.

4.2.6.2 BR_ENC_PRED_DETAIL

Description	Number of "Encoded" Branches Retired
Max Inc/Cyc	3
MT Capture Type	A
Definition	Counts branches retired for bundles with a branch in slot 0 or bundles which over subscribe MLB
NOTE	Bundles with branches in slot 0 (BBB templates with a real branch in slot 0) are termed "encoded," because the first level branch cache has to compress information from 3 branch slots into a space optimized for 2 branches. MLB oversubscription refers to placing information in the second level (mid-level) branch cache, which is reserved for branch bundle pairs with branches in only one of the bundles, or with less than 3 branches. Bundle pairs which "over-subscribe" are not placed in MLB on eviction from the first level branch cache.
Subevents:	
ENC_ANY_PRED	0x830

Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles, regardless of prediction result
ENC_CORR_PRED	0x831
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles, correctly predicted path and target
ENC_OVRSUB_ANY_PRED	0x838
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles that oversubscribe MLB, regardless of prediction result
ENC_OVRSUB_CORR_PRED	0x839
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles that oversubscribe MLB, correctly predicted path and target
ENC_OVRSUB_WRONG_PATH	0x83a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles that oversubscribe MLB, mispredicted path
ENC_OVRSUB_WRONG_TARGET	0x83b
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles that oversubscribe MLB, mispredicted target, but correctly predicted path
ENC_WRONG_PATH	0x832
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles, mispredicted path
ENC_WRONG_TARGET	0x833
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in encoded bundles, mispredicted target, but correctly predicted path
OVRSUB_ANY_PRED	0x834
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-

IP-EAR	L = 7, P = 2
Definition	branches in bundles that oversubscribe MLB, regardless of prediction result
OVRSUB_CORR_PRED	0x835
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in bundles that oversubscribe MLB, correctly predicted path and target
OVRSUB_WRONG_PATH	0x836
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in bundles that oversubscribe MLB, mispredicted path
OVRSUB_WRONG_TARGET	0x837
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	branches in bundles that oversubscribe MLB, mispredicted target, but correctly predicted path

4.2.6.3 BR_PATH_PRED

Description	Frontend Branch Path Prediction Detail
Max Inc/Cyc	3
MT Capture Type	A
Definition	Counts the number of branches retired based on branch path prediction (taken/not-taken), branch path outcome, and branch subtype (ip-rel, indirect, br.ret, all).
NOTE	This event is intended to work the same as on Intel Itanium Processor 9300 Series.
Subevents:	
ANY_MISPRED_NOT_TAKEN	0x808
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with incorrectly predicted path, and not taken outcome
ANY_MISPRED_TAKEN	0x809
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with incorrectly predicted path, and taken outcome
ANY_OKPRED_NOT_TAKEN	0x80a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with correctly predicted path, and not taken outcome
ANY_OKPRED_TAKEN	0x80b

Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with correctly predicted path, and taken outcome
IPREL_MISPRED_NOT_TAKEN	0x80c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with incorrectly predicted path, and not taken outcome
IPREL_MISPRED_TAKEN	0x80d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with incorrectly predicted path, and taken outcome
IPREL_OKPRED_NOT_TAKEN	0x80e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with correctly predicted path, and not taken outcome
IPREL_OKPRED_TAKEN	0x80f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with correctly predicted path, and taken outcome
NON_RETIND_MISPRED_NOT_TAKEN	0x814
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with incorrectly predicted path, and not taken outcome
NON_RETIND_MISPRED_TAKEN	0x815
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with incorrectly predicted path, and taken outcome
NON_RETIND_OKPRED_NOT_TAKEN	0x816
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with correctly predicted path, and not taken outcome
NON_RETIND_OKPRED_TAKEN	0x817
Counter Affinity	0xa0aa0

IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with correctly predicted path, and taken outcome
RETURN_MISPRED_NOT_TAKEN	0x810
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with incorrectly predicted path, and not taken outcome
RETURN_MISPRED_TAKEN	0x811
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with incorrectly predicted path, and taken outcome
RETURN_OKPRED_NOT_TAKEN	0x812
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with correctly predicted path, and not taken outcome
RETURN_OKPRED_TAKEN	0x813
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with correctly predicted path, and taken outcome

4.2.6.4 BR_PRED_DETAIL

Description	Frontend Branch Misprediction Detail
Max Inc/Cyc	3
MT Capture Type	A
Definition	Counts the number of branches retired based on path prediction result, target prediction result, and branch type (ip-rel, indirect, br.ret, any)
NOTE	The path prediction is the taken/not-taken prediction. The target prediction is for the target address of the branch. Target address prediction is checked only on the single predicted taken branch in a bundle, if there is one. This event is intended to work like the corresponding Intel Itanium Processor 9300 Series event. The subevents are defined as follows. ANY_PRED just counts the branch type; CORR_PRED counts branches that had a correctly predicted path and did not have a mispredicted target address; WRONG_PATH counts mispredicted path branches (with the same kind of overcount as for BR_PRED_PATH subevents); WRONG_TARG counts branches with correctly predicted path and a mispredicted target address (implies it was predicted taken and was taken).
Subevents:	
ANY_ANY_PRED	0x820
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types, regardless of prediction result

ANY_CORR_PRED	0x821
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types, correctly predicted path and target
ANY_WRONG_PATH	0x822
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types, mispredicted path
ANY_WRONG_TARGET	0x823
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types, mispredicted target, but correctly predicted path
IPREL_ANY_PRED	0x824
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types, regardless of prediction result
IPREL_CORR_PRED	0x825
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types, correctly predicted path and target
IPREL_WRONG_PATH	0x826
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types, mispredicted path
IPREL_WRONG_TARGET	0x827
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types, mispredicted target, but correctly predicted path
NON_RETIND_ANY_PRED	0x82c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types, regardless of prediction result
NON_RETIND_CORR_PRED	0x82d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-

IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types, correctly predicted path and target
NON_RETIND_WRONG_PATH	0x82e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types, mispredicted path
NON_RETIND_WRONG_TARGET	0x82f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types, mispredicted target, but correctly predicted path
RETURN_ANY_PRED	0x828
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types, regardless of prediction result
RETURN_CORR_PRED	0x829
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types, correctly predicted path and target
RETURN_WRONG_PATH	0x82a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types, mispredicted path
RETURN_WRONG_TARGET	0x82b
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types, mispredicted target, but correctly predicted path

4.2.6.5 BR_PRED_UNKNOWN

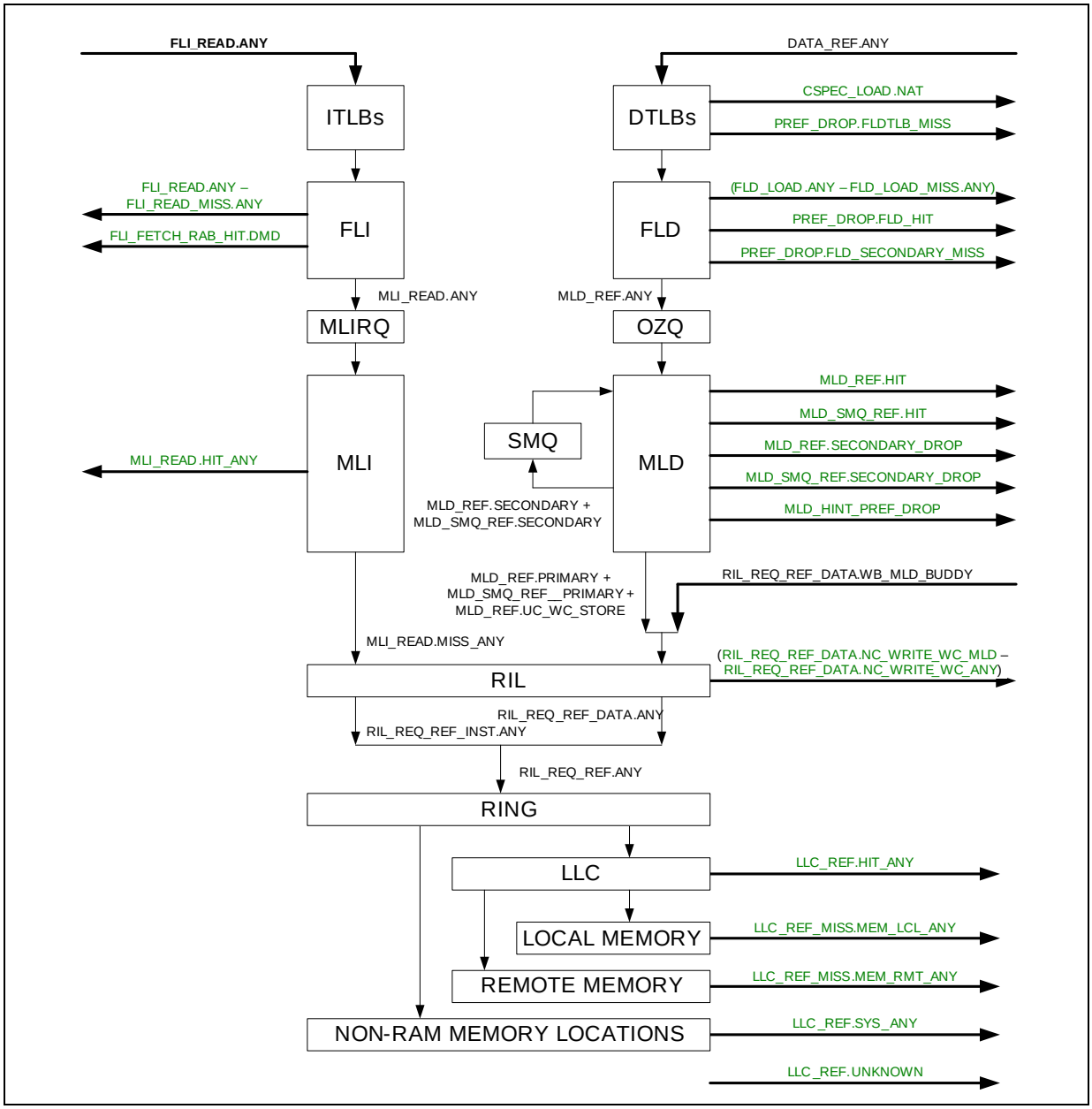
Description	Frontend Branch Path Prediction Unknown
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts retirements of slot 1 and slot 2 branches in which a prior slot was predicted taken
NOTE	When a prior slot in a branch bundle was predicted taken, the remaining slots have an "unknown" predicted path (taken/not-taken). These branches, when retiring, are always counted as correctly predicted, because they do not contribute to a misprediction penalty. When slot 0 is predicted taken, slot 1 and slot 2 are unknown. When slot 1 is predicted taken, slot 2 is unknown.

Subevents:	
ANY	0x818
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with unknown predicted path, and taken or not-taken outcome
ANY_TAKEN	0x819
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	all branch types with unknown predicted path, and taken outcome
IPREL	0x81a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with unknown predicted path, and taken or not-taken outcome
IPREL_TAKEN	0x81b
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	ip-relative branch types with unknown predicted path, and taken outcome
NON_RETIND	0x81e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with unknown predicted path, and taken or not-taken outcome
NON_RETIND_TAKEN	0x81f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	non-return indirect branch types with unknown predicted path, and taken outcome
RETURN	0x81c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with unknown predicted path, and taken or not-taken outcome
RETURN_TAKEN	0x81d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 7, P = 2
Definition	return branch types with unknown predicted path, and taken outcome

4.2.7 Memory Hierarchy Events

The following sections contain the PMU events for the processor memory hierarchy events. The figure below gives an overview of the memory hierarchy and key pertinent performance monitors. The diagram should be read as a kind of flowchart. The three arrows entering the diagram (labeled FLI_READ.ANY, DATA_REF.ANY, and RIL_REQ_REF_DATA.WB_MLD_BUDDY) represent all of the memory references initiated in the core, and the number of occurrences of each type of memory reference can be measured by the corresponding event. The arrows leaving the diagram on the right and left sides represent the memory reference outcomes, and the number of occurrences of each outcome can be measured by the corresponding event. The internal arrows represent memory references flowing from one part of the memory hierarchy to another, and the number of references flowing at each part of the hierarchy can be measured by the corresponding events.

Figure 4-1. Processor Memory Hierarchy



4.2.7.1 Primary Data Reference Types

Much care has been taken to define and count a consistent set of primary data reference types across a wide range of pmu events. All of the events in the preceding diagram count data references if and only if they are from this set of primary data reference types:

- LOAD_INT: integer loads (see DATA_REF.LOAD_INT)
- LOAD_FP: floating point loads (see DATA_REF.LOAD_FP)
- LOAD_RSE: RSE loads (see DATA_REF.LOAD_RSE)

- LOAD_HPW: hardware page walker loads (see DATA_REF.LOAD_HPW)
- STORE_INT: integer stores (see DATA_REF.STORE_INT)
- STORE_FP: floating point stores (see DATA_REF.STORE_FP)
- STORE_RSE: RSE stores (see DATA_REF.STORE_RSE)
- SEMAPHORE: semaphores (see DATA_REF.SEMAPHORE)
- LFETCH: software prefetches (see DATA_REF.LFETCH)
- HW_PREF: hardware prefetches (see DATA_REF.HW_PREF and RIL_REQ_REF_DATA.WB_MLD_BUDDY)

Any reference to LOAD_INT, LOAD_FP, LOAD_RSE, LOAD_HPW, STORE_INT, STORE_FP, STORE_RSE, SEMAPHORE, LFETCH, or HW_PREF in an event name that also contains FLD, MLD, RIL, or LLC implies that the event is measuring the specified subset of these primary data reference types (unless specifically noted otherwise in the event description.) Also, references to LOAD(_ANY)/STORE(_ANY) in this same set of events implies that the event is measuring the LOAD_*/STORE_* subset of these primary data reference types. For example, the event name MLD_LOAD_MISS implies that this event is measuring all of the data references of types LOAD_INT, LOAD_FP, LOAD_RSE, or LOAD_HPW that access and miss the MLD. Furthermore, references to REF in any event name that also contains FLD, MLD, RIL*DATA*, or LLC*DATA* implies that the event is measuring this set of primary data reference types.

This consistency in definition of and usage of primary data reference types has enabled definition of the following equations with further clarify and define what various data cache hierarchy events measure and their relationships to each other:

- DATA_REF.ANY == (PREF_DROP.FLD_HIT + PREF_DROP.FLD_SECONDARY_MISS + PREF_DROP.FLDTLB_MISS + CSPEC_LOAD.NAT + (FLD_LOAD.ANY – FLD_LOAD_MISS.ANY) + MLD_REF.ANY)
- MLD_REF.ANY == (MLD_REF.HIT + MLD_HINT_PREF_DROP + MLD_REF.SECONDARY_DROP + MLD_SMQ_REF.SECONDARY_DROP + MLD_SMQ_REF.HIT + MLD_REF.PRIMARY + MLD_SMQ_REF.PRIMARY + MLD_REF.UC_WC_STORE)
- (MLD_REF.PRIMARY + MLD_SMQ_REF.PRIMARY + MLD_REF.UC_WC_STORE + RIL_REQ_REF_DATA.WB_MLD_BUDDY) == (RIL_REQ_REF_DATA.ANY + (RIL_REQ_REF_DATA.NC_WRITE_WC_MLD - RIL_REQ_REF_DATA.NC_WRITE_WC_ANY))
- (RIL_REQ_REF_INST.ANY + RIL_REQ_REF_DATA.ANY - RIL_REQ_REF_DATA.NC_WRITE_ANY) == (LLC_REF_HIT.ANY + LLC_REF_MISS.MEM_LCL_ANY + LLC_REF_MISS.MEM_RMT_ANY + LLC_REF.SYS_ANY + LLC_REF_UNKNOWN)

Note that the last equation above includes more than just data references. However, the data references could be distinguished by using data reference matching to filter out all data references and then subtracting the non-data references values from the values including data references.

4.2.7.2 Asynchronous Data References and Event Matching Constraints

Asynchronous data references (e.g. LOAD_RSE, LOAD_HPW, STORE_RSE, LFETCH, and HW_PREF) are counted by many PMU events. These references pose a quandary with respect to instruction matching (address and opcode) because unlike synchronous data references, they do not have a one-to-one mapping with instructions. These references could either ignore instruction matching or they could inherit instruction matching from

their triggering instruction. HW_PREF data references that originate in the MLD hardware prefetcher and HW_PREF data references that are MLD buddy line prefetches ignore instruction matching. LOAD_RSE, LOAD_HPW, STORE_RSE, LFETCH, and all other types of HW_PREF data references inherit instruction address matching from their triggering instruction.

Therefore, instruction matching can be used on events that include asynchronous data references with some care as noted in the following guidelines:

- Data reference type matching should typically be used to filter out HW_PREF type data references
- Data reference type matching should be used to specifically include or exclude the other types of asynchronous data references desired in the measurement.

MLD buddy line prefetches can also lead to some confusion with respect to data reference type matching alone. MLD buddy line prefetches are hardware prefetches that are inserted between MLD and RIL. These hardware prefetches respond to HW_PREF data reference type matching. In RIL* events, MLD buddy line prefetches only respond to the HW_PREF data reference type matching. Therefore, although LOAD_INT, for example, may trigger an MLD buddy line prefetch, for RIL* events it is the data reference type of HW_PREF that will determine whether or not this buddy line prefetch is counted. So, if LOAD_INT were excluded by data reference type matching but HW_PREF were not, then the MLD buddy line prefetches triggered by the LOAD_INT will still be counted.

However, for all other PMU events that include MLD buddy line prefetches (e.g. LLC events, MLD fill side events), the triggering data reference also plays a role in data reference type matching. Specifically, MLD buddy line prefetches will be counted only when data reference type matching is configured to allow counting of the HW_PREF type AND the data reference type of the triggering instruction. For example, if data reference type matching is configured to count only LOAD_INT and HW_PREF data reference types, then MLD buddy line prefetches that are triggered by LOAD_INT or HW_PREF data references will be counted and buddy line prefetches triggered by other types of data references will not be counted.

4.2.7.3 DATA_REF

Description	Data references to the memory hierarchy.
Max Inc/Cyc	2
MT Capture Type	A
Definition	Includes loads, stores, semaphore instructions, virtual mode lfetches, virtual and physical mode hardware initiated prefetches other than MLD buddy line prefetches, RSE loads, RSE stores, and VHPT loads. Does NOT include accesses from predicated off instructions, TLB only accesses (e.g. TLB purges), flush caches operations, snoops, write-backs, ld.c's that hit in the ALAT, physical mode lfetches, or MLD buddy line prefetches.
Subevents:	
ANY	0x0dd
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	ANY == SUM(LOAD_INT, LOAD_FP, LOAD_RSE, LOAD_HPW, STORE_INT, STORE_FP, STORE_RSE, SEMAPHORE, LFETCH, HW_PREF)
LOAD_INT	0x0de
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on, integer load instructions that retire
LOAD_FP	0x0df
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on floating point load instructions that retire
LOAD_RSE	0x0e0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from RSE load operations
LOAD_HPW	0x0e7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from vhpt loads that are accepted by the MLD OZQ
STORE_INT	0x0e1
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on integer store instructions that retire
STORE_FP	0x0e2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on floating point store instructions that retire
STORE_RSE	0x0e3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from RSE store operations
SEMAPHORE	0x0e5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on semaphore instructions (xchg, cmpxhg, fetchadd) that retire
LFETCH	0x0e4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1

Definition	All LFETCH data references are to addresses requested by lfetch instructions. (Data prefetches to other addresses are HW_PREF.) However, there are several issues that can prevent a 1-to-1 correspondence with predicated on retired lfetch instructions: 1) Some lfetches issued to A-ports that don't retire can still generate data references. 2) Some lfetches issues to A-ports can be dropped before they generate data references. 3) Lfetches can be dropped due to missing the DTB and not counted as data references. 4) Lfetch.cnt can, obviously, generate more than one data reference. 5) Lfetch instructions that miss the FLDTLB often generate 2 data references to the same address. 6) Data references triggered by lfetch instructions are not counted when data address translation is disabled.
HW_PREF	0x0e6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	Data references from hardware initiated prefetches that are injected into the M-pipeline as a_dpf asynchronous operations but are not re-inserted into the DPFQ due to being rejected by a full MLD OZQ and are not dropped due to missing the DTB. Specifically, this does NOT say that only hardware initiated prefetches that are accepted by the OZQ are counted. For instance, hardware initiated prefetches that are dropped due to an FLDTLB miss before they reach the OZQ are counted.
LOAD_ANY	0x0e8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	LOAD_ANY = SUM(LOAD_INT, LOAD_FP, LOAD_RSE, LOAD_HPW)
STORE_ANY	0x0e9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	STORE_ANY = SUM(STORE_INT, STORE_FP, STORE_RSE)
LOAD_UC	0xca4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on uncacheable load instructions that retire
STORE_UC	0xca5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	data references from predicated on uncacheable store instructions that retire

4.2.8 FLI Events

This section enumerates FLI performance monitoring events.

4.2.8.1 FLI_FETCH_JIT_HIT

Description	FLI demand fetch fill hit
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x86b

Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Bundle data provided by just-in-time fill data that was not yet written to the FLI cache. Note that this event counts even when the bundles from the just-in-time bypass are not actually written into the instruction buffer due to replays or flushes.

4.2.8.2 FLI_FETCH_RAB_HIT

Description	FLI fetch RAB hit
Max Inc/Cyc	1
MT Capture Type	A
Definition	FLI access to an address that has already been requested to the MLI but it has not yet been filled.
Subevents:	
DMND	0x869
Counter Affinity	0xa0aa0
Definition	FLI demand fetch RAB hit
PREF	0x86a
Counter Affinity	0x50550
Definition	FLI prefetch fetch RAB hit

4.2.8.3 FLI_FILL

Description	FLI fills
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x85c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	A 32 byte chunk of data has been written to the FLI cache.

4.2.8.4 FLI_INST_INSERT_RAB

Description	FLI instruction bundles inserted into the instruction bundle queue
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0x872
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	The number of bundles valid out of the front end this cycle (0,1 or 2).

4.2.8.5 FLI_PREF_STALL

Description	FLI Prefetch stall
Max Inc/Cyc	1
MT Capture Type	A

Definition	-
NOTE	-
Subevents:	
ANY	0x861
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLI Prefetch stall
FLOW	0x862
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FLI Prefetch stall due to flow

4.2.8.6 FLI_PURGE

Description	FLI TLB purges
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x865
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any purge but ptc.e was detected in the IPC, an IPC miss occurred or a software insert into the IPC took place.

4.2.8.7 FLI_PVAB_OVERFLOW

Description	FLI PVAB lost
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x86c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Prefetch virtual address buffer has overflowed such that a requested stream will be lost.

4.2.8.8 FLI_RAB_ALMOST_FULL

Description	FLI RAB almost full
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x868
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	The RAB table has 6 or more entries allocated.
NOTE	If RAB entries have been deallocated for debug, those entries are marked as allocated so this event will still function.

4.2.8.9 FLI_RAB_FULL

Description	FLI RAB full
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x867
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	The RAB table has no free entries. No more MLI requests can be generated until an entry frees up.

4.2.8.10 FLI_READ

Description	FLI reads all
Max Inc/Cyc	2
MT Capture Type	A
Definition	The following sub-events describe read access to the FLI cache structures. Note that these events only increment when valid instructions are written into the instruction buffer.
Subevents:	
ANY	0x859
Counter Affinity	0xa0aa0
Definition	FLI reads all
DMND	0x85b
Counter Affinity	0xa0aa0
Definition	FLI demand fetch reads
PREF	0x85e
Counter Affinity	0x50550
Definition	FLI prefetch requests
SNOOP	0x863
Counter Affinity	0xa0aa0
Definition	FLI snoop requests
SNOOP_HIT	0x864
Counter Affinity	0x50550
Definition	FLI snoop hits

4.2.8.11 FLI_READ_MISS

Description	FLI read misses all
Max Inc/Cyc	2
MT Capture Type	A
Definition	The following sub-events account for accesses to the FLI cache that return a miss.
Subevents:	
ANY	0x85a
Counter Affinity	0x50550
Definition	FLI read misses all

DMND	0x85d
Counter Affinity	0x50550
Definition	FLI demand fetch misses
PREF	0x85f
Counter Affinity	0xa0aa0
Definition	FLI prefetch request misses

4.2.8.12 FLI_STEPPING

Description	Frontend FET pipestage is enabled
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x873
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Cycles that the frontend FET pipestage is enabled
NOTE	

4.2.8.13 FLI_STREAM_PREF

Description	FLI stream prefetch requests
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x866
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	A streaming prefetch request has been generated.
NOTE	There is one event per MLI request of the stream.

4.2.8.14 MLI_RETURN_LINE

Description	FLI L1 lines ISB
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x860
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	A 32 byte chunk of data has been returned by the MLI.
NOTE	This event includes chunks that will and will not be written to the FLI.

4.2.9 MLI Events

This section enumerates MLI performance monitoring events.

4.2.9.1 MLI_HIT_CONFLICT

Description	MLI hit conflicts
Max Inc/Cyc	1
MT Capture Type	C
Definition	The number of times a tagged request from IFR to MLI would have hit a valid line in the MLI cache, except that MLI was forced to abandon the lookup, either because the internal MLI read buffers don't have enough room left to buffer the read data (due to simultaneous higher-priority LLC/RIL fill data returns), or because a simultaneous snoop might be invalidating the line. In rare cases, this event may be signalled multiple times for a single request from IFR. The final time the request moves down the MLI pipeline, it also reports a single MLI_DEMAND/PREFETCH_HITS or MLI_DEMAND/ PREFETCH_(NO_)FILLS event, as appropriate
NOTE	
Subevents:	
ANY	0x84e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any MLI hit conflict
DMND	0x84f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Demand MLI hit conflict
PREF	0x850
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Prefetch MLI hit conflict

4.2.9.2 MLI_READ

Description	MLI reads
Max Inc/Cyc	1
MT Capture Type	A
Definition	IFR requests to MLI that result in hit or that missed
NOTE	This event does include uncacheable accesses
Subevents:	
ANY_ANY	0x83c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read all: demand or prefetch
ANY_DMND	0x83d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read all demand
ANY_PREF	0x83e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-

Definition	MLI read all prefetch
HIT_ANY	0x83f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read hit: demand or prefetch with or with out LRU Hint sent
HIT_DMND_LRU	0x842
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read hit demand with LRU Hint Sent
HIT_DMND_NOLRU	0x840
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read hit demand with no LRU Hint Sent
HIT_PREF_LRU	0x843
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read hit prefetch with LRU Hint Sent
HIT_PREF_NOLRU	0x841
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read hit prefetch with no LRU Hint Sent
MISS_ANY	0x844
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read miss: demand or prefetch
MISS_DMND	0x845
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read miss demand
MISS_PREF	0x846
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI read miss prefetch

HIT RATE = 100% * HIT_ANY/ANY_ANY

4.2.9.3 MLI_READ_UC

Description	MLI uncacheable reads
Max Inc/Cyc	1
MT Capture Type	A
Definition	The number of times a tagged demand uncacheable request from IFR to MLI misses in the MLI cache and is issued to LLC/RIL, when the data returned from LLC/RIL will only be bypassed to IFR, and will not fill into MLI.

NOTE	It is also reported for cacheable requests which have no available way to fill into as a result of MLI being deconfigured (via disabling the entire MLI cache, disabling one or more ways, or forcing individual entries into the killed state)
Subevents:	
ANY	0x847
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI UC read all
DMND	0x848
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI UC read demand
PREF	0x849
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI UC read prefetch

4.2.9.4 MLI_RECIRCULATE

Description	MLI recirculates
Max Inc/Cyc	1
MT Capture Type	C
Definition	The number of times a tagged request from IFR to MLI matches a line marked pending in the MLI cache. This means that MLI has already sent another request for this same address (but via a different RABID) to LLC/RIL, and is still waiting for its data to return. Thus, this new request cannot yet return data to IFR, as the MLI does not have the data. So the request must enter a loop of waiting for an event (snoop or fill) which might change the status of the pending line, and then reissuing through the MLI pipeline; the cycle repeats until the request no longer hits pending. Note that this event is signalled every time a request reissues, and so it might fire multiple times for a single request from IFR. The final time the request moves down the MLI pipeline, it also reports either a single MLI_DEMAND_HITS or MLI_DEMAND_MISS event.
Subevents:	
ANY	0x84a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	MLI recirculates
DMND	0x84b
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Demand MLI recirculates
PREF	0x84c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Prefetch MLI recirculates

4.2.9.5 MLI_SNOOP_HIT

Description	MLI snoop hits
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x852
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops that hit in the MLI that cause that line to be Invalidated
NOTE	Snoops are not associated with a Thread and the thread id is what the active thread was at that time. Also, snoop are not tagged, so this is for all snoops

4.2.9.6 MLI_SNOOP_INVAL_BLK_LOOKUP

Description	MLI invalidating snoop hold off lookup
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x84d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	The number of times a snoop that will invalidate a valid line block up a lookup from the IFR.
NOTE	Snoops are not associated with a Thread and the thread id is what the active thread was at that time. Also, snoop are not tagged, so this is for all snoops

4.2.9.7 MLI_SPEC_ABORT

Description	MLI aborted speculative look-up
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x851
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	The number of times a tagged change-priority-to-demand request from IFR to MLI starts down the MLI pipeline, but MLI finds that it has already been looked up, and aborts the new lookup. This can occur because, in order to satisfy demand requests as quickly as possible, MLI starts to look up a demand request without waiting to check if that request has already been handled (as is possible if it was originally issued as a prefetch, and later the IFR changes its priority to a demand)

4.2.10 FLITLB Events

This section enumerates FLITLB performance monitoring events.

4.2.10.1 FLITLB_INSERT_HPW

Description	FLI HPW insert
Max Inc/Cyc	1
MT Capture Type	C

Event Code	0x86f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Demand fetch missed in the FLITLB and caused an IPC request that resulted in an IPC miss that was filled with and HPW insert.

4.2.10.2 FLITLB_MISS

Description	FLI TLB miss
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x86d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Demand fetch missed in the FLITLB and caused an IPC lookup.

4.2.11 MLITLB Events

This section enumerates MLITLB performance monitoring events.

4.2.11.1 MLITLB_HPW_ABORTS

Description	FLI HPW abort
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x870
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	-
NOTE	-

4.2.11.2 MLITLB_MISS

Description	FLI MLI TLB miss
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0x86e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	I/-/-/-
Definition	-
NOTE	-

4.2.12 FLD Events

This section enumerates FLD performance monitoring events. Figure 0-1, “Some Asynchronous Operation Definitions” and Figure 0-2, “FLD Operation Type Definitions” contain some definitions used in the event descriptions in this section.

Table 0-1. Some Asynchronous Operation Definitions

Name	Description
a_snp	Snoop (not snoop to shared)
a_snps	Snoop to shared
a_dpf	data prefetch
a_flsh_stinv	Invalidation of an FLD cache line that was updated by a non-committing store

Table 0-2. FLD Operation Type Definitions

Op Type	instructions	async ops
FLDLD	integer loads other than ld16 and ldc.acq; RSE loads; (note: includes NATed ld.s)	none
FLDPF	lfetch	a_dpf
FLDCANFILL	FLDLD + FLDPF instructions	FLDLD + FLDPF ops
FLDST	any integer store other than st16; RSE stores	none
FLDINVLDT	stf, st16, fc, xchg, cmpxchg, fetchadd	a_snp, a_snps, a_flsh_stinv
FLDWR	FLDST + FLDINVLDT instructions	FLDST + FLDINVLDT ops

4.2.12.1 FLD_FILL

Description	FLD fill
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x101
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a cache line is written to the FLD.
NOTE	-

4.2.12.2 FLD_FILL_CANCEL

Description	FLD fill cancels
Max Inc/Cyc	1
MT Capture Type	A
Definition	This is a subset of FLD_FILL_REQS. FLD_FILL_CANCELS == FLD_FILL_REQ.ANY – FLD_FILL_CANCELS
Subevents:	
ANY	0x0fd
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R

Definition	FLD_FILL_CANCEL.ANY == (FLD_FILL_CANCEL.MLD + FLD_FILL_CANCEL.INFAB + FLD_FILL_CANCEL.POSTFAB)
INFAB	0x0ff
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times FLD fills are cancelled while in the FLD FAB. These cancellations can be caused by FLDWRs to the same cache line as the pending FLD fill, or replacement of an FLDTLB entry with which the FLD fill is associated.
MLD	0x0fe
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	counts the number of times MLD cancelled an FLD fill
POSTFAB	0x100
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times FLD fills are cancelled due to some sort of fill conflict around the time of the fill. Writes, snoops, and FLDTLB inserts around the time of the FLD fill can cause these fill cancellations.

4.2.12.3 FLD_FILL_LRU

Description	FLD fills to "not recently used" state
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x102
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	This is the subset of FLD_FILLS that fill to the "not recently used" state. FLD_FILLS – FLD_FILLED_NRU computes the FLD_FILLS that fill to the "recently used" state.

4.2.12.4 FLD_FILL_REQ

Description	FLD fill requests
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts data references (see .DATA_REF event) that request an FLD. (FLD_FILL_REQS.ANY ≤ DATA_REF.ANY). Only FLDCANFILL operations can request an FLD fill.
NOTE	-
Subevents:	
ANY	0x0f8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD_FILL_REQS.ANY == (FLD_FILL_REQS.LD_INT + FLD_FILL_REQS.LD_RSE + FLD_FILL_REQS.LFETCH + FLD_FILL_REQS.HWPF)
HW_PREF	0x0fc
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD fill requests by hardware initiated prefetches
LFETCH	0x0fb
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD fill requests by LFETCHes
LOAD_INT	0x0f9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD fill requests by integer loads (must be FLDLD to request a fill)
LOAD_RSE	0x0fa
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD fill requests by RSE loads

4.2.12.5 FLD_HINT_NOALLOC

Description	FLD no alloc hinted
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0f6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	Counts the number of times an FLD_ALLOCATE hint of NO_ALLOCATE prevents an FLD fill request.

4.2.12.6 FLD_HINT_NO_MULTI_HWPREF

Description	FLD no multi HW prefetch hinted
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0d8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	-
NOTE	-

4.2.12.7 FLD_HIT

Description	FLD hits
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0ef
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	Counts the data references (see .DATA_REF event description) that hit in the FLD cache. (FLD_HITS ≤ .DATA_REF.ANY) Note that only data references that are FLDLDs, FLDPFs, or FLDWRs can hit in the FLD. Note that snoops, fc instructions, and a_flsh_stinv async ops are not considered data references. See FLD_SPEC_INVLDTS to count non-data reference FLDINVLDTs.

4.2.12.8 FLD_LINE_DEMOTE

Description	FLD line demoted to NRU
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0x0f7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLDLDs, FLDSTs, and non-hardware initiated FLDPFs that have an FLD_LOCALITY hint of MARK_NRU and hit in the FLD cause the accessed line to be marked "not recently used" (instead of marking them most recently used). This event counts how many lines are marked NRU in this manner.

4.2.12.9 FLD_LOAD

Description	FLD loads
Max Inc/Cyc	2
MT Capture Type	A
Definition	FLD_LOAD is the subset of DATA_REF that is composed of FLD LDs. (FLD_LOAD ≤ DATA_REF).
Subevents:	
ANY	0x0ea
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD loads
INT	0x0eb
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD integer loads

4.2.12.10 FLD_LOAD_MISS

Description	FLD load misses
Max Inc/Cyc	2
MT Capture Type	A
Definition	FLD_LOAD_MISS is the subset of FLD_LOAD that missed the FLD. (FLD_LOAD_MISS ≤ FLD_LOAD)
Subevents:	
ANY	0x0ec
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD_LOAD_MISSES_ANY == FLD_LOAD_MISSES.INT + FLD_LOAD_MISSES.RSE
INT	0x0ed
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	The subset of FLD_LOAD_MISSES that comes from integer loads.
RSE	0x0ee
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	The subset of FLD_LOAD_MISSES that comes from RSE loads.

4.2.12.11 FLD_SPEC_INVAL

Description	Operations that invalidate the FLD
Max Inc/Cyc	2
MT Capture Type	A
Definition	This operation counts all FLDINVLDT operations that invalidate the FLD, including speculative invalidates due to non-retiring instructions.
NOTE	FLD_SPEC_INVAL events that occur simultaneous with and on the same M-port as FLD_SPEC_INVAL.FLUSH_STORE events will not be counted.
Subevents:	
ANY	0x0f0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD_SPEC_INVAL.ANY = (FLD_SPEC_INVAL.INSTR + FLD_SPEC_INVAL.FLUSH_STORE + FLD_SPEC_INVAL.SNOOP)
FLUSH_STORE	0x0f2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD invalidations caused by flushed stores.
INST	0x0f1

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	FLD invalidations caused by FLDINVLDT instructions.
SNOOP	0x0f3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/D/R
Definition	FLD invalidations caused by snoops.

4.2.13 MLD Events

This section enumerates MLD performance monitoring events.

4.2.13.1 MLD_BWMODE_CYC

Description	Cycles MLD is in bandwidth mode
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xcd6
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	-
NOTE	-

4.2.13.2 MLD_BYPASS

Description	MLD op that successfully bypassed
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd4
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 2
Definition	Counts the number of ops that bypassed around the OZQ. Only ops that were allowed to bypass are counted.
NOTE	Since ops are bypassed speculatively, this may count ops that do not retire.

4.2.13.3 MLD_BYPASS_ATTEMPT

Description	MLD op that attempted to bypass
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd3
Counter Affinity	0x49200

IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 2
Definition	Counts the number of ops that started to bypass around the OZQ due to the OZQ being empty. May count ops that are not allowed to bypass due to ordering or structural constraints.
NOTE	Since ops are bypassed speculatively, this may count ops that do not retire.

4.2.13.4 MLD_CYC_STALL

Description	Any MLD stall
Max Inc/Cyc	1
MT Capture Type	F
Definition	Counts and categorizes the causes of MLD pipeline stalls.
Subevents:	
ANY	0xcd8
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Any MLD stall
CRIT_BYP	0xcdd
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Return port conflict between P0 and a critical bypass.
FILL_W	0xcdc
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Conflict between fill and older st (P0 only)
HPW	0xcdf
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	MLD stalled due to back to back HPW loads
RAW	0xcda
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	RAW hazard between a multi-bank read on P0 and an older store
RW_BANK	0xcd9
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Bank conflict between a read on P0 and an older store
SEMAPHORE	0xcdb

Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Semaphore stall (P0 only)
SPLIT_CRIT_BYP	0xce6
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Return port conflict between P1 and a critical bypass.
SPLIT_OVERSUB	0xce5
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Return port conflict between P0 and P1
SPLIT_RAW	0xce4
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	RAW hazard between a multi-bank read P1 and an older store
SPLIT_RR_BANK	0xce2
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Bank conflict between reads on P0 and P1
SPLIT_RW_BANK	0xce3
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Bank conflict between a read on P1 and an older store
SPLIT_WW_BANK	0xce1
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Bank conflict between writes on P0 and P1
TAG_ERR	0xce0
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Tag error correction stall
WB_FIFO	0xcde
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Write buffer FIFO full stall

4.2.13.5 MLD_FAB_COUNT

Description	Number of valid FAB entries
Max Inc/Cyc	1
MT Capture Type	C
Definition	Counts the number of valid FAB entries per cycle
NOTE	The maximal number is 16 entries The full count is computed as follows: MLD_FAB_COUNT = MLD_FAB_COUNT.MSB << 1 + MLD_FAB_COUNT.LSB
Subevents:	
LSB	0xcf7
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid FAB entries
MSB	0xcf8
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid FAB entries

4.2.13.6 MLD_FAB_OVERFLOW

Description	MLD FAB overflowed
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd1
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	-
NOTE	-

4.2.13.7 MLD_FILL_MESI_STATE_BUDDY

Description	MLD fill op to non-crit buddy
Max Inc/Cyc	1
MT Capture Type	F
Definition	Counts the number of fills for a cacheable MLD miss.
NOTE	Does not count "no-allocate" fills.
Subevents:	
ANY	0xcc4
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to non-crit buddy
E	0xcc5

Counter Affinity	0x22200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to non-crit buddy with MESI=E
I	0xcc7
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to non-crit buddy with MESI=I
S	0xcc6
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to non-crit buddy with MESI=S

4.2.13.8 MLD_FILL_MESI_STATE_PRIMARY

Description	MLD fill op to crit
Max Inc/Cyc	1
MT Capture Type	F
Definition	-
NOTE	-
Subevents:	
ANY	0xcbf
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to crit buddy
E	0xcc1
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to crit buddy with MESI=E
I	0xcc3
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to crit buddy with MESI=I
M	0xcc0
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to crit buddy with MESI=M
S	0xcc2
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD fill op to crit buddy with MESI=S

4.2.13.9 MLD_HINT_DEFER

Description	MLD ld.s w/miss deferred due to hint
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd0
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of retired loads that were spontaneously deferred on an MLD miss.

4.2.13.10 MLD_HINT_NOALLOC

Description	MLD no-allocate fill due to a hint
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xccb
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a cacheable load, store or semapohore was not filled into the MLD due to an explicit "no allocate" hint

4.2.13.11 MLD_HINT_NO_BUDDY

Description	MLD fill w/o buddy due to a hint
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xccd
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times an implicit buddy prefetch was suppressed on a load, store or semaphore miss to the RIL due to an explicit "no buddy prefetch" hint.

4.2.13.12 MLD_HINT_PREF_DROP

Description	MLD lfetch w/miss dropped due to hint
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xccf
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a prefetch was dropped by the MLD on an MLD miss due to an explicit "drop prefetch on MLD miss" hint.

4.2.13.13 MLD_HINT_NRU

Description	MLD NRU fill due to a hint
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xccc
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a line was filled into the MLD but not marked as used due to an explicit "NRU" hint.

4.2.13.14 MLD_ISSUE_SRC

Description	Valid MLD op by queue source
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts and categorizes valid MLD ops by queue source.
NOTE	Only counts ops that "retire" in MLD pipeline. Won't count bypassed but aborted ops. Scrubs are issued from OZQ despite not being inserted into the OZQ.
Subevents:	
ANY	0xca7
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Any Valid MLD op
BYPASS	0xca8
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Valid Bypassed MLD op
FAB	0xcab
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Valid MLD op issued from FAB
OZQ	0xca9
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Valid MLD op issued from OZQ or a scrub
SMQ	0xcaa
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Valid MLD op issued from SMQ

SNOOP	0xcac
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Valid MLD op issued from SNQ

4.2.13.15 MLD_LOAD

Description	Valid MLD load issued from OZQ/bypass
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts and categorizes the number of load ops issued from the OZQ or bypassed around it.
NOTE	Counts "pure" loads only, not semaphores and not prefetches.
Subevents:	
ANY	0xcb4
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any Valid MLD load issued from OZQ/bypass
HIT	0xcb5
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD load that hit MLD issued from OZQ/bypass
MISS	0xcb6
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD load that missed MLD issued from OZQ/bypass
PRIMARY	0xcb7
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD load that missed the MLD and was a primary miss issued from OZQ/bypass
SECONDARY	0xcb8
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD load that missed the MLD and was a secondary miss issued from OZQ/bypass

4.2.13.16 MLD_LOST_BW

Description	Invalid ops or stalls
Max Inc/Cyc	2
MT Capture Type	C
Definition	Counts and categorizes the number of potential MLD ops that could have "retired" out of the MLD pipeline but didn't.
NOTE	
Subevents:	

ANY	0xce7
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Invalid ops or stalls
FAB_NOP	0xcef
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Ready FAB entry did not issue
NOP	0xcf4
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	No valid ops for MLD
NOP_STALL	0xcf2
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	MLD stall on invalid op
OZQ_FAB_FULL	0xcea
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Full FAB/WLB prevented valid OZQ entries from issuing
OZQ_FAB_PREEMPT	0xcec
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	FAB prevented valid OZQ entries from issuing
OZQ_NOP	0xce8
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Ordering or hazards prevented valid OZQ entries from issuing
OZQ_NOP_ACQ	0xce9
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Outstanding acquire prevented valid OZQ entries from issuing
OZQ_SMQ_FULL	0xceb
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1

Definition	Full SMQ prevented valid OZQ entries from issuing
OZQ_SMQ_PREEMPT	0xced
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	SMQ prevented valid OZQ entry from issuing
OZQ_SNOOP_PREEMPT	0xcee
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Snoop prevented valid OZQ entry from issuing
SMQ_NOP	0xcf0
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Ready SMQ entry did not issue
SPLIT_BUBBLE	0xcf1
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	MLD split stall forced P0 to be invalid
STALL	0xcf3
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	MLD stall prevented valid ops from retiring

4.2.13.17 MLD_NOALLOC_CASTOUT

Description	MLD no-allocate castout
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xcca
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a write-back transaction was issued to the RIL on behalf of a no-allocate store or semaphore.
NOTE	

4.2.13.18 MLD_NOALLOC_FILL

Description	MLD no-allocate fill
Max Inc/Cyc	1
MT Capture Type	C

Event Code	0xcc9
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of times a cacheable load, store or semaphore miss was not filled into the MLD, due to either a no-allocate hint or due to microarchitectural reasons.
NOTE	

4.2.13.19 MLD_OZDATA_COUNT

Description	Number of valid OZDATA entries
Max Inc/Cyc	3
MT Capture Type	C
Definition	Counts the number of valid OZDATA buffer entries per cycle
NOTE	The maximal number is 32 entries The full count is computed as follows: MLD_OZDATA_COUNT = MLD_OZDATA_COUNT.MSB << 2 + MLD_OZDATA_COUNT.LSB
Subevents:	
LSB	0xcfe
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid OZDATA entries
MSB	0xcff
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid OZDATA entries

4.2.13.20 MLD_OZQ_COUNT

Description	Number of reserved OZQ entries
Max Inc/Cyc	1
MT Capture Type	C
Definition	Counts the number of occupied OZQ entries per cycle
NOTE	The maximal number is 16 entries The full count is computed as follows: MLD_OZQ_COUNT = MLD_OZQ_COUNT.MSB << 1 + MLD_OZQ_COUNT.LSB
Subevents:	
LSB	0xcf5
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of reserved OZQ entries
MSB	0xcf6
Counter Affinity	0x24900

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of reserved OZQ entries

4.2.13.21 MLD_OZQ_INSERT

Description	MLD op inserted into OZQ (incl. bypasses)
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd2
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 2
Definition	Counts the number of ops inserted into the OZQ
NOTE	This will count bypassed ops and it can also count ops that do not retire. It will not count integer loads that are satisfied out of the FLD.

4.2.13.22 MLD_OZQ_PREEMPTED

Description	Valid OZQ op was preempted by the SMQ, FAB or SNQ
Max Inc/Cyc	2
MT Capture Type	C
Event Code	0xcd5
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Counts the number of ops that were ready to issue out of the OZQ but could not due to ready entries in the SMQ, FAB or SNQ.
NOTE	

4.2.13.23 MLD_REF

Description	Valid MLD memory reference issued from OZQ/bypass
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts and categorizes the number of valid MLD ops that constitute a memory reference issued by the OZQ or bypassed around it. A memory reference consists of lfetches, hardware data prefetches, loads, stores and semaphores.
NOTE	Does not count flushes
Subevents:	
ANY	0xcad
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any Valid MLD prefetch, ld, st or sem issued from OZQ/bypass
HIT	0xcae
Counter Affinity	0x24900

IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that hit the MLD issued from OZQ/bypass
MISS	0xcaf
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that missed the MLD issued from OZQ/bypass
PRIMARY	0xcb0
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that missed the MLD and was a primary miss issued from OZQ/bypass
SECONDARY	0xcb1
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that missed the MLD was a secondary miss issued from OZQ/bypass
SECONDARY_DROP	0xcb2
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD prefetch, dropped due to secondary miss issued from OZQ/bypass
UC_WC_STORE	0xcb3
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD st to UC or WC space (required to have been issued from OZQ)

4.2.13.24 MLD_SMQ_COUNT

Description	Number of valid SMQ entries
Max Inc/Cyc	1
MT Capture Type	C
Definition	Counts the number of valid SMQ entries per cycle
NOTE	Max number is 16 entries The full count is computed as follows: MLD_SMQ_COUNT = MLD_SMQ_COUNT.MSB << 1 + MLD_SMQ_COUNT.LSB
Subevents:	
LSB	0xcf9
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid SMQ entries
MSB	0xcfa
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Number of valid SMQ entries

4.2.13.25 MLD_SMQ_PRIORITY

Description	MLD SMQ has priority over RIL issue
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xcd7
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Counts the number of cycles a transaction in FAB is prevented from issuing to the RIL due to "priority" entries in SMQ.
NOTE	Priority mechanism is needed to preserve proper ordering within the SMQ.

4.2.13.26 MLD_SMQ_REF

Description	Valid MLD memory reference issued from SMQ
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts and categorizes the number of valid MLD ops that constitute a memory reference issued by the SMQ. A memory reference consists of lfetches, hardware data prefetches, loads, stores and semaphores.
NOTE	Does not count flushes
Subevents:	
ANY	0xcb9
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any Valid MLD memory reference issued from SMQ
HIT	0xcba
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that hits the MLD issued from SMQ
MISS	0xcbb
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that misses the MLD issued from SMQ
PRIMARY	0xcbc
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that misses the MLD and is a primary miss issued from SMQ
SECONDARY	0xcbd
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD memory reference that misses the MLD and is a secondary miss issued from SMQ
SECONDARY_DROP	0xcbe

Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Valid MLD prefetch, dropped due to secondary miss issued from SMQ

4.2.13.27 MLD_SNOOP_DEFER

Description	Deferred MLD snoop
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xcc8
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Counts the number of snoops whose completion was deferred due to a match in the FAB.
NOTE	Only FAB entries that have received a GO will cause a snoop to be deferred.

4.2.13.28 MLD_WCB_CREDIT

Description	Number of reserved WCB entries
Max Inc/Cyc	8
MT Capture Type	C
Event Code	0xcfd
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 7, P = 1
Definition	Counts the number of WCB credits available per cycle
NOTE	Max number is 8 credits

4.2.13.29 MLD_WLB_COUNT

Description	Number of reserved WLB entries
Max Inc/Cyc	8
MT Capture Type	C
Definition	Counts the number of reserved WLB entries per cycle
NOTE	Max number is 32 entries The full count is computed as follows: MLD_WLB_COUNT = MLD_WLB_COUNT.MSB << 2 + MLD_WLB_COUNT.LSB
Subevents:	
LSB	0xcfb
Counter Affinity	0x92400
IP-EAR	L = 7, P = 1
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Number of reserved WLB entries
MSB	0xcfc
Counter Affinity	0x24900

IP-EAR	L = 7, P = 1
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Number of reserved WLB entries

4.2.14 FLDTLB Events

This section enumerates FLDTLB performance monitoring events.

4.2.14.1 FLDTLB_INS_REQ

FLDTLB_INS_REQ events can only be triggered by FLDCANFILL operations (see the FLD Events section A.2.12). New requests will not be triggered until previous requests are finished. Therefore, secondary FLDTLB misses and possibly some primary FLDTLB misses will not trigger this event.

Description	FLDTLB insert requests
Max Inc/Cyc	1
MT Capture Type	A
Definition	Counts the number of times a transfer from DTB transfer to FLDTLB is requested.
NOTE	The overall event is not measured. Instead, sub-events that break down the overall event in two ways are measured. The overall event can thus be calculated from the sub-events in two ways: FLDTLB_INS_REQ.ANY = FLDTLB_INS_REQ.COMPLETE + FLDTLB_INS_REQ.CANCEL ; FLDTLB_INS_REQ.ANY = FLDTLB_INS_REQ.RETIRED + FLDTLB_INS_REQ.NON_RETIRED
Subevents:	
ANY	Calculated, not measured.
Definition	The ANY event is not measured. Instead, sub-events that break down the overall event in two ways are measured. The ANY event can thus be calculated from the sub-events in two ways: FLDTLB_INS_REQ.ANY = FLDTLB_INS_REQ.COMPLETE + FLDTLB_INS_REQ.CANCEL ; FLDTLB_INS_REQ.ANY = FLDTLB_INS_REQ.RETIRED + FLDTLB_INS_REQ.NON_RETIRED
CANCEL	0x109
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 5, P = 0
Definition	DTB transfer was cancelled.
COMPLETE	0x106
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 5, P = 0
Definition	DTB transfer completed successfully.
NON_RETIRED	0x108
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 5, P = 0
Definition	DTB transfer request came from something other than a retired instruction.
RETIRED	0x107
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 5, P = 0
Definition	DTB transfer request came from retired instructions

4.2.14.2 FLDTLB_LOAD_MISS

Description	FLD TLB load miss
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts FLDLD operations that miss the FLDTLB.
NOTE	Note that secondary FLDTLB misses are counted. An FLDTLB load miss rate (including secondary misses) can be calculated as follows: FLDTLB_LOAD_MISS / FLD_LOAD.
Subevents:	
ANY	0x103
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 6, P = 0
Definition	Counts any FLDLD operation that misses the FLDTLB.
INT	0x104
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 6, P = 0
Definition	Counts integer FLDLD operations that miss the FLDTLB
RSE	0x105
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	Y/Y/Y/Y
IP-EAR	L = 6, P = 0
Definition	Counts RSE loads that miss the FLDTLB.

4.2.15 MLDTLB Events

This section enumerates MLDTLB performance monitoring events.

4.2.15.1 DTLB_HPWHINT_BLK

Description	speculative blocking miss launches hpw req
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0xca3
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 0
Definition	This per-port signal indicates that what would have been a speculative non-blocking HPW request has been turned into a blocking HPW request due either to DAHR hints or the itlb.ed or psr.it being low. Only asserted for one port or the other since a M0 miss causes the M1 instruction to not be reached and hence not be able to do a blocking request on the same clock. The speculative instruction includes both ld.s and lfetch.none.

4.2.15.2 DTLB_HPWREQ_BLK_MISS

Description	blocking inst misses DTB, launches HPW request
Max Inc/Cyc	2
MT Capture Type	A
Definition	This per-port signal indicates that a blocking HPW request has been made. This signal is asserted when the instruction which misses the DTB is reached, so therefore it will be asserted for only one port at a time, since if the M0 instruction misses, M1 will not be reached. This count includes RSE ops.
Subevents:	
SUCCEED	0xc9c
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 0
Definition	blocking inst misses DTB, HPW walk succeeded
COAL	0xc9f
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 0
Definition	Blocking walk missed the DTB, coalesced with non-blocking walk
FAIL	0xca0
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 0
Definition	Blocking walk missed the DTB, HPW walk failed

4.2.15.3 DTLB_HPWREQ_SPEC_MISS

Description	speculative inst commits and misses the DTB
Max Inc/Cyc	2
MT Capture Type	A
Event Code	0xc9e
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 0
Definition	This per-port signal indicates that a ld.s instruction was valid, missed the DTB, and committed. A non-blocking walk may or may not have been taken since the HPW will cancel non-blocking walk requests if a non-blocking request is already busy. This signal can be asserted for both ports on the same clock.

4.2.15.4 DTLB_REF

Description	Committed instruction used the DTB, no spec lfetch, ld.s with spontaneous defer and dcr.dm = 1
Max Inc/Cyc	2
MT Capture Type	A

Definition	This is a per-port signal which is asserted for instructions which are translated by the DTB. It does not include hardware prefetches, lfetch instructions issued to the 'A' pipe, or the actual VHPT request. It does not included instructions issued in the physical mode (psr.dt or psr.rt == 0) which bypass the DTB. It does include the tpa, tak and probe instructions even if psr.dt == 0. These events include RSE ops as well.
Subevents:	
ANY	0xca6
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 1
Definition	Committed instruction used the DTB
NONSPEC	0xc9d
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 7, P = 1
Definition	Committed instruction used the DTB, no spec lfetch, ld.s with spontaneous defer and dcr.dm=1

4.2.16 DPF Events

This section enumerates DPF performance monitoring events.

There are 4 main event groupings:

- ASB events, which report on actions that invalidate hints stored in the ASB
- DPF dequeue events, which report on prefetches issued out of the DPFQ
- DPF enqueue events, which report on prefetches inserted into the DPFQ
- FLD events, which report on actions that FLD takes on behalf of the prefetcher

4.2.16.1 DAHS_UNDERFLOW

Description	Data prefetch ASB stack underflowed
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x0c9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Because the ASB contains a finite number of entries (8), a sequence of more than 7 calls will overflow the stack and overwrite the oldest DAHS stack frame. This event counts when a subsequent return underflows, detects that it has no valid DAHS stack frame to reload the DAHRs from, and must reinitialize the DAHRs to their default values instead.

4.2.16.2 DPFQ_DEQ

Description	Data prefetch queue dequeue
Max Inc/Cyc	2
MT Capture Type	A

Definition	This event reports when a prefetch is issued out of the DPFQ (with normal priority), and is inserted by FLD into the pipeline. The subevents can select only certain types of prefetches to be reported. See the following events instead for counts of prefetches that are issued with preempt priority and/or which are rejected by FLD.
NOTE	A dequeue event is reported for each individual prefetch. Thus, for example, if there is a single counted lfetch entry in the DPFQ which issues 7 fetches, then 7 dequeue events will be reported.
Subevents:	
ANY	0x0a3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue (includes all of the following subevents)
FLD_ANY	0x0ab
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue any FLD-initiated prefetch (includes FLD_TGT, FLD_FWD, FLD_BWD, and FLD_BIDI subevents)
FLD_BIDI	0x0af
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue FLD bidirectional
FLD_BWD	0x0ae
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue FLD backward
FLD_FWD	0x0ad
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue FLD forward
FLD_TARGET	0x0ac
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue FLD target
INST_ANY	0x0a4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue any instruction-generated prefetch (includes LFETCH, LFETCH_CNT, and MV_BSPST subevents)
LFETCH	0x0a5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-

IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue lfetch
LFETCH_COUNT	0x0a6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue counted lfetch
MLD	0x0b0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue MLD hardware prefetch
MOV_BSPST	0x0a7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue move to bspstore
RSE_ANY	0x0a8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue any RSE-initiated prefetch (includes RSE_LD and RSE_ST subevents)
RSE_LOAD	0x0a9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue RSE load
RSE_STORE	0x0aa
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue RSE store

4.2.16.3 DPFQ_DEQ_PREEMPT

Description	Data prefetch queue dequeue with preempt attribute
Max Inc/Cyc	2
MT Capture Type	A
Definition	This is like the DPFQ_DEQUEUE event, except that it reports prefetches that are issued with preempt priority. Note that only software prefetches may preempt, and they begin requesting preemption after they have been rejected a configurable number of times. After a hardware prefetch has been rejected for that number of times, it signals a time-out event and gets dropped.
NOTE	A dequeue event is reported for each individual prefetch. Thus, for example, if there is a single counted lfetch entry in the DPFQ which issues 7 fetches, then 7 dequeue events will be reported.
Subevents:	
ANY	0x0b1

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute (includes all of the following subevents)
INST_ANY	0x0b2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute any instruction (includes LFETCH, LFETCH_CNT, and MV_BSPST subevents)
LFETCH	0x0b3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute lfetch
LFETCH_COUNT	0x0b4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute counted lfetch
MOV_BSPST	0x0b5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute move to bspstore
TIMEOUT	0x0b6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt time-out

4.2.16.4 DPFQ_DEQ_PREEMPT_REJECT

Description	Data prefetch queue dequeue with preempt attribute rejected
Max Inc/Cyc	2
MT Capture Type	A
Definition	This is like the DPFQ_DEQ_PREEMPT event, except that it counts prefetch requests that are rejected by FLD instead of accepted into the pipeline.
NOTE	A single fetch can generate multiple reject events, one for each time it attempts to issue into the pipeline but gets rejected.
Subevents:	
ANY	0x0c5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1

Definition	Data prefetch queue dequeue with preempt attribute rejected (includes all of the following subevents)
LFETCH	0x0c6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute rejected lfetch
LFETCH_COUNT	0x0c7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute rejected counted lfetch
MOV_BSPST	0x0c8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue with preempt attribute rejected move to bspstore

4.2.16.5 DPFQ_DEQ_REJECT

Description	Data prefetch queue dequeue rejected
Max Inc/Cyc	2
MT Capture Type	A
Definition	This is like the DPFQ_DEQUEUE event, except that it counts prefetch requests that are rejected by FLD instead of accepted into the pipeline.
NOTE	A single fetch can generate multiple reject events, one for each time it attempts to issue into the pipeline but gets rejected.
Subevents:	
ANY	0x0b7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected (includes all of the following subevents)
FLD_ANY	0x0bf
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected any FLD-initiated prefetch (includes FLD_TGT, FLD_FWD, FLD_BWD, and FLD_BIDI subevents)
FLD_BIDI	0x0c3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected FLD bidirectional
FLD_BWD	0x0c2
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected FLD backward
FLD_FWD	0x0c1
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected FLD forward
FLD_TARGET	0x0c0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected FLD target
INST_ANY	0x0b8
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected any instruction (includes LFETCH, LFETCH_CNT, and MV_BSPST subevents)
LFETCH	0x0b9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected lfetch
LFETCH_COUNT	0x0ba
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected counted lfetch
MLD	0x0c4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected MLD hardware prefetch
MOV_BSPST	0x0bb
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected move to bspstore
RSE_ANY	0x0bc
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected any RSE (includes RSE_LD and RSE_ST subevents)

RSE_LOAD	0x0bd
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected RSE load
RSE_STORE	0x0be
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/-/-
IP-EAR	L = 6, P = 1
Definition	Data prefetch queue dequeue rejected RSE store

4.2.16.6 DPFQ_ENQ

Description	Data prefetch queue enqueue
Max Inc/Cyc	2
MT Capture Type	A
Definition	This event reports when a prefetch is inserted into the DPFQ (and the DPFQ has room for it). The subevents can select only certain types of prefetches to be reported. See the following event instead for counts of prefetches that are inserted into the DPFQ when it is full and thus must kick out an old entry.
NOTE	Only a single enqueue event is reported when an entry is inserted into the DPFQ, even if it will expand into multiple individual fetches. Thus, for example, if an entry is inserted into the DPFQ for a counted lfetch which will issue 7 fetches, then only 1 enqueue event will be reported.
Subevents:	
ANY	0x087
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue (includes all of the following subevents)
FLD_ANY	0x08f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue any FLD-initiated prefetch (includes the FLD_TGT, FLD_FWD, FLD_BWD, and FLD_BIDI subevents)
FLD_BIDI	0x093
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue FLD bidirectional
FLD_BWD	0x092
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue FLD backward
FLD_FWD	0x091

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue FLD forward
FLD_TARGET	0x090
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue FLD target
INST_ANY	0x088
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue any instruction (includes the LFETCH, LFETCH_CNT, and MV_BSPST subevents)
LFETCH	0x089
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue lfetch
LFETCH_COUNT	0x08a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue counted lfetch
MLD	0x094
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue MLD hardware prefetch. WARNING: THE THREAD ASSIGNMENT FOR THIS EVENT IS FAIRLY RANDOM.
MOV_BSPST	0x08b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue move to bspstore
RSE_ANY	0x08c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue any RSE (includes the RSE_LD and RSE_ST subevents)
RSE_LOAD	0x08d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2

Definition	Data prefetch queue enqueue RSE load
RSE_STORE	0x08e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue RSE store

4.2.16.7 DPFQ_ENQ_OVERFLOW

Description	Data prefetch queue enqueue overflow
Max Inc/Cyc	2
MT Capture Type	A
Definition	This is like the DPFQ_ENQUEUE event, except that it reports on prefetches inserted into the DPFQ when the DPFQ is full, indicating that the oldest existing entry was deleted to make room. The subevent still reports the type of prefetch being inserted; no indication is available for the type of the old prefetch being deleted.
NOTE	Only a single enqueue event is reported when an entry is inserted into the DPFQ, even if it will expand into multiple individual fetches. Thus, for example, if an entry is inserted into the DPFQ for a counted lfetch which will issue 7 fetches, then only 1 enqueue event will be reported.
Subevents:	
ANY	0x095
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow (includes all of the following subevents)
FLD_ANY	0x09d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow any FLD-initiated prefetch (includes FLD_TGT, FLD_FWD, FLD_BWD, and FLD_BIDI subevents)
FLD_BIDI	0x0a1
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow FLD bidirectional
FLD_BWD	0x0a0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow FLD backward
FLD_FWD	0x09f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow FLD forward

FLD_TARGET	0x09e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow FLD target
INST_ANY	0x096
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow any instruction (includes LFETCH, LFETCH_CNT, and MV_BSPST subevents)
LFETCH	0x097
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow lfetch
LFETCH_COUNT	0x098
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow counted lfetch
MLD	0x0a2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow MLD hardware prefetch
MOV_BSPST	0x099
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow move to bspstore
RSE_ANY	0x09a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow any RSE (includes RSE_LD and RSE_ST subevents)
RSE_LOAD	0x09b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow RSE load
RSE_STORE	0x09c
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 2
Definition	Data prefetch queue enqueue overflow RSE store

4.2.16.8 FLD_HWPREF_INS

Description	Any FLD HW prefetch inserted
Max Inc/Cyc	2
MT Capture Type	A
Definition	This event provides more detail on an FLD type hardware prefetch that is enqueued in the DPFQ.
Subevents:	
ACQ_PEND	0x0d3
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to acquire pending
ANY	0x0cb
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any FLD HW prefetch inserted (includes all of the following subevents)
CANCEL_FILL	0x0cc
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to cancelled fill
DTLB_MISS	0x0cd
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to DTB miss
DTLB_MISS_LFETCH	0x0d5
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to lfetch DTB miss
FLDTLB_MISS	0x0ce
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to FLD TLB miss
FLDTLB_MISS_LFETCH	0x0d6
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to lfetch FLD TLB miss
FLUSH_STORE	0x0d2
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to flushed store
NEIGHBOR	0x0cf

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to neighbor line
OZQ_FULL	0x0d1
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to OZQ full
OZQ_FULL_LFETCH	0x0d7
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to lfetch OZQ full
REL_OP	0x0d4
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to release op
STORE_ALIAS	0x0d0
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
Definition	FLD HW prefetch inserted due to store alias

4.2.16.9 PREF_DROP

Description	HW prefetch dropped
Max Inc/Cyc	2
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
DTLB_MISS	0x0da
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	This event reports when a potential prefetch is not inserted into the DPFQ because it missed the DTB.
FLDTLB_MISS	0x0d9
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	This event reports when a potential prefetch is not inserted into the DPFQ because it missed the FLD TLB.
FLD_HIT	0x0db
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0

Definition	This event reports when a potential prefetch is not inserted into the DPFQ because it hit the FLD. This event also includes prefetches dropped due to address NAT consumption.
FLD_SECONDARY_MISS	0x0dc
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 0
Definition	HW prefetch dropped due to secondary miss

4.2.17 RIL Events

This section enumerates RIL performance monitoring events. The following abbreviations are widely used:

- ARB = arbiter or arbitration
- AD = AD ring - transaction requests (think ADdress)
- AK = AK ring (AcKnowledge snoop responses from core to LLC logic)
- ATTR = attribute
- BGF = bubble generator fifo (lives between uncore and core clock domains)
- BL = BL ring - data (think BLock data)
- BLK = block
- BUDDY = buddy line prefetch
- BYP = bypass
- CBQ = coalescing buffer queue (queue of writes to coalescing buffer - includes UC, WC stores, and PTC requests)
- CC = cleanse cache
- CRD = code read (more properly, bias shared read; D-side can issue this as well to request a line be returned in S state even if available in E or M.)
- CRDT = credit
- DMND = demand
- DFRQ = Defer Queue (When MLD has to defer the response to a snoop because it has seen GO but is still waiting for data for use-once, RIL gets this response. It will be followed by a CMP snoop completion.)
- DRD = data read (biased to return data in E state)
- DRQ = data request queue (load, store misses, FC's)
- DQ = data queue
- E = exclusive (from MESI)
- FC = flush cache
- FCI = flush cache with .i (instruction) attribute
- FDB = MLD fill data buffer (used to collect data for MLD fill)
- FWD_PROG = forward progress
- FRQ = instruction fetch request queue
- GO = global observability
- M = modified (from MESI)

- MSB = most significant bits (of multi-event count)
- NC = non-cacheable (UC + WC)
- I = invalid (from MESI)
- INST = instruction fetch
- INVAL = invalidate
- LRUHINT = referring to hardware hint from MLD or MLI that causes LLC replacement state update
- LSB = least significant bits (of multi-event count)
- NTA = non-temporal access
- PEND = pending
- PRI = Ring interface (connects processor cores to ring)
- PTC = purge TC (sometimes synonymous with PTCG)
- PTCG = referring to purge caused by ptc.g instruction
- RECOV = recovery (PRI in recovery ramp after being throttled)
- REQ = request
- REF = reference - specifically a primary data reference
- RESP = response
- RFO = request for ownership (expects E or M state response)
- RRQ = ring request queue (includes requests sent from FRQ, DRQ, WRQ, and/or CBQ)
- RSPQ = response queue
- S = shared (from MESI)
- SEB = serial event bus (power messages, quiesce requests, etc.)
- SIBLING - snoop due to request that originated from a different core on the same socket (vs. SELF - snoop due to request from same core or EXT - snoop that originated off socket)
- SLB = snoop line buffer (buffer for snoop data due to a dirty hit)
- SNQ = snoop queue
- SRLZ = serialize
- UC = uncacheable TLB attribute
- WB = writeback TLB attribute
- WC = write coalescing TLB attribute
- WLB = write line buffer (for dirty MLD evictions and LRU hints)
- WRQ = write request queue for writebacks and LRU hints
- WRTBCK = writeback (eviction from MLD - NOT referring to WB TLB attribute)

4.2.17.1 RIL_ARB_PRI_LOST

Description	An arbiter for the AD or BL ring lost arbitration (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	C

Definition	An arbiter for the AD or BL ring lost arbitration (RIL->PRI)
Subevents:	
AD	0xc60
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	An arbiter for the AD ring lost arbitration (RIL->PRI)
AD_FWD_PROG	0xc61
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	One or more AD-ring arbiters are in fwd-prog mode (RIL->PRI)
BL	0xc62
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	An arbiter for the BL ring lost arbitration (RIL->PRI)
BL_FWD_PROG	0xc63
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	One or more BL-ring arbiters are in fwd-prog mode (RIL->PRI)

4.2.17.2 RIL_BL_WRITE

Description	Information on various data transactions sent to RIL on the BL ring(RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	C
Definition	Information on various data transactions sent to RIL on the BL ring(RIL->PRI)
Subevents:	
ANY	0xc5c
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Data sent from RIL on BL ring (RIL->PRI)
SLB	0xc5f
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Data sent from SLB on BL ring (RIL->PRI)
WLB	0xc5d
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Data sent from WLB on BL ring (RIL->PRI)
WLB_BOGUS	0xc5e
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Bogus Data transactions sent (RIL->PRI)

4.2.17.3 RIL_CBQ_EVICT

Description	CBQ is flushing due to a MLD signalling flush instruction (PRI->RIL)
Max Inc/Cyc	1
MT Capture Type	C
Definition	CBQ is flushing (PRI->RIL)
Subevents:	
FULL	0xc53
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	CBQ is flushing due to being full (PRI->RIL)
WCB_FLUSH	0xc52
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	CBQ is flushing due to a MLD signalling flush instruction (PRI->RIL)

4.2.17.4 RIL_CRDT_MLD_FDB_FULL

Description	MLD FDB is out of credits (RIL->MLD)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc6e
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLD FDB is out of credits (RIL->MLD)

4.2.17.5 RIL_CRDT_MLD_FDB_FULL_BLK

Description	MLD FDB is out of credits and data is pending (RIL->MLD)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc6f
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLD FDB is out of credits and data is pending (RIL->MLD)

4.2.17.6 RIL_CRDT_PRI_BLK

Description	Transaction issue to PRI is blocked due to lack of credits
Max Inc/Cyc	1
MT Capture Type	C
Definition	Transaction issue to PRI is blocked due to lack of credits
Subevents:	
AD_ALL	0xc64

Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AD ring is blocked for all active requestors (RIL->PRI)
AD_CBQ	0xc68
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AD ring is blocked for CBQ head (RIL->PRI)
AD_DRQ	0xc66
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AD ring is blocked for DRQ head (RIL->PRI)
AD_FRQ	0xc65
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AD ring is blocked for FRQ head (RIL->PRI)
AD_WRQ	0xc67
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AD ring is blocked for WRQ head (RIL->PRI)
AK_ALL	0xc69
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	AK ring is blocked for all active requestors (RIL->PRI)
BL_ALL	0xc6a
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BL ring is blocked for all active requestors (RIL->PRI)
BL_CBQ	0xc6d
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BL ring is blocked for CBQ head (RIL->PRI)
BL_SNQ	0xc6b
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BL ring is blocked for SNQ head (RIL->PRI)
BL_WRQ	0xc6c
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BL ring is blocked for WRQ head (RIL->PRI)

4.2.17.7 RIL_CRDT_SNQ_BLK

Description	Snoops ready but blocked for any reason into the core (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	C
Definition	-
NOTE	-
Subevents:	
ANY	0xc70
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops ready but blocked for any reason into the core (RIL->PRI)
ANY_Q_FULL	0xc7d
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No Mli/Mld Full, DFRQ/RSPQ/SLBQ/WLBQ full (RIL->CORE)
DFRQ	0xc79
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No Mli/Mld Full, no credits for MLD DfrQ (RIL->CORE)
HALT	0xc71
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> halt.dg or diag block (RIL->CORE)
MLD_FULL	0xc77
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No FwdProg, blocked for MLD max in core (RIL->CORE)
MLD_FWD_PROG	0xc74
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> MLD Fwd Prog limit (RIL->CORE)
MLI_FULL	0xc76
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No FwdProg, blocked for MLI max in core (RIL->CORE)
MLI_FWD_PROG	0xc73
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> MLI Fwd Prog limit (RIL->CORE)
MLI_OR_MLD_FULL	0xc78
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No FwdProg, blocked for MLI or MLD max in core (RIL->CORE)

MLI_OR_MLD_FWD_PROG	0xc75
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> MLI or MLD Fwd Prog limit (RIL->CORE)
RSPQ	0xc7a
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No Mli/Mld Full, no credits for PRI RspQ (RIL->CORE)
SLB_DQ	0xc7b
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No Mli/Mld Full, no credits for SLB DataQ (RIL->CORE)
SRLZ	0xc72
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> no pipelined snoops (RIL->CORE)
USEMANY_ANY	0xc96
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> blocked by UseMany
USEMANY_BYP	0xc97
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> blocked due to a snoop bypass
WLB_DQ	0xc7c
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoops blocked -> No Mli/Mld Full, no credits for WLB DataQ (RIL->CORE)

4.2.17.8 RIL_DATA_RETURN

Description	Fill Data return accounting
Max Inc/Cyc	1
MT Capture Type	C
Definition	Fill Data return accounting
Subevents:	
EARLY_FILL_EM	0xc58
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Early-fill to E/M state (PRI->RIL)
EARLY_FILL_S	0xc59
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-

Definition	Early-fill to S state (PRI->RIL)
MLD_ANY	0xc56
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Data sent to MLD (RIL->MLD)
MLD_CRIT	0xc57
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Crit-bypass data for WB returns sent to MLD (RIL->MLD)
PRI_ANY	0xc54
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Data returning from PRI (PRI->RIL)
PRI_MLD	0xc55
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLD Fill return data from PRI (PRI->RIL)

4.2.17.9 RIL_DRQ

Description	DRQ Issue Queue status events
Max Inc/Cyc	1
MT Capture Type	C
Definition	DRQ Issue Queue status events
Subevents:	
EMPTY	0xc80
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	DRQ Issue Queue is empty
LIMIT_HIT	0xc81
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	DRQ Issue Queue throttled due to hitting the issue limit
NOTE	

4.2.17.10 RIL_DRQ_PACE_BUBBLE

Description	DRQ Pacing prevented DRQ from nominating (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc8f
Counter Affinity	0x92400

IAR/OPC/DAR/DREF	I/O/D/R
Definition	-
NOTE	-

4.2.17.11 RIL_DRQ_VALID

Description	DRQ Transactions in flight on PRI
Max Inc/Cyc	8
MT Capture Type	C
Definition	DRQ Transactions in flight on PRI
NOTE	The maximal number is 32 entries The full count is computed as follows: RIL_DRQ_VALID = RIL_DRQ_VALID.MSB << 2 + RIL_DRQ_VALID.LSB
Subevents:	
LSB	0xc8a
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	DRQ Transactions in flight on PRI (count[1:0])
MSB	0xc89
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	DRQ Transactions in flight on PRI (count[5:2])

4.2.17.12 RIL_FRQ

Description	FRQ Issue Queue status events
Max Inc/Cyc	1
MT Capture Type	C
Definition	-
NOTE	-
Subevents:	
EMPTY	0xc7e
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FRQ Issue Queue is empty
LIMIT_HIT	0xc7f
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FRQ Issue Queue throttled due to hitting the issue limit
NOTE	

4.2.17.13 RIL_FRQ_VALID

Description	FRQ Transactions in flight on PRI
Max Inc/Cyc	8
MT Capture Type	C
Definition	FRQ Issue Queue events
NOTE	The maximal number is 16 entries The full count is computed as follows: RIL_FRQ_VALID = RIL_FRQ_VALID.MSB << 1 + RIL_FRQ_VALID.LSB
Subevents:	
LSB	0xc88
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FRQ Transactions in flight on PRI (count[0])
MSB	0xc87
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FRQ Transactions in flight on PRI (count[4:1])

4.2.17.14 RIL_INTERRUPT

Description	Incoming Interrupt (PRI->RIL)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc51
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Incoming Interrupt (PRI->RIL)

4.2.17.15 RIL_PRI_THROTTLE_ASSERTED

Description	AD issue throttling asserted
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc90
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	-
NOTE	-

4.2.17.16 RIL_PRI_THROTTLE_RECOV

Description	AD issue throttling is not asserted but still recovering
Max Inc/Cyc	1
MT Capture Type	C

Event Code	0xc91
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	-
NOTE	-

4.2.17.17 RIL_REQ.ANY

Description	Any PRI Request sent on the AD ring (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc12
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Any PRI Request sent on the AD ring (RIL->PRI)

4.2.17.18 RIL_REQ_HINT_NRU

Description	PRI Requests with the NRU hint (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc3b
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	-

4.2.17.19 RIL_REQ_OTHER

Description	Miscellaneous non-reference requests issued from RIL
Max Inc/Cyc	1
MT Capture Type	A
Definition	Misc. non-reference requests issued from RIL
Subevents:	
CC	0xc32
Counter Affinity	0x92400
Definition	CC operations issued to PRI (RIL->PRI)
DRQ_ANY	0xc33
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MISC DRQ transactions (FC/FCI/CC) (RIL->PRI)
FC	0xc30
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI FC transactions due to an FC instruction (RIL->PRI)

FCI	0xc31
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI FCI transactions due to an FCI instruction (RIL->PRI)
LRUHINT_ANY	0xc37
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLI or MLD LRUHints issued to PRI (RIL->PRI)
LRUHINT_FROM_MLD	0xc36
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	LRUHints issued from MLD (MLD->RIL)
LRUHINT_MISS_ANY	0xc39
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLI or MLD LRUHints that miss in LLC (PRI->RIL)
LRUHINT_MISS_MLD	0xc3a
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLD LRUHints that miss in LLC (PRI->RIL)
LRUHINT_MLD	0xc38
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD LRUHints issued to PRI (RIL->PRI)
PTCG	0xc34
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PTC.G transactions issued to PRI (RIL->PRI)
PTCG_PEND	0xc35
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	A PTC.G is outstanding (RIL)
WRQ_FC_FCI	0xc2b
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	WRQ-issued FC/FCI transactions (RIL->PRI)
WRQ_SKIP_LRUHINT	0xc2d
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ LRU hint not issued (too many outstanding) (RIL->PRI)
WRTBCK_MLD_EVICT	0xc2e
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Writebacks due to an MLD capacity eviction issued from MLD (MLD->RIL)

WRTBCK_MLD_FC	0xc2f
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Writebacks due to an FC instruction issued from MLD (MLD->RIL)
WRTBCK_WRQ	0xc2a
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Writebacks issued to PRI as WbMtoS (RIL->PRI)
WRTBCK_WRQ_SKIP	0xc2c
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ Writeback not issued (hit by snoop) (RIL->PRI)

4.2.17.20 RIL_REQ_REF.ANY

Description	Any instruction or data reference (RIL->PRI)
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0xc13
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Any instruction or data reference (RIL->PRI)

4.2.17.21 RIL_REQ_REF_DATA

Description	PRI data references (CRd, DRd, RFO, UC/WC read/write)
Max Inc/Cyc	1
MT Capture Type	A
Definition	PRI data references (CRd, DRd, RFO, UC/WC read/write)
Subevents:	
ANY	0xc18
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any PRI data ref (CRd, DRd, RFO, UC/WC read/write) (RIL->PRI)
DRQ_ANY	0xc28
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	All MLD requests that write to the DRQ (MLD->RIL)
NC_ANY	0xc20
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any PRI WC or UC data read or write (RIL->PRI)
NC_READ_ANY	0xc21
Counter Affinity	0x22200

IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any PRI WC or UC data read (RIL->PRI)
NC_READ_UC	0xc22
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI UC data reads (RIL->PRI)
NC_WRITE_ANY	0xc23
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any PRI WC or UC data write (RIL->PRI)
NC_WRITE_UC	0xc26
Counter Affinity	0x44400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI UC data writes (RIL->PRI)
NC_WRITE_WC_ANY	0xc24
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any PRI WC data write (RIL->PRI)
NC_WRITE_WC_FULL	0xc25
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI full-line WC data writes (RIL->PRI)
NC_WRITE_WC_MLD	0xc27
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD WC transaction issued from MLD (before coalescing) (MLD->RIL)
WB_ANY	0xc19
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any CRd, DRd, or RFO issued to PRI (RIL->PRI)
WB_CRD	0xc1c
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI CRd requests (data-side only) (RIL->PRI)
WB_DRD	0xc1d
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI DRd requests (RIL->PRI)
WB_MLD_ANY	0xc1a
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	cacheable MLD requests (not incl. buddy requests) (MLD->RIL)
WB_MLD_BUDDY	0xc1b
Counter Affinity	0x24900

IAR/OPC/DAR/DREF	I/O/D/R
Definition	cacheable MLD buddy requests (MLD->RIL)
WB_RFO	0xc1e
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	PRI RFO requests (RIL->PRI)
WB_SELF_SNOOP	0xc1f
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	MLD requests that hit in the WRQ (RIL)
WRQ_ANY	0xc29
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/O/D/R
Definition	All MLD requests that write to the WRQ (MLD->RIL)

4.2.17.22 RIL_REQ_REF_INST

Description	Instruction references (CRd, PRdCode)
Max Inc/Cyc	1
MT Capture Type	A
Definition	Instruction references (CRd, PRdCode)
Subevents:	
ANY	0xc14
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any instruction ref (CRd, PRdCode) (RIL->PRI)
NC	0xc15
Counter Affinity	0x22200
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any PRI UC or WC code read (RIL->PRI)
WB_ANY	0xc16
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/-/-/-
Definition	PRI WB code reads (RIL->PRI)
WB_DMND	0xc17
IAR/OPC/DAR/DREF	I/-/-/-
Counter Affinity	0x92400
Definition	PRI demand WB requests (RIL->PRI)

4.2.17.23 RIL_RESP

Description	PRI responses seen by the core (PRI->RIL)
Max Inc/Cyc	1
MT Capture Type	C

Definition	PRI responses seen by the core (PRI->RIL)
Subevents:	
GO	0xc5a
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI GO response (PRI->RIL)
WRITEPULL	0xc5b
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI Writepull response (PRI->RIL)

4.2.17.24 RIL_RRQ.LIMIT_HIT

Description	RRQ Issue Queue throttled due to hitting the issue limit
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc84
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	RRQ Issue Queue throttled due to hitting the issue limit
NOTE	

4.2.17.25 RIL_SEB

Description	SEB quiesce events
Max Inc/Cyc	1
MT Capture Type	C
Definition	-
NOTE	-
Subevents:	
BGF_QUIESCE_ACTIVE	0xc94
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BGF quiesce active
LDST_QUIESCE_PEND	0xc93
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Load/store quiesce still pending
PTC_QUIESCE_PEND	0xc92
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PTC Quiesce still pending

4.2.17.26 RIL_SHOOTDOWN

Description	Incoming shootdowns (PRI->RIL)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc4f
Counter Affinity	0x88800
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Incoming shootdowns (PRI->RIL)
NOTE	-

4.2.17.27 RIL_SHOOTDOWN_PEND_CYC

Description	Shootdown is pending (RIL)
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc50
Counter Affinity	0x11100
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Cycles of pending shootdown

4.2.17.28 RIL_SNOOP_REQ

Description	PRI snoop requests (PRI->RIL)
Max Inc/Cyc	1
MT Capture Type	C
Definition	PRI snoop requests (PRI->RIL)
Subevents:	
ANY	0xc01
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any PRI snoop request (PRI->RIL)
CODE_ANY	0xc02
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any PRI SnpCode request (PRI->RIL)
CODE_SELF	0xc03
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpCode due to request from this core (PRI->RIL)
CODE_SIBLING	0xc04
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpCode due to sibling core request (PRI->RIL)

DATA_ANY	0xc05
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any SnpData request (PRI->RIL)
DATA_SELF	0xc06
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpData due to request from this core (PRI->RIL)
DATA_SIBLING	0xc07
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpData due to sibling core request (PRI->RIL)
INVAL_ANY	0xc08
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any SnpInv request (PRI->RIL)
INVAL_LLC_EVICT	0xc0b
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpInv due to an LLC eviction (PRI->RIL)
INVAL_SELF	0xc09
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpInv due to request from this core (PRI->RIL)
INVAL_SIBLING	0xc0a
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	PRI SnpInv due to sibling core request (PRI->RIL)

4.2.17.29 RIL_SNOOP_RESP

Description	Snoop missed MLD (MLD->RIL)
Max Inc/Cyc	1
MT Capture Type	C
Definition	Core Snoop Responses (MLD->RIL)
Subevents:	
MLD_DEFER	0xc11
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	MLD Snoop was deferred (MLD->RIL)
MLD_HIT_E	0xc0f
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoop hit MLD Snoop exclusive (MLD->RIL)

MLD_HIT_M	0xc10
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoop hit MLD Snoop modified (dirty) (MLD->RIL)
MLD_HIT_S	0xc0e
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoop hit MLD Snoop shared (MLD->RIL)
MLD_MISS	0xc0c
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoop missed MLD (MLD->RIL)
WRQ_HIT_M	0xc0d
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Snoop missed MLD, hit WRQ dirty (MLD->RIL)

4.2.17.30 RIL_SNQ

Description	Snoop queue status events
Max Inc/Cyc	1
MT Capture Type	C
Definition	SNQ events
Subevents:	
EMPTY	0xc85
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	No snoops in the issue queue or active in the core
LIMIT_HIT	0xc86
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	SNQ throttled due to hitting the issue limit
NOTE	

4.2.17.31 RIL_SNQ_VALID

Description	SNQ Transactions in flight in Core
Max Inc/Cyc	15
MT Capture Type	C
Definition	SNQ Transactions in flight in Core
NOTE	The maximal number is 16 entries The full count is computed as follows: RIL_SNQ_VALID = RIL_SNQ_VALID.MSB << 1 + RIL_SNQ_VALID.LSB
Subevents:	
LSB	0xc8e

Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	SNQ Transactions in flight in Core (count[0])
MSB	0xc8d
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	-/-/-/-
Definition	SNQ Transactions in flight in Core (count[4:1])

4.2.17.32 RIL_WRQ

Description	WRQ Issue Queue is empty
Max Inc/Cyc	1
MT Capture Type	C
Definition	-
NOTE	-
Subevents:	
EMPTY	0xc82
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ Issue Queue is empty
LIMIT_HIT	0xc83
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ Issue Queue throttled due to hitting the issue limit

4.2.17.33 RIL_WRQ_VALID

Description	WRQ Transactions in flight on PRI
Max Inc/Cyc	8
MT Capture Type	C
Definition	WRQ events
NOTE	The maximal number is 32 entries The full count is computed as follows: RIL_WRQ_VALID = RIL_WRQ_VALID.MSB << 2 + RIL_WRQ_VALID.LSB
Subevents:	
LSB	0xc8c
Counter Affinity	0x94200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ Transactions in flight on PRI (count[1:0])
MSB	0xc8b
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	-/-/-/-
Definition	WRQ Transactions in flight on PRI (count[5:2])

4.2.18 RSE Events

This section enumerates RSE performance monitoring events.

4.2.18.1 RSE_CURRENT_REG

Description	RSE current registers
Max Inc/Cyc	12
MT Capture Type	A
Definition	This event counts the number of current frame registers at each RSE_EVENT_RETIRED event
NOTE	This event adds to the count the current frame marker size of frame (CFM.sof) just prior to each retirement event instead of just after. Either way can work. However, this event is dumb, because it counts two different quantities: CFM.sof and CFM.soo (size of outputs). Due to a call instruction, CFM.sof ≤ CFM.soo. At the next EVENT, that CFM.soo will be counted. The next event is likely an alloc instruction setting CFM.sof to a desired value. Also, alloc optimizations may set CFM.sof smaller prior to a call, and at the call, this will cause another count of a value not really in use. This event works the same as on prior processor models. The full count is computed as follows: RSE_CURRENT_REG = RSE_CURRENT_REG.MSB << 3 + RSE_CURRENT_REG.LSB
Subevents:	
LSB	0x154
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	RSE current registers lower bits (count[2:0])
MSB	0x153
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	RSE current registers upper bits (count[2:0])

4.2.18.2 RSE_DIRTY_REG

Description	RSE dirty registers
Max Inc/Cyc	15
MT Capture Type	A
Definition	This event counts the number of dirty registers on the register stack at each RSE_EVENT_RETIRED event.
NOTE	This event adds to the count just prior to each retirement event instead of just after. Either way can work. This event has a similar frailty to RSE_CURRENT_REGS in that any RSE_EVENT_RETIRED causes a count irrespective of whether it should. However, from examining a few examples, this event seems to perform better in allowing a calculation of the average number of dirty registers. This event works the same as on prior processor models. The full count is computed as follows: RSE_DIRTY_REG = RSE_DIRTY_REG.MSB << 4 + RSE_DIRTY_REG.LSB
Subevents:	
LSB	0x156
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	RSE dirty registers lower bits (count [3:0])

MSB	0x155
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	RSE dirty registers upper bits (count [7:4])

4.2.18.3 RSE_REF_RETIRED

Description	RSE reference retired
Max Inc/Cyc	2
MT Capture Type	A
Definition	Counts RSE HW generated loads and stores retiring/committing (sometimes called fills and spills).
NOTE	RSE generates one or two loads, or one or two stores per cycle. Uses per-instruction PMU tags (instruction address and opcode matching) on the initiating instruction (alloc, flushrs, loadrs, or return, but not rfi) which get further qualified by Data EAR matching. The PMU tag on rfi for RSE loads is always set to true, and then does get qualified with D-EAR.
Subevents:	
ANY	0x158
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	RSE reference retired
LOAD	0x159
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	RSE load reference retired
STORE	0x15a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	I/O/D/R
IP-EAR	L = 6, P = 1
Definition	RSE store reference retired

4.2.19 LLC Events

This section enumerates LLC performance monitoring events measured in the core. For detailed LLC characterization, refer to the Cbox PMU documentation.

4.2.19.1 LLC_REF_HIT

Description	Any LLC hit (PRI->RIL) (DRQ/FRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Definition	-
NOTE	-
Subevents:	

ANY	0xc3c
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Any LLC hit (PRI->RIL) (DRQ/FRQ misses only)
NO_SNOOP	0xc3d
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/o/d/r
Definition	LLC hit requiring no snoops (PRI->RIL) (DRQ/FRQ misses only)
SNOOP	0xc3e
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/o/d/r
Definition	LLC hit requiring snoops, not resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)
SNOOP_FWD	0xc3f
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/o/d/r
Definition	LLC hit requiring snoops, resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)

4.2.19.2 LLC_REF_MISS

Description	LLC misses (DRQ/FRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Definition	These events measure LLC responses to tagged events when the GO message returns from the LLC on PRI.
Subevents:	
ANY	0xc41
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Any LLC miss (PRI->RIL) (DRQ/FRQ misses only)
MEM_LCL_ANY	0xc42
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Miss to local RAM (PRI->RIL) (DRQ/FRQ misses only)
MEM_LCL_NO_SNOOP	0xc44
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Miss to local RAM requiring no snoops (PRI->RIL) (DRQ/FRQ misses only)
MEM_LCL_SNOOP	0xc45
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Miss to local RAM requiring snoops, not resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)
MEM_LCL_SNOOP_FWD	0xc46
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/o/d/r

Definition	Miss to local RAM requiring snoops, resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)
MEM_RMT_ANY	0xc43
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Miss to remote RAM (PRI->RIL) (DRQ/FRQ misses only)
MEM_RMT_NO_SNOOP	0xc47
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/o/d/r
Definition	Miss to remote RAM requiring no snoops (PRI->RIL) (DRQ/FRQ misses only)
MEM_RMT_SNOOP	0xc48
Counter Affinity	0x24900
Definition	Miss to remote RAM requiring snoops, not resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)
IAR/OPC/DAR/DREF	I/o/d/r
MEM_RMT_SNOOP_FWD	0xc49
Counter Affinity	0x49200
Definition	Miss to remote RAM requiring snoops, resulting in forwarded data (PRI->RIL) (DRQ/FRQ misses only)

4.2.19.3 LLC_REF_MISS_DATA

Description	Data read or write that misses LLC (PRI->RIL) (DRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Definition	-
NOTE	-
Subevents:	
ANY	0xc4b
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any data read or write that misses LLC (PRI->RIL) (DRQ misses only). Note that ld.bias and lfetch.excl are included.
READ	0xc4c
Counter Affinity	0x49200
IAR/OPC/DAR/DREF	I/O/D/R
Definition	Any data read that misses LLC (PRI->RIL) (DRQ misses only). Note that ld.bias and lfetch.excl data references are not included.
Notes:	LLC_REF_MISS_DATA.ANY and LLC_REF_MISS_DATA.READ, when constrained to count lfetches using data reference constraints do not return the same counts. This is due to the fact that LL_REF_MISS_DATA.READ does not count lfetch.excl nor ld.bias ops.

4.2.19.4 LLC_REF_MISS_INST

Description	Instruction miss that misses LLC (PRI->RIL) (FRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Definition	-

NOTE	-
Subevents:	
ANY	0xc4d
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any instruction miss that misses LLC (PRI->RIL) (FRQ misses only)
PRIMARY	0xc4e
Counter Affinity	0x24900
IAR/OPC/DAR/DREF	I/-/-/-
Definition	Any primary instruction miss that misses LLC (PRI->RIL) (FRQ misses only)

4.2.19.5 LLC_REF_SYS_ANY

Description	Request satisfied by non-RAM system address (PRI->RIL) (DRQ/FRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Event Code	0xc40
Counter Affinity	0x55500
IAR/OPC/DAR/DREF	I/o/d/r
Definition	-
NOTE	-

4.2.19.6 LLC_REF_UNKNOWN

Description	Hit/Miss response unknown/NA (DRQ/FRQ misses only)
Max Inc/Cyc	1
MT Capture Type	F
Event Code	0xc4a
Counter Affinity	0x92400
IAR/OPC/DAR/DREF	I/o/d/r
Definition	-
NOTE	-

4.2.20 System Events

This section enumerates System performance monitoring events.

4.2.20.1 CPU_CPL_CHANGE

Description	CPU privilege level changes
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	

ANY	0x01a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	Any CPU privilege level change
FROM0	0x01b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	CPU privilege level change from level 0
FROM1	0x01c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	CPU privilege level change from level 1
FROM2	0x01d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	CPU privilege level change from level 2
FROM3	0x01e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	CPU privilege level change from level 3

4.2.20.2 INTERRUPT_EVENT

Description	IVA interrupt event
Max Inc/Cyc	1
MT Capture Type	A
Definition	Counts the number of IVA interrupts
Note	This monitor will not work correctly unless one of the ETB modes is enabled. If the ETB/IP-EAR or MT-EAR are not used, the least intrusive way to accomplish this is by setting either PMC_ETB_CFG.plm or PMC_IPEAR_CFG.plm to a non-zero value.
Subevents:	
MASKED	0x01f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Masked IVA interrupt event
NOTE	INTERRUPT_EVENT.MASKED is further qualified by the address/match values programmed in PMC_IVAEV_CFG
UNMASKED	0x020
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 0
Definition	Unmasked IVA interrupt event

4.2.20.3 SERIALIZATION_EVENT

Description	Serialize event
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x021
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 1
Definition	-
NOTE	-

4.2.20.4 UNCORE_FREEZE

Description	UNCORE PMU freeze signal from the uncore seen
Max Inc/Cyc	1
MT Capture Type	C
Event Code	0xc9b
Counter Affinity	0xaaa00
IAR/OPC/DAR/DREF	-/-/-/-
Definition	This event counts every time the uncore PMU global enable transitions low

4.2.21 Multithreading Events

This section enumerates Multithreading performance monitoring events.

4.2.21.1 CYC_FE_FWPROG

Description	Number of cycles in forward progress screen
Max Inc/Cyc	1
MT Capture Type	F
Event Code	0x8a1
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Number of cycles in forward progress screen
NOTE	This event counts the number of cycles the frontend is deciding to lock onto a thread that is not making forward progress. The frontend measures lack of forward progress for a particular thread when it has N switchout events without a valid instruction having been delivered to the instruction buffers. See the Instruction Fetch MT section for further details on the forward progress screen.

4.2.21.2 MT_BE_BGND_CYC_IN_STATE

Description	BE background thread state
Max Inc/Cyc	1
MT Capture Type	A
Definition	Counts the number of cycles the background thread spends in a particular thread state, further constrained by the setting of PMC_BEMT_CTL.
NOTE	This event is qualified by PMC_BEMT_CTL
Subevents:	
HU	0x143
Counter Affinity	0xaaa20
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in high unstalled state
HW	0x144
Counter Affinity	0x54550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in high stalled state
LU	0x147
Counter Affinity	0xaaa20
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in low unstalled state
LW	0x148
Counter Affinity	0x54550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in low stalled state
NU	0x145
Counter Affinity	0xaaa20
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in nominal unstalled state
NW	0x146
Counter Affinity	0x54550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread in background in nominal stalled state

4.2.21.3 MT_BE_FAIR_STATE

Description	Fairness counter state
Max Inc/Cyc	1
MT Capture Type	A (GREEN)/F(YELLOW,ORANGE,RED)
Definition	Counts the number of cycles the fairness counter spends in the respective fairness state.
NOTE	“Threading” for this event reflects the state, rather than the current thread. GREEN will be measured (and qualified) by either thread, whereas YELLOW, ORANGE and RED will be associated (and counted by) the respective victimized thread.
Subevents:	
GREEN	0x149

Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness counter green state cycles
ORANGE	0x14b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness counter orange state cycles
RED	0x14c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness counter red state cycles
YELLOW	0x14a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness counter yellow state cycles

Following relationship holds true:

MT_BE_FAIR_STATE.GREEN.all +
MT_BE_FAIR_STATE.YELLOW.T0 + MT_BE_FAIR_STATE.YELLOW.T1 +
MT_BE_FAIR_STATE.ORANGE.T0 + MT_BE_FAIR_STATE.ORANGE.T1+
MT_BE_FAIR_STATE.RED.T0 + MT_BE_FAIR_STATE.RED.T1 == CPU_OP_CYCLES.all

4.2.21.4 MT_BE_FAIR_TRANSITION

Description	Fairness state transitions
Max Inc/Cyc	1
MT Capture Type	F
Definition	This monitor counts the number of state transitions
NOTE	-
Subevents:	
GRN0	0x150
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness green 0 transitions - counts 0 crossings within the green state
GRN2YLW	0x14d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness transitions green to yellow
ORN2RED	0x14f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness transitions orange to red
YLW2ORN	0x14e

Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Fairness transitions yellow to orange

4.2.21.5 MT_BE_THRSW_ACTUAL_IN

Description	BE thread switch in events
Max Inc/Cyc	1
MT Capture Type	A
Definition	BE thread switch in events
NOTE	These are events that caused the background thread to be switched into the foreground, and thus are applied/counted to the thread transitioning into the foreground.
Subevents:	
ALAT_INVAL	0x136
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to ALAT invalidate
HPWINS	0x134
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to HPW insert
IBQ_NOTEMPTY	0x135
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to IBQ not empty
LP_EXIT	0x137
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to LP exit
MLDRTN	0x133
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to MLD return
TIMEOUT	0x132
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to time-out
FAIR	0x138
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	BE thread switch in due to fairness

4.2.21.6 MT_BE_THRSW_ACTUAL_OUT

Description	BE thread switch out events
Max Inc/Cyc	1
MT Capture Type	F
Definition	BE thread switch out events
NOTE	These are events that caused the foreground thread to be switched into the background, and thus are applied/counted to the thread transitioning into the background. Summation of this thread switch event set satisfies the following relationship MT_BE_THRSW_ACTUAL_OUT.ANY.me = sum(MT_BE_THRSW_ACUTAL_OUT.*.me) + sum(MT_BE_THRSW_ACUTAL_IN.*.other)
Subevents:	
ANY	0x12a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	Any BE thread switch, due to either the foreground thread switching out or the background thread switching in. Use MT_BE_THRSW_ACTUAL_OUT.ANY, as EAR_EVENT_ETB_IP_MT does not increment
ATPAUSE	0x12e
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due to hint \\@pause
HPW_MISS	0x12c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due to HPW miss
IBQ_EMPTY	0x12d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due to IBQ empty
INJ_DBG	0x131
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due random injection/debug
LP_ENTER	0x12f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due to LP enter
MLD_USE	0x12b
Counter Affinity	0xaaaa0

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out due to mld use
RFIX	0x130
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch out for rfi.x

4.2.21.7 MT_BE_THRSW_DISABLE

Description	BE thread switch disabled cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
EXPL	0x13a
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Explicitly disabled BE thread switch cycles, i.e., at least one of the threads is in LP
IMPL	0x13b
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Implicitly disabled BE thread switch cycles, i.e., PCR.sd is 1

4.2.21.8 MT_BE_THRSW_DROP

Description	BE dropped thread switch
Max Inc/Cyc	1
MT Capture Type	A
Event Code	0x139
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	Thread switch opportunities that were dropped due to switching-table/fairness policy, full-slice mode, and in switch wait window
NOTE	-

4.2.21.9 MT_BE_THRSW_HOLD

Description	BE thread switch held cycles
Max Inc/Cyc	1
MT Capture Type	A

Event Code	0x13c
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Cycles when the switch is held up due to BE forward progress camping or other rare pipeline conditions.
NOTE	-

4.2.21.10 MT_BE_THRSW_STALL

Description	BE thread switch stall cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	-
NOTE	-
Subevents:	
ANY	0x13d
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles
CRAB	0x141
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles, switch blocked by CRAB
FLD	0x142
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles, switch blocked by FLD
PIPE	0x13f
Counter Affinity	0xaaaa0
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles, switch blocked by pipeline
RSE	0x140
Counter Affinity	0x55550
IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles, switch blocked by RSE
SWITCH	0x13e
Counter Affinity	0x55550

IAR/OPC/DAR/DREF	-/-/-/-
IP-EAR	L = 5, P = 3
Definition	BE thread switch stall cycles, switch blocked, in switch

4.2.21.11 MT_FE_BE_IN_SAME_THREAD

Description	Cycles front and back end are in same thread
Max Inc/Cyc	1
MT Capture Type	F
Event Code	0x874
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	This event counts the cycles during which front and back-end are in the same thread domain, and form a continuous pipeline.
NOTE	-

4.2.21.12 MT_FE_BGND_CYC_IN_STATE

Description	FE thread state
Max Inc/Cyc	1
MT Capture Type	F
Definition	-
NOTE	-
Subevents:	
HIGH	0x8a0
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FE thread in background with high urgency
LOW	0x89e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FE thread in background with low urgency
NOMINAL	0x89f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	FE thread in background with nominal urgency

4.2.21.13 MT_FE_THRSW_ACTUAL_IN

Description	Front end thread switch in events
Max Inc/Cyc	1
MT Capture Type	A
Definition	Front end thread switch in events
NOTE	These are events that caused the background thread to be switched into the foreground, and thus are applied/counted to the thread transitioning into the foreground.

Subevents:	
BRQ_NON_BLK	0x881
Counter Affinity	0xa0aa0
Definition	Front end thread switch in due to BRQ non blocking
IBQ_EMPTY	0x87a
Counter Affinity	0x50550
Definition	Front end thread switch in due to IBQ empty
IBQ_NOTFULL	0x879
Counter Affinity	0xa0aa0
Definition	Front end thread switch in due to IBQ not full
MLI_UCRTN	0x87f
Counter Affinity	0xa0aa0
Definition	Front end thread switch in due to MLI UC return
MLI_WBRTN	0x87d
Counter Affinity	0xa0aa0
Definition	Front end thread switch in due to MLI write back return

4.2.21.14 MT_FE_THRSW_ACTUAL_OUT

Description	Front end thread switch out events
Max Inc/Cyc	1
MT Capture Type	F
Definition	Front end thread switch out events
NOTE	These are events that caused the foreground thread to be switched into the background, and thus are applied/counted to the thread transitioning into the background. Summation of this thread switch event set satisfies the following relationship MT_FE_THRSW_ACTUAL_OUT.ANY.me = sum(MT_FE_THRSW_ACUTAL_OUT.*.me) + sum(MT_fE_THRSW_ACUTAL_IN.*.other)
Subevents:	
ANY	0x876
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any FE thread switch, due to either the foreground thread switching out or the background thread switching in.
BE_FOLLOW	0x883
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to back-end follow
BRQ_BLK	0x880
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to BRQ blocking
HINT_BSWT	0x882
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to hint bswt

IBQ_FULL	0x878
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to IBQ full
IBQ_NOTEMPTY	0x87b
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to IBQ not empty
LOCKED	0x884
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to locked to BE
MLI_UCMISS	0x87e
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to MLI UC miss
MLI_WBMISS	0x87c
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to MLI write back miss
TIMEOUT	0x877
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Front end thread switch out due to switch time-out

4.2.21.15 MT_FE_THRSW_MISS_IN

Description	FE missed thread switch in opportunity cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	This event counts the number of cycles a thread switch event is not acted on.
Subevents:	
ANY	0x886
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	Any FE missed thread switch in opportunity cycles
BRQ_NON_BLK	0x891
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	brqnblk
IBQ_EMPTY	0x88a
Counter Affinity	0x50550
IAR/OPC/DAR/DREF	-/-/-/-
Definition	qempty

IBQ_NOTFULL	0x889
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	qnfull
MLI_UCRTN	0x88f
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	mliucrtn_act
MLI_WBRTN	0x88d
Counter Affinity	0xa0aa0
IAR/OPC/DAR/DREF	-/-/-/-
Definition	mliwbrtn

4.2.21.16 MT_FE_THRSW_MISS_OUT

Description	FE missed thread switch out opportunity cycles
Max Inc/Cyc	1
MT Capture Type	A
Definition	This event counts the number of cycles a thread switch event is not acted on.
Subevents:	
ANY	0x885
Counter Affinity	0xa0aa0
Definition	Any FE missed thread switch out opportunity cycles
BE_FOLLOW	0x893
Counter Affinity	0xa0aa0
Definition	BE follow
BRQ_BLK	0x890
Counter Affinity	0x50550
Definition	brqblk
HINT_BSWT	0x892
Counter Affinity	0x50550
Definition	hintbswt
IBQ_FULL	0x888
Counter Affinity	0x50550
Definition	qfull
IBQ_NOTEMPTY	0x88b
Counter Affinity	0xa0aa0
Definition	qnempty
LOCKED	0x894
Counter Affinity	0x50550
Definition	locked to BE
MLI_UCMISS	0x88e
Counter Affinity	0x50550
Definition	mliucmiss

MLI_WBMISS	0x88c
Counter Affinity	0x50550
Definition	mliwbmiss
TIMEOUT	0x887
Counter Affinity	0xa0aa0
Definition	tmout

4.2.21.17 MT_FE_THRSW_STALL

Description	Cycles the frontend threadswitch pipe is stalled
Max Inc/Cyc	1
MT Capture Type	A
Definition	Cycles the frontend threadswitch pipe is stalled and inhibited from issuing a pending threadswitch
Subevents:	
ANY	0x895
Counter Affinity	0xa0aa0
Definition	Any
BLK_ANY	0x899
Counter Affinity	0xa0aa0
Definition	Switch blocked
BLK_FW_PROG	0x89c
Counter Affinity	0x50550
Definition	Switch blocked, forward progress screen. The frontend forward progress screen is enabled and blocking a thread switch to the inactive thread.
BLK_IN_PROG	0x89b
Counter Affinity	0xa0aa0
Definition	Switch blocked, other switch in progress
BLK_IPC_MISS	0x89a
Counter Affinity	0x50550
Definition	Switch blocked for IPC miss
BLK_THRESH	0x89d
Counter Affinity	0xa0aa0
Definition	Switch blocked for threshold
EXPL	0x897
Counter Affinity	0xa0aa0
Definition	Explicitly disabled
IMPL	0x898
Counter Affinity	0x50550
Definition	Implicitly disabled
MTLCK	0x896
Counter Affinity	0x50550
Definition	MT locked

§

5 Uncore Performance Monitoring

Intel Itanium® processor uncore performance monitoring is supported by PMUs local to each of the boxes; Bbox, Cbox, Rbox, Sbox, Wbox and Zboxes. PMUs may be frozen locally (per box) or globally, either manually or due to uncore PMU counter overflow. In other words, any of the available uncore PMU 'counters' may be configured to freeze, upon overflow, either the other PMUs that reside within its box, or all uncore PMU counters.

All of the Itanium® processor uncore performance monitoring features are accessed through privileged Chip Set Registers (CSRs). SAL or the OS may access CSRs directly rather than asking SAL for access.

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

Table 5-1. Per-Box Performance Monitoring Capabilities

Box	# Boxes	# Counters/ Box	Generic Counters?	Packet Match/ Mask Filters?	Bit Width
BBox	2	8	N, subctl	Y	48
CBox	8	6	Y	N	48
RBox	1 (L/R sides)	8 (assign to same side port)	N, subctl	Y	48
SBox	2	4	Y	N	48
WBox	1	4	Y	Y	48
ZBox	2	6	N, subctl	N	48

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

- Section 5.3, "Global Performance Monitoring Control"
- Section 5.4, "Bbox Performance Monitoring"
- Section 5.5, "Cbox Performance Monitoring"
- Section 5.6, "Rbox Performance Monitoring"
- Section 5.7, "Sbox Performance Monitoring"
- Section 5.8, "Wbox Performance Monitoring"
- Section 5.9, "Zbox Performance Monitoring"
- Section 5.10, "Packet Matching Reference"

5.1 Processor Overview

The Intel® Itanium® processor 9500 series processor extends the multi-core family of Intel® Itanium® processors into the next generation with significant enhancements in both the core and system interface. The Intel® Itanium® processor 9500 series processor is upward compatible with the Intel® Itanium® Processor 9300 Series , providing the same Intel® QuickPath Interconnect (Intel® QPI) point-to-point links

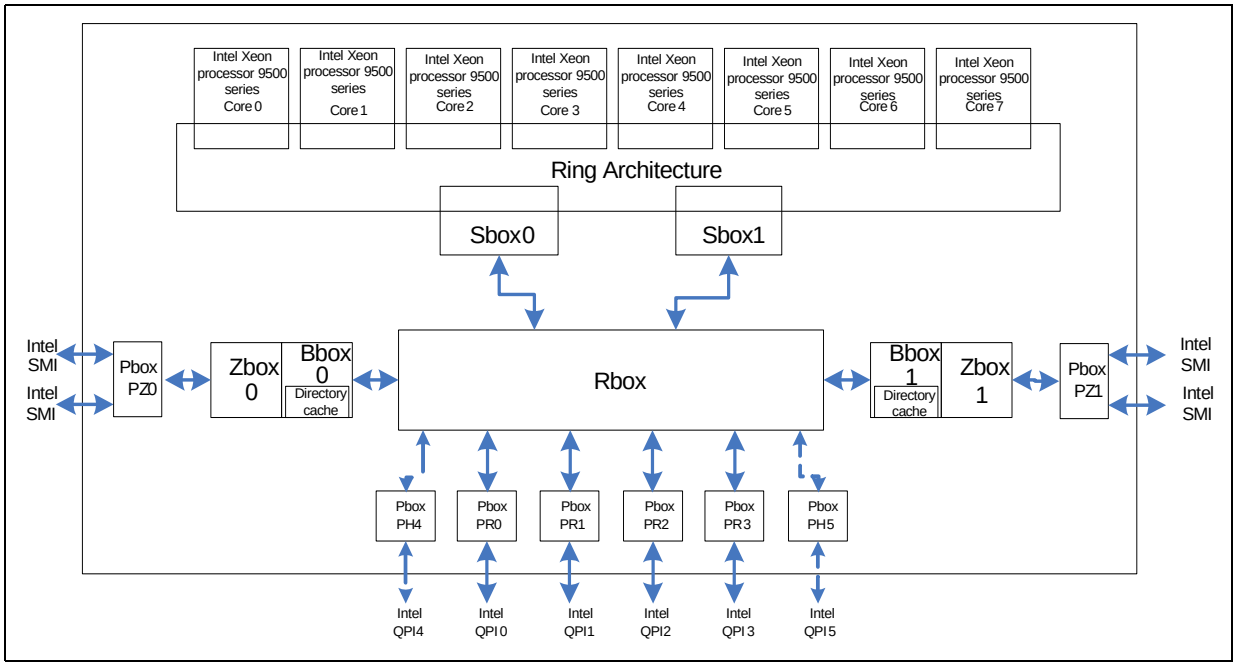
based system design, buffer-based memory technology (Intel® 7500 Scalable Memory Buffer) and interoperability with Intel® Itanium® Processor 9300 Series based chipsets.

The processor key features include:

- Fully compatible with binaries for the Intel Itanium processor family
- Implemented on 32-nm process technology
- Chip multiprocessor architecture with up to eight cores per die; two threads per core
- 64-bit virtual addressing; 50-bit physical addressing
- Each core provides in-order issue and execution of up to twelve instructions per cycle
- First-level cache (FLC) per core:
  - 16 KB FLI instruction cache
  - 16 KB FLD write-through data cache
- Mid-level cache (MLC) cache per core:
  - 512 KB MLI instruction cache
  - 256 KB MLD writeback data cache
- On-chip Last-level cache (LLC):
  - Up to 32 MB writeback cache
- Intel® Cache Safe Technology logic to disable cache entries that have become hard errors
- High-bandwidth, low-latency point-to-point link interface based on the Intel® QuickPath Interconnect (Intel® QPI):
  - Four full width and two half width Intel QuickPath Interconnects
  - Aggregate data bandwidth of 19.2 GB/s per each 4.8 GT/s full width Intel QuickPath Interconnect
  - Aggregate data bandwidth of 28.8 GB/s per each 6.4 GT/s full width Intel QuickPath Interconnect
- Self-healing Intel QuickPath Interconnect via width reduction:
  - Automatically configured during reset
  - Independent link width per direction
- Support for hot-add and hot-removal at the Intel QuickPath Interconnect
- Two Integrated Memory Controllers
  - Each memory controller supports two Intel® Scalable Memory Interconnects (Intel® SMI)
  - Support for one Intel 7500 Scalable Memory Buffer per Intel Scalable Memory Interconnect; four Intel 7500 Scalable Memory Buffers per processor
- Peak memory bandwidth (DDR3-1067)
  - 34.1 GB/s read (plus up to 11.2 GB/s concurrent write)
  - 17.0 GB/s write (plus up to 17.0 GB/s concurrent read)
- Memory reliability, availability, and serviceability (RAS) features including:
  - Extensive memory ECC support including Single Device Data Correction (SDDC) for x8 and x4 chip failure and Intel Double Device Data Correction (Intel DDDC) for x4 chip failure

  - — Failover mode to operate with a single lane failure per channel, per direction
  - — Support for Rank Sparing, Memory Mirroring, Memory Scrubbing, and Memory Migration
- Support for hot-add and hot-removal at the Intel Scalable Memory Interconnect
- ECC or parity Error protection on all major structures
- In-band and out-of-band system and configuration management support
- Power Management Controller for performance, power, and thermal management
- Power management support including memory thermal throttling
- Support for ACPI C-states, P-states, and S-states
- Support for Virtualization (soft-partitioning at the sub-socket level) with hardware optimizations to increase Virtual Machine performance
- Support for Dynamic Domain Partitioning (hard partitioning at the processor socket)

Figure 5-1. Intel® Itanium® Processor 9500 Series Processor Block Diagram



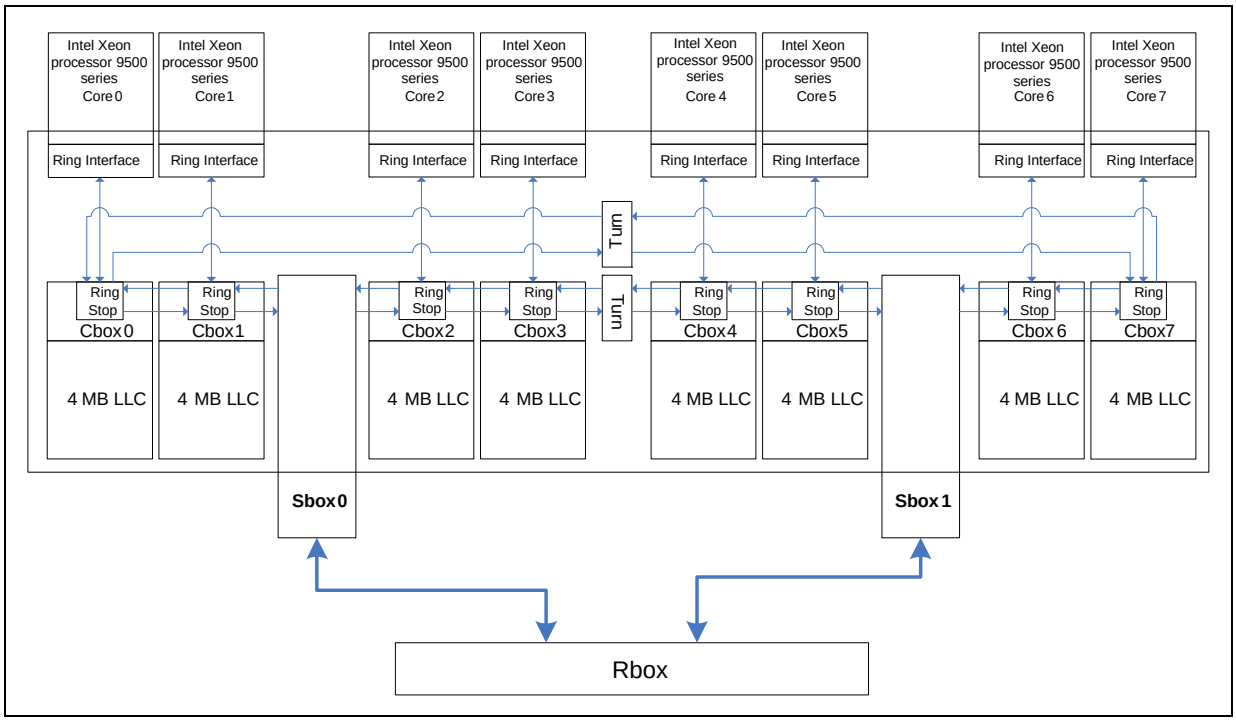
5.1.1 Ring Interconnect Overview

The Ring Interconnect is the communication channel connecting cores to the components of the Shared Cache. The Ring Interconnect is comprised of eight separately controlled rings which are grouped into four pairs of counter rotating, clockwise and counter-clockwise, rings. The AD ring pair carries addresses for request and snoop messages. The BL ring pair carries 32 Byte data blocks, and byte enables. The AK ring pair carries various coherence messages, and the IV ring pair carries multi-cast snoop messages from the Cboxes to the cores.

The Ring Interconnect is fully pipelined with a Ring Stop at each Cbox and Sbox. Each Ring Stop has buffering and arbitration facilities for messages entering the Ring and messages exiting the Ring. Messages are placed on either the clockwise or counter

clockwise ring depending on which path is the shortest route from source to destination. Messages on the Ring have priority over new messages attempting to enter the Ring.

Figure 5-2. Intel® Itanium® Processor 9500 Series Ring Architecture



5.1.2 Cache Control (Cbox) Overview

There are eight Cbox instances, of which each is associated with a processor core and a 4 MB slice of the Last-Level Cache. Collectively, the eight Cboxes and two Sboxes comprise the two Intel QPI caching agents for the socket.

The Ring Interface section of the Cbox contains the PRI (similar to IDI but tweaked to accommodate Intel Itanium processor requirements) interface to the core and frequency/voltage domain crossing circuitry (BGF) to allow for frequency/voltage scaling of the cores relative to the Shared Cache infrastructure.

Each Cbox can have 16 transactions in-flight at a time, which can be a mix of Mid-Level cache misses from the cores, LLC cache misses outstanding to Home Agents, and LLC write-backs outstanding to Home Agents. In addition the Cbox can perform LLC transactions in response to snoops initiated by Home Agents to maintain Global Cache Coherence.

Mid-Level Cache misses are distributed amongst the individual slices of the Shared Cache by the application of a Hash Function to the physical address of each transaction. The Hash Function attempts to ensure that programs access the LLC slices uniformly. Additionally, the four Cbox instances on the left side of the die communicate with the Sbox instance on the left side, and the same for the right side Cbox and Sbox instances respectively.

The Cbox implements key Intel QPI caching agent functionality which includes the Intel QPI Source Address Decoder (SAD), outgoing request buffer (ORB), and probe request queue (PRQ).

The processor contains eight instances of the Cbox numbered 0 through 7. Each Cbox manages a 4 MB 32-way set-associative last level cache (LLC) slice as well as a processor interface. The four instances of the Cbox on the left of the floorplan (C0–3) are associated with the four left processors (P0–3) The other four instances of the Cbox on the right of the floorplan (C4–7) are associated with the three right processors (P4–7). The Sbox on the left (S0) handles the left Cboxes. The Sbox on the right (S1), handles the right Cboxes. The Cboxes, Sboxes, and processors are connected using the Ring Interconnect. The eight Cboxes, together with the two Sboxes can be viewed as two Intel® QPI caching agents each with a node id. Each caching agent handles half of the cacheable address space.

Table 5-2. LLC Parameters

Data Type Supported	Policy	Size	Line Size	Ways	Index	Queueing	Latency
Instruction, Integer, FP	WB	32MB	64B	32	Hash(PA[49:6])	Variable: depends on system configuration, max 96 (12 per Cbox)	Variable: depends on distance along the Ring between requesting core and destination Cbox, also depends on Uncore Frequency

- Address Hash
  - The eight 4 MB LLC slices in the processor are addressed via an address hash function. This function is designed to evenly distribute accesses among the slices, even if the access pattern is a regular stride. It is also designed to evenly distribute accesses among the sets (indexes) of each slice, again, even if the access pattern is a regular stride.
  - The hash function takes as an input bits [49:6] of the physical address. It generates a 14-bit hash, of which 3 bits are used to address the 8 instances of the Cbox, and 11 bits are used to address the 2048 indexes in the target Cbox.
- LLC Arrays
  - Each CDAT unit contains half of a 64B cache line for 2048 sets by 16 ways of data. Overall, the Cbox can read or write 32B per cycle. The data array is protected by a DECTED ECC code.
  - Each CTAG unit corresponds to 16 ways of tag, LRU, and state/core-valid arrays for 2048 sets. Each array is protected by a SECTED ECC code.
- Coherence
  - Each cache line in the LLC maintains a Intel QPI global coherence state; one of Modified, Exclusive, Shared or Invalid (MESI).
  - Along with the global coherence state for each line, an 8-bit core-valid mask (directory) is maintained. This mask indicates which cores may also have the same line in their caches. The core-valid mask may have bits set corresponding to cores which no longer have the cache line, since cores can silently displace Exclusive or Shared cache lines.

5.1.3 Last-Level Cache (LLC) Overview

The processor Last-Level Cache is a 32 MB, 64 B per line, write-back cache which is shared by all eight cores and is inclusive of the data in all caches in the eight cores. The LLC is organized as eight 4 MB slices. The tag array for the LLC contains address tags, MESI state bits, and core-valid bits which are used to filter snoop requests to the cores. The LLC can read or write 32 B per cycle. The data array provides row and column redundancy and uses DECTED (double-error correct, triple-error detect) for error

detection and correction. (Like the Intel Itanium processor 9300 series) The processor implements Intel® Cache Safe Technology which allows individual lines that are found to be bad during operation to be disabled on the fly.

5.1.4 System Bridge (Sbox) Overview

The Sbox is the interface between the on-die Ring Interconnect domain and the Intel QPI domain. It buffers and reformats messages in both the Ring-Bound and System-Bound directions. There are two Sbox instances each of which services messages from four of the eight Cbox instances. Each Sbox instance has a connection to the Rbox. The Sboxes have separate nodeIDs to simplify routing but are actually partitioned by address with each Sbox servicing half of the address space of the shared ring cache. Refer to the Cbox chapter on the Hashing function for more information on how addresses are mapped.

5.1.5 Global Coherence Engine (Bbox) Overview

The primary function of the Bbox is to be the coherent Home Agent for the Intel QPI cache coherence protocol. The Home Agent algorithm requires the Bbox to track outstanding requests, log snoop responses and other control messages, and make certain algorithmic decisions on how to respond to requests. The Bbox has additional requirements on managing interactions with the Memory Controller, including RAS flows. All requests that are in-flight in the Memory Controller are tracked in the Bbox.

The Bbox is responsible for the protocol side of memory interactions, including the coherent and non-coherent Home Agent protocols. Additionally, the Bbox is responsible for ordering memory reads/writes to a given address so that the Memory Controller does not also have to do this conflict checking. All requests for memory attached to the coupled Memory Controller must first be ordered through the Bbox.

The Bbox implements directory coherency using a directory stored in main memory. The Bbox also implements a directory cache to improve the performance of directory coherency. The Bbox supports up to 64 directory caching agents but has no support for Source Snoop caching agents.

The Bbox has a 512 entry Tracker File and allows up to 48 operations in-flight to the Memory Controller from a caching agent.

5.1.6 Integrated Memory Controller (Zbox) Overview

The Zbox is the interface to memory for the processor. In general terms it translates read and write commands into specific memory commands and schedules them with respect to memory timing. The other main function of the Zbox is advanced ECC support. There are two Zboxes implemented in each die, one for each pair of Intel SMI channels.

The Zbox interfaces to the Rbox through the Bbox. It also interfaces to the Pbox pads.

A memory controller manages two Intel SMI channels operating in lockstep, for a total of four Intel SMI channels per socket. The processor supports up to 32 DIMMs per socket.

The Memory Controller supports RAS features including retry on transient errors, Intel Single Device Data Correction (SDDC) for a single x4 or x8 device, Intel Double Device Data Correction (DDDC) for two x4 devices, Rank Sparing, and Intel SMI lane fail-over mode.

The Zbox PMU is reserved if memory mirroring is enabled on the platform.

5.1.7 Inter-processor Router (Rbox) Overview

The Rbox on the processor is a 10-port switch/router implementing the Intel QPI Routing and Link layer. It is responsible for routing and transmitting all intra- and inter-processor communication and it provides concurrent full duplex bandwidth. The Rbox is connected directly to the Bbox, Sbox, and Pboxes via 80b interfaces. The Ubox taps off the Bbox connection. Most boxes can handle the data at full bandwidth but the local and external Intel QPI links operate at different frequencies and selectively enable cycles for data transfer in order to match the port bandwidth to the physical link. The Rbox consists of ten identical ports and a wire crossbar that connects the ports together. Route-through functionality (data transferred between Intel QPI ports) is supported by the Crossbar Router. The router is ECC-protected for enhanced reliability, and supports Link-Level Retry for handling transient link errors.

The Rbox supports the HOM, SNP, DRS, NDR, NCS and NCB message channels and the VN0, VN1 and VNA virtual routing channels. An 80 entry Route Table (per input port) supports routing of messages with 10b node-ids.

5.1.8 Port Physical Interfaces (Pbox) Overview

The Pbox contains the Physical layer (PHY) for the processor Intel QPI and Intel SMI ports. The PHY consists of an electrical sub-block as well as a logical sub-block. The electrical sub-block contains all the circuitry (mostly analog in nature) related to transmitting and receiving clock and data signals to/from another component. The logical sub-block (mostly digital in nature) consists primarily of control logic, serialization and de-serialization structures, control and status registers, and other digital logic found in the datapath. In summary, the Pbox is essentially a combination of digital control logic, digital datapath logic and sophisticated analog circuitry, supporting high speed differential point-to-point signaling.

There are eight separate Pbox instances (or ports) in the processor, each providing a communications path to/from the Link-layer agents (Rbox, Zbox) and the off-socket links.

Four of the Pbox instances are associated exclusively with the Rbox and provide for full-width, 20-lane, ports for inter-socket connectivity. Two of the Pbox instances are associated exclusively with the Rbox and provide for half-width, 10-lane, ports for inter-socket connectivity. The remaining two instances of the Pbox are associated with the Bbox and Zbox and provide the Intel SMI memory interface. The Intel SMI instance of the Pbox connects to two Intel SMI channels that are accessed in parallel.

5.1.9 System Utilities Controller (Ubox) Overview

The Ubox is the Intel QPI configuration agent and provides access mechanisms for a variety of on-die, on-socket, and off-socket resources and communication paths, including In-Band Configuration, Out-of-Band Configuration, Off-Chip ROM, SMBus access, and Inter-processor Interrupts.

- In-Band Configuration Access: This feature allows platform firmware running on the processor to access the processor's Configuration and Status Registers (CSRs) through Intel QPI.
- Out-of-Band Configuration Access through SMBus and JTAG: This feature includes the SMBus and JTAG Slave engines which allows a service processor or system

management controller to access the CSRs in the Ubox address space via the SMBus or JTAG pins.

- Off-chip Flash ROM Interface: The off-chip flash ROM interface allows the processor to be connected to up to four SPI Flash ROM devices. These ROM devices are used to store PAL code, SAL code and error logs.
- Out-of-Band SMBus Master Engine for Platform Configuration: The SMBus Master engine allows the processor to be connected to other platform resources (such as Intel 7500 Scalable Memory Buffer and memory) using the SMBus protocol. The interface allows firmware or system management controllers to access the SPD registers on any DIMMs connected to the processor.
- Interruption Utility: The interruption utility facilitates interruptions to the on-chip processor cores.

5.2 Uncore PMU Programming Overview

Software can access the PMU CSR registers which contain the uncore PMUs.

To access the a specific register, the address is built in hierarchical fashion beginning with the aperture base address, adding in the socket, then the uncore box of choice and finally the specific CSR residing within the box.

```
SAL Aperture Base - 0x8003FFF3.xxxxxxxx

SAL Aperture Length - 0x100000000
```

Table 5-3. Physical / Virtual Address Offsets for Socket Access

Window	Offset [32:0]	Description
Current Socket	0xFEB0xxxx	This window provides access to the uncore PMU registers for the current socket, or the socket containing the core that is executing the CSR access.
Socket 0	0xFC00xxxx	This window provides access to the uncore PMU registers in socket 0.
Socket 1	0xFC01xxxx	This window provides access to the uncore PMU registers in socket 1.
...		
Socket 255	0xFCFFxxxx	This window provides access to the uncore PMU registers in socket 255.

Table 5-4. Address Offsets for Per-Box Access (Sheet 1 of 2)

Window	Offset [15:0]	Description
Ubox	--	N/A - LS 16b of address specified.
Bbox 0/Zbox 0	0x3xxx	Window to Bbox 0 and Zbox 0 CSRs
Bbox 1/Zbox 1	0xBxxx	Window to Bbox 1 and Zbox 1 CSRs
Sbox 0/Wbox	0x7xxx	Window to Sbox 0 and the Wbox
Sbox 1	0xFxxx	Window to Sbox 1
Rbox	0x2xxx,0x8xxx	These windows provide access to Rbox CSRs (see Rbox Uncore PMU Summary Table for details)
Cbox 0	0x40xx	Window to Cbox 0
Cbox 1	0x48xx	Window to Cbox 1
Cbox 2	0x50xx	Window to Cbox 2
Cbox 3	0x58xx	Window to Cbox 3
Cbox 4	0xD8xx	Window to Cbox 4

Table 5-4. Address Offsets for Per-Box Access (Sheet 2 of 2)

Window	Offset [15:0]	Description
Cbox 5	0xD0xx	Window to Cbox 5
Cbox 6	0xC8xx	Window to Cbox 6
Cbox 7	0xC0xx	Window to Cbox 7

So, by example, the physical address for B_CSR_PERF_CTL0 in Bbox 0 on the current socket is:

```
0x8003FFF3.00000000 + // SAL aperture

0x00000000.FEB00000 + // current socket

0x00000000.00003000 + // Bbox 0

0x00000000.00000808 = // offset for B_CSR_PERF_CTL0

0x8003FFF3.FEB03808
```

5.2.1 On Accessing Uncore PMUs by Virtual Addresses (Win/Linux*)

Driver software typically does not deal with physical addresses, but remaps these address ranges to the virtual address space so CSR registers can be more conveniently accessed with memory-mapped input/output (MMIO) operations. The following C code samples show how these physical addresses can be mapped and un-mapped in the virtual address space for Windows and Linux. These addresses must be mapped as un-cacheable memory.

```
#define TKWUNC_SAL_APERTURE 0x8003FFF300000000

#define TKWUNC_APERTURE_LEN 0x100000000

void* BaseVirtAddr;

#ifdef linux

static void MapTkwUncCSRsLinux(void)

{

    // map the MMIO space using the SAL aperture

    BaseVirtAddr =

        ioremap_nocache(TKWUNC_SAL_APERTURE, TKWUNC_APERTURE_LEN);

}

static void UnmapTkwUncCSRsLinux(void)

{

    // unmap the MMIO space

    if (0 != BaseVirtAddr) {

        iounmap(BaseVirtAddr);

        BasVirtAddr = 0;

    }
```

```
}
#else  // linux
static void MapTkwUncCSRsWindows(void)
{
    PHYSICAL_ADDRESS baseaddr;
    // create base physical address
    baseaddr.QuadPart = TKWUNC_SAL_APERTURE;
    BaseVirtAddr =
        MmMapIoSpace(baseaddr, TKWUNC_APERTURE_LEN, MmNonCached);
}
static void UnmapTkwUncCSRsWindows(void)
{
    // unmap the MMIO spce
    if (0 != BaseVirtAddr) {
        MmUnmapIoSpace(BaseVirtAddr, TKWUNC_APERTURE_LEN);
        BaseVirtAddr = 0;
    }
}
#endif
```

Using the code listed above, we can map the physical addresses of the SAL CSR aperture into the virtual address space. Assume that the address mapping operation yields a base virtual address of 0xFB00D800.00000000. Then, using the example from the previous section, the B_CSR_PERF_CTL0 CSR register can be accessed in virtual address space at virtual address:

```
0xFB00D800.00000000 + // base virtual address
0x00000000.FEB00000 + // current socket
0x00000000.00003000 + // Bbox 0
0x00000000.00000808
= // offset for B_CSR_PERF_CTL0
0xFB00D800.FEB03808
```

To read the CSR, the following C code may be executed.

```
unsigned long long val;
val = *((unsigned long long*) 0xFB00D800FEB03808);
```

and to write the register:

```
*((unsigned long long*) 0xFB00D800FEB03808) = val;
```

Since the uncore performance monitors represent socket-wide resources that are not context switched by the OS, it is highly recommended that only one piece of software (per-socket) attempt to program and extract information from the monitors. To keep things simple, it is also recommended that the monitoring software communicate with the OS such that it can be executed on coreId = 0, threadId = 0. Although recommended, this step is not necessary. Software may be notified of an overflowing uncore counter on any core.

5.2.2 Uncore PMU Summary Tables

Table 5-5. Uncore Performance Monitoring CSRs (Sheet 1 of 3)

Box	CSR Addresses	Description
Sbox Counters		
Sbox 1	0xF8F8	Intel QPI Configuration Register
	0xF8F0-0xF8E8	Intel QPI Data Registers
	0xF8E0-0xF8D8	Global (Control/Status/Ovf Control)
	0xF8B0-0xF878	Counter/Config Registers
	0xF858-0xF840	Match/Mask Registers
Sbox 0	0x78F8	Intel QPI Configuration Register
	0x78F0-0x78E8	Intel QPI Data Registers
	0x78E0-0x78D8	Global (Control/Status/Ovf Control)
	0x78B0-0x7878	Counter/Config Registers
	0x7858-0x7840	Match/Mask Registers
Wbox Counters	**AddrOffset [15:0]**	
Wbox	0x76D0-0x76C0	Global (Control/Status/Ovf Control)
	0x74F8-0x74A0	Counter/Config Registers
	0x7040	Box Master Register
CBox Counters	**AddrOffset [16:0]**	
Cbox 7	0xD8B8-0xD8A0	Box Configuration Registers
	0xD898-0xD870	Counter/Control Registers
Cbox 6	0xD0B8-0xD0A0	Box Configuration Registers
	0xD098-0xD070	Counter/Control Registers
Cbox 5	0xC8B8-0xC8A0	Box Configuration Registers
	0xC898-0xC870	Counter/Control Registers
Cbox 4	0xC0B8-0xC0A0	Box Configuration Registers
	0xC098-0xC070	Counter/Control Registers
Cbox 3	0x58B8-0x58A0	Box Configuration Registers

Table 5-5. Uncore Performance Monitoring CSRs (Sheet 2 of 3)

Box	CSR Addresses	Description
	0x5898-0x5870	Counter/Control Registers
Cbox 2	0x50B8-0x50A0	Box Configuration Registers
	0x5098-0x5070	Counter/Control Registers
Cbox 1	0x48B8-0x48A0	Box Configuration Registers
	0x4898-0x4870	Counter/Control Registers
Cbox 0	0x40B8-0x40A0	Box Configuration Registers
	0x4098-0x4070	Counter/Control Registers
BBox Counters	**AddrOffset [15:0]**	
Bbox 1	0xB9C0-0xB980	DC Registers (Counters/Control/MatchMask)
	0xB8F8-0xB8C0	Counter Registers
	0xB8A0-0xB890	Subcounters (IMT, ARB)
	0xB880	Box Master Register
	0xB878-0xB840	Counter Control Registers
	0xB830-0xB828	Subcounters (BZ, IOB)
	0xB800	Box Control Registers (subcounters, mask/match)
Bbox 0	0x39C0-0x3980	DC Registers (Counters/Control/MatchMask)
	0x38F8-0x38C0	Counter Registers
	0x38A0-0x3890	Subcounters (IMT, ARB)
	0x3880	Box Master Register
	0x3878-0x3840	Counter Control Registers
	0x3830-0x3828	Subcounters (BZ, IOB)
	0x3800	Box Control Registers (subcounters, mask/match)
ZBox Counters	**AddrOffset [15:0]**	
Zbox 1	0xB0D8	Box Master Register
	0xB0D8-0xBA8	Subconfig Registers (FVC,PLD,PGT,THR,ISS,DSP)
	0xB0A0	Box Overflow Status
	0xB98-0xB40	Counter/Config Registers
Zbox 0	0x30D8	Box Master Register
	0x30D8-0x3A8	Subconfig Registers (FVC,PLD,PGT,THR,ISS,DSP)
	0x30A0	Box Overflow Status
	0x398-0x340	Counter/Config Registers
RBox Counters	**AddrOffset [15:0]**	
Rbox (Both Sides)		
	0x2AC0	Box Master Register

Table 5-5. Uncore Performance Monitoring CSRs (Sheet 3 of 3)

Box	CSR Addresses	Description
Rbox R	0x8AB8-0x8A40	Counter/Config Registers(15-8)
	0x{88,86,84,82,80} C8-80	Mask/Match Registers for Ports 9-5
	0x{88,86,84,82,80} 78-70	IPERF SubConfig Registers for Ports 9-5 (RIX Events)
	0x{88,86,84,82,80} 10	ARB SubConfig Registers for Ports 9-5 (QLX Events)
Rbox L	0x2AB8-0x2A40	Counter/Config Registers(7-0)
	0x{28,26,24,22,20} C8-80	Mask/Match Registers for Ports 9-5
	0x{28,26,24,22,20} 78-70	IPERF SubConfig Registers for Ports 4-0 (RIX Events)
	0x{28,26,24,22,20} 10	ARB SubConfig Registers for Ports 4-0 (QLX Events)
UBox Counter	**AddrOffset [15:0]**	
	0x1588	**Ubox PMON Global Configuration Register**
	0x1540	**Uncore PMON Interrupt Register**

5.3 Global Performance Monitoring Control

The Itanium processor 9500 series uncore PMUs support two PMU freeze (or disable) modes: box-level (local) and processor-level (global).

If an uncore counter (for example, in the Zbox) is configured to generate a box-level freeze when an overflow is detected from the counter, only the PMUs within the box (for example, the other Zbox PMUs) are frozen. The overflow will be sent upstream (to the Sbox) to be recorded, but the PMUs in other boxes remain unaffected by this local freeze. If, however, the uncore counter is configured to generated a socket-wide freeze, when a overflow is detected from the counter, all uncore PMUs within the same socket will be frozen simultaneously.

If a counter is configured to trip an uncore-wide freeze, an interrupt will be sent from the Ubox as the freeze (disable) signal is sent out.

In socket-level freeze mode, if a second overflow is detected in the window of time it takes the first overflow to freeze all the uncore PMUs, the additional overflow will be recorded in the Sbox.

Global control of uncore PMU enabling and disabling is contained within the Ubox. However, each box has a *PERF_MASTER.glb_lcl bit which allows a box to choose whether to participate in the global control or stick solely with box-level control.

5.3.1 Global Enable / Disable

For each box that accepts global control (*PERF_MASTER.glb_lcl is set to 1), U_CSR_PERF_CTL.glb_en allows a user to enable/disable uncore performance monitoring with a single bit.

If a participating (allows global control) box's counter overflows, and SW set the box's mask bit in the appropriate S_CSR_PMON_FRZ_EN register, hardware will clear the U_CSR_PERF_CTL.glb_en bit to suspend all uncore performance monitoring participating in the global session.

Similarly, software may manually disable all participating uncore PMUs by writing 0 to U_CSR_PERF_CTL.glb_en.

Note: When a global uncore disable signal is sent out to the uncore boxes, it is also sent to each core. Each core PMU capture it as an event (UNCORE_FREEZE - Event Code 0xC9 in the core counters).

5.3.2 Setting Up a Global Monitoring Session

Use the global freeze mechanism (refer to Section 5.3.1, "Global Enable/Disable") to ensure that no counting is taking place while the session is being configured:

Global 1) Disable uncore monitoring

- Set U_CSR_PERF_CTL.glb_en to 0.

Global 2) For each box in which SW wants to measure events and have the session globally controlled:

- Set *PERF_MASTER.glb_lcl to 1

Now, set up monitoring in each box.

Box 1) Select event(s) to monitor:

Determine what events should be captured and program the control registers to capture them (i.e. refer to individual box sections for details).

- that is, Set B_CSR_PERF_CTL1_3 to 0x1b to capture SNP_REQ_ALL.

Box 2) If necessary, enable counting locally by setting the appropriate enable bits

- For events captured in the C & W boxes, there is an enable bit (b22) in each individual counter control register as well as an enable bit for each counter in the C/W_CSR_PMON_GLOBAL_CTL register.

Box 3) Select how to gather data. If polling, skip to *Global 3*. If sampling:

- To set up a **sample interval**, software can:
  a. pre-program the data register with a value of [2^48- sample interval length]
  b. and mask the overflow coming from the box to trigger an interrupt. Doing so allows software to be **notified** when the number of events in the sample have been captured. Capturing a performance monitoring sample every 'X cycles' (i.e. B_CYCLES) is a common use of this mechanism.

That is, to stop counting and receive notification when the 1,000th SNP_REQ_ALL has been detected,

a. Set B_CSR_PERF_CNT[3] to (2^48- 1000)

b. Set S_CSR_PMON_FRZ_EN.msk_b to 1 in the associated Sbox (S0 for B0, S1 for B1), else set to 0.

Note: For Sbox counters, the overflow has to be specifically enabled by setting the S_CSR_PMON_CTLx.ovf_en bit to 1 for counter x.

Note: Even if the box is 'isolated' from global control, in sending out the interrupt when an overflow is detected, the Ubox will also disable counters that are globally enabled.

Global 3) Enable all uncore PMU counters

- Set U_CSR_PERF_CTL.glb_en to 1.

And with that, counting will begin.

5.3.2.1 How Setting Up a Box-level Monitoring Session Differs

If a user wishes to isolate a specific box from global control, the following steps will differ from those outlined in Section 5.3.2, "Setting Up a Global Monitoring Session".

The three steps marked *Global* will be replaced as follows:

1. Make sure the box is isolated from the global network:
- Set *PERF_MASTER.glb_lcl to 0
2. Be sure counting has been stopped in the box:
- Set *PERF_MASTER.en to 0.

Perform steps *Box* #1-3 found in Section 5.3.2, "Setting Up a Global Monitoring Session".

Finally, instead of enabling counting at the global level, enable counting at the box level:

- Set *PERF_MASTER.en to 1

And with that, counting will begin.

Note: If the Box's .glb_lcl is 0, a local overflow will **always** stop the counters within the box. There is one exception to this rule: In the Sbox, the overflow must be explicitly enabled in each counter's control register by setting S_CSR_PMON_CTLx.ovf_en to 1.

5.3.3 Reading the Sample Interval

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

a) **Polling** - before reading, it is recommended that software freeze the counters

- If the box is participating in global control, set U_CSR_PERF_CTRL.glb_en to 0
- Else set the box's *PERF_MASTER.en to 0

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

Overflow bits are stored within each box and summarized in the Sbox. First, software should read S_CSR_PMON_SUMMARY.ov_{box} bits to determine which box(es) contain the overflowing counter. Once the box(es) has been identified, read that box's

overflow bits (i.e. {C,S,W}_CSR_PMON_GLOBAL_STATUS.ov, B_CSR_PERF_CNTx.ov, R_CSR_PERF_CNT_CTRL_x.ov and Z_CSR_PMU_CNT_STATUS.cntX_ov) to find the overflowing counter.

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

5.3.4 Enabling a New Sample Interval from Frozen Counters.

a) Clear all relevant uncore counters.

- For each box monitored, set the box's *PERF_MASTER.clr to 1

b) Clear all relevant overflow bits.

- If monitoring session was globally controlled, clear S_CSR_PMON_SUMMARY.
- All but the Rbox have a box-level status register with overflow bits that should be cleared (i.e. B_CSR_PMON_PERF_STATUS.ov {C,S,W}_CSR_PMON_GLOBAL_STATUS.ov, Z_CSR_PMU_CNT_STATUS.ov). The R and Bbox have overflow bits in the data registers that will need to be cleared. Yes, there are 2 overflow bits for each Bbox register.

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 5.3.2, "Setting Up a Global Monitoring Session".

d) Re-enable counting:

- If monitoring session was globally controlled, set U_CSR_PERF_CTL.glb_en to 1
- Else set *PERF_MASTER.en to 1 for relevant boxes.

5.3.5 Global Performance Monitors

5.3.5.1 Global PMU Global Control/Status Registers

The following register represents one bit of state governing all PMUs in the uncore - the global enable bit. Each box may be programmed (through it's local *PERF_MASTER.glb_lcl bit) to be governed by this bit global control bit or to ignore it.

Overflow indications from each box are collected in each of the Sboxes (S_CSR_PMON_SUMMARY).

Note: When a global uncore disable signal is sent out to the uncore boxes, it is also sent to each core. Each core PMU capture it as an event (UNCORE_FREEZE - Event Code 0xC9 in the core counters).

Table 5-6. U_CSR_PERF_CTL Register Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	31:1	RO_NA	0	Read zero; writes ignored.

Table 5-6. U_CSR_PERF_CTL Register Field Definitions

Field	Bits	Access	HW Reset Val	Description
glb_en	0	RW_RW	0	Global Enable 1: Enable uncore PMON. Send enable signal to all boxes. 0:Disable uncore PMON. Send disable signal to all boxes. For any box participating in global control (*PERF_MASTER.glb_lcl == 1): - HW will reflect updates to this bit in the box's *PERF_MASTER.en bit. - HW will clear this bit on detection of an overflow from a performance monitor from the box.

As previously mentioned, if an overflow is detected by the Ubox and U_CSR_PERF_CTL.glb_lcl == 1, the Ubox will signal an interrupt. The following register governs how Uncore PMU interrupt is relayed.

Table 5-7. U_CSR_IDR[14] (for PMON interrupts) Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:34	RO_NA	0	Read zero; writes ignored. (?)
idle	33	RO_WO	0	If 0, there is something in queue to be delivered to core
pend	32	RW_RW	0	Interrupt pending If 1: Ubox will remember that it saw this class of interrupt, but will not process it until this bit is set back to 0
ig	31:29	RO_NA	0	Read zero; writes ignored. (?)
dop	28	RW_RW	0	Delivery operation 0 - NOP 1 - Deliver to core using fields pic and v
pic	27:12	RW_RW	0	Target PIC bit-vector (can be multi-hot). One bit per-thread. Each bit corresponds to a Core/Thread as specified: 27 - Core 7, Thread 1 26 - Core 7, Thread 0 25 - Core 6, Thread 1 24 - Core 6, Thread 0 ... 13 - Core 0, Thread 1 12 - Core 0, Thread 0
ig	11:8	RO_NA	0	Read zero; writes ignored. (?)
v	7:0	RW_RW	0	Vector

5.4 Bbox Performance Monitoring

5.4.1 Overview of the Bbox

The Bbox is responsible for the protocol side of memory interactions, including the coherent and non-coherent home agent protocols (as defined in the *Intel® QuickPath Interconnect Specification*). Additionally, the Bbox is responsible for ordering memory

reads/writes to a given address so that the Zbox does not also have to do this conflict checking. All requests for memory attached to the coupled Zbox must first be ordered through the Bbox.

The Bbox has additional requirements on managing interactions with the Zbox, including RAS flows. All requests in-flight in the Zbox are tracked in the Bbox. The primary function of the Bbox, is as the coherent home agent for the QuickPath Interconnect cache coherence protocol. The home agent algorithm requires the Bbox to track outstanding requests, log snoop responses and other control messages, and make certain algorithmic decisions about how to respond to requests.

The Bbox only supports directory caching agents. The source snoopy QuickPath Interconnect protocol flows are not supported.

5.4.2 Bbox Performance Monitoring Overview

The Bbox PMU supports event monitoring through eight 48-bit wide counters (B_CSR_PERF_CNT{7:0}). Each of these eight counters operates orthogonally, such that each can be configured to monitor any of the available primary Bbox events. These counters will increment by a maximum of 1 per cycle.

The count values of all 8 counters can be cleared by writing the B_CSR_PMON_PERF_MASTER.clr bit.

Bbox PMU allows users to monitor latency related events. Traditionally, latency related events have been calculated by measuring the number of live transactions per cycle and accumulating this value in one of the PMU counters. The Bbox offers a different approach. A number of live counters are dedicated to track live entries in different queues and structures. Rather than directly accumulate the live counter values in the PMU counters, they are fed to a number of accumulators (widths ranging from 6 to 11bits). Overflow of these accumulator values are then fed into the main PMU counters. In order to capture an accurate live entry count for a specific queue, the accumulator assigned to it will have to be shifted in as the LSB of the count captured in the PMU.

5.4.2.1 Bbox PMU - Overflow, Freeze and Unfreeze

Bbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Bbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, “Global Enable/Disable”).

Bbox PMU can be frozen due to one of three reasons

- *Globally:* Ubox sends a disable signal (and B_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, “Global Enable/Disable”) or *local* (SW writes 0 to B_CSR_PMON_PERF_MASTER.en when B_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.
- *Locally:* The Bbox was set to local control (B_CSR_PMON_PERF_MASTER.glb_lcl = 0) and a Bbox counter overflowed.

If an overflow is detected from a Bbox performance counter, the overflow bit is set in the data register (B_CSR_PERF_CNT_x.ov), accumulated at the box level (B_CSR_PMON_PERF_STATUS.ovX), and forwarded up the chain towards the Sbox (S_CSR_PMON_SUMMARY.ov_b). Refer to Table , “” to determine how each Bbox’s overflow bit is accumulated in the attached Sbox.

The Ubox may be configured to freeze all uncore counting (refer to Table , “”) when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field(s) responsible for the freeze, must be cleared by setting them to 0. Once the overflow bit(s) has been cleared, the Bbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, “Enabling a New Sample Interval from Frozen Counters.”), counting will resume.

5.4.3 Bbox Performance Monitoring CSRs

Table 5-8. Bbox PMU Summary (Sheet 1 of 2)

CSR Name	AddrOffset [11:0]	Priv Lvl	Description
Directory Cache			
B_CSR_DC_PERF_MASK_WR2	0x9C8	none	DC performance write mask 2
B_CSR_DC_PERF_MATCH_WR2	0x9C0	none	DC performance write match 2
B_CSR_DC_PERF_WMARK	0x9B8	none	DC performance watermark
B_CSR_DC_PERF_CNT	0x9B0	none	DC performance counters
B_CSR_DC_PERF_MASK_WR1	0x9A8	none	DC performance write mask 1
B_CSR_DC_PERF_MATCH_WR1	0x9A0	none	DC performance write match
B_CSR_DC_PERF_MASK_RD	0x998	none	DC performance read mask
B_CSR_DC_PERF_MATCH_RD	0x990	none	DC performance read match
B_CSR_DC_PERF_CTL1	0x988	none	performance control 1
B_CSR_DC_PERF_CTL0	0x980	none	performance control 0
Generic Counters			
B_CSR_PERF_CNT_7	0x8F8	none	Performance Counter 7
B_CSR_PERF_CNT_6	0x8F0	none	Performance Counter 6
B_CSR_PERF_CNT_5	0x8E8	none	Performance Counter 5
B_CSR_PERF_CNT_4	0x8E0	none	Performance Counter 4
B_CSR_PERF_CNT_3	0x8D8	none	Performance Counter 3
B_CSR_PERF_CNT_2	0x8D0	none	Performance Counter 2
B_CSR_PERF_CNT_1	0x8C8	none	Performance Counter 1
B_CSR_PERF_CNT_0	0x8C0	none	Performance Counter 0
SubCounter (for Queues) Registers			
B_CSR_IMT_PERF_CNT	0x8A0	none	In-Flight Memory Table (IMT) performance counter
B_CSR_ARB_PERF_CNT_1	0x898	none	Arbiter PMU sub-counter 1
B_CSR_ARB_PERF_CNT_0	0x890	none	Arbiter PMU sub-counter 0
Box-Level Control/Status			
B_CSR_PMON_PERF_MASTER	0x880	none	Performance Master Control Register
B_CSR_PMON_PERF_STATUS	0x800	none	Performance Status Register

Table 5-8. Bbox PMU Summary (Sheet 2 of 2)

CSR Name	AddrOffset [11:0]	Priv Lvl	Description
Generic Control for Counters			
B_CSR_PERF_CTL1_7	0x878	none	Performance Counter Control 7
B_CSR_PERF_CTL1_6	0x870	none	Performance Counter Control 6
B_CSR_PERF_CTL1_5	0x868	none	Performance Counter Control 5
B_CSR_PERF_CTL1_4	0x860	none	Performance Counter Control 4
B_CSR_PERF_CTL1_3	0x858	none	Performance Counter Control 3
B_CSR_PERF_CTL1_2	0x850	none	Performance Counter Control 2
B_CSR_PERF_CTL1_1	0x848	none	Performance Counter Control 1
B_CSR_PERF_CTL1_0	0x840	none	Performance Counter Control 0
SubControl/SubCounters			
B_CSR_BZ_PERF_CNT	0x830	none	B-Z Interface counter
B_CSR_IOB_PERF_CNT	0x828	none	Input/Output Block (IOB) performance counters
B_CSR_PERF_CTL3	0x818	none	Performance Control
B_CSR_PERF_CTL2	0x810	none	Performance Control
B_CSR_PERF_CTL0	0x808	none	Performance Control 0

5.4.3.1 Bbox Box Level PMU State

The following registers represent the state governing all PMUs in the Bbox.

B_CSR_PMON_PERF_MASTER controls the general characteristics of the Bbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

Note: SW MUST set .clk_en to 1 to enable Bbox performance monitoring state, by turning on the clock, in addition to the normal enable mechanisms.

If an overflow is detected from one of the Bbox PMON registers, the corresponding bit in the _PERF_STATUS.ov field will be set. A user must clear the corresponding bits in the __PERF_STATUS.ov field before beginning a new sample interval.

And the _CTL0 register contains the bits used to enable queue occupancy monitoring.

Table 5-9. B_CSR_PMON_PERF_MASTER Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	63:4	RO_NA	0	Read zero; writes ignored. (?)
ck_en	3	RW_RW	0	Enables Bbox PMU clock. ***Note:*** Must be set to 1 in order for Bbox counters to capture events!

Table 5-9. B_CSR_PMON_PERF_MASTER Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Type	HW Reset Val	Description
clr	2	RW_RW	0	Writing 1 clears all the Bbox counters. The overflow bits found in B_CSR_PMON_PERF_STATUS will be cleared in the same operation since they are copies of the overflow bits found in the counters.
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: **Global**: Enable/Disable of counters in Bbox will track U_CSR_PERF_CTL.glb_en. Local overflows will be passed on to Ubox without freezing local counters. 0: **Local**: Enable/Disable of counters in Bbox will NOT track U_CSR_PERF_CTL.glb_en. Allows SW to write the .en bit. Disables Counters on any local counter overflow.
en	0	RW_RW	0	Enable/disable Bbox PMU counters. This bit is dependent on the setting of the .glb_lcl bit. If .glb_lcl is set to 1, SW writes to this bit are ignored and only HW may affect it's state. If .glb_lcl is set to 0, SW may exert control by setting the bit. In either case, since HW may alter this bit, (due to tracking the global enable or a local overflow) SW may read it to determine the state of the Bbox counters. 1: Enable Bbox PMU counting. 0: Disable (freeze) Bbox PMU counters.

Table 5-10. B_CSR_PMON_PERF_STATUS Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:8	RO_NA	0	Read zero; writes ignored. (?)
ov	7:0	RW_RO	0	If an overflow is detected from the corresponding Bbox PMON register, it's overflow bit will be set. ***Note:*** These are copies of the bits found in the individual registers. Therefore, these bits will be cleared when the counters are cleared.

It is is necessary to set the appropriate .*_*en* to 1 before the subcounter will track the occupancy of the queue it's associate with (i.e. set *.iob_en* to 1 to track IOB_OVFL).

Table 5-11. B_CSR_PERF_CTL0 Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Access	HW Reset Val	Description
ig	63:32	RO_NA	0	Read zero; writes ignored.
rsv	31:16	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.
ackcnflt_en	15	RW_RW	0	Include ackcnflt packets in IOB input events. NOTE: Does not have a completion packet sent. Therefore, this bit should be kept clear by default.
wrdata_en	14	RW_RW	0	1- include wr*data* (WB{I,S,E} data type) packets in IOB events. ***Note:*** For each write from the core, the Bbox gets two write packets. Therefore, this bit should be kept clear by default to prevent double counting.
iob_en	13	RW_RW	0	Enable IOB PMU sub counters. Only affects IOB overflow.

Table 5-11. B_CSR_PERF_CTL0 Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Access	HW Reset Val	Description
bz_en	12	RW_RW	0	Enable BZ PMU sub counters - needed to measure events 0x21,0x22, 0x24, 0x31 & 0x32 (overflow, first acknowledge and watermark)
dc_en	11	RW_RW	0	Enable DC PMU sub counters. Only affects DC overflow.
arb_en	10	RW_RW	0	enable arbiter PMU sub counters. Only affects arbiter overflow.
imt_en	9	RW_RW	0	enable IMT PMU sub counters.Only affects IMT overflow.
ig	8:0	RO_NA	0	Read zero; writes ignored.

Note: A safe default value for this register is 0x37e0 (or 0x3e00 should SW choose to manually handle enabling the appropriate sub-counters according to the events selected for the run).

5.4.3.2 Bbox PMU state - Counter/Control Pairs

The control for each of 8 Bbox performance monitors resides in a paired control register - B_CSR_PERF_CTL1_x. The task of this register is to select the event to be monitored by its corresponding data counter. Setting the *.ev_sel* performs the event selection.

Table 5-12. B_CSR_PERF_CTL1_{7-0} Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:6	RO_NA	0	Read zero; writes ignored.
ev_sel	5:0	RW_RW	0	Performance Counter select. The counter increments by 1 every clock the input is '1. B_CSR_PERF_CTL2 and B_CSR_PERF_CTL3 provide additional controls. See Table 5-11 for event definitions.

The Bbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-13. B_CSR_PERF_CNT{7-0} Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:49	RO_NA	0	Read zero; writes ignored.
ov	48	RW_RW	0	Overflow of the performance counter. NOTE: Writing 1 to this field may cause a freeze. Writing capability is for debug purposes and not expected to be used. See Section 5.4.2 for more information.
cnt	47:0	RW_RW	0	48-bit performance event counter

5.4.3.3 Bbox Performance Monitoring SubControl Register

The _CTL2 register is used to select subevents for the ARB and IOB events (refer to Section 5.4.5, “BBox Events Ordered By Code” for more detail).

Table 5-14. B_CSR_PERF_CTL2 Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Access	HW Reset Val	Description
ig	63:47	RO_NA	0	Read zero; writes ignored.
z_wmark_ge_le	46	RW_RW	0	0: BZ watermark is active when z_cnt <= z_wmark 1: BZ watermark is active when z_cnt >= z_wmark
z_wmark	45:40	RW_RW	0	Watermark of outstanding transactions to z box
z_op	39:24	RW_RW	0	Decoded opcode to Zbox. Fill2b events should be counted separately from any other opcodes.
iob_remote_local	23:22	RW_RW	0	0: count local or remote accesses. This option is only valid for IOB live transactions. 1: count local accesses 2: count remote accesses
iob_id_out	21	RW_RW	0	if 0: DNID of output Intel QPI packet is used to determine local or remote socket if 1: RHNID of output Intel QPI packet is used to determine local or remote socket
iob_id_in	20	RW_RW	0	if 0: RHNID of input Intel QPI packet is used to determine local or remote socket if 1: RSNID of input Intel QPI packet is used to determine local or remote socket
iob_trans_sel	19:18	RW_RW	0	This field affects selection of types of transactions that are used in accumulator of outstanding transactions in IOB. 0: match opcode and class (defined in B_CSR_PERF_CTL3 register) to output and input flits. 1: match opcode for input flits and count all outgoing *cmp* flits that correspond to the input flits. 2: track snoops (class, opcode and iob_id) selection has no effect.
arbq_wmark	17:12	RW_RW	0	Threshold field that can be used in conjunction with event 0xc
arbq_sel1	17:12	RW_RW	0	Arbiter queue selection for Arbiter event set1. The value programmed here selects the queue monitored by the event codes 0x0d to 0x11. However, the combinations are only meaningful for 0x10 (the non-empty case). For the other event the subfield is effectively ignored.

Table 5-14. B_CSR_PERF_CTL2 Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Access	HW Reset Val	Description
arbq_sel0	5:0	RW_RW	0	Arbiter queue selection for Arbiter event set0. The value programmed here selects the queue monitored by the event codes 0x08 to 0x0c. Possible values for this field are 0x00-0x35,0x37-0x39. However, not all of these values are supported by all the events with codes 0x08 to 0x0c. Refer to Table 5-15, "ARBQ_SEL0 - which queues are relevant for ARB_Q0* events." for more details.

Table 5-15. ARBQ_SEL0 - which queues are relevant for ARB_Q0* events.

arbq_sel0	Event 0x8 (OCCUPANCY)	Event 0x9 (INSERTS)	Event 0xA (REMOVE)	Event 0xB (NE_CYCLES)	Event 0xC (Thresholded OCCUPANCY)
0x3f-0x3a: NONE	---	---	---	---	---
0x39: Any Conflict Queue	---	Insert into any of the conflict queues	Remove from any of the conflict queues.	Cycles one or more of the conflict queues is not empty.	---
0x38: SAQ (OR of CLM, DRS, NCB, NCS)	---	Insert into any of the listed queues.	Remove from any of the listed queues.	Cycles one or more of the listed queues is not empty.	---
0x37: CLM	Y	Y	Y	Y	Y
0x36: None	---	---	---	---	---
0x35: COHQ	Y	Y	Y	Y	Y
0x34: NBAQ/NBR	Y	Y	Y	Y	Y
0x33: BRAQ/DRS	Y	Y	Y	Y	Y
0x32: WIAQ/NCB	Y	Y	Y	Y	Y
0x31: RIAQ/NCS	Y	Y	Y	Y	Y
0x30: Any Conflict Queue	Number of non-empty conflict queues.	Insert into any of the conflict queues	Remove from any of the conflict queues.	Cycles one or more of the conflict queues is not empty.	---
0x0-0x2f: Select one of the conflict queues	---	Y	Y	Y	---

NOTE regarding the ***iob_trans_sel***, ***iob_id_in***, ***iob_id_out***, and ***iob_remote_local*** events:

If you want to change *iob_trans_sel* from 0 or 1, you MUST write to PERF_CTL3 immediately afterwards. If *iob_trans_sel* is 0 or 1 and you want to change *iob_id_in*, you must write PERF_CTL3 immediately afterwards.

iob_trans_sel == 1 cannot be used in mirroring configurations.This is because under mirroring, the primary sometimes issues NcRd* to the slave instead of issuing Cmps for the incoming packet. Because iob_trans_sel == 1 causes a decrement of the live counter only on outgoing Cmp* and NOT on outgoing NcRd*, the live counter would end up with too many increments and no corresponding decrements causing the live counter to lose count of live transactions.

iob_trans_sel == 0 - This mode is most useful when mirroring is turned on. However, if you plan to use events 0x16-0x18 in this config when mirroring is turned on, then you need to:

Set PERF_CTL0.ackcnflt_en to 1, PERF_CTL2.remote_local to 03., PERF_CTL3.{*opcode_in*,*opcode_out}* to 0xffff4, PERF_CTL3.*class_in* to 0xfff9 and PERF_CTL3.*class_out* to 0x4014. In effect, if mirroring is enabled, *iob_trans_sel*== 0 can only be used to measure the average latency of all transactions (including AckCnflt) that are serviced by the Bbox. We cannot measure the latency of a subset of opcodes.

5.4.3.4 Bbox Register for IOB Related Mask/Match Facility

For many of the IOB events (that is, IOB_IN_PCKTS), it is possible to filter according to the opcode and message class of the packets. Since the match/mask fields are one-hot encoded, any combination of opcodes/message classes may be tracked simultaneously. Details for how to select among the available message classes and opcodes are included in Table 5-18, “Opcode Match by Message Class for the Bbox” and Table 5-18, “Opcode Match by Message Class for the Bbox”.

Table 5-16. B_CSR_PERF_CTL3 Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
opcode_out	63:48	RW	0	Match output message opcodes (decoded)
class_out	47:32	RW	0	Match output message class (decoded)
opcode_in	31:16	RW	0	Match input message opcodes (decoded)
class_in	15:0	RW	0	Match input message class (decoded). Note.When monitoring IOB latency, this filed should not be set to include message class “Snoop Responses” (MC=1). If MC=1 is selected by making class_in[1]=1, the latency values could be grossly exaggerated. Bbox IOB internal counters could be reset only by another write to this CSR. Until then the IOB live counters could be incorrect.

Note: SW should set this register to all 1s by default so that events are not filtered according to class or opcode.

Table 5-17. Intel® QuickPath Interconnect Packet Message Classes

Code	Name	Definition
0x0001	HOM0	Home - Requests
0x0002	HOM1	Home - Responses
0x0004	NDR	Non-Data Responses
0x0008	SNP	Snoops
0x0010	NCS	Non-Coherent Standard

0x0800	NCB	Non-Coherent Bypass

0x4000	DRS	Data Response

Table 5-18. Opcode Match by Message Class for the Bbox

Decoded Opcode	HOM0 (I)	HOM1 (I)	SNP (O)	DRS
0x0001	RdCur	RspI	SnpCur	(O) DataC_(EIMS)
0x0002	RdCode	RspS	SnpCode	---
0x0004	RdData	---	SnpData	(O) DataC_(EIMS)_Cmp
0x0008	---	---	---	(O) DataNc
0x0010	RdInvOwn	RspCnflt	SnpInvOwn	(I) WbIData
0x0020	InvXtoI	---	SnpInvXtoI	(I) WbSData
0x0040	EvctCln	---	---	(I) WbEData
0x0080	---	---	---	---
0x0100	InvItoE	RspFwd	SnpInvItoE	(I) WbIDataPtl
0x0200	---	RspFwdI	---	---
0x0400	---	RspFwdS	---	(I) WbEDataPtl
0x0800	---	RspFwdIWb	---	---
0x1000	WbMtoI	RspFwdSWb	---	---
0x2000	WbMtoE	RspIWb	---	---
0x4000	WbMtoS	RspSWb	---	---
0x8000	AckCnflt	---	---	---
	NDR	**NCB**	**NCS**	
0x0001	Gnt_Cmp	NcWr[1]	NcRd[1]	
0x0002	---	---	---	
0x0004	---	---	---	
0x0008	---	---	---	
0x0010	---	---	NcRdPtl[1]	
0x0020	---	---	---	
0x0040	---	---	---	
0x0080	---	---	---	
0x0100	Cmp	---	---	
0x0200	---	---	---	
0x0400	Cmp_FwdCode	---	---	
0x0800	Cmp_FwdInvOwn	---	---	
0x1000	Cmp_FwdInvItoE	NcWrPtl[1]	---	
0x2000	---	---	---	
0x4000	---	---	---	
0x8000	---	---	---	

[1] Only with memory mirroring enabled.

(I) indicates this opcode is only valid as input to the Bbox. (O) indicates this opcode is only valid as output from the Bbox. All others can be seen as either input or output. For more information about the opcodes, refer to Table 5-112, “Opcodes (Alphabetical Listing)”.

5.4.3.5 Bbox PMU Subcounter Registers - Subunit descriptions

For many of the Bbox queues, a subcounter register has been implemented to track transactions through the queue. Subsequent sections include tables detailing the subcounters for the following Bbox subunits:

IOB - The Input/Output Block. Tracks events that occur at the interfaces between the Bbox and Intel QPI ports.

BZ - Tracks requests initiated by the Bbox and end with an acknowledgement of the request by the Zbox. Each transaction represents only part of the latency between B and Zbox access. One Bbox read/write request may include more than one ack from its Zbox.

ARB - The ARB (arbitration queue) contains 37 queues: BRAQ, WIAQ, RIAQ, NBRAQ, COHQ and 32 conflict queues which are used to arbitrate amongst tracker/IMT entries for a given resource when the resource is not immediately available. Most ARB queues correspond to an Intel QPI message class. Each ARB queue is either the depth of the Tracker (512) or the depth of the IMT (32). Refer to Section 5.4.4.3, “Arbiter Events” for a more detailed description of each queue.

IMT - The In-flight Memory Table - tracks and serializes in-flight reads and writes to the Zbox. IMT also performs protocol serialization.

DC - Directory Cache

5.4.3.6 Bbox IOB PMU Register

The following table contains the subcounter for tracking transactions through the IOB. Overflows from the subcounter can be captured if the IOB_OVFL event is selected.

Note: Wr*data* (WB{I,S,E} data type) packets are not counted as IOB input events by default. To include Wr*Data* packets in IOB input events, set B_CSR_PERF_CTL0.wrdata_en=1. This needs to be done in addition to selecting the corresponding class_in and opcode_in. When both WbM* and Wb*Data* packets are selected and enabled, B_CSR_IOB_PERF_CNT.live_cnt may be non-0 when Bbox is idle.

IOB transactions have an age state(1bit) to indicate whether the output packet or a completion packet belongs to the current set of transactions that we want to capture. Every write to this register flips the age bit.

Table 5-19. B_CSR_IOB_PERF_CNT Register – Field Definitions

Field	Bits	***Note:*** Access	HW Reset Val	Description
ig	63:41	RO_NA	0	Read zero; writes ignored.
snp_live_cnt_remote	40:31	RO_RW	0	Number of currently outstanding remote snoops

Table 5-19. B_CSR_IOB_PERF_CNT Register – Field Definitions

Field	Bits	Note: Access	HW Reset Val	Description
snp_live_cnt_local	30:21	RO_RW	0	Number of currently outstanding local snoops
iob_live_cnt	20:11	RO_RW	0	Number of currently outstanding IOB transactions. WB{I/S/E}Data* packets do not have a separate acknowledge from Bbox. In order for the counter to become 0 when Bbox becomes idle, these packets are ignored by default. (class=drs, opcode=(4/5/6/7/8/10/11).To enable count of these packets, set B_CSR_.wrdata_en=1and write corresponding B_CSR_PERF_CTL3.class_in and opcode_in fields. Write to B_CSR_PERF_CTL3 clears IOB live counter.
accum_cnt	10:0	RW_RW	0	Accumulated outstanding IOB or snoop transactions

5.4.3.7 Bbox BZ PMU Registers

The following table contains the subcounter for tracking transactions through the BZ. Overflows from the subcounter can be captured if the BZ_OVFL event is selected.

Table 5-20. B_CSR_BZ_PERF_CNT Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:45	RO_NA	0	Read zero; writes ignored.
bz_state	44:31	RO_RW	0	Valid outstanding transactions in bz interface. Cleared when a write to B_CSR_PERFC_CTL2 occurs AND B_CSR_PERFC_CTL2.z_op is different from the previous setting.
live_cnt	12:7	RO_RW	0	Number of currently outstanding IOB transactions. Fill2Bbox requests are not acknowledged by Zbox. If B_CSR_PERF_CTL2.z_op is configured to count FILL2Bbox requests, a PMU acknowledge is generated one clock later in order for the live_cnt to become 0 when Bbox becomes idle. Cleared when a write to B_CSR_PERFC_CTL2 occurs AND B_CSR_PERFC_CTL2.z_op is different from the previous setting.
accum_cnt	6:0	RW_RW	0	Accumulated outstanding BZ transactions. Bit 6 is overflow.

5.4.3.8 Bbox ARB PMU Registers

The following tables contain the subcounters for tracking transactions through selected ARB queues. Overflows from the subcounter are accumulated as events. Refer to the ARB_Q*_OVFL events in Section 5.4.5, “BBox Events Ordered By Code” for more information.

Table 5-21. B_CSR_ARB_PERF_CNT0 Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	31:22	RO_NA	0	Read zero; writes ignored.

Table 5-21. B_CSR_ARB_PERF_CNT0 Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
live_cnt	21:12	RO_RW	0	This counter reflects the number of currently outstanding events determined by B_CSR_PERF_CTL2.arbq_sel0
ig	11	RO_NA	0	Read zero; writes ignored.
accum_cnt	10:0	RW_RW	0	This counter accumulates live_cnt. accum_cnt (n) = accum_cnt(n-1) + live_cnt. MSB is overflow that can trigger main PMU counters.

Table 5-22. B_CSR_ARB_PERF_CNT1 Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	31:23	RO_NA	0	Read zero; writes ignored.
live_cnt	22:12	RO_RW	0	This counter reflects the number of currently outstanding events determined by B_CSR_PERF_CTL2.arbq_sel1
accum_cnt	11:0	RW_RW	0	This counter accumulates live_cnt. accum_cnt (n) = accum_cnt(n-1) + live_cnt. MSB is overflow that can trigger main PMU counters.

5.4.3.9 Bbox IMT PMU Register

The following table contains the subcounter for tracking transactions through the IMT. Overflows from the subcounter can be captured if the IMT_OVFL event is selected.

Table 5-23. B_CSR_IMT_PERF_CNT Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
imt_valid	63:16	RO_RW	0	IMT Valid Bits - indicates which IMT entries are valid (occupied)
ig	15:13	RO_NA		Read zero; writes ignored.
live_cnt	12:7	RO_RW	0	Outstanding IMT transaction counter.
accum_cnt	6:0	RW_R	0	Accumulated IMT transactions. Overflow (bit 6) drives main performance counters.

5.4.3.10 Bbox DC PMU Registers

The following tables represent the state used to gather events related to the Directory Cache. The specified subevents are captured in the generic DC_EVENT* events.

The following table contains the subcounter for tracking transactions through the IMT. Overflows from the subcounter can be captured if one of the DC_EVENT*.TRANS_OCCUPANCY events is selected.

Table 5-24. B_CSR_DC_PERF_CNT Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:22	RO_NA	0	Read zero; writes ignored.
live_cnt	21:16	RW_RW	0	Outstanding dc transaction counter. The live counter is cleared by logic reset or a write to B_CSR_DC_PERF_CTL1. Freeze does not stop the live counter.
accum_cnt	6:0	RW_RW	0	Accumulated dc transactions. Overflow (bit 6) is connected to main performance counters.

Table 5-25. B_CSR_DC_PERF_CTL0 Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:20	RO_NA	0	Read zero; writes ignored.
event3	19:15	RW_RW	0	Select for dc event 3 - same selection as event0
event2	14:10	RW_RW	0	Select for dc event 2 - same selection as event0
event1	9:5	RW_RW	0	Select for dc event 1 - same selection as event0
event0	4:0	RW_RW	0	Select for dc event 0. ***Note:*** **Events 0 through 6 are affected by B_CSR_DC_PERF_CTL1.dc_opcode. B_CSR_PERF_CT0.dc_en needs to be set to 1 to enable events 0 through 7.** See Table 5-27 for a description of selections.

Table 5-26. Mux controls for B_CSR_DC_PERF_CTL1 (Sheet 1 of 2)

Name	Code	Description
TRANS_OVFL	0x0	Accumulator overflow. Outstanding transactions matching opcodes (B_CSR_DC_PERF_CTL1.dc_opcode) are accumulated in a 7-bit counter. Bit 6(oveflow) is connected to the main performance counters. To obtain the number of matching transactions, multiply the result in main counter by 2^6 and add B_CSR_DC_PERF_CNT.accum_cnt[5:0]
TRANS_INSERT	0x1	New transaction entering dc (insert). This event counts number of transactions entering the DC based on opcode matching (B_CSR_DC_PERF_CTL1.dc_opcode). NCRD and NCWR do not enter dc. Some RdData commands go through DC twice. The first pass is for a coarse-share MESI, and the second pass (if there is one) is for an exclusive MESI.Both passes correspond to the same entry in IMT. Number of dc inserts may differ from the number of IMT inserts.
TRANS_REMOVE	0x2	Transaction retired from dc (remove). This event counts number of transactions exiting the DC based on opcode matching (B_CSR_DC_PERF_CTL1.dc_opcode).
WMARK	0x3	Watermark event. Indicates that a number of currently outstanding transactions in dc (maximum is 32) is less and equal or greater and equal than a constant. (defined in B_CSR_DC_PERF_WMARK).
FULL	0x4	dc full. Indicates that DC has 32 outstanding transactions
NON_EMPTY	0x5	dc non-empty.

Table 5-26. Mux controls for B_CSR_DC_PERF_CTL1 (Sheet 2 of 2)

Name	Code	Description
MATCH_READ	0x6	Matching dc read event. Upon receiving a request, dc reads it's cache. The data read from the cache is matched to the criteria defined in B_CSR_DC_PERF_MATCH_RD. The selected events are masked with B_CSR_DC_PERF_MASK_RD. Affected by B_CSR_DC_PERF_CTL1.dc_opcode. dc_read_event = ((DC_CACHE_DATA_RD xnor B_CSR_DC_PERF_MATCH_RD) or B_CSR_DC_PERF_MASK_RD) and MATCH_DC_OPCODES AND NOT 2nd_LOOKUP
MATCH_WRITE	0x7	Matching dc write event. Upon receiving a request dc may write to it's cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR. The selected events are masked with B_CSR_DC_PERF_MASK_WR. dc_write_event = (DC_CACHE_DATA_WR xnor B_CSR_DC_PERF_MATCH_WR) or B_CSR_DC_PERF_MASK_WR.
NEED_SNOOPS	0x8	new transaction requires snoops. Indicates that single or multiple snoops are required. No matching is performed.
LINE_EVICT	0x9	DC line eviction event
MATCH_READ_2ND_LOOK	0xa	Matching DC read second lookup. Works the same way as 6, with the exception that second DC read lookups cause this event.
SNP_LOC	0xb	Measure snoops caused by cell local memory accesses.
MATCH_WRITE	0xc	Matching dc write event 2. This utilizes a second set of match/match registers. Upon receiving a request dc may write to it's cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR2. The selected events are masked with B_CSR_DC_PERF_MASK_WR2. dc_write_event2 = (DC_CACHE_DATA_WR2 xnor B_CSR_DC_PERF_MATCH_WR2) or B_CSR_DC_PERF_MASK_WR2.
LINE_ALLOCATE	0xd	DC line allocation event

Table 5-27. B_CSR_DC_PERF_CTL1 Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:16	RO_NA		Read zero; writes ignored.
dc_opcode	15:0	RW_RW	0	Match opcode(decoded) of input/output transactions entering and leaving dc to be counted by live counter of outstanding transactions. The same opcode is used to match opcodes within matching dc read and write events. The opcode types are defined in Intel QPI. Since the opcode field represents a decoded opcode, multiple opcodes can be selected simultaneously. Affects event{0-3} values 0 through 6. NOTE: Since this acts as a filter, SW should set the default value of this field to ALL 1s. The DC uses the same Message Class, Opcode encoding as QP.

Table 5-28. B_CSR_DC_PERF_MATCH_RD Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Access	HW Reset Val	Description
ig	63:37	RO_NA	0	Read zero; writes ignored.
match_2nd_read_pass	36	RW_RW	0	match 2nd read pass
match_buddy	35	RW_RW	0	match buddy bit

Table 5-28. B_CSR_DC_PERF_MATCH_RD Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Access	HW Reset Val	Description
match_dir	34:23	RW_RW	0	match 11 bits of directory info
match_bank	22	RW_RW	0	match bank bit
match_hit	21:1	RW_RW	0	match dc hit bits
match_hit_or	0	RW_RW	0	match hit (bit OR of 16 dc hits)

Table 5-29. B_CSR_DC_PERF_MASK_RD Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:37	RO_NA	0	Read zero; writes ignored.
mask_2nd_read_pass	36	RW_RW	0	mask 2nd read pass
mask_buddy	35	RW_RW	0	mask buddy bit
mask_dir	34:23	RW_RW	0	mask 11 bits of directory info
mask_bank	22	RW_RW	0	mask bank bit
mask_hit	21:1	RW_RW	0	mask DC hit bits
mask_hit_or	0	RW_RW	0	mask hit (bit OR of 16 DC hits)

Table 5-30. B_CSR_DC_PERF_MATCH_WR{1,2} Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:37	RO_NA	0	Read zero; writes ignored.
match_snp_gen	36	RW_RW	0	match if snoop is generated
match_imt	30	RW_RW	0	match IMT update
match_bank	29	RW_RW	0	match bank
match_buddy	28	RW_RW	0	match buddy bit
match_dir	27:16	RW_RW	0	match directory bits
match_way	15:0	RW_RW	0	match cache way write

Table 5-31. B_CSR_DC_PERF_MASK_WR{1,2} Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:37	RO_NA	0	Read zero; writes ignored.
mask_snp_gen	36	RW_RW	0	mask if snoop is generated
mask_imt	35	RW_RW	0	mask IMT update
mask_bank	34	RW_RW	0	mask bank
mask_buddy	33	RW_RW	0	mask buddy bit
mask_dir	32:21	RW_RW	0	mask directory bits
mask_way	20:0	RW_RW	0	mask cache way write

Table 5-32. B_CSR_DC_PERF_WMARK Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:7	RO_NA	0	Read zero; writes ignored.
wmark_ge_le	6	RW_RW	0	0: Watermark event occurs when DC trans_cnt <= wmark 1: Watermark event occurs when DC trans_cnt >= wmark
wmark	5:0	RW_RW	0	Watermark of outstanding events in DC. B_CSR_DC_PERF_CNT.trans_cnt is compared to the value in this field.

5.4.4 Bbox Performance Monitoring Events

5.4.4.1 IOB Events

IOB events are events that occur at the Bbox *Intel QuickPath Interconnect* interfaces. These events can provide information that relates to overall Bbox performance.

Events generated by IOB are configured with the B_CSR_PERF_CTL1 and B_CSR_PERF_CTL2 registers.

IOB related latency monitoring is made possible by the existence of two sub-counters inside the IOB. The first is a 10b counter that maintains the current number of live outstanding transactions that match the criteria determined by B_CSR_CTL3.{class_in,opcode_in}. The second is an 11b counter which accumulates these outstanding transactions every cycle. The overflow of this second counter is connected to the main PMU counters as an IOB event. The two IOB sub-counters are accessible through the B_CSR_IOB_PERF_CNT register.

IOB events are selected by opcode, class and locality (remote/local/any). Any set of different class and opcode packets can be selected for counting. To calculate average latency, snoop requests and responses need to be selected separately. To determine the latency of non-snoop packets, any completion packet can be selected by setting B_CSR_PERF_CTL2.iob_trans_sel to 1. Wr*data* (WB{I,S,E} Data Type) packets are not selected by default in B_CSR_PERF_CTL3. This is done to obtain correct latency results for WR* packets.

Input packet locality is determined by either **RHNID** (Requester Home Node ID) or **RSNID** (Requester Snoop Node ID). Output packet locality is determined by either **DNID** (Destination ID) or RHNID.

Four types of response forward packets can be counted, local to local, remote to local, local to remote and remote to remote.

5.4.4.2 BZ Events

BZ requests are initiated by the Bbox and are completed when the Zbox acknowledges (‘ack’s) the request. This is a partial latency of a Bbox to Zbox access. One Bbox read or write request may include more than one ack from the Zbox.

Two counters exist to compute the average latency of BZ requests. The first is a 6b that maintains the current number of outstanding requests. The second is a 7b counter which accumulates these outstanding requests each cycles. The overflow of the second counter is connected to the main PMU counters as a BZ event (BZ_OVFL).

Since FILL2B opcodes are not acknowledged by the Zbox, this opcode should be excluded from the Zbox opcode selection when average latency data is required.

Note: To determine average Zbox Latency, one can measure BZ_OVFL / BZ_ACK. These events can be further filtered by Zbox opcode. See appropriate event descriptions for more information.

5.4.4.3 Arbiter Events

The arbiter contains 37 queues: BRAQ, WIAQ, RIAQ, NBRAQ, COHQ and 32 conflict queues. The arbiter sub counters are 10 and 11b respectively and can either accumulate the outstanding transactions in one of the queues or accumulate the number of non-empty arbiter queues. B_CSR_PERF_ARB_PERF_CNT{1-0} provide access to these sub-counters. The B_CSR_PERF_CTL2 CSR contains fields controlling the counters. Overflows of the two counters are connected to main PMU counters as arbiter events ARB_Q{0,1}_OVFL.

The arbitration queues are used to arbitrate amongst Tracker/IMT entries for a given resource when the resource is not immediately available. Examples of unavailable resources are: QuickPath Interconnect credits to send a packet out of the Bbox, the Rbox output port is busy or the IMT is full. Most ARB queues correspond to QuickPath Interconnect message classes. Two of the ARB queues (the COHQ and the CNFLTQ) are used to arbitrate for access to the IMT itself.

Following arbQs are present in the design:

- 1) RIAQ (Read ARBq) – this queue holds failover reads that target a Bbox slave. It is used only in mirroring flows. 32 entries.
- 2) WIAQ (Write ARBq) – this queue holds writes that target a Bbox slave. It is used only in migration/mirroring flows. 32 entries.
- 3) BRAQ (Block Response ARBq)- this queue holds requests that are waiting to send DataC_x_Cmp to the requestor. The reason these are queued up in the BRAQ is either due to lack of credits or output port (to Rbox) business. 32 entries.
- 4) NBRAQ (Non-Block Response ARBq) – this queue holds requests that are waiting to send a Cmp (NDR) packet to the requestor. The reason these are queued up in the NBRAQ is either due to lack of credits or output port (to Rbox) business. Note that in the case of NBRAQ, the IMT entry may already have been de-allocated (such is not the case for BRAQ though). 512 entries.
- 5) COHQ (aka RSRCQ) – this queue holds requests that encountered a full IMT. Since the home message class (new requests) must be sunk due to tracker pre-allocation, a queue must keep track of entries that could not be immediately allocated into the IMT. The COHQ serves this purpose. 512 entries.
- 6) CLMQ – (Cell Local Memory Queue) this queue is used to implement the CLM-mode3 optimization. Writebacks from IOH agents are pushed into this queue till the DC has written out the hint indicating who the next consumer owner of the line would be. 32 entries.
- 7) CNFLTQ (Conflict Queues) – this is a collection of 32 separate queues (one per each IMT entry) of 64 entry deep each. The queue holds all conflictors (max 64 since the Bbox only supports a 6b nid, and each caching agent can have at most one request to the same address).

5.4.4.4 Directory Cache Events

Two counters have been provided to calculate the average latency of transactions that enter and leave directory cache. The first is a 6b counter that maintains the current number of outstanding transactions, and the second is a 7b counter that accumulates these outstanding transactions. B_CSR_DC_PERF_CNT register provides access to the counters. The transactions that are counted are gated with transaction opcodes (subset of QuickPath Interconnect opcodes). Two CSR registers, B_CSR_DC_PERF_CTL0 and B_CSR_DC_PERF_CTL1 provide control of the DC events. Four DC events are generated. The four events DC_EVENT{3-0} are connected to the main PMU counters.

In addition, directory cache read and write events can be generated. The read and write events are gated with match and mask registers(B_CSR_DC_PERF_MATCH_RD, B_CSR_DC_PERF_MATCH_RD, B_CSR_DC_PERF_MATCH_WR, B_CSR_DC_PERF_MATCH_WR).

Additional DC events can be found in the B_CSR_DC_ CSR definition.

5.4.4.5 IMT Events

In-flight memory table events enable calculation of average occupation of IMT. There are 5 IMT events: sub-counter overflow, pop conflict, allocate, non-empty and full. The number of IMT inserts is a sum of pop conflict and allocate events. All IMT events are defined in B_CSR_PERF_CTL1.

5.4.5 BBox Events Ordered By Code

Table 5-33 summarizes the directly-measured BBox events.

Table 5-33. Performance Monitor Events for BBox Events (Sheet 1 of 2)

Symbol Name	Event Code	SubCtl Dep?	Max Inc/Cyc	Description
RFP_LOC_LOC	0x00		1	RFP Local/Local
RFP_REM_LOC	0x01		1	RFP Remote/Local
RFP_LOC_REM	0x02		1	RFP Local/Remote
RFP_REM_REM	0x03		1	RFP Remote/Remote
IMT_OCCUPANCY	0x04		1	IMT Occupancy
IMT_ALLOC	0x05		1	IMT Allocations
IMT_POP_CFL	0x06		1	IMT Pop Conflicts
IMT_NE_CYCLES	0x07		1	Cycles IMT Not Empty
ARB_Q0_OCCUPANCY	0x08	CTL2	1	Arb Queue 0 Occupancy
ARB_Q0_INSERTS	0x09	CTL2	1	Arb Queue 0 Inserts
ARB_Q0_REMOVE	0x0A	CTL2	1	Arb Queue 0 Remove
ARB_Q0_NE_CYCES	0x0B	CTL2	1	Cycles Arb Queue 0 Not Empty
ARB_Q0_THOCCUPANCY	0x0C	CTL2	1	Arb Queue 0 Occupancy (Thresholded)
ARB_Q1_OCCUPANCY	0x0D	CTL2	1	Arb Queue 1 Occupancy
ARB_Q1_INSERTS	0x0E	CTL2	1	Arb Queue 1 Inserts
ARB_Q1_REMOVE	0x0F	CTL2	1	Arb Queue 1 Remove
ARB_Q1_NE_CYCES	0x10	CTL2	1	Cycles Arb Queue 1 Not Empty
ARB_Q1_EMPTY_INSERT	0x11	CTL2	1	Insert to Empty Arbiter Queue

Table 5-33. Performance Monitor Events for BBox Events (Sheet 2 of 2)

Symbol Name	Event Code	SubCtl Dep?	Max Inc/Cyc	Description
DC_EVENT0	0x12	DC_CTL0 ,DC_WM ARK	1	Directory Cache Event 0
DC_EVENT1	0x13	DC_CTL0 ,DC_WM ARK	1	Directory Cache Event 1
DC_EVENT2	0x14	DC_CTL0 ,DC_WM ARK	1	Directory Cache Event 2
DC_EVENT3	0x15	DC_CTL0 ,DC_WM ARK	1	Directory Cache Event 3
IOB_OCCUPANCY	0x16		1	IOB Occupancy
IOB_INSERTS	0x17	CTL2	1	IOB Inserts of Live Trans
IOB_REMOVES	0x18	CTL2	1	IOB Removals of Live Trans
IOB_IN_PKTS	0x19	CTL2	1	IOB Input Packets
IOB_OUT_PKTS	0x1A	CTL2	1	IOB Output Packets
SNP_REQ_ALL	0x1B		1	All Snoop Requests
SNP_REQ_LOC	0x1C		1	Local Snoop Requests
SNP_REQ_REM	0x1D		1	Remote Snoop Requests
SNP_RSP_ALL	0x1E		1	All Snoop Responses
SNP_RSP_LOC	0x1F		1	Local Snoop Responses
SNP_RSP_REM	0x20		1	Remote Snoop Responses
BZ_OCCUPANCY	0x21		1	BZ Occupancy
BZ_ACK	0x22		1	BZ Acknowledge
BZ_ACK_ALL	0x23		1	All BZ Acknowledges
BZ_CYCLES_TRANS	0x24	CTL2	1	Cycles Z Transactions Outstanding
B_CYCLES	0x25		1	Bbox Clock Cycles
NSL_SUCC	0x26	1	1	NSL Success
TRACKER_IMT_HAZARD	0x27	1	1	Tracker/IMT Hazard
NSL_REJ	0x28	1	1	NSL Reject
IMT_FULL_CURR_PIPE	0x29	1	1	Cycles IMT Full for Current Pipe Pass
IMT_FULL	0x2A	1	1	Cycles IMT Full
NSL_EVENT0	0x2B	1	1	NSL Event 0
NSL_EVENT1	0x2C	1	1	NSL Event 1
MEM_HINT_DC_LUP	0x2D	1	1	Memory Hint in DC Lookup
MEM_HINT_Z_RESP	0x2E	1	1	Memory Hint in Zbox Response
POISON_RECV	0x2F	1	1	Poison Packet Received
POISON_SENT	0x30	1	1	Poison Packet Sent
BZ_OP_MATCH	0x31	1	1	BZ Request Opcode Match
Z_OPT_V2V	0x32	1	1	Zbox V2V Optimized Requests

5.4.6 Bbox Performance Monitor Event List

This section enumerates Itanium® processor 9500 series uncore performance monitoring events for the Bbox.

ARB_Q0_INSERTS

- **Title:** Arb Queue 0 Inserts
- **Category:** ARB
- **Event Code:** 0x9, **Max. Inc/Cyc:** 1,
- **Definition:** An insert (write) to the selected ARB queue.

Extension	CTL2[5:0]	Description
CFL0	0x00	Conflict Queue 0
CFL1	0x01	Conflict Queue 1
CFL2	0x02	Conflict Queue 2
CFL3	0x03	Conflict Queue 3
CFL4	0x04	Conflict Queue 4
CFL5	0x05	Conflict Queue 5
CFL6	0x06	Conflict Queue 6
CFL7	0x07	Conflict Queue 7
CFL8	0x08	Conflict Queue 8
CFL9	0x09	Conflict Queue 9
CFL10	0x0a	Conflict Queue 10
CFL11	0x0b	Conflict Queue 11
CFL12	0x0c	Conflict Queue 12
CFL13	0x0d	Conflict Queue 13
CFL14	0x0e	Conflict Queue 14
CFL15	0x0f	Conflict Queue15
CFL16	0x10	Conflict Queue 16
CFL17	0x11	Conflict Queue 17
CFL18	0x12	Conflict Queue 18
CFL19	0x13	Conflict Queue 19
CFL20	0x14	Conflict Queue 20
CFL21	0x15	Conflict Queue 21
CFL22	0x16	Conflict Queue 22
CFL23	0x17	Conflict Queue 23
CFL24	0x18	Conflict Queue 24
CFL25	0x19	Conflict Queue 25
CFL26	0x1a	Conflict Queue 26
CFL27	0x1b	Conflict Queue 27
CFL28	0x1c	Conflict Queue 28
CFL29	0x1d	Conflict Queue 29
CFL30	0x1e	Conflict Queue 30
CFL31	0x1f	Conflict Queue 31
CFL32	0x20	Conflict Queue 33

Extension	CTL2[5:0]	Description
CFL33	0x21	Conflict Queue 32
CFL34	0x22	Conflict Queue 34
CFL35	0x23	Conflict Queue 35
CFL36	0x24	Conflict Queue 36
CFL37	0x25	Conflict Queue 37
CFL38	0x26	Conflict Queue 38
CFL39	0x27	Conflict Queue 39
CFL40	0x28	Conflict Queue 40
CFL41	0x29	Conflict Queue 41
CFL42	0x2a	Conflict Queue 44
CFL44	0x2b	Conflict Queue 42
CFL44	0x2c	Conflict Queue 44
CFL45	0x2d	Conflict Queue 45
CFL46	0x2e	Conflict Queue 46
CFL47	0x2f	Conflict Queue 47
ANY_CFL	0x30	Any Conflict Queue
RIAQ	0x31	RIAQ(NCS) Queue
WIAQ	0x32	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue
NBRAQ	0x34	NBAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
SAQ	0x38	Or of RIAQ,WIAQ,BRAQ and CLM Queues
ANY_CFL2	0x39	Any Conflict Queue
NONE	0x3f-0x3a	(* nothing will be counted *)

ARB_Q0_NE_CYCLES

- **Title:** Cycles Arb Queue 0 Not Empty
- **Category:** ARB
- **Event Code:** 0xb, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles when the selected queue(s) is not empty.

Extension	CTL2[5:0]	Description
CFL0	0x00	Conflict Queue 0
CFL1	0x01	Conflict Queue 1
CFL2	0x02	Conflict Queue 2
CFL3	0x03	Conflict Queue 3
CFL4	0x04	Conflict Queue 4
CFL5	0x05	Conflict Queue 5
CFL6	0x06	Conflict Queue 6
CFL7	0x07	Conflict Queue 7
CFL8	0x08	Conflict Queue 8

Extension	CTL2[5:0]	Description
CFL9	0x09	Conflict Queue 9
CFL10	0x0a	Conflict Queue 10
CFL11	0x0b	Conflict Queue 11
CFL12	0x0c	Conflict Queue 12
CFL13	0x0d	Conflict Queue 13
CFL14	0x0e	Conflict Queue 14
CFL15	0x0f	Conflict Queue15
CFL16	0x10	Conflict Queue 16
CFL17	0x11	Conflict Queue 17
CFL18	0x12	Conflict Queue 18
CFL19	0x13	Conflict Queue 19
CFL20	0x14	Conflict Queue 20
CFL21	0x15	Conflict Queue 21
CFL22	0x16	Conflict Queue 22
CFL23	0x17	Conflict Queue 23
CFL24	0x18	Conflict Queue 24
CFL25	0x19	Conflict Queue 25
CFL26	0x1a	Conflict Queue 26
CFL27	0x1b	Conflict Queue 27
CFL28	0x1c	Conflict Queue 28
CFL29	0x1d	Conflict Queue 29
CFL30	0x1e	Conflict Queue 30
CFL31	0x1f	Conflict Queue 31
CFL32	0x20	Conflict Queue 33
CFL33	0x21	Conflict Queue 32
CFL34	0x22	Conflict Queue 34
CFL35	0x23	Conflict Queue 35
CFL36	0x24	Conflict Queue 36
CFL37	0x25	Conflict Queue 37
CFL38	0x26	Conflict Queue 38
CFL39	0x27	Conflict Queue 39
CFL40	0x28	Conflict Queue 40
CFL41	0x29	Conflict Queue 41
CFL42	0x2a	Conflict Queue 44
CFL44	0x2b	Conflict Queue 42
CFL44	0x2c	Conflict Queue 44
CFL45	0x2d	Conflict Queue 45
CFL46	0x2e	Conflict Queue 46
CFL47	0x2f	Conflict Queue 47
ANY_CFL	0x30	Any Conflict Queue
RIAQ	0x31	RIAQ(NCS) Queue
WIAQ	0x32	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue

Extension	CTL2[5:0]	Description
NBRAQ	0x34	NBAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
SAQ	0x38	Or of RIAQ,WIAQ,BRAQ and CLM Queues
ANY_CFL2	0x39	Any Conflict Queue
NONE	0x3f-0x3a	(* nothing will be counted *)

ARB_Q0_OCCUPANCY

- **Title:** Arb Queue 0 Occupancy
- **Category:** ARB
- **Event Code:** 0x8, **Max. Inc/Cyc:** 1,
- **Definition:** Overflow from arbiter subcounter accumulating live events B_CSR_PERF_CTL2.arbq_sel0 selects which arbiter queue(s) to monitor.
- **NOTE:** Set B_CSR_PERF_CTL0.arb_en to enable the subcounter which, in turn, enables this event. To obtain the number of arbiter transactions, multiply by 2^10 and add B_CSR_ARB_PERF_CNT0.accum_cnt[9:0]

Extension	CTL2[5:0]	Description
NONE	0x2f-0x00	(* nothing will be counted *)
NE_CFL	0x30	Number of non-empty conflict queues.
RIAQ	0x31	RIAQ(NCS) Queue
WIAQ	0x33	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue
NBRAQ	0x34	NBAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
NONE2	0x3f-0x38	(* nothing will be counted *)

ARB_Q0_REMOVE

- **Title:** Arb Queue 0 Remove
- **Category:** ARB
- **Event Code:** 0xa, **Max. Inc/Cyc:** 1,
- **Definition:** A remove (read) from the selected ARB queue.

Extension	CTL2[5:0]	Description
CFL0	0x00	Conflict Queue 0
CFL1	0x01	Conflict Queue 1
CFL2	0x02	Conflict Queue 2
CFL3	0x03	Conflict Queue 3
CFL4	0x04	Conflict Queue 4
CFL5	0x05	Conflict Queue 5
CFL6	0x06	Conflict Queue 6

Extension	CTL2[5:0]	Description
CFL7	0x07	Conflict Queue 7
CFL8	0x08	Conflict Queue 8
CFL9	0x09	Conflict Queue 9
CFL10	0x0a	Conflict Queue 10
CFL11	0x0b	Conflict Queue 11
CFL12	0x0c	Conflict Queue 12
CFL13	0x0d	Conflict Queue 13
CFL14	0x0e	Conflict Queue 14
CFL15	0x0f	Conflict Queue15
CFL16	0x10	Conflict Queue 16
CFL17	0x11	Conflict Queue 17
CFL18	0x12	Conflict Queue 18
CFL19	0x13	Conflict Queue 19
CFL20	0x14	Conflict Queue 20
CFL21	0x15	Conflict Queue 21
CFL22	0x16	Conflict Queue 22
CFL23	0x17	Conflict Queue 23
CFL24	0x18	Conflict Queue 24
CFL25	0x19	Conflict Queue 25
CFL26	0x1a	Conflict Queue 26
CFL27	0x1b	Conflict Queue 27
CFL28	0x1c	Conflict Queue 28
CFL29	0x1d	Conflict Queue 29
CFL30	0x1e	Conflict Queue 30
CFL31	0x1f	Conflict Queue 31
CFL32	0x20	Conflict Queue 33
CFL33	0x21	Conflict Queue 32
CFL34	0x22	Conflict Queue 34
CFL35	0x23	Conflict Queue 35
CFL36	0x24	Conflict Queue 36
CFL37	0x25	Conflict Queue 37
CFL38	0x26	Conflict Queue 38
CFL39	0x27	Conflict Queue 39
CFL40	0x28	Conflict Queue 40
CFL41	0x29	Conflict Queue 41
CFL42	0x2a	Conflict Queue 44
CFL44	0x2b	Conflict Queue 42
CFL44	0x2c	Conflict Queue 44
CFL45	0x2d	Conflict Queue 45
CFL46	0x2e	Conflict Queue 46
CFL47	0x2f	Conflict Queue 47
ANY_CFL	0x30	Any Conflict Queue
RIAQ	0x31	RIAQ(NCS) Queue

Extension	CTL2[5:0]	Description
WIAQ	0x32	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue
NBRAQ	0x34	NBAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
SAQ	0x38	Or of RIAQ,WIAQ,BRAQ and CLM Queues
ANY_CFL2	0x39	Any Conflict Queue
NONE	0x3f-0x3a	(* nothing will be counted *)

ARB_Q0_THOCCUPANCY

- **Title:** Arbiter Queue 0 Occupancy (Thresholded)
- **Category:** ARB
- **Event Code:** 0xc, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles selected arbiter queue's occupancy exceeds threshold.
- **NOTE:** Set B_CSR_PERF_CTL0.arb_en to enable the subcounter which, in turn, enables this event. To obtain the number of arbiter transactions, multiply by 2^10 and add B_CSR_ARB_PERF_CNT0.accum_cnt[9:0]. The occupancy of the selected ARBQ is tested against the B_CSR_PERF_CTL2.arb_thresh (bits [17:12]). In any given cycle, if the occupancy exceeds the threshold, this event will be recorded.

Extension	CTL2[5:0]	Description
NONE	0x30-0x00	(* nothing will be counted *)
RIAQ	0x31	RIAQ(NCS) Queue NOTE: Number of entries in the queue equals or exceeds the number set in B_CSR_PERF_CTL2.ARBQ_WMARK
WIAQ	0x32	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue
NBRAQ	0x34	NBRAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE2	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
SAQ	0x38	Or of CLM ,BRAQ, RIAQ and WIAQ Queues
ANY_CFL2	0x39	First Insert into any empty conflict queue
NONE	0x3f-0x3a	(* nothing will be counted *)

ARB_Q1_EMPTY_INSERT

- **Title:** Insert to Empty Arbiter Queue
- **Category:** ARB
- **Event Code:** 0x11, **Max. Inc/Cyc:** 1,
- **Definition:** Insert to any empty arb counter(s).

ARB_Q1_INSERTS

- **Title:** Arb Queue 1 Inserts
- **Category:** ARB
- **Event Code:** 0xe, **Max. Inc/Cyc:** 1,
- **Definition:** An insert (write) to any conflict queue.

ARB_Q1_NE_CYCLES

- **Title:** Cycles Arb Queue 1 Not Empty
- **Category:** ARB
- **Event Code:** 0x10, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles when the selected queue(s) is not empty.

Extension	CTL2[11:6]	Description
CFL0	0x00	Conflict Queue 0
CFL1	0x01	Conflict Queue 1
CFL2	0x02	Conflict Queue 2
CFL3	0x03	Conflict Queue 3
CFL4	0x04	Conflict Queue 4
CFL5	0x05	Conflict Queue 5
CFL6	0x06	Conflict Queue 6
CFL7	0x07	Conflict Queue 7
CFL8	0x08	Conflict Queue 8
CFL9	0x09	Conflict Queue 9
CFL10	0x0a	Conflict Queue 10
CFL11	0x0b	Conflict Queue 11
CFL12	0x0c	Conflict Queue 12
CFL13	0x0d	Conflict Queue 13
CFL14	0x0e	Conflict Queue 14
CFL15	0x0f	Conflict Queue15
CFL16	0x10	Conflict Queue 16
CFL17	0x11	Conflict Queue 17
CFL18	0x12	Conflict Queue 18
CFL19	0x13	Conflict Queue 19
CFL20	0x14	Conflict Queue 20
CFL21	0x15	Conflict Queue 21
CFL22	0x16	Conflict Queue 22
CFL23	0x17	Conflict Queue 23
CFL24	0x18	Conflict Queue 24
CFL25	0x19	Conflict Queue 25
CFL26	0x1a	Conflict Queue 26
CFL27	0x1b	Conflict Queue 27
CFL28	0x1c	Conflict Queue 28
CFL29	0x1d	Conflict Queue 29
CFL30	0x1e	Conflict Queue 30
CFL31	0x1f	Conflict Queue 31

Extension	CTL2[11:6]	Description
CFL32	0x20	Conflict Queue 33
CFL33	0x21	Conflict Queue 32
CFL34	0x22	Conflict Queue 34
CFL35	0x23	Conflict Queue 35
CFL36	0x24	Conflict Queue 36
CFL37	0x25	Conflict Queue 37
CFL38	0x26	Conflict Queue 38
CFL39	0x27	Conflict Queue 39
CFL40	0x28	Conflict Queue 40
CFL41	0x29	Conflict Queue 41
CFL42	0x2a	Conflict Queue 44
CFL44	0x2b	Conflict Queue 42
CFL44	0x2c	Conflict Queue 44
CFL45	0x2d	Conflict Queue 45
CFL46	0x2e	Conflict Queue 46
CFL47	0x2f	Conflict Queue 47
ANY_CFL	0x30	Any Conflict Queue
RIAQ	0x31	RIAQ(NCS) Queue
WIAQ	0x33	WIAQ(NCB) Queue
BRAQ	0x33	BRAQ(DRS) Queue
NBRAQ	0x34	NBAQ(NBR) Queue
COHQ	0x35	COHQ Queue
NONE	0x36	(* nothing will be counted *)
CLM	0x37	CLM Queue
SAQ	0x38	Or of RIAQ,WIAQ,BRAQ and CLM Queues
ANY_CFL2	0x39	Any Conflict Queue
NONE	0x3f-0x3a	(* nothing will be counted *)

ARB_Q1_OCCUPANCY

- **Title:** Arb Queue 1 Occupancy
- **Category:** ARB
- **Event Code:** 0xd, **Max. Inc/Cyc:** 1,
- **Definition:** Overflow from arbiter subcounter accumulating live events. The live counter is fed by the number of entries occupied by all conflict queues.
- **NOTE:** Set B_CSR_PERF_CTL0.arb_en to enable the subcounter which, in turn, enables this event. To obtain the number of arbiter transactions, multiply by 2^11 and add B_CSR_ARB_PERF_CNT0.accum_cnt[10:0]

ARB_Q1_REMOVE

- **Title:** Arb Queue 1 Remove
- **Category:** ARB
- **Event Code:** 0xf, **Max. Inc/Cyc:** 1,
- **Definition:** A remove (read) from any conflict queue.

B_CYCLES

- **Title:** Bbox Clock Cycles
- **Category:** Miscellaneous
- **Event Code:** 0x25, **Max. Inc/Cyc:** 1,
- **Definition:** Increment by 1 every clock (when the counter is enabled)

BZ_ACK

- **Title:** BZ Acknowledge
- **Category:** BZ
- **Event Code:** 0x22, **Max. Inc/Cyc:** 1,
- **Definition:** First BZ acknowledge that acknowledges a BZ command that matched B_CSR_PERF_CTL2.z_op.
- **NOTE:** Enabled by B_CSR_PERF_CTL0.bz_en. Since fill2b requests are not acknowledged by Zbox, a local acknowledge is created so count of all requests equals count of all acknowledges. Fill2b events should be counted separately from any other opcodes (configured by B_CSR_PERF_CTL2.z_op (CTL2[36:21])). Used to compute avg. Zbox latency; (if this event is used to monitor latency need to exclude fill2b).

BZ_ACK_ALL

- **Title:** All BZ Acknowledges
- **Category:** BZ
- **Event Code:** 0x23, **Max. Inc/Cyc:** 1,
- **Definition:** Number of BZ commands acknowledged by Zbox. Not possible to pick commands; Possible that Zbox sent multiple acks per command it received.
- **NOTE:** Fill2b requests from the Bbox are not acknowledges by Zbox and therefore are not included in this event

BZ_OP_MATCH

- **Title:** BZ Request Opcode Match
- **Category:** BZ
- **Event Code:** 0x31, **Max. Inc/Cyc:** 1,
- **Definition:** BZ requests that match opcodes selected in B_CSR_PERF_CTL2.z_op

BZ_OCCUPANCY

- **Title:** BZ Occupancy
- **Category:** BZ
- **Event Code:** 0x21, **Max. Inc/Cyc:** 1,
- **Definition:** Cumulative number of selected BZ commands outstanding in Zbox. Fully decoded bits are present to select combination of opcodes
- **NOTE:** Enabled by B_CSR_PERF_CTL0.bz_en. To obtain correct number of BZ Transactions, multiply by 2^6 and add B_CSR_BZ_PERF_CNT.accum_cnt[5:0]. Since fill2b are not acknowledged by Zbox, fill2b opcode should be excluded from B_CSR_PERF_CTL2.z_op (CTL2[36:21]). Used to compute avg. Zbox latency; Partial latency of Bbox included (customers should program correctly to exclude fill2b)

BZ_CYCLES_TRANS_OUT

- **Title:** Cycles Z Trans Outstanding
- **Category:** BZ
- **Event Code:** 0x24, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles there are outstanding transactions destined for the Zbox within a given threshold.
- **NOTE:** Enabled by B_CSR_PERF_CTL0.bz_en.

Extension	CTL2[43]	Description
LT_THRESH	0x0	less than threshold ***Note:*** Threshold set in CTL2[42:37]
GE_THRESH	0x1	greater than or equal to threshold ***Note:*** Threshold set in CTL2[42:37]

DC_EVENT0

- **Title:** Directory Cache Event 0
- **Category:** DC
- **Event Code:** 0x12, **Max. Inc/Cyc:** 1,
- **Definition:** Directory Cache Event 0

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
TRANS_OCCUPANCY	0x0		Outstanding transactions matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) that have overflowed the accumulator (B_CSR_DC_PERF_CNT). ***Note:*** Set B_CSR_PERF_CTL0.dc_en to enable the subcounter which, in turn, enables this event. To obtain the number of matching transactions, multiply the result in main counter by 2^6 and add B-CSR_DC_PERF_CNT.accum_cnt[5:0].
TRANS_INSERT	0x1		Number of transactions entering DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) ***Note:*** NCRD and NCWR do not enter DC. Some RdData commands go through DC twice. The first pass is for a coarse-share MESI, and the second pass (if there is one) is for an exclusive MESI. Both passes correspond to the same entry in IMT. Number of DC inserts may differ from the number of IMT inserts.
TRANS_REMOVE	0x2		Number of transactions removed from DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode)
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x0	Cycles that number of outstanding DC transactions is less than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x1	Cycles that number of outstanding DC transactions is greater than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
FULL	0x4		DC is full. ***Note:*** Indicates the DC has 32 outstanding transactions.
NON_EMPTY	0x5		DC is not empty. DC has some outstanding transactions.

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
MATCH_READ	0x6		Matching DC read event. ***Note:*** Upon receiving a request, DC reads it's cache. The data read from the cache is matched to the criteria defined in B_CSR_DC_PERF_MATCH_RD. The selected events are masked with B_CSR_DC_PERF_MASK_RD. Affected by B_CSR_DC_PERF_CTL1.dc_opcode. DC_READ_EVENT = ((DC_CACHE_DATA_RD xnor B_CSR_DC_PERF_MATCH_RD) or (B_CSR_DC_PERF_MASK_RD) and MATCH_DC_OPCODES AND NOT 2nd_LOOKUP
MATCH_WRITE	0x7		Matching DC write event. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR. The selected events are masked with B_CSR_DC_PERF_MASK_WR. DC_WRITE_EVENT = (DC_CACHE_DATA_WR xnor B_CSR_DC_PERF_MATCH_WR) or B_CSR_DC_PERF_MASK_WR.
NEED_SNOOPS	0x8		New transaction requires snoops. ***Note:*** Indicates that single or multiple snoops are required. No matching is performed.
LINE_EVICT	0x9		DC line eviction event.
MATCH_READ_2ND_LOOK	0xa		Matching DC read second lookup. ***Note:*** Works the same way as the MATCH_READ event, with the exception that second DC read lookups cause this event.
SNP_LOC	0xb		Snoops caused by cell local memory accesses.
MATCH_WRITE	0xc		Matching DC write event 2. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR2. The selected events are masked with B_CSR_DC_PERF_MASK_WR2. DC_WRITE_EVENT2 = (DC_CACHE_DATA_WR2 xnor B_CSR_DC_PERF_MATCH_WR2) or B_CSR_DC_PERF_MASK_WR2.
LINE_ALLOC	0xd		DC line allocation event.

DC_EVENT1

- **Title:** Directory Cache Event 1
- **Category:** DC
- **Event Code:** 0x13, **Max. Inc/Cyc:** 1,
- **Definition:** Directory Cache Event 1.

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
TRANS_OCCUPANCY	0x0		Outstanding transactions matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) that have overflowed the accumulator (B_CSR_DC_PERF_CNT). ***Note:*** Set B_CSR_PERF_CTL0.dc_en to enable the subcounter which, in turn, enables this event. To obtain the number of matching transactions, multiply the result in main counter by 2^6 and add B-CSR_DC_PERF_CNT.accum_cnt[5:0].
TRANS_INSERT	0x1		Number of transactions entering DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) ***Note:*** NCRD and NCWR do not enter DC. Some RdData commands go through DC twice. The first pass is for a coarse-share MESI, and the second pass (if there is one) is for an exclusive MESI. Both passes correspond to the same entry in IMT. Number of DC inserts may differ from the number of IMT inserts.
TRANS_REMOVE	0x2		Number of transactions removed from DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode)
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x0	Cycles that number of outstanding DC transactions is less than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x1	Cycles that number of outstanding DC transactions is greater than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
FULL	0x4		DC is full. ***Note:*** Indicates the DC has 32 outstanding transactions.
NON_EMPTY	0x5		DC is not empty. DC has some outstanding transactions.
MATCH_READ	0x6		Matching DC read event. ***Note:*** Upon receiving a request, DC reads it's cache. The data read from the cache is matched to the criteria defined in B_CSR_DC_PERF_MATCH_RD. The selected events are masked with B_CSR_DC_PERF_MASK_RD. Affected by B_CSR_DC_PERF_CTL1.dc_opcode. DC_READ_EVENT = ((DC_CACHE_DATA_RD xnor B_CSR_DC_PERF_MATCH_RD) or (B_CSR_DC_PERF_MASK_RD) and MATCH_DC_OPCODES AND NOT 2nd_LOOKUP
MATCH_WRITE	0x7		Matching DC write event. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR. The selected events are masked with B_CSR_DC_PERF_MASK_WR. DC_WRITE_EVENT = (DC_CACHE_DATA_WR xnor B_CSR_DC_PERF_MATCH_WR) or B_CSR_DC_PERF_MASK_WR.
NEED_SNOOPS	0x8		New transaction requires snoops. ***Note:*** Indicates that single or multiple snoops are required. No matching is performed.
LINE_EVICT	0x9		DC line eviction event.
MATCH_READ_2ND_LOOK	0xa		Matching DC read second lookup. ***Note:*** Works the same way as the MATCH_READ event, with the exception that second DC read lookups cause this event.
SNP_LOC	0xb		Snoops caused by cell local memory accesses.

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
MATCH_WRITE	0xc		Matching DC write event 2. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR2. The selected events are masked with B_CSR_DC_PERF_MASK_WR2. DC_WRITE_EVENT2 = (DC_CACHE_DATA_WR2 xnor B_CSR_DC_PERF_MATCH_WR2) or B_CSR_DC_PERF_MASK_WR2.
LINE_ALLOC	0xd		DC line allocation event.

DC_EVENT2

- **Title:** Directory Cache Event 2
- **Category:** DC
- **Event Code:** 0x14, **Max. Inc/Cyc:** 1,
- **Definition:** Directory Cache Event 2

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
TRANS_OCCUPANCY	0x0		Outstanding transactions matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) that have overflowed the accumulator (B_CSR_DC_PERF_CNT). ***Note:*** NOTE: Set B_CSR_PERF_CTL0.dc_en to enable the subcounter which, in turn, enables this event. To obtain the number of matching transactions, multiply the result in main counter by 2^6 and add B-CSR_DC_PERF_CNT.accum_cnt[5:0].
TRANS_INSERT	0x1		Number of transactions entering DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) ***Note:*** NOTE: NCRD and NCWR do not enter DC. Some RdData commands go through DC twice. The first pass is for a coarse-share MESI, and the second pass (if there is one) is for an exclusive MESI. Both passes correspond to the same entry in IMT. Number of DC inserts may differ from the number of IMT inserts.
TRANS_REMOVE	0x2		Number of transactions removed from DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode)
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x0	Cycles that number of outstanding DC transactions is less than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x1	Cycles that number of outstanding DC transactions is greater than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
FULL	0x4		DC is full. Indicates the DC has 32 outstanding transactions.
NON_EMPTY	0x5		DC is not empty. DC has some outstanding transactions.
MATCH_READ	0x6		Matching DC read event. ***Note:*** Upon receiving a request, DC reads it's cache. The data read from the cache is matched to the criteria defined in B_CSR_DC_PERF_MATCH_RD. The selected events are masked with B_CSR_DC_PERF_MASK_RD. Affected by B_CSR_DC_PERF_CTL1.dc_opcode. DC_READ_EVENT = ((DC_CACHE_DATA_RD xnor B_CSR_DC_PERF_MATCH_RD) or (B_CSR_DC_PERF_MASK_RD) and MATCH_DC_OPCODES AND NOT 2nd_LOOKUP

Extension	DC_CTL0 [4:0]	DC_WM ARK[6]	Description
MATCH_WRITE	0x7		Matching DC write event. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR. The selected events are masked with B_CSR_DC_PERF_MASK_WR. DC_WRITE_EVENT = (DC_CACHE_DATA_WR xnor B_CSR_DC_PERF_MATCH_WR) or B_CSR_DC_PERF_MASK_WR.
NEED_SNOOPS	0x8		New transaction requires snoops. ***Note:*** Indicates that single or multiple snoops are required. No matching is performed.
LINE_EVICT	0x9		DC line eviction event.
MATCH_READ_2ND_LOOK	0xa		Matching DC read second lookup. ***Note:*** Works the same way as the MATCH_READ event, with the exception that second DC read lookups cause this event.
SNP_LOC	0xb		Snoops caused by cell local memory accesses.
MATCH_WRITE	0xc		Matching DC write event 2. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR2. The selected events are masked with B_CSR_DC_PERF_MASK_WR2. DC_WRITE_EVENT2 = (DC_CACHE_DATA_WR2 xnor B_CSR_DC_PERF_MATCH_WR2) or B_CSR_DC_PERF_MASK_WR2.
LINE_ALLOC	0xd		DC line allocation event.

DC_EVENT3

- **Title:** Directory Cache Event 3
- **Category:** DC
- **Event Code:** 0x15, **Max. Inc/Cyc:** 1,
- **Definition:** Directory Cache Event 3.

Extension	DC_CTL0 [4:0]	DC_WM ARK[6]	Description
TRANS_OCCUPANCY	0x0		Outstanding transactions matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) that have overflowed the accumulator (B_CSR_DC_PERF_CNT). ***Note:*** Set B_CSR_PERF_CTL0.dc_en to enable the subcounter which, in turn, enables this event. To obtain the number of matching transactions, multiply the result in main counter by 2^6 and add B-CSR_DC_PERF_CNT.accum_cnt[5:0].
TRANS_INSERT	0x1		Number of transactions entering DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode) ***Note:*** NOTE: NCRD and NCWR do not enter DC. Some RdData commands go through DC twice. The first pass is for a coarse-share MESI, and the second pass (if there is one) is for an exclusive MESI. Both passes correspond to the same entry in IMT. Number of DC inserts may differ from the number of IMT inserts.
TRANS_REMOVE	0x2		Number of transactions removed from DC matching DC Opcode (B_CSR_DC_PERF_CTL1.dc_opcode)
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x0	Cycles that number of outstanding DC transactions is less than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].

Extension	DC_CTL0 [4:0]	DC_WMARK[6]	Description
CYCLES_TRANS_OUT.THRESH_LE	0x3	0x1	Cycles that number of outstanding DC transactions is greater than or equal to number specified in B_CSR_DC_PERF_WMARK[5:0].
FULL	0x4		DC is full. ***Note:*** Indicates the DC has 32 outstanding transactions.
NON_EMPTY	0x5		DC is not empty. DC has some outstanding transactions.
MATCH_READ	0x6		Matching DC read event. ***Note:*** Upon receiving a request, DC reads it's cache. The data read from the cache is matched to the criteria defined in B_CSR_DC_PERF_MATCH_RD. The selected events are masked with B_CSR_DC_PERF_MASK_RD. Affected by B_CSR_DC_PERF_CTL1.dc_opcode. DC_READ_EVENT = ((DC_CACHE_DATA_RD xnor B_CSR_DC_PERF_MATCH_RD) or (B_CSR_DC_PERF_MASK_RD) and MATCH_DC_OPCODES AND NOT 2nd_LOOKUP
MATCH_WRITE	0x7		Matching DC write event. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR. The selected events are masked with B_CSR_DC_PERF_MASK_WR. DC_WRITE_EVENT = (DC_CACHE_DATA_WR xnor B_CSR_DC_PERF_MATCH_WR) or B_CSR_DC_PERF_MASK_WR.
NEED_SNOOPS	0x8		New transaction requires snoops. ***Note:*** Indicates that single or multiple snoops are required. No matching is performed.
LINE_EVICT	0x9		DC line eviction event.
MATCH_READ_2ND_LOOK	0xa		Matching DC read second lookup. ***Note:*** Works the same way as the MATCH_READ event, with the exception that second DC read lookups cause this event.
SNP_LOC	0xb		Snoops caused by cell local memory accesses.
MATCH_WRITE	0xc		Matching DC write event 2. ***Note:*** Upon receiving a request, DC may write to its cache. Writes are matched to the criteria defined in B_CSR_DC_PERF_MATCH_WR2. The selected events are masked with B_CSR_DC_PERF_MASK_WR2. DC_WRITE_EVENT2 = (DC_CACHE_DATA_WR2 xnor B_CSR_DC_PERF_MATCH_WR2) or B_CSR_DC_PERF_MASK_WR2.
LINE_ALLOC	0xd		DC line allocation event.

IMT_ALLOC

- **Title:** IMT Allocations
- **Category:** IMT
- **Event Code:** 0x05, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IMT entries allocated.
- **NOTE:** No rd/wr breakdown.

IMT_FULL

- **Title:** Cycles IMT Full
- **Category:** IMT
- **Event Code:** 0x2a, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles the IMT is full.

IMT_FULL_CURR_PIPE

- **Title:** Cycles IMT Full for Current Pipe Pass
- **Category:** IMT
- **Event Code:** 0x29, **Max. Inc/Cyc:** 1,
- **Definition:** Indicates that the half of the IMT selected for the current pipeline is full.

IMT_NE_CYCLES

- **Title:** Cycles IMT Not Empty
- **Category:** IMT
- **Event Code:** 0x07, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles the IMT is not empty.

IMT_OCCUPANCY

- **Title:** IMT Occupancy
- **Category:** IMT
- **Event Code:** 0x04, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IMT counter overflows.
- **NOTE:** Set B_CSR_PERF_CTL0.imt_en to enable the subcounter which, in turn, enables this event. No rd/wr breakdown. To calculate the number of IMT events, multiply the counter value by 2^6 and add B_CSR_IMT_PERF_CNT.accum_cnt[5:0].

IMT_POP_CFL

- **Title:** IMT Pop Conflicts
- **Category:** IMT
- **Event Code:** 0x6, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IMT pop conflicts.

IOB_IN_PKTS

- **Title:** IOB Input Packets
- **Category:** IOB
- **Event Code:** 0x19, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IOB input packets.
- **NOTE:** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_in, opcode_in}.

Extension	CTL2 [20:19], [17]	Description
RHNID	0x1,0x0	RHNID of input Intel QPI packet is used to determine local or remote socket
RSNID	0x1,0x1	RSNID of input Intel QPI packet is used to determine local or remote socket

IOB_INSERTS

- **Title:** IOB Inserts of Live Trans
- **Category:** IOB
- **Event Code:** 0x17, **Max. Inc/Cyc:** 1,
- **Definition:** IOB live transaction increment.

Extension	CTL2 Dep Bits	Description
IN_PKTS.RHNID	[16]0x0 && [17]0x0 && [20:19]x1	New input packets; RHNID of input Intel QPI packet is used to determine local or remote socket ***Note:*** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_in, opcode_in}
IN_PKTS.RSNID	[16]0x0 && [17]0x1 && [20:19]x1	New input packets; RSNID of input Intel QPI packet is used to determine local or remote socket ***Note:*** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_in, opcode_in}
SNOOPS	[16:15]0x2	Snoop packets launched by Bbox; RHNID of input Intel QPI packet is used to determine local or remote socket ***Note:*** Not designed to count average latency of snoops. If snoops responses are selected (B_CSR_PERF_CTL2.class_in[1]=1), live_cnt is not going to be 0 at the end of the test when Bbox is idle.

IOB_OUT_PKTS

- **Title:** IOB Output Packets
- **Category:** IOB
- **Event Code:** 0x1a, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IOB output packets.
- **NOTE:** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_out, opcode_out}.

Extension	CTL2 [20:19], [18]	Description
RHNID	0x1,0x0	RHNID of output Intel QPI packet is used to determine local or remote socket
RSNID	0x1,0x1	RSNID of output Intel QPI packet is used to determine local or remote socket

IOB_OCCUPANCY

- **Title:** IOB Occupancy
- **Category:** IOB
- **Event Code:** 0x16, **Max. Inc/Cyc:** 1,
- **Definition:** Number of IOB counter overflows.
- **NOTE:** Set B_CSR_PERF_CTL0.iob_en to enable the subcounter which, in turn, enables this event. To calculate the number of IOB events, multiply the counter value by 2^11 and add B_CSR_IOB_PERF_CNT.accum_cnt[10:0].

IOB_REMOVES

- **Title:** IOB Removals of Live Trans
- **Category:** IOB
- **Event Code:** 0x18, **Max. Inc/Cyc:** 1,
- **Definition:** IOB live transaction decrement.

Extension	CTL2 Dep Bits	Description
OUT_PKTS.RHNID	[16]0x0 && [17]0x0 && [20:19]x1	New output packets; RHNID of input Intel QPI packet is used to determine local or remote socket ***Note:*** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_in, opcode_in}
OUT_PKTS.RSNID	[16]0x0 && [17]0x1 && [20:19]x1	New output packets; RSNID of input Intel QPI packet is used to determine local or remote socket ***Note:*** User can filter according to class or opcode by setting bits in B_CSR_PERF_CTL3.{class_in, opcode_in}
IN_PKT_COMPLETES	[16:15]0x1	All completions to packets that matched input packet criteria in IOB_INSERTS.
SNOOPS	[16:15]0x2	Snoop response packets received by Bbox.

MEM_HINT_DC_LUP

- **Title:** Memory Hint in DC Lookup
- **Category:** Mem Hint
- **Event Code:** 0x2d, **Max. Inc/Cyc:** 1,
- **Definition:** Lookup in DC a request that has local memory hint bit set.

MEM_HINT_Z_RESP

- **Title:** Memory Hint in Zbox Response
- **Category:** Mem Hint
- **Event Code:** 0x2e, **Max. Inc/Cyc:** 1,
- **Definition:** Received a response from the Zbox that has local memory hint bit set.

NSL_REJ

- **Title:** NSL Reject
- **Category:** NSL
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Rejected NSL pipeline pass

NSL_SUCC

- **Title:** NSL Success
- **Category:** NSL
- **Event Code:** 0x26, **Max. Inc/Cyc:** 1,
- **Definition:** Successful NSL pipeline pass

POISON_RECV

- **Title:** Poison Packet Received
- **Category:** Misc
- **Event Code:** 0x2f, **Max. Inc/Cyc:** 1,
- **Definition:** Received a packet with poison. The received packet poison errors need to be enabled in B_CSR_ERR_MAS_0.erecv_poison.

POISON_SENT

- **Title:** Poison Packet Sent
- **Category:** Misc
- **Event Code:** 0x30, **Max. Inc/Cyc:** 1,
- **Definition:** Sent a packet with poison. The sent packet poison errors need to be enabled in B_CSR_ERR_MAS_0.esent_poison.

RFP_LOC_LOC

- **Title:** RFP Local/Local
- **Category:** RFP
- **Event Code:** 0x00, **Max. Inc/Cyc:** 1,
- **Definition:** Response forward packet arriving in Bbox.Local requestor and local responder.

RFP_LOC_REM

- **Title:** RFP Local/Remote
- **Category:** RFP
- **Event Code:** 0x02, **Max. Inc/Cyc:** 1,
- **Definition:** Response forward packet arriving in Bbox.Local requestor and remote responder.

RFP_REM_LOC

- **Title:** RFP Remote/Local
- **Category:** RFP
- **Event Code:** 0x01, **Max. Inc/Cyc:** 1,
- **Definition:** Response forward packet arriving in Bbox. Remote requestor and local responder.

RFP_REM_REM

- **Title:** RFP Remote/Remote
- **Category:** RFP
- **Event Code:** 0x03, **Max. Inc/Cyc:** 1,
- **Definition:** Response forward packet arriving in Bbox. Remote requestor and remote responder.

SNP_REQ_ALL

- **Title:** All Snoop Requests
- **Category:** SNP
- **Event Code:** 0x1b, **Max. Inc/Cyc:** 1,
- **Definition:** All snoop requests (remote or local).

SNP_REQ_LOC

- **Title:** Local Snoop Requests
- **Category:** SNP
- **Event Code:** 0x1c, **Max. Inc/Cyc:** 1,
- **Definition:** Local snoop requests.

SNP_REQ_REM

- **Title:** Remote Snoop Requests
- **Category:** SNP
- **Event Code:** 0x1d, **Max. Inc/Cyc:** 1,
- **Definition:** Remote snoop requests.

SNP_RSP_ALL

- **Title:** All Snoop Responses
- **Category:** SNP
- **Event Code:** 0x1e, **Max. Inc/Cyc:** 1,
- **Definition:** All snoop responses (remote or local).

SNP_RSP_LOC

- **Title:** Local Snoop Responses
- **Category:** SNP
- **Event Code:** 0x1f, **Max. Inc/Cyc:** 1,
- **Definition:** Local snoop responses.

SNP_RSP_REM

- **Title:** Remote Snoop Responses
- **Category:** SNP
- **Event Code:** 0x20, **Max. Inc/Cyc:** 1,
- **Definition:** Remote snoop responses.

TRACKER_IMT_HAZARD

- **Title:** Tracker/IMT Hazard
- **Category:** Miscellaneous
- **Event Code:** 0x27, **Max. Inc/Cyc:** 1,
- **Definition:** Tracker/IMT Hazard.

Z_OPT_V2V

- **Title:** Zbox V2V Optimized Requests
- **Category:** Misc
- **Event Code:** 0x32, **Max. Inc/Cyc:** 1,
- **Definition:** Zbox V2V requests to Zbox when read optimization is performed.

5.5 Cbox Performance Monitoring

5.5.1 Overview of the Cbox

For the Intel Itanium® Processor 9500 Series, the LLC coherence engine (Cbox) manages the interface between the core and the last level cache (LLC). All core transactions that access the LLC are directed from the core to a Cbox via the ring interconnect. The Cbox is responsible for managing data delivery from the LLC to the requesting core. It is also responsible for maintaining coherence between the cores within the socket that share the LLC; generating snoops and collecting snoop responses to the local cores when the MESI protocol requires it.

The Cbox is also the gate keeper for all Intel® QuickPath Interconnect (Intel® QPI) messages that originate in the core and is responsible for ensuring that all Intel QuickPath Interconnect messages that pass through the socket's LLC remain coherent.

The Intel Itanium processor 9500 series contains eight instances of the Cbox, each assigned to manage a distinct 3MB, 24-way set associative slice of the processor's total LLC capacity. For processors with fewer than 8 3MB LLC slices, the Cboxes for missing slices will still be active and track ring traffic caused by their co-located core even if they have no LLC related traffic to track (i.e. hits/misses/snoops).

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

5.5.2 Cbox Performance Monitoring Overview

Each of the Cboxes in the Itanium processor 9500 series supports event monitoring through six 48-bit wide counters (C_CSR_PMON_CTR{5:0}). Each of these six counters can be programmed to count any Cbox event. The Cbox counters can increment by a maximum of 5b per cycle.

The count values of all 6 counters can be cleared by writing the C_CSR_PMON_PERF_MASTER.clr bit.

For information on how to setup a monitoring session, refer to Section 5.3, "Global Performance Monitoring Control".

5.5.2.1 Cbox PMU - Overflow, Freeze and Unfreeze

Cbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Cbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, "Global Enable/Disable").

Cbox PMU can be frozen due to one of three reasons:

- *Globally:* Ubox sends a disable signal (and C_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, "Global Enable/Disable") or *local* (SW writes 0 to C_CSR_PMON_PERF_MASTER.en when C_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.

- *Locally:* The Cbox was set to local control (C_CSR_PMON_PERF_MASTER.glb_lcl = 0) and a Cbox counter overflowed.

If an overflow is detected from a Cbox performance counter, the overflow bit is set at the box level (C_CSR_PMON_GLOBAL_STATUS.ov), and forwarded up the chain towards the Sbox. i.e. If a Cbox0 counter overflows, a notification is sent and stored in Sbox (S_CSR_PMON_SUMMARY.ov_c0_7). Refer to Table 5-64, "S_CSR_PMON_SUMMARY Register Fields" to determine how each Cbox's overflow bit is accumulated in the attached Sbox.

The Ubox may be configured to freeze all uncore counting (refer to Table 5-65, "S_CSR_PMON_FRZ_EN Register Fields") when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field responsible for the freeze, must be cleared. Assuming all the counters have been locally enabled (.en bit in data registers meant to monitor events) and the overflow bit(s) has been cleared, the Cbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, "Enabling a New Sample Interval from Frozen Counters."), counting will resume.

5.5.3 Cbox Performance Monitors

Table 5-34. Cbox Performance Monitoring CSRs

CSR Name	CSR Address [7:0]	Size (bits)	Description
Box-Level Control / Status			
C_CSR_PMON_PERF_MASTER	0xB8	32	Cbox [x] PMON Performance Master
Legacy Box-Level Control / Status			
C_CSR_PMON_GLOBAL_STATUS	0xB0	32	Cbox [x] PMON Global Status
C_CSR_PMON_GLOBAL_CTL	0xA0	32	Cbox [x] PMON Global Control
Generic Counter / Control			
C_CSR_PMON_EVT_SEL_5	0x98	64	Cbox [x] PMON Event Select 5
C_CSR_PMON_EVT_SEL_4	0x90	64	Cbox [x] PMON Event Select 4
C_CSR_PMON_EVT_SEL_3	0x88	64	Cbox [x] PMON Event Select 3
C_CSR_PMON_EVT_SEL_2	0x80	64	Cbox [x] PMON Event Select 2
C_CSR_PMON_EVT_SEL_1	0x78	64	Cbox [x] PMON Event Select 1
C_CSR_PMON_EVT_SEL_0	0x70	64	Cbox [x] PMON Event Select 0
C_CSR_PMON_CTR_5	0x68	64	Cbox [x] PMON Counter 5
C_CSR_PMON_CTR_4	0x60	64	Cbox [x] PMON Counter 4
C_CSR_PMON_CTR_3	0x58	64	Cbox [x] PMON Counter 3
C_CSR_PMON_CTR_2	0x50	64	Cbox [x] PMON Counter 2
C_CSR_PMON_CTR_1	0x48	64	Cbox [x] PMON Counter 1
C_CSR_PMON_CTR_0	0x40	64	Cbox [x] PMON Counter 0

5.5.3.1 Cbox Box Level PMON State

The following registers represent the state governing all box-level PMUs in the Cbox.

C_CSR_PMON_PERF_MASTER controls the general characteristics of the Cbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

The _GLOBAL_CTL register contains the bits used to enable monitoring. It is necessary to set the .ctr_en bit to 1 before the corresponding data register can collect events.

If an overflow is detected from one of the Cbox PMON registers, the corresponding bit in the _GLOBAL_STATUS.ov field will be set.

Table 5-35. C_CSR_PMON_PERF_MASTER Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:4	RO_NA	0	Read zero; writes ignored. (?)
ck_en	3	RW_RW	0	Enables Cbox PMU clock.
clr	2	RW_RW	0	Writing 1 clears all Cbox counters as well as the overflow bits in C_CSR_PMON_GLBOAL_STATUS
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: **Global**: Enable/Disable of counters in Cbox will track U_CSR_PERF_CTL.glb_en. Local overflows will be passed on to Ubox without freezing local counters. 0: **Local**: Enable/Disable of counters in Cbox will NOT track U_CSR_PERF_CTL.glb_en. Allows SW to write the .en bit. Disables Counters on any local counter overflow.
en	0	RW_RW	0	Enable/disable Cbox PMU counters. This bit is dependent on the setting of the .glb_lcl bit. If .glb_lcl is set to 1, SW writes to this bit are ignored and only HW may affect it's state. If .glb_lcl is set to 0, SW may exert control by setting the bit. In either case, since HW may alter this bit, (due to tracking the global enable or a local overflow) SW may read it to determine the state of the Cbox counters. 1: Enable Cbox PMU counting. 0: Disable (freeze) Cbox PMU counters.

Table 5-36. C_CSR_PMON_GLOBAL_CTL Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	31:6	RO_NA	0	Read zero; writes ignored. (?)
ctr_en	5:0	RW_RO	0	Must be set to enable each Cbox counter. ***Note:*** Ubox/PERF_MASTER enable and per counter enable must also be set to fully enable the counter.

Table 5-37. C_CSR_PMON_GLOBAL_STATUS Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	31:6	RO_NA	0	Read zero; writes ignored. (?)
ov	5:0	RW_RO	0	If an overflow is detected from the corresponding CBOX PMON register, it's overflow bit will be set.

5.5.3.2 Cbox PMON state - Counter/Control Pairs

The following table defines the layout of the Cbox performance monitor control registers. The main task of these configuration registers is to select the event to be monitored by their respective data counter. Setting the *.ev_sel* and *.umask* fields performs the event selection. The *.en* bit must be set to 1 to enable counting.

Additional control bits include:

- *.threshold* - since Cbox counters can increment by a value greater than 1, a threshold can be applied. If the *.threshold* 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.

- *.invert* - Changes the .threshold test condition to '<'

- *.edge_detect* - 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.

Table 5-38. C_CSR_PMON_EVT_SEL{5-0} Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	63	RO_NA	0	Read zero; writes ignored. (?)
rsv	62:61	0	0	Reserved; Must write to 0 else behavior is undefined.
ig	60:51	RO_NA	0	Read zero; writes ignored. (?)
rsv	50	0	0	Reserved; Must write to 0 else behavior is undefined.
ig	49:32	RO_NA	0	Read zero; writes ignored. (?)
threshold	31:24	RW_RO	0	Threshold used in counter comparison.
invert	23	RW_RO	0	When 0, the comparison that will be done is threshold <= event. When set to 1, the comparison that is inverted (i.e. threshold < event)
en	22	RW_RO	0	Local Counter Enable. When set, the associated counter is locally enabled. NOTE: It must also be enabled in C_CSR_PMON_GLOBAL_CTL and the Ubox to be fully enabled.
ig	21:19	RO_NA	0	Read zero; writes ignored. (?)
edge_detect	18	RW_RO	0	When asserted, the 0 to 1 transition edge of a 1 bit event input will cause the corresponding counter to increment. When 0, the counter will increment for however long the event is asserted. NOTE: .edge_detect is in series following threshold and invert, so it can be applied to multi-increment events that have been filtered by the threshold field.
reset_occ_count	17	RW_RO	0	Reset Occupancy Counter

Table 5-38. C_CSR_PMON_EVT_SEL{5-0} Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	16	RO_NA	0	Read zero; writes ignored. (?)
umask	15:8	RW_RO	0	Select subevents to be counted within the selected event.
ev_sel	7:0	RW_RO	0	Select event to be counted.

The Cbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-39. C_CSR_PMON_CTR{5-0} Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored. (?)
event_count	47:0	RW_RW	0	48-bit performance event counter

5.5.4 Cbox Performance Monitoring Events

5.5.4.1 An Overview:

The performance monitoring events within the Cbox include all events internal to the LLC as well as events which track ring related activity at the Cbox/Core ring stops. The only ring specific events that are not tracked by the Cbox PMUs are those events that track ring activity at the Sbox ring stop (see the Sbox chapter for details on those events).

Cbox performance monitoring events can be used to track LLC access rates, LLC hit/ miss rates, LLC eviction and fill rates, and to detect evidence of back pressure on the LLC pipelines. In addition, the Cbox has performance monitoring events for tracking MESI state transitions that occur as a result of data sharing across sockets in a multi-socket system. And finally, there are events in the Cbox for tracking ring traffic at the Cbox/Core sink inject points.

Every event in the Cbox (with the exception of the P2C inject and *2P sink counts) are from the point of view of the LLC and cannot be associated with any specific core since all cores in the socket send their LLC transactions to all Cboxes in the socket. The P2C inject and *2P sink counts serve as the exception since those events are tracking ring activity at the cores' ring inject/sink points.

There are separate sets of counters for each Cbox instance. For any event, to get an aggregate count of that event for the entire LLC, the counts across the Cbox instances must be added together. The counts can be averaged across the Cbox instances to get

a view of the typical count of an event from the perspective of the individual Cboxes. Individual per-Cbox deviations from the average can be used to identify hot-spotting across the Cboxes or other evidences of non-uniformity in LLC behavior across the Cboxes. 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 Cboxes would indicate hot-spotting.

5.5.4.2 Acronyms frequently used in Cbox Events:

The Rings:

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

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

AK (Acknowledge) Ring - Acknowledges Sbox to Cbox and Cbox to Core. Carries snoop responses from Core to Cbox.

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

Internal Cbox Queues:

IRQ - Ingress Request Queue on AD Ring. Associated with requests from core.

IPQ - Ingress Probe Queue on AD Ring. Associated with snoops from Sbox.

VIQ - Victim Queue internal to Cbox.

IDQ - Ingress Data Queue on BL Ring. For data from either Core or Sbox.

ICQ - Sbox Ingress Complete Queue on AK Ring

SRQ - Processor Snoop Response Queue on AK ring

IGQ - Ingress GO-pending (tracking GO's to core) Queue

MAF - Miss Address File. Intel QPI ordering buffer that also tracks local coherence.

5.5.4.3 The Queues:

There are four internal occupancy queue counters, each of which is 5bits wide and dedicated to its queue: IRQ, IPQ, VIQ, MAF.

Note: IDQ, ICQ, SRQ and IGQ occupancies are not tracked since they are mapped 1:1 to the MAF and, therefore, can not create back pressure.

There is no need to explicitly reset the occupancy counters in the Cbox since they are counting from reset de-assertion.

5.5.4.4 Detecting Performance Problems in the Cbox Pipeline:

IRQ occupancy counters should be used to track if the Cbox pipeline is exerting back pressure on the Core-request path. There is a one-to-one correspondence between the LLC requests generated by the cores and the IRQ allocations. IPQ occupancy counters should be used to track if the Cbox pipeline is exerting back pressure on the Intel QPI-snoop path. There is a one-to-one correspondence between the Intel QPI snoops received by the socket, and the IPQ allocations in the Cboxes. In both cases, if the message is in the IRQ/IPQ then the Cbox hasn't acknowledged it yet and the request

hasn't yet entered the LLC's "coherence domain". It deallocates from the IRQ/IPQ at the moment that the Cbox does acknowledge it. In optimal performance scenarios, where there are minimal conflicts between transactions and loads are low enough to keep latencies relatively near to idle, IRQ and IPQ occupancies should remain very low.

One relatively common scenario in which IRQ back pressure will be high is worth mentioning: The IRQ will backup when software is demanding data from memory at a rate that exceeds the available memory BW. The IRQ is designed to be the place where the extra transactions wait Ubox's RTIDs to become available when memory becomes saturated. IRQ back pressure becomes interesting in a scenario where memory is not operating at or near peak sustainable BW. That can be a sign of a performance problem that may be correctable with software tuning.

One final warning on LLC pipeline congestion: Care should be taken not to blindly sum events across Cboxes without also checking the deviation across individual Cboxes when investigating performance issues that are concentrated in the Cbox pipelines. Performance problems where congestion in the Cbox pipelines is the cause should be rare, but if they do occur, the event counts may not be homogeneous across the Cboxes in the socket. The average count across the Cboxes may be misleading. If performance issues are found in this area it will be useful to know if they are or are not localized to specific Cboxes.

5.5.5 Cbox Events Ordered By Code

Table 5-40 summarizes the directly-measured Cbox events.

Table 5-40. Performance Monitor Events for Cbox Events (Sheet 1 of 2)

Symbol Name	Event Code	Max Inc/Cyc	Description
Ring Events			
BOUNCES_P2C_AD	0x01	1	Number of P2C AD bounces.
BOUNCES_C2P_AK	0x02	1	Number of C2P AK bounces.
BOUNCES_C2P_BL	0x03	1	Number of C2P BL bounces.
BOUNCES_C2P_IV	0x04	1	Number of C2P IV bounces.
SINKS_P2C	0x05	3	Number of P2C sinks.
SINKS_C2P	0x06	3	Number of C2P sinks.
SINKS_S2C	0x07	3	Number of S2C sinks.
SINKS_S2P_BL	0x08	1	Number of S2P sinks (BL only).
ARB_WINS	0x09	7	Number of ARB wins.
ARB_LOSSES	0x0A	7	Number of ARB losses.
Local Events			
STARVED_EGRESS	0x0B	8	Increment on EGR queue starvation
EGRESS_BYPASS_WINS	0x0C	7	Egress Bypass Wins
INGRESS_BYPASS_WINS_AD	0x0E	1	Ingress Sbox/Non-Sbox Bypass Wins
MAF_ACK	0x10	1	MAF Acknowledges
MAF_NACK1	0x11	1	MAF Non-Acknowledgements (First Set)
MAF_NACK2	0x12	1	MAF Non-Acknowledgements (Second Set)
LLC_MISSES	0x14	1	LLC Misses
LLC_HITS	0x15	1	LLC Hits
LLC_S_FILLS	0x16	1	LLC lines filled from Sbox

Table 5-40. Performance Monitor Events for Cbox Events (Sheet 2 of 2)

Symbol Name	Event Code	Max Inc/Cyc	Description
LLC_VICTIMS	0x17	1	LLC lines victimized
Queue Occupancy Events			
OCCUPANCY_IRQ	0x18	24	IRQ Occupancy
TRANS_IRQ	0x19	1	IRQ Transactions (dealloc)
OCCUPANCY_IPQ	0x1A	8	IPQ Occupancy
TRANS_IPQ	0x1B	1	IPQ Transactions
OCCUPANCY_VIQ	0x1C	8	VIQ Occupancy
TRANS_VIQ	0x1D	1	VIQ Transactions
OCCUPANCY_MAF	0x1E	16	MAF Occupancy
TRANS_MAF	0x1F	1	MAF Transactions
Local Events			
SEMAPHORE_NACKS	0x20	1	Semaphore NACKs
SNPS	0x27	1	Snoops to LLC
SNP_HITS	0x28	1	Snoop Hits in LLC
LLC_READS_I	0x29	1	LLC Inst Reads
LLC_READS_D	0x2A	1	LLC Data Reads
LLC_WRITES	0x2B	1	LLC Writes

5.5.6 Cbox Performance Monitor Event List

This section enumerates Itanium processor 9500 series uncore performance monitoring events for the Cbox.

ARB_LOSSES

- **Title:** Arbiter Losses.
- **Category:** Ring - Egress
- **Event Code:** 0x0A, **Max. Inc/Cyc:** 7,
- **Definition:** Number of Ring arbitration losses. A loss occurs when a message injection on to the ring fails. This could occur either because there was another message resident on the ring at that ring stop or because the co-located ring agent issued a message onto the ring in the same cycle.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
AD_SB	b00000001	AD ring in the direction that points toward the nearest Sbox
AD_NSB	b00000010	AD ring in the direction that points away from the nearest Sbox
AD_ALL	b00000011	AD ring in either direction.
AK_SB	b00000100	AK ring in the direction that points toward the nearest Sbox
AK_NSB	b00001000	AK ring in the direction that points away from the nearest Sbox
AK_ALL	b00001100	AK ring in either direction.
BL_SB	b00010000	BL ring in the direction that points toward the nearest Sbox
BL_NSB	b00100000	BL ring in the direction that points away from the nearest Sbox
BL_ALL	b00110000	BL ring in either direction.

Extension	umask [15:8]	Description
IV	b01000000	IV ring
ALL	b01111111	All rings

ARB_WINS

- **Title:** Arbiter Wins
- **Category:** Ring - Egress
- **Event Code:** 0x09, **Max. Inc/Cyc:** 7,
- **Definition:** Number of Ring arbitration wins. A win is when a message was successfully injected onto the ring.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
AD_SB	b00000001	AD ring in the direction that points toward the nearest Sbox
AD_NSB	b00000010	AD ring in the direction that points away from the nearest Sbox
AD_ALL	b00000011	AD ring in either direction.
AK_SB	b00000100	AK ring in the direction that points toward the nearest Sbox
AK_NSB	b00001000	AK ring in the direction that points away from the nearest Sbox
AK_ALL	b00001100	AK ring in either direction.
BL_SB	b00010000	BL ring in the direction that points toward the nearest Sbox
BL_NSB	b00100000	BL ring in the direction that points away from the nearest Sbox
BL_ALL	b00110000	BL ring in either direction.
IV	b01000000	IV ring
ALL	b01111111	All rings

BOUNCES_C2P_AK

- **Title:** C2P AK Bounces
- **Category:** Ring - WIR
- **Event Code:** 0x02, **Max. Inc/Cyc:** 1,
- **Definition:** Number of LLC Ack responses to the core that bounced on the AK ring.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
SB	b000000x1	Direction that points toward the nearest Sbox
NSB	b0000001x	Direction that points away from the nearest Sbox
ALL	b00000011	Either direction

BOUNCES_C2P_BL

- **Title:** C2P BL Bounces
- **Category:** Ring - WIR
- **Event Code:** 0x03, **Max. Inc/Cyc:** 1,
- **Definition:** Number of LLC data responses to the core that bounced on the BL ring.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
SB	b000000x1	Direction that points toward the nearest Sbox
NSB	b0000001x	Direction that points away from the nearest Sbox
ALL	b00000011	Either direction

BOUNCES_C2P_IV

- **Title:** C2P IV Bounces
- **Category:** Ring - WIR
- **Event Code:** 0x04, **Max. Inc/Cyc:** 1,
- **Definition:** Number of Cbox snoops of a processor's cache that bounced on the IV ring.

BOUNCES_P2C_AD

- **Title:** P2C AD Bounces
- **Category:** Ring - WIR
- **Event Code:** 0x01, **Max. Inc/Cyc:** ,
- **Definition:** Core request to LLC bounces on AD ring.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
SB	b000000x1	Direction that points toward the nearest Sbox
NSB	b0000001x	Direction that points away from the nearest Sbox
ALL	b00000011	Either direction

EGRESS_BYPASS_WINS

- **Title:** Egress Bypass Wins
- **Category:** Local - Egress
- **Event Code:** 0x0C, **Max. Inc/Cyc:** 7,
- **Definition:** Number of times a ring egress bypass was taken when a message was injected onto the ring. The subevent field allows tracking of each available egress queue bypass path, including both 0 and 1 cycle versions.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
AD_BYP0	b00000001	0 cycle AD egress bypass
AD_BYP1	b00000010	1 cycle AD egress bypass
AK_BYP0	b00000100	0 cycle AK egress bypass
AK_BYP1	b00001000	1 cycle AK egress bypass

Extension	umask [15:8]	Description
BL_BYP0	b00010000	0 cycle BL egress bypass
BL_BYP1	b00100000	1 cycle BL egress bypass
IV_BYP0	b01000000	0 cycle IV egress bypass
IV_BYP1	b10000000	1 cycle IV egress bypass

INGRESS_BYPASS_WINS_AD

- **Title:** Ingress Sbox/Non Sbox Bypass Wins
- **Category:** Local - Egress
- **Event Code:** 0x0E, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times that a message, off the AD ring, sunk by the Cbox took one of the ingress queue bypasses. The subevent field allows tracking of each available ingress queue bypass path, including both 0 and 1 cycle versions.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
IRQ_BYP0	b00000001	0 cycle Ingress Request Queue bypass
IRQ_BYP1	b00000010	1 cycle Ingress Request Queue bypass
IPQ_BYP0	b00000100	0 cycle Ingress Probe Queue bypass
IPQ_BYP1	b00001000	1 cycle Ingress Probe Queue bypass

LLC_HITS

- **Title:** LLC Hits
- **Category:** Local - LLC
- **Event Code:** 0x15, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache Hits
- **NOTE:** LRU hints are included in count.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
M	b00000xx1	Modified
E	b00000x1x	Exclusive
S	b000001xx	Shared
ALL	b00000111	All hits (to any cacheline state)

LLC_MISSES

- **Title:** LLC Misses
- **Category:** Local - LLC
- **Event Code:** 0x14, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache Misses

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
S	b000000x1	Shared - request requires S line to be upgraded (due to RFO)

Extension	umask [15:8]	Description
I	b0000001x	Invalid - address not found
ALL	b0000011	All misses

LLC_READS_D

- **Title:** LLC Data Reads
- **Category:** Local - LLC
- **Event Code:** 0x2A, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache Data Reads (Opcode is DRd or RFO)

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
HIT	b0000xxx1	Hit
MISS	b0000xx1x	Miss
FILL	b0000x1xx	Fill
VICTIM	b00001xxx	Victim
ALL	b00001111	All misses

LLC_READS_I

- **Title:** LLC Inst Reads
- **Category:** Local - LLC
- **Event Code:** 0x29, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache Instruction Reads (Opcode is CRd)

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
HIT	b0000xxx1	Hit
MISS	b0000xx1x	Miss
FILL	b0000x1xx	Fill
VICTIM	b00001xxx	Victim
ALL	b00001111	All misses

LLC_S_FILLS

- **Title:** LLC Sbox Fills
- **Category:** Local - LLC
- **Event Code:** 0x16, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache lines filled from Sbox

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
M	b00000xx1	Filled to LLC in Modified (remote socket forwarded M data without writing back to memory controller)
E	b00000x1x	Filled to LLC in Exclusive

Extension	umask [15:8]	Description
S	b000001xx	Filled to LLC in Shared
ALL	b00000111	All fills to LLC

LLC_VICTIMS

- **Title:** LLC Lines Victimized
- **Category:** Local - LLC
- **Event Code:** 0x17, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache lines victimized

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
M	b0000xxx1	Modified data victimized (explicit WB to memory)
E	b0000xx1x	Exclusive data victimized
S	b0000x1xx	Shared data victimized
I	b00001xxx	LLC fill that occurred without victimizing any data

LLC_WRITES

- **Title:** LLC Writes
- **Category:** Local - LLC
- **Event Code:** 0x2B, **Max. Inc/Cyc:** 1,
- **Definition:** Last Level Cache Writes (Opcode is WBMtoS or FC)

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
HIT	b000000x1	Hit
MISS	b0000001x	Miss
ALL	b00000011	All misses

MAF_ACK

- **Title:** MAF ACK
- **Category:** Local - MAF
- **Event Code:** 0x10, **Max. Inc/Cyc:** 1,
- **Definition:** Miss Address File Acknowledgements.

MAF_NACK1

- **Title:** MAF NACK1
- **Category:** Local - MAF
- **Event Code:** 0x11, **Max. Inc/Cyc:** 1,
- **Definition:** Rejected (not-acknowledged) LLC pipeline passes (Set 1).

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
GO_PENDING	bxxxxxxx1	A message associated with a transaction monitored by the MAF was delayed because the transaction had a GO pending in the requesting core.
VIC_PENDING	bxxxxxx1x	An LLC fill was delayed because the victimized data in the LLC was still being processed.
SNP_PENDING	bxxxxx1xx	A message associated with a transaction monitored by the MAF was delayed because the transaction had a snoop pending.
AC_PENDING	bxxxx1xxx	An incoming remote Intel QPI snoop was delayed because it conflicted with an existing MAF transaction that had an Ack Conflict pending.
SCIDX_BLOCK	bxxx1xxxx	Side-Cam Index Blocking
PA_BLOCK	bxx1xxxxx	If this count is very high, it likely means that software is frequently issuing requests to the same physical address from disparate threads simultaneously. Though there will also sometimes be a small number of PA_BLOCK nacks in the background due to cases when a pair of messages associated with the same transaction happen to arrive at the LLC at the same time and one of them gets delayed.
IDLE_QPI	bx1xxxxxx	Idle Intel QPI State
ALL_MAF_NACK2	b1xxxxxxx	A message was rejected when one or more of the sub-events under MAF_NACK2 was true. This is included in MAF_NACK1 so that MAF_NACK1 with sub-event 0xFF will count the total number of Nacks.
TOTAL_MAF_NACKS	b11111111	Total number of LLC pipeline passes that were nacked.

MAF_NACK2

- **Title:** MAF NACK2
- **Category:** Local - MAF
- **Event Code:** 0x12, **Max. Inc/Cyc:** 1,
- **Definition:** Rejected (not-acknowledged) LLC pipeline passes (Set 2).

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
MAF_FULL	bxxxxxxx1	An incoming local processor RD/WR or remote Intel QPI snoop request that required a MAF entry was delayed because no MAF entry was available.
EGRESS_FULL	bxxxxxx1x	Some incoming message to the LLC that needed to generate a response message for transmission onto the ring was delayed due to ring back pressure.
VIQ_FULL	bxxxxx1xx	An incoming local processor RD request that missed the LLC was delayed because the LLC victim buffer was full.

Extension	umask [15:8]	Description
NO_TRACKER_CREDITS	bxxxx1xxx	An incoming local processor RD or WR request was delayed because it required a Home tracker credit (for example, LLC RD Miss) and no credit was available.
NO_S_FIFO_CREDITS	bxxx1xxxx	Some incoming message to the LLC that needed to generate a message to the Sbox was delayed due to lack of available buffering resources in the Sbox.
NO_S_REQTBL_ENTRIES	bxx1xxxxx	An incoming local processor Rd or WR that needed to generate a transaction to Home (for example, LLC RD Miss) was delayed because the Sbox Request Table was full.
WB_PENDING	bx1xxxxxx	An incoming remote Intel QPI snoop request to the LLC was delayed because it conflicted with an existing transaction that had a WB to Home pending.
NACK2_ELSE	b1xxxxxxx	Some incoming message to the LLC was delayed for a reason not covered by any of the other MAF_NACK1 or MAF_NACK2 sub-events.

OCCUPANCY_IPQ

- **Title:** IPQ Occupancy
- **Category:** Queue Occupancy
- **Event Code:** 0x1A, **Max. Inc/Cyc:** 8,
- **Definition:** Cumulative count of occupancy in the LLC's Ingress Probe Queue.

OCCUPANCY_IRQ

- **Title:** IRQ Occupancy
- **Category:** Queue Occupancy
- **Event Code:** 0x18, **Max. Inc/Cyc:** 24,
- **Definition:** Cumulative count of occupancy in the LLC's Ingress Response Queue.

OCCUPANCY_MAF

- **Title:** MAF Occupancy
- **Category:** Queue Occupancy
- **Event Code:** 0x1E, **Max. Inc/Cyc:** 16,
- **Definition:** Cumulative count of occupancy in the LLC's Miss Address File.

OCCUPANCY_VIQ

- **Title:** VIQ Occupancy
- **Category:** Queue Occupancy
- **Event Code:** 0x1C, **Max. Inc/Cyc:** 8,
- **Definition:** Cumulative count of the occupancy in the Victim Ingress Queue.

SEMAPHORE_NACKS

- **Title:** Semaphore NACKs.
- **Category:** Local - CC
- **Event Code:** 0x20, **Max. Inc/Cyc:** 1,
- **Definition:** Number of Semaphore NACKS. Gives extent of local socket degradation generated by the fairness logic.

SINKS_C2P

- **Title:** C2P Sinks
- **Category:** Ring - WIR
- **Event Code:** 0x06, **Max. Inc/Cyc:** 3,
- **Definition:** Number of messages sunk by the processor that were sent by one of the Cboxes.
- **NOTE:** Each sink represents the transfer of 32 bytes, or 2 sinks per cache line.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
IV	b00000001	IV (Cbox snoops of a processor's cache)
AK	b00000010	AK (GO messages send to the processor)
BL	b00000100	BL (LLC data sent back to processor)

SINKS_P2C

- **Title:** P2C Sinks
- **Category:** Ring - WIR
- **Event Code:** 0x05, **Max. Inc/Cyc:** 3,
- **Definition:** Number of messages sunk from the ring at the Cbox that were sent by one of the local processors.
- **NOTE:** Each sink represents the transfer of 32 bytes, or 2 sinks per cache line.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
AD	b00000001	AD (Core RD/WR requests to the LLC)
AK	b00000010	AK (Core snoop responses to the LLC)
BL	b00000100	BL (explicit and implicit WB data from the core to the LLC)

SINKS_S2C

- **Title:** S2C Sinks
- **Category:** Ring - WIR
- **Event Code:** 0x07, **Max. Inc/Cyc:** 3,
- **Definition:** Number of messages sunk from the ring at the Cbox that were sent by the Sbox.
- **NOTE:** Each sink represents the transfer of 32 bytes, or 2 sinks per cache line.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
AD	b00000001	AD (Intel QPI snoop request of LLC)
AK	b00000010	AK (Intel QPI completions sent to LLC)
BL	b00000100	BL (Data Fills sent to the LLC in response to RD requests)

SINKS_S2P_BL

- **Title:** S2P Sinks
- **Category:** Ring - WIRa
- **Event Code:** 0x08, **Max. Inc/Cyc:** 1,
- **Definition:** Number BL ring messages sunk by the processor that were sent from the Sbox. This covers BL only, because that is the only kind of message the Sbox can send to a processor.
- **NOTE:** Each sink represents the transfer of 32 bytes, or 2 sinks per cache line.

SNP_HITS

- **Title:** Snoop Hits in LLC
- **Category:** Local - CC
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Number of Intel QPI snoops that hit in the LLC according to state of LLC when the hit occurred. GotoS: LLC Data or Code Read Snoop Hit 'x' state in remote cache. GotoI: LLC Data Read for Ownership Snoop Hit 'x' state in remote cache.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
REMOTE_RD_HITM	b00xxxxx1	Intel QPI SnpData or SnpCode hit M line in LLC
REMOTE_RD_HITE	b00xxxx1x	Intel QPI SnpData or SnpCode hit E line in LLC
REMOTE_RD_HITS	b00xxx1xx	Intel QPI SnpData or SnpCode hit S line in LLC
REMOTE_RFO_HITM	b00xx1xxx	Intel QPI SnpInvOwn or SnpInvItoE hit M line in LLC
REMOTE_RFO_HITE	b00x1xxxx	Intel QPI SnpInvOwn or SnpInvItoE hit E line in LLC
REMOTE_RFO_HITS	b001xxxxx	Intel QPI SnpInvOwn or SnpInvItoE hit S line in LLC
REMOTE_HITM	b00xx1xx1	Intel QPI Snoops that hit M line in LLC
REMOTE_HITE	b00x1xx1x	Intel QPI Snoops that hit E line in LLC
REMOTE_HITS	b001xx1xx	Intel QPI Snoops that hit S line in LLC
REMOTE_ANY	b11111111	Intel QPI Snoops that hit in LLC (any line state)

SNPS

- **Title:** Snoops to LLC
- **Category:** Local - CC
- **Event Code:** 0x27, **Max. Inc/Cyc:** 1,
- **Definition:** Number of Intel QPI snoops seen by the LLC.
- **NOTE:** Subtract CACHE_CHAR_QUAL.ANY_HIT from this event to determine how many snoops missed the LLC.

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
REMOTE_RD	b000000x1	Remote Read - Goto S State. Intel QPI snoops (SnpData or SnpCode) to LLC that caused a transition to S in the cache. NOTE: ALL SnpData and SnpCode transactions are counted. If SnpData HITM policy is M->I, this subevent will capture those snoops.
REMOTE_RFO	b0000001x	Remote RFO - Goto I State. Intel QPI snoops (SnpInvOwn or SnpInvItoE) to LLC that caused an invalidate of a cache line.
REMOTE_ANY	b00000011	Intel QPI snoops to LLC that hit in the cache line

STARVED_EGRESS

- **Title:** Egress Queue Starved
- **Category:** Local - EGR
- **Event Code:** 0x0B, **Max. Inc/Cyc:** 8,
- **Definition:** Number of cycles that an Egress Queue is in starvation

Extension	umask [15:8]	Description
---	b00000000	(*nothing will be counted*)
P2C_AD_SB	b00000001	Processor-to-Cbox AD Egress that injects in the direction toward the nearest Sbox
C2S_AD_SB	b00000010	Cbox to Sbox AD Egress.
AD_SB	b00000011	Sum of AD Egresses that injects in the direction toward the nearest Sbox
AD_NSB	b00000100	Sum across both AD Egress that inject in the direction away from the nearest Sbox.
AD	b00000111	Sum across all AD Egresses
AK_SB	b00001000	AK Egress that injects in the direction toward the nearest Sbox.
AK_NSB	b00010000	AK Egress that injects in the direction away from the nearest Sbox.
AK	b00011000	Sum across all AK Egresses.
BL_SB	b00100000	BL Egress that injects in the direction toward the nearest Sbox.
BL_NSB	b01000000	BL Egress that injects in the direction away from the nearest Sbox.
BL	b01100000	Sum across all BL Egresses.
IV	b10000000	IV Egress

TRANS_IPQ

- **Title:** IPQ Transactions
- **Category:** Queue Occupancy
- **Event Code:** 0x1B, **Max. Inc/Cyc:** 1,
- **Definition:** Number of Intel QPI snoop probes that entered the LLC's Ingress Probe Queue.

TRANS_IRQ

- **Title:** IRQ Transactions
- **Category:** Queue Occupancy
- **Event Code:** 0x19, **Max. Inc/Cyc:** 1,
- **Definition:** Number of processor RD and/or WR requests to the LLC that entered the Ingress Response Queue.

TRANS_MAF

- **Title:** MAF Transactions
- **Category:** Queue Occupancy
- **Event Code:** 0x1F, **Max. Inc/Cyc:** 1,
- **Definition:** Number of transactions to allocate entries in LLC's Miss Address File.

TRANS_VIQ

- **Title:** VIQ Transactions
- **Category:** Queue Occupancy
- **Event Code:** 0x1D, **Max. Inc/Cyc:** 1,
- **Definition:** Number of LLC victims to enter the Victim Ingress Queue. All LLC victims pass through this queue. Including those that end up not requiring a WB.

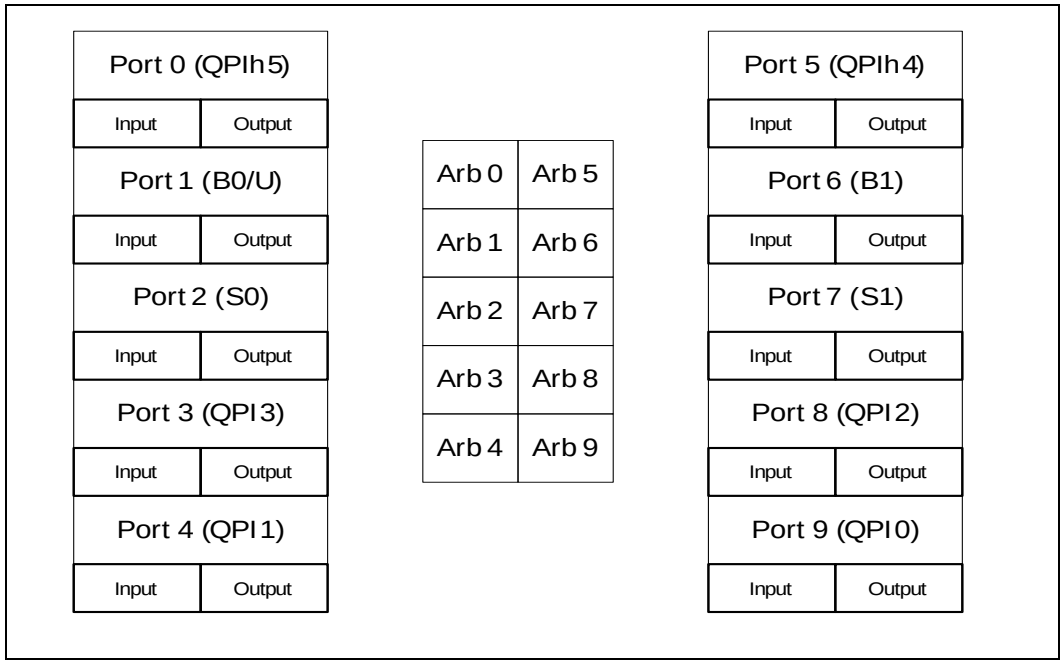
5.6 Rbox Performance Monitoring

5.6.1 Overview of the Rbox

The Crossbar Router (Rbox) is a 10 port switch/router implementing the QuickPath Interconnect Link layers. The Rbox is responsible for routing and transmitting all intra- and inter-processor communication. The Rbox is connected to 4 full 80b and 2 half-width 40b QuickPath Interconnect links. The Rbox is also connected to both Bboxes (one connection is shared with a connection to the Ubox) and both Sboxes by 80b interfaces.

The Rbox consists of 10 identical ports and a wire crossbar that connects the ports together. Each port contains three main sections as shown in Figure 5-3: the input port, the output port, and the arbitration control.

Figure 5-3. Rbox Block Diagram



5.6.1.1 Rbox Input Port

The Rbox input port is responsible for storing incoming packets from the Bbox/Sbox/Uboxs, and Intel QPI Links using the Intel QuickPath Interconnect protocol. Data from each packet header is consolidated and sent to the Rbox arbiter.

Rbox input ports have two structures of important to performance monitoring; Entry overflow table (EOT) and Entry Table (ET). Rbox PMU supports performance monitoring in these two structures.

5.6.1.2 Rbox Arbitration Control

The Rbox arbitration control is responsible for selecting when packets move from the input ports to the output ports and which output port they go to if there are multiple options specified.

Rbox arbitration does not have any storage structures. This part of the logic basically determines which port to route the packet and then arbitrate to secure a route to that port through the cross-bar.

The arbitration is done at 3 levels: queue, port and global arbitration. Rbox PMUs support performance monitoring at the arbitration control.

5.6.1.3 Rbox Output Port

The Rbox output port acts as a virtual wire that is responsible for de-coupling the crossbar from further downstream paths to on-chip or off-chip ports while carrying out the Link layer functions.

5.6.1.4 Rbox Link Layer Resources

Each Rbox port supports up to three virtual networks (VN0, VN1, and VNA) as defined by the *Intel® QuickPath Interconnect Specification*.

Table 5-41. Input Buffering Per Port

Message Class	Abbr	VNA		VN0		VN1	
		Pkts	Flits	Pkts	Flits	Pkts	Flits
Home	HOM		96	1	up to 2	1	up to 2
Snoop	SNP			1	up to 2	1	up to 2
Non-Data Response	NDR			1	up to 2	1	up to 2
Data Response	DRS			1	up to 11	1	up to 11
Non-Coherent Standard	NCS			1	up to 3	1	up to 3
Non-Coherent Bypass	NCB			1	up to 11	1	up to 11

5.6.2 Rbox Performance Monitoring Overview

The Rbox supports performance event monitoring through its Performance Monitoring Unit (PMU). At a high level, the Rbox PMU supports features comparable to other uncore PMUs. Rbox PMUs support both Global and Local PMU freeze/unfreeze. Rbox PMUs are accessible through Chip-Set Registers (CSRs).

Rbox PMU consists of 16 48b-wide performance monitoring data counters and a collection of other peripheral control registers. The count values of all 16 counters can be cleared by writing the R_CSR_PMON_PERF_MASTER.clr bit.

The counters, along with the control register paired with each one, are split. Half of the counters (0-7) can monitor events occurring on the ‘left’ side of the RBox (ports 0-4) and the other half (8-15) monitor the ‘right’ side (ports 5-10).

Since the Rbox consists of 10 almost identical ports, Rbox perfmon events consist of an identical set of events for each port. The Rbox perfmon usage model allows monitoring of multiple ports at the same time. Rbox PMUs do not provide any global performance monitoring events.

Unlike other boxes, event programming in the Rbox is hierarchical. It is necessary to program multiple CSRs to select the event to be monitored. In order to program an event, each of the control registers for its accompanying counter must be redirected to a subcontrol register attached to a specific port by setting the *.pt_sel* field in the control register. Once a port is chosen, each control register can then be redirected to one of 2 IPERF control registers (for Rbox Input Port or RIX events), one of 2 fields in a QLX control register (for Rbox Arbitration Queue or QLX events) or one of 2 mask/match registers. Therefore, it is possible to monitor up to two of any event per port.

The Rbox also includes a pair of mask/match registers on each port that allow a user to match packets serviced (packet is transferred from input to output port) by the Rbox according to various standard packet fields such as message class, opcode, and so forth.

5.6.2.1 Choosing An Event To Monitor - Example

1) Pick an event to monitor (for example, FLITS_SENT)

2) Pick a port to monitor on (for example, QPI3)

3) Pick a generic counter (control+data) that can monitor an event on that port. (for example, R_CSR_PERF_CNT/CNT_CTRL4) and set the control to point to it (R_CSR_PERF_CNT_CTRL_4.pt_sel == 0x3).

4) Pick one of the two sub counters that allow a user to monitor the event (R_CSR_P03_IPERF1), program it to monitor the chosen event (R_CSR_P03_IPERF1[31] = 0x1) and set the generic control to point to it (R_CSR_PERF_CNT_CTRL4.ev_sel == 0x1).

5.6.2.2 Rbox PMU - Overflow, Freeze and Unfreeze

Rbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Rbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, “Global Enable/Disable”).

Rbox PMU can be frozen due to one of three reasons:

- *Globally:* Ubox sends a disable signal (and R_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, “Global Enable/Disable”) or *local* (SW writes 0 to R_CSR_PMON_PERF_MASTER.en when R_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.
- *Locally:* The Rbox was set to local control (R_CSR_PMON_PERF_MASTER.glb_lcl = 0) and a Rbox counter overflowed.

If an overflow is detected from a Rbox performance counter, the overflow bit is set in the overflowing data register (R_CSR_PERF_CNT_CTRL_{15-0}.ov), and forwarded up the chain towards the Sbox. That is, if an Rbox counter overflows, a notification is sent and stored in Sbox (S_CSR_PMON_SUMMARY.ov_r). Refer to Table 5-64, “S_CSR_PMON_SUMMARY Register Fields” to determine how each Rbox’s overflow bit is accumulated in the attached Sbox.

The Ubox may be configured to freeze all uncore counting (refer to Table 5-65, “S_CSR_PMON_FRZ_EN Register Fields”) when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field responsible for the freeze, must be cleared at which point the Rbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, "Enabling a New Sample Interval from Frozen Counters."), counting will resume.

5.6.3 Rbox Performance Monitoring CSRs

Table 5-42. Rbox PMU Summary (Sheet 1 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
Counters 15-8		
R_CSR_PERF_CNT_15	0x8AB8	Performance Counter 15
R_CSR_PERF_CNT_14	0x8AB0	Performance Counter 14
R_CSR_PERF_CNT_13	0x8AA8	Performance Counter 13
R_CSR_PERF_CNT_12	0x8AA0	Performance Counter 12
R_CSR_PERF_CNT_11	0x8A98	Performance Counter 11
R_CSR_PERF_CNT_10	0x8A90	Performance Counter 10
R_CSR_PERF_CNT_9	0x8A88	Performance Counter 9
R_CSR_PERF_CNT_8	0x8A80	Performance Counter 8
Control for Counters 15-8		
R_CSR_PERF_CNT_CTRL_15	0x8A78	Performance Counter 15 Control
R_CSR_PERF_CNT_CTRL_14	0x8A70	Performance Counter 14 Control
R_CSR_PERF_CNT_CTRL_13	0x8A68	Performance Counter 13 Control
R_CSR_PERF_CNT_CTRL_12	0x8A60	Performance Counter 12 Control
R_CSR_PERF_CNT_CTRL_11	0x8A58	Performance Counter 11 Control
R_CSR_PERF_CNT_CTRL_10	0x8A50	Performance Counter 10 Control
R_CSR_PERF_CNT_CTRL_9	0x8A48	Performance Counter 9 Control
R_CSR_PERF_CNT_CTRL_8	0x8A40	Performance Counter 8 Control
IPERF Registers (for RIX Events) for Ports 9-5		
R_CSR_P09_IPERF1	0x8878	RIX Performance Counter Event Config 1 (Port 9)
R_CSR_P09_IPERF0	0x8870	RIX Performance Counter Event Config 0 (Port 9)
R_CSR_P08_IPERF1	0x8678	RIX Performance Counter Event Config 1 (Port 8)
R_CSR_P08_IPERF0	0x8670	RIX Performance Counter Event Config 0 (Port 8)
R_CSR_P07_IPERF1	0x8478	RIX Performance Counter Event Config 1 (Port 7)
R_CSR_P07_IPERF0	0x8470	RIX Performance Counter Event Config 0 (Port 7)
R_CSR_P06_IPERF1	0x8278	RIX Performance Counter Event Config 1 (Port 6)
R_CSR_P06_IPERF0	0x8270	RIX Performance Counter Event Config 0 (Port 6)
R_CSR_P05_IPERF1	0x8078	RIX Performance Counter Event Config 1 (Port 5)
R_CSR_P05_IPERF0	0x8070	RIX Performance Counter Event Config 0 (Port 5)

Table 5-42. Rbox PMU Summary (Sheet 2 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
ARB Registers (for QLX Events) for Ports 9-5		
R_CSR_P04_ARB_PERF9	0x8810	Arbiter PMU Configuration Register for Port 9
R_CSR_P03_ARB_PERF8	0x8610	Arbiter PMU Configuration Register for Port 8
R_CSR_P02_ARB_PERF7	0x8410	Arbiter PMU Configuration Register for Port 7
R_CSR_P01_ARB_PERF6	0x8210	Arbiter PMU Configuration Register for Port 6
R_CSR_P00_ARB_PERF5	0x8010	Arbiter PMU Configuration Register for Port 5
Port 9 Match / Mask Registers		
R_CSR_P09_MM_CFG_1	0x88C8	Match/Mask Set 1 Configuration Register for Port 09
R_CSR_P09_MM_CFG_0	0x88C0	Match/Mask Set 0 Configuration Register for Port 09
R_CSR_P09_MATCH_MSB_1	0x88B8	Match Set 1 MSB {63:32} for Port 09
R_CSR_P09_MATCH_MSB_0	0x88B0	Match Set 0 MSB {63:32} for Port 09
R_CSR_P09_MATCH_LSB_1	0x88A8	Match Set 1 LSB {31:0} for Port 09
R_CSR_P09_MATCH_LSB_0	0x88A0	Match Set 0 LSB {31:0} for Port 09
R_CSR_P09_MASK_MSB_1	0x8898	Mask Set 1 MSB {63:32} for Port 09
R_CSR_P09_MASK_MSB_0	0x8890	Mask Set 0 MSB {63:32} for Port 09
R_CSR_P09_MASK_LSB_1	0x8888	Mask Set 1 LSB {31:0} for Port 09
R_CSR_P09_MASK_LSB_0	0x8880	Mask Set 0 LSB {31:0} for Port 09
Port 8 Match / Mask Registers		
R_CSR_P08_MM_CFG_1	0x86C8	Match/Mask Set 1 Configuration Register for Port 08
R_CSR_P08_MM_CFG_0	0x86C0	Match/Mask Set 0 Configuration Register for Port 08
R_CSR_P08_MATCH_MSB_1	0x86B8	Match Set 1 MSB {63:32} for Port 08
R_CSR_P08_MATCH_MSB_0	0x86B0	Match Set 0 MSB {63:32} for Port 08
R_CSR_P08_MATCH_LSB_1	0x86A8	Match Set 1 LSB {31:0} for Port 08
R_CSR_P08_MATCH_LSB_0	0x86A0	Match Set 0 LSB {31:0} for Port 08
R_CSR_P08_MASK_MSB_1	0x8698	Mask Set 1 MSB {63:32} for Port 08
R_CSR_P08_MASK_MSB_0	0x8690	Mask Set 0 MSB {63:32} for Port 08
R_CSR_P08_MASK_LSB_1	0x8688	Mask Set 1 LSB {31:0} for Port 08
R_CSR_P08_MASK_LSB_0	0x8680	Mask Set 0 LSB {31:0} for Port 08
Port 7 Match / Mask Registers		
R_CSR_P07_MM_CFG_1	0x84C8	Match/Mask Set 1 Configuration Register for Port 07
R_CSR_P07_MM_CFG_0	0x84C0	Match/Mask Set 0 Configuration Register for Port 07
R_CSR_P07_MATCH_MSB_1	0x84B8	Match Set 1 MSB {63:32} for Port 07
R_CSR_P07_MATCH_MSB_0	0x84B0	Match Set 0 MSB {63:32} for Port 07
R_CSR_P07_MATCH_LSB_1	0x84A8	Match Set 1 LSB {31:0} for Port 07
R_CSR_P07_MATCH_LSB_0	0x84A0	Match Set 0 LSB {31:0} for Port 07
R_CSR_P07_MASK_MSB_1	0x8498	Mask Set 1 MSB {63:32} for Port 07
R_CSR_P07_MASK_MSB_0	0x8490	Mask Set 0 MSB {63:32} for Port 07

Table 5-42. Rbox PMU Summary (Sheet 3 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
R_CSR_P07_MASK_LSB_1	0x8488	Mask Set 1 LSB {31:0} for Port 07
R_CSR_P07_MASK_LSB_0	0x8480	Mask Set 0 LSB {31:0} for Port 07
Port 6 Match/Mask Registers		
R_CSR_P06_MM_CFG_1	0x82C8	Match/Mask Set 1 Configuration Register for Port 06
R_CSR_P06_MM_CFG_0	0x82C0	Match/Mask Set 0 Configuration Register for Port 06
R_CSR_P06_MATCH_MSB_1	0x82B8	Match Set 1 MSB {63:32} for Port 06
R_CSR_P06_MATCH_MSB_0	0x82B0	Match Set 0 MSB {63:32} for Port 06
R_CSR_P06_MATCH_LSB_1	0x82A8	Match Set 1 LSB {31:0} for Port 06
R_CSR_P06_MATCH_LSB_0	0x82A0	Match Set 0 LSB {31:0} for Port 06
R_CSR_P06_MASK_MSB_1	0x8298	Mask Set 1 MSB {63:32} for Port 06
R_CSR_P06_MASK_MSB_0	0x8290	Mask Set 0 MSB {63:32} for Port 06
R_CSR_P06_MASK_LSB_1	0x8288	Mask Set 1 LSB {31:0} for Port 06
R_CSR_P06_MASK_LSB_0	0x8280	Mask Set 0 LSB {31:0} for Port 06
Port 5 Match/Mask Registers		
R_CSR_P05_MM_CFG_1	0x80C8	Match/Mask Set 1 Configuration Register for Port 05
R_CSR_P05_MM_CFG_0	0x80C0	Match/Mask Set 0 Configuration Register for Port 05
R_CSR_P05_MATCH_MSB_1	0x80B8	Match Set 1 MSB {63:32} for Port 05
R_CSR_P05_MATCH_MSB_0	0x80B0	Match Set 0 MSB {63:32} for Port 05
R_CSR_P05_MATCH_LSB_1	0x80A8	Match Set 1 LSB {31:0} for Port 05
R_CSR_P05_MATCH_LSB_0	0x80A0	Match Set 0 LSB {31:0} for Port 05
R_CSR_P05_MASK_MSB_1	0x8098	Mask Set 1 MSB {63:32} for Port 05
R_CSR_P05_MASK_MSB_0	0x8090	Mask Set 0 MSB {63:32} for Port 05
R_CSR_P05_MASK_LSB_1	0x8088	Mask Set 1 LSB {31:0} for Port 05
R_CSR_P05_MASK_LSB_0	0x8080	Mask Set 0 LSB {31:0} for Port 05
Global Control Register		
R_CSR_PMON_PERF_MASTER	0x2AC0	Performance Monitor Master Control
Right Side Counters		
Counters 7-0		
R_CSR_PERF_CNT_7	0x2AB8	Performance Counter 7
R_CSR_PERF_CNT_6	0x2AB0	Performance Counter 6
R_CSR_PERF_CNT_5	0x2AA8	Performance Counter 5
R_CSR_PERF_CNT_4	0x2AA0	Performance Counter 4
R_CSR_PERF_CNT_3	0x2A98	Performance Counter 3
R_CSR_PERF_CNT_2	0x2A90	Performance Counter 2
R_CSR_PERF_CNT_1	0x2A88	Performance Counter 1

Table 5-42. Rbox PMU Summary (Sheet 4 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
R_CSR_PERF_CNT_0	0x2A80	Performance Counter 0
Control for Counters 7-0		
R_CSR_PERF_CNT_CTRL_7	0x2A78	Performance Counter 7 Control
R_CSR_PERF_CNT_CTRL_6	0x2A70	Performance Counter 6 Control
R_CSR_PERF_CNT_CTRL_5	0x2A68	Performance Counter 5 Control
R_CSR_PERF_CNT_CTRL_4	0x2A60	Performance Counter 4 Control
R_CSR_PERF_CNT_CTRL_3	0x2A58	Performance Counter 3 Control
R_CSR_PERF_CNT_CTRL_2	0x2A50	Performance Counter 2 Control
R_CSR_PERF_CNT_CTRL_1	0x2A48	Performance Counter 1 Control
R_CSR_PERF_CNT_CTRL_0	0x2A40	Performance Counter 0 Control
IPERF Registers (for RIX Events) for Ports 4-0		
R_CSR_P04_IPERF1	0x2878	RIX Performance Counter Event Config 1 (Port 4)
R_CSR_P04_IPERF0	0x2870	RIX Performance Counter Event Config 0 (Port 4)
R_CSR_P03_IPERF1	0x2678	RIX Performance Counter Event Config 1 (Port 3)
R_CSR_P03_IPERF0	0x2670	RIX Performance Counter Event Config 0 (Port 3)
R_CSR_P02_IPERF1	0x2478	RIX Performance Counter Event Config 1 (Port 2)
R_CSR_P02_IPERF0	0x2470	RIX Performance Counter Event Config 0 (Port 2)
R_CSR_P01_IPERF1	0x2278	RIX Performance Counter Event Config 1 (Port 1)
R_CSR_P01_IPERF0	0x2270	RIX Performance Counter Event Config 0 (Port 1)
R_CSR_P00_IPERF1	0x2078	RIX Performance Counter Event Config 1 (Port 0)
R_CSR_P00_IPERF0	0x2070	RIX Performance Counter Event Config 0 (Port 0)
ARB Registers (for QLX Events) for Ports 9-5		
R_CSR_P04_ARB_PERF4	0x2810	Arbiter PMU Configuration Register for Port 4
R_CSR_P03_ARB_PERF3	0x2610	Arbiter PMU Configuration Register for Port 3
R_CSR_P02_ARB_PERF2	0x2410	Arbiter PMU Configuration Register for Port 2
R_CSR_P01_ARB_PERF1	0x2210	Arbiter PMU Configuration Register for Port 1
R_CSR_P00_ARB_PERF0	0x2010	Arbiter PMU Configuration Register for Port 0
Port 4 Match / Mask Registers		
R_CSR_P04_MM_CFG_1	0x28C8	Match/Mask Set 1 Configuration Register for Port 04
R_CSR_P04_MM_CFG_0	0x28C0	Match/Mask Set 0 Configuration Register for Port 04
R_CSR_P04_MATCH_MSB_1	0x28B8	Match Set 1 MSB (63:32} for Port 04
R_CSR_P04_MATCH_MSB_0	0x28B0	Match Set 0 MSB (63:32} for Port 04
R_CSR_P04_MATCH_LSB_1	0x28A8	Match Set 1 LSB (31:0} for Port 04

Table 5-42. Rbox PMU Summary (Sheet 5 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
R_CSR_P04_MATCH_LSB_0	0x28A0	Match Set 0 LSB (31:0} for Port 04
R_CSR_P04_MASK_MSB_1	0x2898	Mask Set 1 MSB (63:32} for Port 04
R_CSR_P04_MASK_MSB_0	0x2890	Mask Set 0 MSB (63:32} for Port 04
R_CSR_P04_MASK_LSB_1	0x2888	Mask Set 1 LSB (31:0} for Port 04
R_CSR_P04_MASK_LSB_0	0x2880	Mask Set 0 LSB (31:0} for Port 04
Port 3 Match/Mask Registers		
R_CSR_P03_MM_CFG_1	0x26C8	Match/Mask Set 1 Configuration Register for Port 03
R_CSR_P03_MM_CFG_0	0x26C0	Match/Mask Set 0 Configuration Register for Port 03
R_CSR_P03_MATCH_MSB_1	0x26B8	Match Set 1 MSB (63:32} for Port 03
R_CSR_P03_MATCH_MSB_0	0x26B0	Match Set 0 MSB (63:32} for Port 03
R_CSR_P03_MATCH_LSB_1	0x26A8	Match Set 1 LSB (31:0} for Port 03
R_CSR_P03_MATCH_LSB_0	0x26A0	Match Set 0 LSB (31:0} for Port 03
R_CSR_P03_MASK_MSB_1	0x2698	Mask Set 1 MSB (63:32} for Port 03
R_CSR_P03_MASK_MSB_0	0x2690	Mask Set 0 MSB (63:32} for Port 03
R_CSR_P03_MASK_LSB_1	0x2688	Mask Set 1 LSB (31:0} for Port 03
R_CSR_P03_MASK_LSB_0	0x2680	Mask Set 0 LSB (31:0} for Port 03
Port 2 Match/Mask Registers		
R_CSR_P02_MM_CFG_1	0x24C8	Match/Mask Set 1 Configuration Register for Port 02
R_CSR_P02_MM_CFG_0	0x24C0	Match/Mask Set 0 Configuration Register for Port 02
R_CSR_P02_MATCH_MSB_1	0x24B8	Match Set 1 MSB (63:32} for Port 02
R_CSR_P02_MATCH_MSB_0	0x24B0	Match Set 0 MSB (63:32} for Port 02
R_CSR_P02_MATCH_LSB_1	0x24A8	Match Set 1 LSB (31:0} for Port 02
R_CSR_P02_MATCH_LSB_0	0x24A0	Match Set 0 LSB (31:0} for Port 02
R_CSR_P02_MASK_MSB_1	0x2498	Mask Set 1 MSB (63:32} for Port 02
R_CSR_P02_MASK_MSB_0	0x2490	Mask Set 0 MSB (63:32} for Port 02
R_CSR_P02_MASK_LSB_1	0x2488	Mask Set 1 LSB (31:0} for Port 02
R_CSR_P02_MASK_LSB_0	0x2480	Mask Set 0 LSB (31:0} for Port 02
Port 1 Match/Mask Registers		
R_CSR_P01_MM_CFG_1	0x22C8	Match/Mask Set 1 Configuration Register for Port 01
R_CSR_P01_MM_CFG_0	0x22C0	Match/Mask Set 0 Configuration Register for Port 01
R_CSR_P01_MATCH_MSB_1	0x22B8	Match Set 1 MSB (63:32} for Port 01
R_CSR_P01_MATCH_MSB_0	0x22B0	Match Set 0 MSB (63:32} for Port 01
R_CSR_P01_MATCH_LSB_1	0x22A8	Match Set 1 LSB (31:0} for Port 01
R_CSR_P01_MATCH_LSB_0	0x22A0	Match Set 0 LSB (31:0} for Port 01
R_CSR_P01_MASK_MSB_1	0x2298	Mask Set 1 MSB (63:32} for Port 01
R_CSR_P01_MASK_MSB_0	0x2290	Mask Set 0 MSB (63:32} for Port 01
R_CSR_P01_MASK_LSB_1	0x2288	Mask Set 1 LSB (31:0} for Port 01
R_CSR_P01_MASK_LSB_0	0x2280	Mask Set 0 LSB (31:0} for Port 01
Port 0 Match/Mask Registers		

Table 5-42. Rbox PMU Summary (Sheet 6 of 6)

CSR Name	AddrOffset [15:0]	Description
Left Side Counters		
R_CSR_P00_MM_CFG_1	0x20C8	Match/Mask Set 1 Configuration Register for Port 00
R_CSR_P00_MM_CFG_0	0x20C0	Match/Mask Set 0 Configuration Register for Port 00
R_CSR_P00_MATCH_MSB_1	0x20B8	Match Set 1 MSB (63:32} for Port 00
R_CSR_P00_MATCH_MSB_0	0x20B0	Match Set 0 MSB (63:32} for Port 00
R_CSR_P00_MATCH_LSB_1	0x20A8	Match Set 1 LSB (31:0} for Port 00
R_CSR_P00_MATCH_LSB_0	0x20A0	Match Set 0 LSB (31:0} for Port 00
R_CSR_P00_MASK_MSB_1	0x2098	Mask Set 1 MSB (63:32} for Port 00
R_CSR_P00_MASK_MSB_0	0x2090	Mask Set 0 MSB (63:32} for Port 00
R_CSR_P00_MASK_LSB_1	0x2088	Mask Set 1 LSB (31:0} for Port 00
R_CSR_P00_MASK_LSB_0	0x2080	Mask Set 0 LSB (31:0} for Port 00

5.6.3.1 Rbox Performance Monitors To Port Mapping

Table 5-43. Rbox Port Map

Port ID	Rbox Port#	L/R Box	PMU Cnt 0-7	PMU Cnt 8-15	IPERF Addresses	ARB_PERF Addresses	Match/Mask Addresses
QPIh5	0	L	b000	NA	0x2078,0x2070	0x2010	0x2080-0x20C8
B0U	1	L	b001	NA	0x2278,0x2270	0x2210	0x2280-0x22C8
S0	2	L	b010	NA	0x2478,0x2470	0x2410	0x2480-0x24C8
QPI3	3	L	b011	NA	0x2678,0x2670	0x2610	0x2680-0x26C8
QPI1	4	L	b100	NA	0x2878,0x2870	0x2810	0x2880-0x28C8
QPIh4	5	R	NA	b000	0x8078,0x8070	0x8010	0x8080-0x80C8
B1	6	R	NA	b001	0x8278,0x8270	0x8210	0x8280-0x82C8
S1	7	R	NA	b010	0x8478,0x8470	0x8410	0x8480-0x84C8
QPI2	8	R	NA	b011	0x8688,0x8680	0x8610	0x8680-0x86C8
QPI0	9	R	NA	b100	0x8878,0x8870	0x8810	0x8880-0x88C8

5.6.3.2 Rbox Box Level PMON state

R_CSR_PMON_PERF_MASTER controls the general characteristics of the Rbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

Table 5-44. R_CSR_PMON_PERF_MASTER Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	63:3	RO_NA	0	Read zero; writes ignored. (?)
clr	2	RW_RW	0	Writing 1 clears all the Rbox counters as well as the associated overflow bits found in the control registers (R_CSR_PERF_CNT_CTRLx.ov)
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: Global. Disable counters on Ubox PMU enable 0: Local. Disables Counters on any local counter overflow.

Table 5-44. R_CSR_PMON_PERF_MASTER Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Type	HW Reset Val	Description
freeze	0	RW_RW	0	Freeze/unfreeze Rbox PMU counters. Read of this field shows the freeze state of the PMU counters. PMU counters can be frozen via CSR write or PMU overflow. See Freeze and Unfreeze section for more information. 1: write '1 to freeze Rbox PMU counters 0: Write '0 to unfreeze Rbox PMU counters. If the counters are not frozen, writing '0 has no effect.

5.6.3.3 Rbox PMON state - Counter/Control Pairs + Filters

The following table defines the layout of the Itanium processor 9500 series Rbox performance monitor control registers. The main task of these configuration registers is to select the port (*.pt_sel*) to monitor events on and choose the subcontrol register (*.ev_sel*) that selects the event to be monitored by the respective data counter.

Table 5-45. R_CSR_PERF_CNT_CTRL_{15-0} Fields

Field	Bits	Access	HW Reset Val	Description
ig	63:25	RO_NA	0	Read zero; writes ignored.
ov	24	RW_RW	0	Performance Counter overflow status bit.
rsv	23	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.
ig	22	RO_NA	0	Read zero; writes ignored.
pt_sel	21:19	RW_RW	0	Count Port Select. Values of 000-100 are valid for the Rbox ports. 111-101 are Reserved.
ev_sel	18:16	RW_RW	0	Count Event Select: 000: Port X RIX Performance Event 0 001: Port X RIX Performance Event 1 NOTE: RIX events are set with R_CSR_IPERF* 010: Port X QLX/GBX Performance Event 0 011: Port X QLX/GBX Performance Event 1 NOTE: QLX events are set with R_CSR_ARB_PERF* 100: Port X Mask & Match 0 101: Port X Mask & Match 1 NOTE: Mask/Match events are set with R_CSR_MATCH R_CSR 11x: Reserved
ig	15:0	RO_NA	0	Read zero; writes ignored.

The Rbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-46. R_CSR_PERF_CNT_{15-0} Fields

Field	Bits	Access	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored.
cnt	47:0	RW_RW	0	Performance Counter Value (47:0)

5.6.3.4 Rbox RIX Performance Monitoring Control Registers

The following table contains the events that can be monitored if one of the RIX (IPERF) registers was chosen to select the event.

Table 5-47. R_CSR_P{9-0}_IPERF{1-0} Registers (Sheet 1 of 2)

Field	Bits	Access	HW Reset Val	Description
FLT_SENT	31	RW_RW	0x0	Flit Sent
FLT_RCVD	30	RW_RW	0x0	Flit Received
NSP_SENT	29	RW_RW	0x0	Non-special Flit Sent
NSP_RCVD	28	RW_RW	0x0	Non-special Flit Received
OUTB_OV	27	RW_RW	0x0	Running Counter of Output Buffer depth Overflow
OUTB_WR	26	RW_RW	0x0	Flit written into Output Buffer
OUTB_NE	25	RW_RW	0x0	Output Buffer Not Empty
RETQ_RD	24	RW_RW	0x0	Retry Buffer Read Count
INPCKTERR	23	RW_RW	0x0	Incoming Packet Error
FSTENOUGH	22	RW_RW	0x0	Fast Enough path used
EOT_OV	21	RW_RW	0x0	Running Counter of EOT Depth Overflow
EOT_WR	20	RW_RW	0x0	EOT Entry inserted

Table 5-47. R_CSR_P{9-0}_IPERF{1-0} Registers (Sheet 2 of 2)

Field	Bits	Access	HW Reset Val	Description
EOTMSGSEL	19:17	RW_RW	0x0	Message Class select for EOT Depth Overflow and EOT Entry inserted events. 001: Home0 010: Home1 011: Snoop 100: Non-Data Response 101: Data Response 110: Non-Coherent Standard 111: Non-Coherent Bypass
EOT_NE	16	RW_RW	0x0	Count cycles EOT is not Empty
ARB_MSGSEL	15:9	RW_RW	0x04	Allocation to Arb Select Bit Mask: 0b1XXXXXX: Home 0bX1XXXXX: Home1 0bXX1XXXX: Snoop 0bXXX1XXX: Non-Data Response 0bXXXX1XX: Data Response 0bXXXXX1X: Non-Coherent Standard 0bXXXXXX1: Non-Coherent Bypass
INQ_DEALLOC	8	RW_RW	0x0	De-allocation from Input Queue
PKT_VNSEL	7:6	RW_RW	0x0	New Packet VN Select: Anded with result of New Packet Class Bit Mask. 11: VNA | VN1 | VN0 10: VNA 01: VN1 00: VN0
PKT_MSGSEL	5:0	RW_RW	0x0	New Packet Class Bit Mask: Bit mask to select which packet types to count. Anded with New Packet VN Select. b1XXXXX: Home bX1XXXX: Snoop bXX1XXX: Non-Data Response bXXX1XX: Data Response bXXXX1X: Non-Coherent Standard bXXXXX1: Non-Coherent Bypass

5.6.3.5 Rbox ARB Performance Monitoring Control Registers

The following table contains the events that can be monitored if one of the QLX (ARB) registers was chosen to select the event.

Table 5-48. R_CSR_ARB_PERF_CTR{4-0} Register Fields (Sheet 1 of 2)

Field	Bits	Access	HW Reset Val	Description
ig	31:24	RO_NA		Read zero; writes ignored.
ev1_sub	23	RW_RW	0x0	Performance Event 1 Sub-Class Select: 0: VN0 1: VN1
ev1_cls	22:20	RW_RW	0x0	Performance Event 1 Class Select: 000: HOM 001: SNP 010: NDR 011: NCS 100: DRS 101: NCB 110: VNA - Large (9,10,11 flits) == data packets 111: VNA - Small (1,2,3 flits) == non-data packets
ev1_type	19:16	RW_RW	0x0	Performance Event 1 Type Select: 0000: Queue Arb Bid 0001: Queue Arb Fail 0010: Queue Home Order Kill 0011: Reserved 0100: Local Arb Bid 0101: Local Arb Fail 0110: Local Home Order Kill 0111: Reserved 1000: Global Arb Bid 1001: Global Arb Fail 1010: Global Home Order Kill 1011: Target Available 1100: Entry Table Depth Overflow 1101-1111: Reserved
ig	15:8	RO_NA		Read zero; writes ignored.

Table 5-48. R_CSR_ARB_PERF_CTR{4-0} Register Fields (Sheet 2 of 2)

Field	Bits	Access	HW Reset Val	Description
ev0_sub	7	RW_RW	0x0	Performance Event 0 Sub-Class Select: 0: VN0 1: VN1
ev0_cls	6:4	RW_RW	0x0	Performance Event 0 Class Select: 000: HOM 001: SNP 010: NDR 011: NCS 100: DRS 101: NCB 110: VNA - Large (9,10,11 flits) == data packets 111: VNA - Small (1,2,3 flits) == non-data packets
ev0_type	3:0	RW_RW	0x0	Performance Event 0 Type Select: 0000: Queue Arb Bid 0001: Queue Arb Fail 0010: Queue Home Order Kill 0011: Reserved 0100: Local Arb Bid 0101: Local Arb Fail 0110: Local Home Order Kill 0111: Reserved 1000: Global Arb Bid 1001: Global Arb Fail 1010: Global Home Order Kill 1011: Target Available 1100: Entry Table Depth Overflow 1101-1111: Reserved

5.6.3.6 Rbox Registers for Mask/Match Facility

In addition to generic event counting, each port of the Rbox provides two pairs of MATCH/MASK registers pair that allow a user to filter packet traffic serviced (crossing from an input port to an output port) by the Rbox according to the packet Opcode, Message Class, Response, HNID and Physical Address. Program the selected Rbox counter's IPERF register to capture matched flits as events.

To use the match/mask facility :

a) Set the MM_CFG (see Table 5-49, "R_CSR_PORT{4-0}_MM_CFG_{1-0} Registers") *.mm_en* (bit 21) to 1.

NOTE: In order to monitor packet traffic, instead of the flit traffic associated with each packet, set *.match_flt_cnt* to 0x1.

b) Program the match/mask regs (see Table 5-50, "R_CSR_PORT{4-0}_MATCH_{1-0}_MSB Registers"/Table 5-51, "R_CSR_PORT{4-0}_MATCH_{1-0}_LSB Registers" and Table 5-52, "R_CSR_PORT{4-0}_MASK_{1-0}_MSB Registers"/Table 5-53, "R_CSR_PORT{4-0}_MASK_{1-0}_LSB Registers").

c) Set the counter's control register event select to the appropriate IPERF subcontrol register and set the IPERF register's event select (0x5 for MM1 and 0x4 for MM0) to capture the mask/match as a performance event.

Table 5-49. R_CSR_PORT{4-0}_MM_CFG_{1-0} Registers

Field	Bits	HW Reset Val	Description
ig	31:22	0x0	Read zero; writes ignored. (?)
mm_en	21	0x0	Match/Mask enable Set to 1 to enable mask/match
ig_flt_cnt	20	0x0	Ignore flit count (Mask any flit). Set to to ignore match_flt_cnt field
match_start	19:16	0x0	Match flit count relative to start of new packet Set number of flit count in a packet on which to trigger a match event. Ex: Set to '0000' to match on first flit in header.
match_71_64	15:8	0x0	upper 8 bits [71:64] of match data
mask_71_64	7:0	0x0	upper 8 bits [71:64] of mask data

The following tables contain the packet traffic that can be monitored if one of the mask/match registers was chosen to select the event.

Table 5-50. R_CSR_PORT{4-0}_MATCH_{1-0}_MSB Registers

Field	Bits	HW Reset Val	Description
rsv	31:20	0x0	Reserved; Must write to 0 else behavior is undefined.
RDS	19:16	0x0	Response Data State (valid when MC == DRS and Opcode == 0x0-2). Bit settings are mutually exclusive. b1000 - Modified b0100 - Exclusive b0010 - Shared b0001 - Forwarding b0000 - Invalid (Non-Coherent)
rsv	15:3	0x0	Reserved; Must write to 0 else behavior is undefined.
RNID_4_2	2:0	0x0	Remote Node ID[4:2]

Table 5-51. R_CSR_PORT{4-0}_MATCH_{1-0}_LSB Registers

Field	Bits	HW Reset Val	Description
RNID_1_0	32:31	0x0	Remote Node ID[1:0]
rsv	30:18	0x0	Reserved; Must write to 0 else behavior is undefined.
DNID	17:13	0x0	Destination Node ID
MC	12:9	0x0	Message Class b0000 HOM - Requests b0001 HOM - Responses b0010 NDR b0011 SNP b0100 NCS --- b1100 NCB --- b1110 DRS
OPC	8:5	0x0	Opcode DRS,NCB: [8] Packet Size, 0 == 9 flits, 1 == 11 flits NCS: [8] Packet Size, 0 == 1 or 2 flits, 1 == 3 flits See Section 5.10, “Packet Matching Reference” for a listing of opcodes that may be filtered per message class.
VNW	4:3	0x0	Virtual Network b00 - VN0 b01 - VN1 b1x - VNA
rsv	2:0	0x0	Reserved; Must write to 0 else behavior is undefined.

Table 5-52. R_CSR_PORT{4-0}_MASK_{1-0}_MSB Registers

Field	Bits	HW Reset Val	Description
rsv	63:52	0x0	Reserved; Must write to 0 else behavior is undefined.
RDS	51:48	0x0	Response Data State (for certain DRS messages)
rsv	47:36	0x0	Reserved; Must write to 0 else behavior is undefined.
RNID	35:33	0x0	Remote Node ID[4:2]

Table 5-53. R_CSR_PORT{4-0}_MASK_{1-0}_LSB Registers

Field	Bits	HW Reset Val	Description
RNID_1_0	32:31	0x0	Remote Node ID[1:0]
rsv	30:18	0x0	Reserved; Must write to 0 else behavior is undefined.
DNID	17:13	0x0	Destination Node ID
MC	12:9	0x0	Message Class
OPC	8:5	0x0	Opcode See Section 5.10, "Packet Matching Reference" for a listing of opcodes that may be filtered per message class.
VNW	4:3	0x0	Virtual Network
rsv	2:0	0x0	Reserved; Must write to 0 else behavior is undefined.

Following is a selection of common events that may be derived by using the Rbox packet matching facility.

Table 5-54. Message Events Derived from the Match/Mask filters (Sheet 1 of 2)

Field	Match [15:0]	Mask [15:0]	Description
HOM.AnyReq	0x0000	0x1E00	Any HOM request message. A HOM request message is 1 flit. There are no HOM messages sent to a Sbox.
HOM.AnyResp	0x0200	0x1E00	Any HOM response message. A HOM response message is 1 flit. There are no HOM messages sent to a Sbox.
SNP.AnySnp	0x0600	0x1E00	Any Snoop message. A Snoop message is 1 flit. There are no snoop messages sent to a Bbox.
DRS.AnyDataC	0x1C00	0x1F80	Any Data Response message containing a cache line in response to a core request. The AnyDataC messages are only sent to an Sbox. The metric DRS.AnyResp - DRS.AnyDataC will compute the number of DRS writeback and non snoop write messages.
DRS.DataC_M	0x1C00 && Match [51:48] 0x8	0x1FE0 && Mask [51:48] 0xF	Data Response message of a cache line in M state that is response to a core request. The DRS.DataC_M messages are only sent to Sboxes.
DRS.WbIData	0x1C80	0x1FE0	Data Response message for Write Back data where cachline is set to the I state.
DRS.WbSData	0x1CA0	0x1FE0	Data Response message for Write Back data where cachline is set to the S state.
DRS.WbEData	0x1CC0	0x1FE0	Data Response message for Write Back data where cachline is set to the E state.

Table 5-54. Message Events Derived from the Match/Mask filters (Sheet 2 of 2)

Field	Match [15:0]	Mask [15:0]	Description
DRS.AnyResp	0x1C00	0x1E00	Any Data Response message. A DRS message can be either 9 flits for a full cache line or 11 flits for partial data.
DRS.AnyResp9flits	0x1C00	0x1F00	Any Data Response message that is 11 flits in length. An 11 flit DRS message contains partial data. Each 8 byte chunk contains an enable field that specifies if the data is valid.
DRS.AnyResp11flits	0x1D00	0x1F00	Any Non Data Response completion message. A NDR message is 1 on flit.
NDR.AnyCmp	0x0400	0x1E00	Non-Coherent Standard read messages. The Non-Coherent read messages is the only NCS message that is 1 flit in length.
NCS.NcRd	0x0800	0x1FE0	Any HOM request message. A HOM request message is 1 flit. There are no HOM messages sent to a Sbox.
NCS.AnyMsg1or2flits	0x0800	0x1F00	Any Non-Coherent Standard message that is 1 or 2 flits in length.
NCS.AnyMsg3flits	0x0900	0x1F00	Any Non-Coherent Standard message that is 3 flits in length.
NCB.AnyMsg9flits	0x1800	0x1F00	Any Non-Coherent Bypass message that is 9 flits in length. A 9 flit NCB message contains a full 64 byte cache line.
NCB.AnyMsg11flits	0x1900	0x1F00	Any Non-Coherent Bypass message that is 11 flits in length. An 11 flit NCB message contains either partial data or an interrupt. For NCB 11 flit data messages, each 8 byte chunk contains an enable field that specifies if the data is valid.
NCB.AnyInt	0x1900	0x1F80	Any Non-Coherent Bypass interrupt message. NCB interrupt messages are 11 flits in length.

Note: Bits 71:16 of the match/mask must be 0 in order to derive these events (except where noted - see DRS.DataC_M). Also the match/mask configuration register should be set to 0x00210000 (bits 21 and 16 set).

5.6.4 Rbox Performance Monitoring Events

5.6.4.1 An Overview:

The Rbox events provide information on topics such as: a breakdown of traffic as it flows through each of the Rbox's ports (NEW_PACKETS_RECV) , raw flit traffic (that is, FLITS_REC_NON_SPEC or FLITS_SENT), incoming transactions entered into arbitration for outgoing ports (ALLOC_TO_ARB), transactions that fail arbitration (GLOBAL_ARB_BID_FAIL), tracking status of various queues (OUTPUTQ_NE), and so forth.

In addition, the Rbox provides the ability to match/mask against ALL flit traffic that leaves the Rbox. This is particularly useful for calculating link utilization, throughput and packet traffic broken down by opcode and message class.

5.6.5 RBox Events Ordered By Code

Table 5-80 summarizes the directly-measured RBox events.

Table 5-55. Performance Monitor Events for RBox Events

Symbol Name	Event Code	Max Inc/Cyc	Description
RIX (Ring Input Port) Events	**IPERF**		
NEW_PACKETS_RECV	[5:0]0xX	1	New Packets Received by Port
INQUE_READ_WIN	[8]0x1	1	Input Queue Read Win
ALLOC_TO_ARB	[15:9]0xX	1	Transactions allocated to Arb
EOT_NE_CYCLES	[16]0x1	1	Cycles EOT Not Empty
EOT_INSERTS	[20]0x1	1	Number of Inserts into EOT
EOT_OCCUPANCY	[21]0x1	1	EOT Occupancy
FAST_ENOUGH	[22]0x1	1	Fast Packets
PCKT_ERR_IN	[23]0x1	1	Incoming Packet Errors
RETRYQ_READS	[24]0x1	1	Retry Buffer Reads
OUTPUTQ_NE	[25]0x1	1	Cycles Output Queue Not Empty
OUTPUTQ_IN	[26]0x1	1	Output Queue Inserts
OUTPUTQ_ACC	[27]0x1	1	Output Queue Overflow Accumulator ***Note:*** Multiply by 128 for correct value.
FLITS_RCVD_NSP	[28]0x1	1	Non-Special Flits Received
FLITS_SENT_NSP	[29]0x1	1	Non-Special Flits Sent
FLITS_RCVD	[30]0x1	1	Flits Received
FLITS_SENT	[31]0x1	1	Flits Sent
QLX (Arbitration Queue) Events	**QLX[3:0]**		
QUE_ARB_BID	0x0	1	Queue ARB Bids
QUE_ARB_BID_FAIL	0x1	1	Failed Queue ARB Bids
QUE_HOME_ORDER_KILL	0x2	1	Queue Home Order Kills
LOCAL_ARB_BID	0x4	1	Local ARB Bids
LOCAL_ARB_BID_FAIL	0x5	1	Failed Local ARB Bids
LOCAL_HOME_ORDER_KILL	0x6	1	Local Home Order Kills
GLOBAL_ARB_BID	0x8	1	Global ARB Bids
GLOBAL_ARB_BID_FAIL	0x9	1	Failed Global ARB Bids
GLOBAL_HOME_ORDER_KILL	0xA	1	Global Home Order Kills
TARGET_AVAILABLE	0xB	1	Target Available
ET_ACC	0xB	1	Accumulated Depth of ET ***Note:*** Multiply by 32 for correct value.

5.6.6 Rbox Performance Monitor Event List

This section enumerates Intel® Itanium® processor 9500 series uncore performance monitoring events for the Rbox.

ALLOC_TO_ARB

- **Title:** Transactions allocated to Arb
- **Category:** RIX
- **[Bit(s)] Value:** See Note, **Max. Inc/Cyc:** 1,
- **Definition:** Transactions entered into Entry Table; This also means they are now available.
- **NOTE:** Program IPERF bits [15:9] to represent the Message Class(es) to be measured

. Message Class:

0b1XXXXXX: Home

0bX1XXXXX: Home1

0bXX1XXXX: Snoop

0bXXX1XXX: Non-Data Response

0bXXXX1XX: Data Response

0bXXXXX1X: Non-Coherent Standard

0bXXXXXX1: Non-Coherent Bypass

EOT_OCCUPANCY

- **Title:** EOT Occupancy
- **Category:** RIX
- **[Bit(s)] Value:** [21]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Accumulated depth Entry Overflow Table which tracks packets that overflowed the Entry Table (*see NOTE for clarification).
- **NOTE:** This basically counts VNA. This event collects the overflow of a subcounter that accurately tracks EOT depth. Multiply this event by 32 to determine the correct EOT depth count. Unlike other packets, ALL HOM packets go through EOT only to stay ordered. HOM0 corresponds to VN0/VNA0 packets while HOM1 corresponds to VN1/VNA1. They are independent meaning HOM0 only need to stay ordered within HOM0 packets and same for HOM1.

Table 5-56. Unit Masks for EOT_DEPTH_ACC

Extension	IPERF Bit Values [19:17]	Description
---	b000	(*nothing will be counted*)
HM0	b001	Home0 Messages
HM1	b010	Home1 Messages
SNP	b011	Snoop Messages
NDR	b100	Non-Data Response Messages
DRS	b101	Data Response Messages
NCS	b110	Non-Coherent Standard Messages
NCB	b111	Non-Coherent Bypass Messages

EOT_INSERTS

- **Title:** Number of Inserts Into EOT
- **Category:** RIX
- **[Bit(s)] Value:** [20]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Used with MC field. Accumulated depth of packets captured in the Entry Overflow Table for specified message types (i.e. EOT count by MC)
- **NOTE:** This basically counts VNA.

Table 5-57. Unit Masks for EOT_INSERTS

Extension	IPERF Bit Values [19:17]	Description
---	b000	(*nothing will be counted*)
HM0	b001	Home0 Messages
HM1	b010	Home1 Messages
SNP	b011	Snoop Messages
NDR	b100	Non-Data Response Messages
DRS	b101	Data Response Messages
NCS	b110	Non-Coherent Standard Messages
NCB	b111	Non-Coherent Bypass Messages

EOT_NE_CYCLES

- **Title:** Cycles EOT Not Empty
- **Category:** RIX
- **[Bit(s)] Value:** [16]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles the Entry Overflow Table buffer is not empty.
- **NOTE:** Signals only if EOTs for ALL message classes are empty

ET_ACC

- **Title:** Accumulated Depth Of ET
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0xb, **Max. Inc/Cyc:** 1,
- **Definition:** Accumulated Depth of Entry Table for specified message types.
- **NOTE:** This event collects the overflow of a subcounter that accurately tracks ET depth. Multiply this event by 32 to determine the correct ET depth count.

Table 5-58. Unit Masks for ET_DEPTH_ACC

Extension	ARB_PERF Bit Values [6:4]	Description
HOM	b000	Home Messages
SNP	b001	Snoop Messages
NDR	b010	Non-Data Response Messages
NCS	b011	Non-Coherent Standard Messages
DRS	b100	Data Response Messages
NCB	b101	Non-Coherent Bypass Messages
---	b110-b111	(*illegal selection*)

FAST_ENOUGH

- **Title:** Fast Packets
- **Category:** RIX
- **[Bit(s)] Value:** [22]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** How often a multi-flit packet uses the optimized path.

FLITS_RCVD

- **Title:** Flits Received
- **Category:** RIX
- **[Bit(s)] Value:** [30]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Counts all flits received.

FLITS_SENT

- **Title:** Flits Sent
- **Category:** RIX
- **[Bit(s)] Value:** [31]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Counts all flits

FLITS_NSP_RCVD

- **Title:** Non-Special Flits Received
- **Category:** RIX
- **[Bit(s)] Value:** [28]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Counts all non-special flits received.

GLOBAL_ARB_BID

- **Title:** Global ARB Bids
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x8, **Max. Inc/Cyc:** 1,
- **Definition:** Count global arbitration bids from the port. Occurs on output port.

GLOBAL_ARB_BID_FAIL

- **Title:** Failed Global ARB Bids
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x9, **Max. Inc/Cyc:** 1,
- **Definition:** Count failed global arbitration bids from the port. Occurs on output port.
- **NOTE:** Multi-flit packets will only receive a single fail signal.

GLOBAL_HOME_ORDER_KILL

- **Title:** Global Home Order Kills
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0xa, **Max. Inc/Cyc:** 1,
- **Definition:** Arbitration attempts killed in the HOM channel of the global ARB that violate ordering. HOM packets may be issued back to back through the ARBs, but if the first HOM was not chosen by the ARB, the second one will be killed due to ordering rules. This event accounts for how often this situation occurs in the global ARB.

INQUE_READ_WIN

- **Title:** Input Queue Read Win.
- **Category:** RIX
- **[Bit(s)] Value:** [8]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Bid wins arbitration. Counts number of IQA reads and drains to XBAR.

LOCAL_ARB_BID

- **Title:** Local ARB Bids
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x4, **Max. Inc/Cyc:** 1,
- **Definition:** Number of bids to exit port. Occurs on input port.

LOCAL_ARB_BID_FAIL

- **Title:** Failed Local ARB Bids
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x5, **Max. Inc/Cyc:** 1,
- **Definition:** Number of bids to exit port that failed. Occurs on input port.

LOCAL_HOME_ORDER_KILL

- **Title:** Local Home Order Kills
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x6, **Max. Inc/Cyc:** 1,
- **Definition:** Arbitration attempts killed in the HOM channel of the local ARB that violate ordering. HOM packets may be issued back to back through the ARBs, but if the first HOM was not chosen by the ARB, the second one will be killed due to ordering rules. This event accounts for often this situation occurs in the local ARB.

NEW_PACKETS_RECV

- **Title:** New Packets Received by Port
- **Category:** RIX
- **[Bit(s)] Value:** see table, **Max. Inc/Cyc:** 1,
- **Definition:** Counts new packets received according to the Virtual Network and Message Class specified.
- **NOTE:** Program IPERF bits [5:0] to represent the Message Class(es) to be measured. Each packet gets allocated to the ARB exactly once.

Message Class:

0b1XXXXX: Snoop

0bX1XXXX: Home

0bXX1XXX: Non-Data Response

0bXXX1XX: Data Response

0bXXXX1X: Non-Coherent Standard

0bXXXXX1: Non-Coherent Bypass

Table 5-59. Unit Masks for NEW_PACKETS_RECV (Sheet 1 of 2)

Extension	IPERF Bit Values [7:6]	Description
VN0	b00	VN0
VN1	b01	VN1

Table 5-59. Unit Masks for NEW_PACKETS_RECV (Sheet 2 of 2)

Extension	IPERF Bit Values [7:6]	Description
VNA	b10	VNA
ANY	b11	VNA | VN1 | VN0

OUTPUTQ_OVFL

- **Title:** Output Queue Overflow Accumulator
- **Category:** RIX
- **[Bit(s)] Value:** [27]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Running count of number of times the Output Buffer overflows.
- **NOTE:** For a full-width port the output buffer is always by-passed. This event collects the overflow of a subcounter that accurately tracks the output buffer depth. Multiply this event by 128 to determine the correct output buffer depth count.

OUTPUTQ_IN

- **Title:** Output Queue Inserts
- **Category:** RIX
- **[Bit(s)] Value:** [26]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Number of flits inserted to the Output Buffer
- **NOTE:** For a full-width port the output buffer is always by-passed

OUTPUTQ_NE

- **Title:** Cycles Output Queue Not Empty
- **Category:** RIX
- **[Bit(s)] Value:** [25]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles the Output Buffer is not empty.
- **NOTE:** For a full-width port the output buffer is always by-passed

PCKT_ERR_IN

- **Title:** Incoming Packet Errors
- **Category:** RIX
- **[Bit(s)] Value:** [23]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Incoming Packet Error (#times input port requested retry from the sender?)

QUE_ARB_BID

- **Title:** Queue ARB Bids
- **Category:** • **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x0, **Max. Inc/Cyc:** 1,
- **Definition:** Number of queue ARB bids. Each message class has its own queue. Occurs on input port.

Table 5-60. Unit Masks for QUE_ARB_BID

Extension	ARB_PERF Bit Values [6:4]	Description
HOM	b000	Home Messages
SNP	b001	Snoop Messages
NDR	b010	Non-Data Response Messages
NCS	b011	Non-Coherent Standard Messages
DRS	b100	Data Response Messages
NCB	b101	Non-Coherent Bypass Messages
---	b110-b111	(*illegal selection*)

QUE_ARB_BID_FAIL

- **Title:** Failed Queue ARB Bids
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Number of failed queue ARB bids. Each message class has its own queue. Occurs on input port. Home order kill only way to receive failed queue arb bid.

Table 5-61. Unit Masks for QUE_ARB_BID_FAIL

Extension	ARB_PERF Bit Values [6:4]	Description
HOM	b000	Home Messages
SNP	b001	Snoop Messages
NDR	b010	Non-Data Response Messages
NCS	b011	Non-Coherent Standard Messages
DRS	b100	Data Response Messages
NCB	b101	Non-Coherent Bypass Messages
---	b110-b111	(*illegal selection*)

QUE_HOME_ORDER_KILL

- **Title:** Queue Home Order Kills
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0x2, **Max. Inc/Cyc:** 1,
- **Definition:** Arbitration attempts killed in the HOM channel of the queue ARB that violate ordering. HOM packets may be issued back to back through the ARBs, but if the first HOM was not chosen by the ARB, the second one will be killed due to ordering rules. This event accounts for often this situation occurs in the queue ARB.

RETRY_BUF_READS

- **Title:** Retry Buffer Reads
- **Category:** RIX
- **[Bit(s)] Value:** [24]0x1, **Max. Inc/Cyc:** 1,
- **Definition:** Number of flits sent from the retry buffer.

TARGET_AVAILABLE

- **Title:** Target Available
- **Category:** QLX
- **[Bit(s)] Value:** [3:0]0xb, **Max. Inc/Cyc:** 1,
- **Definition:** There are enough credits on the target port to allow a bid. The target is unavailable if the output buffer is full or there are no credits. There are target available bits for each VN0/VN1 per message class as well as one for VNA-small and V

Table 5-62. Unit Masks for TARGET_AVAILABLE

Extension	ARB_PERF Bit Values [7:4]	Description
VN0.HOM	b0000	VN0 Home Messages
VN0.SNP	b0001	VN0 Snoop Messages
VN0.NDR	b0010	VN0 Non-Data Response Messages
VN0.NCS	b0011	VN0 Non-Coherent Standard Messages
VN0.DRS	b0100	VN0 Data Response Messages
VN0.NCB	b0101	VN0 Non-Coherent Bypass Messages
VN0.VSM	b0110	VN0 VNA-small (<= 3 flits) Messages
VN0.VLG	b0111	VN0 VNA-large (9-11 flits) Messages
VN1.HOM	b1000	VN1 Home Messages
VN1.SNP	b1001	VN1Snoop Messages
VN1.NDR	b1010	VN1Non-Data Response Messages
VN1.NCS	b1011	VN1Non-Coherent Standard Messages
VN1.DRS	b1100	VN1Data Response Messages
VN1.NCB	b1101	VN1Non-Coherent Bypass Messages
VN1.VSM	b1110	VN1 VNA-small (<= 3 flits) Messages
VN1.VLG	b1111	VN1 VNA-large (9-11 flits) Messages

5.7 Sbox Performance Monitoring

5.7.1 Overview of the Sbox

The Sbox represents the interface between the last level cache and the system interface. It manages flow control between the Cboxes (forming the ring) and the System (forwarded through the router - the Rbox). The Sbox is broken into system bound (ring to Intel QPI) and ring bound (Intel QPI to ring) connections.

As such, it shares responsibility with the Cbox(es) as the Intel QPI caching agent(s). It is responsible for converting Cbox requests to Intel QPI messages (that is, snoop generation and data response messages from the snoop response) as well as converting/forwarding ring messages to Intel QPI packets and vice versa.

5.7.2 Sbox Performance Monitoring Overview

Each Sbox in the Itanium processor 9500 series supports event monitoring through 4 48b wide counters (S_CSR_PMON_CTR/CTL{3:0}). Each of these four counters can be programmed to count any Sbox event. the Sbox counters can increment by a maximum of 64 per cycle.

The count values of all 4 counters can be cleared by writing the S_CSR_PMON_PERF_MASTER.clr bit.

The Sbox also includes a mask/match register that allows a user to match packets leaving the Sbox according to various standard packet fields such as message class, opcode, etc. (NOTE: specifically goes with event 0, does not effect other events)

For information on how to setup a monitoring session, refer to Section 5.3, "Global Performance Monitoring Control".

5.7.2.1 Sbox PMU - Overflow, Freeze and Unfreeze

Sbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Sbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, "Global Enable/Disable").

Sbox PMU can be frozen due to one of three reasons:

- *Globally:* Ubox sends a disable signal (and S_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, "Global Enable/Disable") or *local* (SW writes 0 to S_CSR_PMON_PERF_MASTER.en when S_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.
- *Locally:* The Sbox was set to local control (S_CSR_PMON_PERF_MASTER.glb_lcl = 0) an Sbox counter overflowed and it's corresponding S_CSR_PMON_CTLx.ovf_en bit was set to 1.

If an overflow is detected from an Sbox performance counter and the overflow has been enabled (S_CSR_PMON_CTLx.ovf_en for counter x has been set to 1), the overflow bit is set at the box level (S_CSR_PMON_GLOBAL_STATUS.ov) and rolled up for global consumption in S_CSR_PMON_SUMMARY.ov_s.

The Ubox may be configured to freeze all uncore counting (refer to Table 5-65, "S_CSR_PMON_FRZ_EN Register Fields") when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field responsible for the freeze, must be cleared. When the overflow bit(s) has been cleared, the Sbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, "Enabling a New Sample Interval from Frozen Counters."), counting will resume.

Note: Due to the nature of the subcounters used in the Sbox, if a queue occupancy count event is set up to be captured, SW should set .reset_occ_cnt in the same write that the corresponding control register is enabled.

5.7.3 Sbox Performance Monitors

Table 5-63. Sbox Performance Monitoring CSRs (Sheet 1 of 2)

CSR Name	CSR Address	Size (bits)	Description
Intel QPI EAR			
S_CSR_QEAR_CTL	0x8F8	32	QPI EAR Control
S_CSR_QEAR_DAT1	0x8F0	32	QPI EAR Data 1
S_CSR_QEAR_DAT0	0x8E8	32	QPI EAR Data 0

Table 5-63. Sbox Performance Monitoring CSRs (Sheet 2 of 2)

CSR Name	CSR Address	Size (bits)	Description
Global Control/Status			
S_CSR_PMON_SUMMARY	0x8E0	32	Sbox PMON Global Summary Records overflows received from boxes associated with this Sbox.
S_CSR_PMON_FRZ_EN	0x8D8	32	Sbox PMON Global Freeze Enable Determines what to do with overflows received from boxes associated with this Sbox.
Box-Level Control/Status			
S_CSR_PMON_GLOBAL_STATUS	0x8D0	32	Sbox PMON Global Overflow Status
S_CSR_PMON_PERF_MASTER	0x8D8	32	Sbox PMON Global Freeze Enable
Generic Counter/Control			
S_CSR_PMON_CTR3	0x8B0	64	Sbox PMON Counter 3
S_CSR_PMON_CTL3	0x8A8	64	Sbox PMON Control 3
S_CSR_PMON_CTR2	0x8A0	64	Sbox PMON Counter 2
S_CSR_PMON_CTL2	0x898	64	Sbox PMON Control 2
S_CSR_PMON_CTR1	0x890	64	Sbox PMON Counter 1
S_CSR_PMON_CTL1	0x888	64	Sbox PMON Control 1
S_CSR_PMON_CTR0	0x880	64	Sbox PMON Counter 0
S_CSR_PMON_CTL0	0x878	64	Sbox PMON Control 0
Event Filtering Control			
S_CSR_MATCH2	0x858	64	Sbox Match2 Register
S_CSR_MASK	0x850	64	Sbox Mask Register
S_CSR_MATCH	0x848	64	Sbox Match Register
S_CSR_MM_CFG	0x840	64	Sbox Enable Match/Mask Config

5.7.3.1 Sbox PMON for Global State

The S_CSR_PMON_SUMMARY register in each Sbox collects overflow bits from cores and uncore boxes directly associated with that Sbox such that, apart from the Wbox which is exclusively managed by Sbox 1, each Sbox manages half of the overflow bits.

And S_CSR_PMON_FRZ_EN allows a user to choose whether the overflow should trigger a global freeze of all uncore PMU state.

When a global uncore disable signal is sent out to the uncore boxes, it is also sent to each core. Each core PMU capture it as an event (UNCORE_FREEZE - Event Code 0xC9 in the core counters).

Note: If a user wishes to isolate a box from global control, be sure to set the corresponding bit in S_CSR_PMON_FRZ_EN to 0.

Table 5-64. S_CSR_PMON_SUMMARY Register Fields

Field	Bits	Type	HW Reset Val	Description
ig	63:17	RO_NA	0	Read zero; writes ignored.
ov_t1_c3_4	16	RO_WO	0	Overflow in Thread 1 of Core 3 (S0) or 4 (S1)
ov_t0_c3_4	15	RO_WO	0	Overflow in Thread 0 of Core 3 (S0) or 4 (S1)
ov_t1_c2_5	14	RO_WO	0	Overflow in Thread 1 of Core 2 (S0) or 5 (S1)
ov_t0_c2_5	13	RO_WO	0	Overflow in Thread 0 of Core 2 (S0) or 5 (S1)
ov_t1_c1_6	12	RO_WO	0	Overflow in Thread 1 of Core 1 (S0) or 6 (S1)
ov_t0_c1_6	11	RO_WO	0	Overflow in Thread 0 of Core 1 (S0) or 6 (S1)
ov_t1_c0_7	10	RO_WO	0	Overflow in Thread 1 of Core 0 (S0) or 7(S1)
ov_t0_c0_7	9	RO_WO	0	Overflow in Thread 0 of Core 0 (S0) or 7(S1)
ov_c3_4	8	RO_WO	0	Overflow in Cbox 3 (S0) or 4 (S1)
ov_c2_5	7	RO_WO	0	Overflow in Cbox 2 (S0) or 5 (S1)
ov_c1_6	6	RO_WO	0	Overflow in Cbox 1 (S0) or 6 (S1)
ov_c0_7	5	RO_WO	0	Overflow in Cbox 0 (S0) or 7 (S1)
ov_w	4	RO_WO	0	Overflow in Wbox (S1 only)
ov_z	3	RO_WO	0	Overflow in Zbox 0 (S0) or 1 (S1)
ov_b	2	RO_WO	0	Overflow in Bbox 0 (S0) or 1 (S1)
ov_r	1	RO_WO	0	Overflow in attached half of Rbox - Counters 0-7 in S0, Counters 8-15 in S1
ov_s	0	RO_WO	0	Overflow in this Sbox

Table 5-65. S_CSR_PMON_FRZ_EN Register Fields (Sheet 1 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	63:17	RO_NA	0	Read zero; writes ignored.
msk_t1_c3_4	16	RW_RO	0	Mask Overflow in Thread 1 of Core 3 (S0) or 4 (S1)
msk_t0_c3_4	15	RW_RO	0	Mask Overflow in Thread 0 of Core 3 (S0) or 4 (S1)
msk_t1_c2_5	14	RW_RO	0	Mask Overflow in Thread 1 of Core 2 (S0) or 5 (S1)
msk_t0_c2_5	13	RW_RO	0	Mask Overflow in Thread 0 of Core 2 (S0) or 5 (S1)
msk_t1_c1_6	12	RW_RO	0	Mask Overflow in Thread 1 of Core 1 (S0) or 6 (S1)
msk_t0_c1_6	11	RW_RO	0	Mask Overflow in Thread 0 of Core 1 (S0) or 6 (S1)
msk_t1_c0_7	10	RW_RO	0	Mask Overflow in Thread 1 of Core 0 (S0) or 7(S1)
msk_t0_c0_7	9	RW_RO	0	Mask Overflow in Thread 0 of Core 0 (S0) or 7(S1)
msk_c3_4	8	RW_RO	0	Mask Overflow in Cbox 3 (S0) or 4 (S1)
msk_c2_5	7	RW_RO	0	Mask Overflow in Cbox 2 (S0) or 5 (S1)
msk_c1_6	6	RW_RO	0	Mask Overflow in Cbox 1 (S0) or 6 (S1)
msk_c0_7	5	RW_RO	0	Mask Overflow in Cbox 0 (S0) or 7 (S1)
msk_w	4	RW_RO	0	Mask Overflow in Wbox (S1 only)
msk_z	3	RW_RO	0	Mask Overflow in Zbox 0 (S0) or 1 (S1)
msk_b	2	RW_RO	0	Mask Overflow in Bbox 0 (S0) or 1 (S1)
msk_r	1	RW_RO	0	Mask Overflow in attached half of Rbox - Counters 0-7 in S0, Counters 8-15 in S1

Table 5-65. S_CSR_PMON_FRZ_EN Register Fields (Sheet 2 of 2)

Field	Bits	Type	HW Reset Val	Description
msk_s	0	RW_RO	0	Mask Overflow in this Sbox

5.7.3.2 Sbox Box Level PMON state

The following registers represent the state governing all box-level PMUs in the Sbox.

S_CSR_PMON_PERF_MASTER controls the general characteristics of the Sbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

If an overflow is detected from one of the Sbox PMON registers, the corresponding bit in the _GLOBAL_STATUS.ov field will be set.

Table 5-66. S_CSR_PMON_PERF_MASTER Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	31:3	RO_NA	0	Read zero; writes ignored. (?)
clr	2	RW_RW	0	Writing 1 clears all the Sbox counters as well as overflow bits in S_CSR_PMON_GLOBAL_STATUS.
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: **Global**: Enable/Disable of counters in Sbox will track U_CSR_PERF_CTL.glb_en. Local overflows will be passed on to Ubox without freezing local counters. 0: **Local**: Enable/Disable of counters in Sbox will NOT track U_CSR_PERF_CTL.glb_en. Allows SW to write the .en bit. Disables Counters on any local counter overflow.
en	0	RW_RW	0	Enable/disable Sbox PMU counters. This bit is dependent on the setting of the .glb_lcl bit. If .glb_lcl is set to 1, SW writes to this bit are ignored and only HW may affect it's state. If .glb_lcl is set to 0, SW may exert control by setting the bit. In either case, since HW may alter this bit, (due to tracking the global enable or a local overflow) SW may read it to determine the state of the Sbox counters. 1: Enable Sbox PMU counting. 0: Disable (freeze) Sbox PMU counters.

Table 5-67. S_CSR_PMON_GLOBAL_STATUS Register Fields

Field	Bits	Type	HW Reset Val	Description
ig	63:4	RO_NA	0	Read zero; writes ignored. (?)
ov	3:0	RW_RO	0	If an overflow is detected from the corresponding SBOX PMON register and the overflow has been enabled (S_CSR_PMON_CTLx.ovf_en for counter x is set to 1), its overflow bit will be set.

5.7.3.3 Sbox PMON state - Counter/Control Pairs + Filters

The following table defines the layout of the Sbox performance monitor control registers. The main task of these configuration registers is to select the event to be monitored by their respective data counter. Setting the *.ev_sel* field performs the event selection. Additional control bits include:

- *.threshold* - If the *.threshold* 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.

- *.invert* - Changes the .threshold test condition to '<'

- *.edge_detect* - 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.

- *.reset_occ_cnt* - Reset 7b occupancy counter associated with this counter.

Table 5-68. S_CSR_PMON_CTL{3-0} Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63	RO_NA	0	Read zero; writes ignored. (?)
rsv	62	RW_RO	0	Reserved; Must write to 0 else behavior is undefined.
ig	61:32	RO_NA	0	Read zero; writes ignored. (?)
threshold	31:24	RW_RO	0	Threshold used for counter comparison.
invert	23	RW_RO	0	Invert threshold comparison. When '0', the comparison will be thresh >= event. When '1', the comparison will be threshold < event.
ovf_en	22	RW_RO	0	On overflow 0: overflow signal is dropped. 1: foveal signal will be sent out.
ig	21:19	RO_NA	0	Read zero; writes ignored. (?)
edge_detect	18	RW_RO	0	Edge Detect. When bit is set, 0->1 transition of a one bit event input will cause counter to increment. When bit is 0, counter will increment for however long event is asserted.
reset_occ_cnt	17	WO_RO	0	Reset Occupancy Counter associated with this counter.
ig	16	RO_NA	0	Read zero; writes ignored. (?)
umask	15:8	RW_RO	0	Unit Mask - select subevent of event.
ev_sel	7:0	RW_RO	0	Event Select

The Sbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events. In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-69. S_CSR_PMON_CTR{3-0} Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored. (?)
event_count	47:0	RW_RW	0	48-bit performance event counter

5.7.3.4 Sbox Registers for Mask/Match Facility

In addition to generic event counting, the uncore provides a MATCH/MASK registers in each Sbox. These MATCH/MASK registers allow a user to filter outgoing packet traffic (system bound) according to the packet Opcode, Message Class, Response, HNID and Physical Address. Program the selected Sbox counter to capture TO_R_PROG_EV to capture the filter match as an event.

To use the match/mask facility :

a) Set MM_CFG (see Table 5-70, "S_CSR_MM_CFG Register – Field Definitions") reg bit 63 to 0.

b) Program match/mask regs (see Table 5-71, "S_CSR_MATCH Register – Field Definitions" and Table 5-72, "S_CSR_MATCH2 Register – Field Definitions"). (if MM_CFG[63] == 1, a write to match/mask will produce a GP fault).

NOTE: The address and the Home Node ID have a mask component in the MASK register. To mask off other fields (for example, opcode or message class), set the field to all 0s.

c) Set the counter's control register event select to 0x0 (TO_R_PROG_EV) to capture the mask/match as a performance event.

d) Set MM_CFG reg bit 63 to 1 to start matching.

Table 5-70. S_CSR_MM_CFG Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
mm_en	63	RW_RW	0	Enable Match/Mask
ig	62:0	RO_NA	0	Read zero; writes ignored.

Table 5-71. S_CSR_MATCH Register – Field Definitions (Sheet 1 of 2)

Field	Bits	Type	HW Reset Val	Description
ig	63	RO_NA	0	Read zero; writes ignored. (?)
resp	62:59	RW_RO	0	Match if returning data is in b1xxx - 'S' state bx1xx - 'E' state. bxx1x - 'M' state. bxxx1 - 'I' state. ***Note:*** Response matching is only done on the DRS response to a HOM0 RdCode, RdData or RdInvOwn request. ***Note:*** Match will be ignored if field set to all 0s.

Table 5-71. S_CSR_MATCH Register – Field Definitions (Sheet 2 of 2)

Field	Bits	Type	HW Reset Val	Description
mc	58:54	RW_RO	0	Match on Message Class b1xxxx - NCB bx1xxx - NCS bxx1xx - NDR bxxx1x - HOM1 bxxxx1 - HOM0 ***Note:*** Match will be ignored if field set to all 0s.
addr	53:10	RW_RO	0	Match on PA address bits [49:6]
hnid	9:0	RW_RO	0	Match on Home NodeID

Table 5-72. S_CSR_MATCH2 Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	62:11	RO_NA	0	Read zero; writes ignored.
opc	10:0	RW_RO	0	Match on Opcode (see Table 5-73, "S_CSR_MATCH2.opc - Opcode Match by Message Class") ***Note:*** Match will be ignored if field set to all 0s.

Refer to Table 5-112, "Opcodes (Alphabetical Listing)" for definitions of the opcodes found in the following table.

Table 5-73. S_CSR_MATCH2.opc - Opcode Match by Message Class

bit	NCB	NCS	NDR	HOM1	HOM0
10	---	---	---	---	---
9	---	NcIOWr	---	RspFwd	EvctCln
8	DebugData	---	---	RspSWb	InvXtoI
7	WcWrPtl	NcCfgWr	---	RspIWb	AckCnflt
6	NcWrPtl	NcIORd	---	RspFwdSWb	---
5	---	---	---	RspFwdIWb	WbMtoI
4	IntPhysical	NcCfgRd	---	RspFwdS	---
3	PurgeTC	NcRdPtl	---	RspFwdI	InvItoE
2	NcMsgB	---	---	RspCnflt	RdInvOwn
1	WcWr	IntAck	---	RspS	RdData
0	---	NcRd	---	RspI	RdCode

Table 5-74. S_CSR_MASK Register – Field Definitions

Field	Bits	HW Reset Val	Description
ig	62:45		Read zero; writes ignored.
addr	44:1	0	Mask PA address bits [49:6]. For each mask bit that is set, the corresponding bit in the address is already considered matched (e.g. it is ignored). If it is clear the it must match the corresponding address match bit in the S_CSR_MATCH register.
hnid	0	0	Disable HNID matching. 1 - HNID is NOT matched 0 - HNID is compared against the match

5.7.4 QEAR

Table 5-75. QEAR Performance Monitoring CSRs

CSR Name	CSR Address [12:0]	Size (bits)	Description
S_CSR_QEAR_CTL	0x8F8	32	QPI EAR Control
S_CSR_QEAR_DAT1	0x8F0	32	QPI EAR Data 1
S_CSR_QEAR_DAT0	0x8E8	32	QPI EAR Data 0

The Intel® QuickPath Interconnect (Intel QPI) Event Address Register (QEAR) provides functionality similar to the Data EAR (DEAR) in the core. On an inbound request to the Sbox, the following information is captured:

1. Physical Address [49:6].
2. Message Class and Opcode
3. NID of the targeted Home Agent

When the Data for the request is returned for a read, or the Cmp is returned for a write, the following is also captured.

1. The latency in core clock cycles from send to end. The cycle counter is 15 bits allowing measurement of latencies up to 32,768 cycles.
2. An overflow bit indicating that the counter overflowed before Data was returned.
3. For Read requests, the QEAR indicates whether the Data was returned with a DataC*_Cmp. Since only Home agents can return Cmps, this indicates that data was returned from the Home agent, rather than as a cache to cache transfer.

Note: Some OEM node controller Home agents may send DataC* and Cmp separately, so having this bit clear may not imply a cache to cache transfer.

The QEAR does not support the measurement of the latency of the Cmp for read requests. On a completed transaction, the QEAR can signal the QEAR event (0x90) to the Sbox counters and it will also respond to the subsequent Sbox freeze to halt sampling. CSR reads may then be used to access the logged information.

Since the QEAR is located in the Sbox, responsible for bundling memory requests into QPI packets, its latency count will not include the latencies induced by any Core or Caching Agent. It will include the latencies of the Sbox packet building and packet decoding logic (including any latencies waiting for Intel QPI credits), the clock synchronizer from Sbox to the sysint, and all sysint and off-socket resources.

5.7.4.1 QEAR Functionality

Aside from the PMU event (0x90 - QEAR_EVENT) and response to PMU freeze, the QEAR is controlled almost entirely by CSR with no connection to the core Sbox PMU block.

The QEAR provides the following configurability:

- **Count threshold** - The QEAR will only signal a PMU event if the captured count is above 128, 256 or 512 core cycles.
- **LFSR** - The QEAR includes an LFSR to randomly select either 1 of 8 requests. The LFSR can be set to be ignored. The LFSR value is not readable or directly writeable. On a write to CPE_QEAR_CSR_CTL, the LFSR is reset to a fixed value. Then it progresses each cycle.
- **Opcode Match & Mask and Message Class Selects** - Latency measurements can be done on any HOM0, NCS, or NCB opcode that the Sbox supports. Opcode Match & Mask and Message Class selects can be used to restrict the operations that are measured.

One limitation of the Intel QPI EAR logic is that when a request is initially targeted for sampling, the address, message class, opcode and homeNID are automatically logged into the QEAR CSRs and the counter is reset. The latency counts in the CSRs are free running until the transaction completes. Thus if software polls the QEAR CSRs while a measurement is in progress, it will see the address and other information for the measurement in progress, and will see the free running latency and overflow values for the current measurement.

Additionally, if the QEAR is not frozen, reads from the various QEAR registers may correspond to measurements for completely different operations if the QEAR triggers on a new request. To avoid this problem the QEAR must be configured to use the Sbox's overflow mechanism. To do this:

1. Program an Sbox control register (e.g. thru S_CSR_PMON_CTL0) to capture the QEAR event (0x90). The QEAR event will trigger once for every measurement.
2. The associated data register must be set with a value and configuration to signal PMD overflow/freeze after the desired number of measurement events.
3. Sampling software should then read the QEAR CSRs only in response to an Uncore PMD overflow event. In this case, the PMD overflow will prevent the QEAR from starting a new measurement, and thus all of the logged information will correspond to the same measurement.
4. No reprogramming of the QEAR CSRs is needed. To re-enable sampling software only needs to clear the PMD overflow condition.

5.7.4.2 Sequence to Enable QEAR - If Changing LFSRMSK

1) Read S_CSR_QEAR_CTL.lfsrmsk and store value.

If a user wishes to change the value of .lfsrmsk:

2) Disable the QEAR by setting S_CSR_QEAR_CTL.mc to 0.

3) Disable Sbox monitoring (set S_MSR_PMON_PERF_MASTER.en to 0).

4) Set S_CSR_QEAR_CTL.lfsrmsk.

5) Re-Enable Sbox monitoring (set S_MSR_PMON_PERF_MASTER.en to 1)

5) Now set the S_CSR_QEAR_CTL to the desired value for sampling, being sure to maintain the same .lfsrmsk value as set in Step 3.

Else if the .lfsrmsk value will remain the same (i.e. the default value is 0 == enabled and SW intends to maintain this value throughout the monitoring session):

2) Disable the QEAR by setting S_CSR_QEAR_CTL.mc to 0.

3) Set the S_CSR_QEAR_CTL to the desired value for sampling, being sure to maintain the .lfsrmsk value as read from Step 1.

5.7.4.3 Sbox PMON state for QEAR

The following tables define the layout of the Sbox QEAR performance monitor registers.

Table 5-76. QEAR Configuration Register Fields (S_CSR_QEAR_CTL)

Field	Bit Range	Access	HW Reset Val	Description
cnt	31:17	RW_WO	0	Latency count in core clock cycles
rsv	16	RW_NA	0	Reserved; Must write to 0 else behavior is undefined.
thresh	15:14	RW_RO	0	Threshold value of QEAR count to signal a CPE QEAR event to PMU. b00 - Any count value b01 - Count >= 128 b10 - Count >= 256 b11 - Count >= 512
opmsk	13:10	RW_RO	0	Intel QPI Opcode Mask
op	9:6	RW_RO	0	Intel QPI Opcode. A transaction will be measured if its QPI Opccode[3:0] makes the following statement true: ((Opcode[3:0] XNOR op[3:0]) OR opmsk[3:0]) = 0xF. NOTE: Refer to Table 5-112, “Opcodes (Alphabetical Listing)” for more information.
mc	5:3	RW_RO	0	Message Class Select: xx1 - HOM x1x - NCB 1xx - NCS If the msgclass bit is set to 3'b000, then CEAR logic matching is disabled and "opcode[3:0]" contains the opcode to match.
rsv	2	RW_NA	0	Reserved; Must write to 0 else behavior is undefined.
lfsr_msk	1	RW_RO	0	If set to 1, the QEAR lfsr is ignored for triggering a capture. Once the QEAR starts fishing for a new sample to watch, it will capture the first one that comes by. Recommended: Set this bit to 0 to randomize QEAR captures by letting some fish (samples) go.
rsv	0	RW_NA	0	Reserved; Must write to 0 else behavior is undefined.

Table 5-77. QEAR Data Register 0 Fields (CPE_CSR_CEAR_DAT0)

Field	Bit Range	Access	HW Reset Val	Description
pa_lo	31:6	RW_WO	0	Intel QPI Physical Address [31:6]
op	5:2	RW_WO	0	Intel QPI opcode of captured request.
ov	1	RW_WO	0	Indicates that the last QEAR capture overflowed the latency counter.
datacmp	0	RW_WO	0	For measured requests that expect return data, the data packet returned is a DataC*_Cmp. This may help indicate whether the data request was satisfied by a home agent or a caching agent. This will not be set for requests returned with DataC_*_FrcAckCnflct, even though those are sent only from home agents.

Table 5-78. QEAR Data Register 1 Fields (CPE_CSR_CEAR_DAT1)

Field	Bit Range	Access	HW Reset Val	Description
ig	31:30	RO_NA	0	Reads zero; Writes are ignored
hnid	29:20	RW_WO	0	Home NID of request
mc	19:18	RW_WO	0	Message Class: 00 - HOM0 01 - NCB 10 - NCS 11 - [shouldn't happen]
pa_hi	17:0	RW_WO	0	Core Physical Address [49:32]

5.7.5 Sbox Performance Monitoring Events

5.7.5.1 An Overview

The Sbox provides events to track incoming (ring bound)/outgoing (system bound) transactions, various queue occupancies/latencies that track those transactions and a variety of static events such as bypass use (that is, EGRESS_BYPASS) and when output credit is unavailable (for example, NO_CREDIT_HOM). Many of these events can be further broken down by message class.

5.7.5.2 On Queue Occupancy Usage

This means two things:

a) none of the physical queues receive more than one entry per cycle

b) The entire 7b from the 'selected' (by the event select) queue occ subcounter is sent to the generic counter each cycle, meaning that the max inc of a generic counter is 64 (for the sys bound HOM buffer).

Associated with each of the four general purpose counters is a 7b queue occupancy counter which supports the various queue occupancy events found in Section 5.7.6, "Sbox Events Ordered By Code".

Each System Bound and Ring Bound data storage structure within the Sbox (queue/FIFO/buffer) has an associated tally counter which can be used to provide input into one of the Sbox performance counters. The data structure the tally counter is 'attached' to then sends increment/decrement signals as it receives/removes new entries. The tally counter then sends its contents to the performance counter each cycle.

The following table summarizes the queues (and their associated events) responsible for buffering traffic into and out of the Sbox.

Table 5-79. Sbox Data Structure Occupancy Events

Structure / Event Name	Max Entries	Instances	Description / Comment
System Bound (Rbox) HOM Message Queue TO_R_HOM_MSGQ_OCCUPANCY	64	1	HOM Packet to System
System Bound DRS Message Queue TO_R_DRS_MSGQ_OCCUPANCY	4	4	DRS Packet to System 1 buffer per attached Cbox
System Bound NCB Message Queue TO_R_NCB_MSGQ_OCCUPANCY	2	4	NCB Packet to System 1 buffer per attached Cbox
System Bound NCS Message Queue TO_R_NCS_MSGQ_OCCUPANCY	2	4	NCS Packet to System 1 buffer per attached Cbox
Ring Bound SNP Message Queue TO_RING_SNP_MSGQ_OCCUPANCY	31	1	SNP Packet from System
Ring Bound NDR Message Queue TO_RING_NDR_MSGQ_OCCUPANCY	32	1	NDR Packet from System
Ring Bound R2S Message Queue TO_RING_R2S_MSGQ_OCCUPANCY	8	1	Packets headed to Ring
Request Table	48	1	System Bound Request

5.7.5.3 On Packet Transmission Events

For the message classes that have variable length messages, the Sbox has separate events which count the number of flits of those message classes sent or received (that is, PKTS_SENT_HOM vs. PKTS/FLITS_SENT_NCB). For message classes that have fixed length messages, the total number of flits can be calculated by multiplying the total messages by the number of flits per message (that is, PKTS_RCVD_SNP).

Message Class	Flits per Msg (SMP)	Flits per Msg (EMP)	Comment
HOM	1	2	
Ring Bound DRS	9	10	R2S DRS messages are always full cacheline messages which are 9 flits. ***Note:*** Flits are variable in the Sys Bound direction.
Ring Bound SNP	1	2	The only ring bound SNP messages.
Ring Bound NDR	1	2	The only ring bound NDR messages

The number of flits sent or received can be divided by the total number of uncore cycles (see Section 5.8.2, "Wbox Performance Monitoring Overview") to calculate the link utilization for each message class. The combined number of flits across message classes can be used to calculate the total link utilization.

Note that for S2R and R2S links, there is no single event which counts the total number of message and credit carrying idle flits sent on the link. The total link utilization can be approximated by adding together the number of flits of the message classes that are expected to be most frequent.

5.7.6 Sbox Events Ordered By Code

Table 5-80 summarizes the directly-measured Sbox events.

Table 5-80. Performance Monitor Events for Sbox Events (Sheet 1 of 2)

Symbol Name	Event Code	Max Inc/Cyc	Description
TO_R_PROG_EV	0x00	1	System Bound Programmable Event
TO_R_B_HOM_MSGQ_CYCLES_FULL	0x03	1	Cycles System Bound HOM Message Queue Full.
TO_R_B_HOM_MSGQ_CYCLES_NE	0x06	1	Cycles System Bound HOM Message Queue Not Empty.
TO_R_B_HOM_MSGQ_OCCUPANCY	0x07	64	System Bound HOM Message Queue Occupancy
TO_R_NDR_MSGQ_OCCUPANCY	0x0D	16	System Bound NDR Message Queue Occupancy
TO_R_DRS_MSGQ_CYCLES_FULL	0x0E	1	Cycles System Bound DRS Message Queue Full
TO_R_DRS_MSGQ_CYCLES_NE	0x0F	1	Cycles System Bound DRS Message Queue Not Empty
TO_R_DRS_MSGQ_OCCUPANCY	0x10	64	System Bound DRS Message Queue Occupancy
TO_R_NCB_MSGQ_CYCLES_FULL	0x11	1	Cycles System Bound NCB Message Queue Full
TO_R_NCB_MSGQ_CYCLES_NE	0x12	1	Cycles System Bound NCB Message Queue Not Empty
TO_R_NCB_MSGQ_OCCUPANCY	0x13	64	System Bound NCB Message Queue Occupancy
TO_R_NCS_MSGQ_CYCLES_FULL	0x14	1	Cycles System Bound NCS Message Queue Full
TO_R_NCS_MSGQ_CYCLES_NE	0x15	1	Cycles System Bound NCS Message Queue Not Empty
TO_R_NCS_MSGQ_OCCUPANCY	0x16	64	System Bound NCS Message Queue Occupancy
TO_RING_SNP_MSGQ_CYCLES_FULL	0x20	1	Cycles Ring Bound SNP Message Queue Full
TO_RING_SNP_MSGQ_CYCLES_NE	0x23	1	Cycles Ring Bound SNP Message Queue Not Empty
TO_RING_MSGQ_OCCUPANCY	0x26	36	Cycles Ring Bound Message Queue Occupancy
TO_RING_NDR_MSGQ_CYCLES_FULL	0x27	1	Cycles Ring Bound NDR Message Queue Full.
TO_RING_NDR_MSGQ_CYCLES_NE	0x28	1	Cycles Ring Bound NDR Message Queue Not Empty
TO_RING_NDR_MSGQ_OCCUPANCY	0x29	32	Ring Bound NDR Message Queue Occupancy
TO_RING_R2S_MSGQ_CYCLES_FULL	0x2A	1	Cycles Ring Bound R2S Message Queue Full.
TO_RING_R2S_MSGQ_CYCLES_NE	0x2C	1	Cycles Ring Bound R2S Message Queue Not Empty
TO_RING_R2S_MSGQ_OCCUPANCY	0x2E	8	Ring Bound R2S Message Queue Occupancy
HALFLINE_BYPASS	0x30	1	Half Cacheline Bypass
REQ_TBL_OCCUPANCY	0x31	48	Request Table Occupancy
EGRESS_BYPASS	0x40	1	Egress Bypass
EGRESS_ARB_WINS	0x41	1	Egress ARB Wins
EGRESS_ARB_LOSSES	0x42	1	Egress ARB Losses
EGRESS_STARVED	0x43	1	Egress Cycles in Starvation
RBOX_HOM_BYPASS	0x50	1	Rbox HOM Bypass
PKTS_SENT_HOM	0x60	1	HOM Packets Sent to System
PKTS_SENT_DRS	0x64	1	DRS Packets Sent to System

Table 5-80. Performance Monitor Events for Sbox Events (Sheet 2 of 2)

Symbol Name	Event Code	Max Inc/Cyc	Description
FLITS_SENT_DRS	0x65	1	DRS Flits Sent to System
PKTS_SENT_NCS	0x66	1	NCS Packets Sent to System
FLITS_SENT_NCS	0x67	1	NCS Flits Sent to System
PKTS_SENT_NCB	0x68	1	NCB Packets Sent to System
FLITS_SENT_NCB	0x69	1	NCB Flits Sent to System
RBOX_CREDIT_RETURNS	0x6A	1	Rbox Credit Returns
PKTS_RCVD_NDR	0x70	1	NDR Packets Received from System
PKTS_RCVD_SNP	0x71	1	SNP Packets Received from System
PKTS_RCVD_DRS_FROM_R	0x72	1	DRS Packets Received from Rbox
RBOX_CREDITS	0x76	1	Rbox Credit Carrying Flits
NO_CREDIT_HOM	0x80	1	HOM Credit Unavailable
NO_CREDIT_DRS	0x82	1	DRS Credit Unavailable
NO_CREDIT_NCS	0x83	1	NCS Credit Unavailable
NO_CREDIT_NCB	0x84	1	NCB Credit Unavailable
NO_CREDIT_VNA	0x86	1	VNA Credit Unavailable
NO_CREDIT_AD	0x87	1	AD Credit Unavailable
NO_CREDIT_AK	0x88	1	AK Credit Unavailable
NO_CREDIT_BL	0x89	1	BL Credit Unavailable
NO_CREDIT_IPQ	0x8A	1	IPQ Credit Unavailable
QEAR_EVENT	0x90	1	QPI EAR Event

5.7.7 Sbox Performance Monitor Event List

This section enumerates Itanium processor 9500 series uncore performance monitoring events for the Sbox.

EGRESS_ARB_LOSSES

- **Title:** Egress ARB Losses
- **Category:** Ring Bound Credits
- **Event Code:** 0x42, **Max. Inc/Cyc:** 1,
- **Definition:** Egress Arbitration Losses.
- **NOTE:** Enabling multiple subevents in this category will result in the counter being increased by the number of selected subevents that occur in a given cycle. Because only one of the even/odd FIFOs can arbitrate to send onto the ring in each cycle, the event for the even/odd FIFOs in each direction are exclusive. The bypass event for each direction is the sum of the bypass events of the even/odd FIFOs.

Extension	umask [15:8]	Description
---	b000000	(*nothing will be counted*)
AD_CW	b000001	AD Clockwise
AD_CCW	b000010	AD Counter-Clockwise
AD	b000011	AD
AK_CW	b000100	AK Clockwise
AK_CCW	b001000	AK Counter-Clockwise

Extension	umask [15:8]	Description
AK	b001100	AK
BL_CW	b010000	BL Clockwise
BL_CCW	b100000	BL Counter-Clockwise
BL	b110000	BL

EGRESS_ARB_WINS

- **Title:** Egress ARB Wins
- **Category:** Ring Bound Transmission
- **Event Code:** 0x41, **Max. Inc/Cyc:** 1,
- **Definition:** Egress Arbitration Wins.
- **NOTE:** Enabling multiple subevents in this category will result in the counter being increased by the number of selected subevents that occur in a given cycle. Because only one of the even/odd FIFOs can arbitrate to send onto the ring in each cycle, the event for the even/odd FIFOs in each direction are exclusive. The bypass event for each direction is the sum of the bypass events of the even/odd FIFOs.

Extension	umask [15:8]	Description
---	b000000	(*nothing will be counted*)
AD_CW	b000001	AD Clockwise
AD_CCW	b000010	AD Counter-Clockwise
AD	b000011	AD
AK_CW	b000100	AK Clockwise
AK_CCW	b001000	AK Counter-Clockwise
AK	b001100	AK
BL_CW	b010000	BL Clockwise
BL_CCW	b100000	BL Counter-Clockwise
BL	b110000	BL

EGRESS_BYPASS

- **Title:** Egress Bypass
- **Category:** Ring Bound Enhancement
- **Event Code:** 0x40, **Max. Inc/Cyc:** 1,
- **Definition:** Egress Bypass optimization utilized.
- **NOTE:** Enabling multiple subevents in this category will result in the counter being increased by the number of selected subevents that occur in a given cycle. Because only one of the even/odd FIFOs can arbitrate to send onto the ring in each cycle, the event for the even/odd FIFOs in each direction are exclusive. The bypass event for each direction is the sum of the bypass events of the even/odd FIFOs.

Extension	umask [15:8]	Description
---	b000000	(*nothing will be counted*)
AD_CW	b000001	AD Clockwise
AD_CCW	b000010	AD Counter-Clockwise
AD	b000011	AD
AK_CW	b000100	AK Clockwise

Extension	umask [15:8]	Description
AK_CCW	b001000	AK Counter-Clockwise
AK	b001100	AK
BL_CW	b010000	BL Clockwise
BL_CCW	b100000	BL Counter-Clockwise
BL	b110000	BL

EGRESS_STARVED

- **Title:** Egress Cycles in Starvation
- **Category:** Ring Bound Credits
- **Event Code:** 0x43, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles the Sbox egress FIFOs are in starvation.
- **NOTE:** Enabling multiple subevents in this category will result in the counter being increased by the number of selected subevents that occur in a given cycle. Because only one of the even/odd FIFOs can arbitrate to send onto the ring in each cycle, the event for the even/odd FIFOs in each direction are exclusive. The bypass event for each direction is the sum of the bypass events of the even/odd FIFOs.

Extension	umask [15:8]	Description
---	b000000	(*nothing will be counted*)
AD_CW	b000001	AD Clockwise
AD_CCW	b000010	AD Counter-Clockwise
AD	b000011	AD
AK_CW	b000100	AK Clockwise
AK_CCW	b001000	AK Counter-Clockwise
AK	b001100	AK
BL_CW	b010000	BL Clockwise
BL_CCW	b100000	BL Counter-Clockwise
BL	b110000	BL

FLITS_SENT_DRS

- **Title:** DRS Flits Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x65, **Max. Inc/Cyc:** 1,
- **Definition:** Number of data response flits the Sbox has transmitted to the system.

FLITS_SENT_NCB

- **Title:** NCB Flits Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x69, **Max. Inc/Cyc:** 11,
- **Definition:** Number of non-coherent bypass flits the Sbox has transmitted to the system.

FLITS_SENT_NCS

- **Title:** NCS Flits Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x67, **Max. Inc/Cyc:** 1,
- **Definition:** Number of non-coherent standard flits the Sbox has transmitted to the system.

HALFLINE_BYPASS

- **Title:** Half Cacheline Bypass
- **Category:** Ring Bound Enhancement
- **Event Code:** 0x30, **Max. Inc/Cyc:** 1,
- **Definition:** Half Cacheline Bypass optimization (where the line is sent early) was utilized.

NO_CREDIT_AD

- **Title:** AD Ring Credit Unavailable
- **Category:** Ring Bound Credits
- **Event Code:** 0x87, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending SNP, NCS or NCB message to send and there is no credit for the target egress FIFO.

NO_CREDIT_AK

- **Title:** AK Ring Credit Unavailable
- **Category:** Ring Bound Credits
- **Event Code:** 0x88, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending NDR or S2C credit return message to send but there is no credit for the target egress FIFO.

NO_CREDIT_BL

- **Title:** BL Ring Credit Unavailable
- **Category:** Ring Bound Credits
- **Event Code:** 0x89, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending DRS or debug message to send and there is no credit for the target egress FIFO.

NO_CREDIT_DRS

- **Title:** DRS Credit Unavailable
- **Category:** System Bound Credits
- **Event Code:** 0x82, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending data response message to send and there is no DRS or VNA credit available.

NO_CREDIT_HOM

- **Title:** HOM Credit Unavailable
- **Category:** System Bound Credits
- **Event Code:** 0x80, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending home message to send and there is no HOM or VNA credit available.

NO_CREDIT_IPQ

- **Title:** IPQ Credit Unavailable
- **Category:** Ring Bound Credits
- **Event Code:** 0x8A, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has an incoming SNP to send but there is no IPQ credit available for the target Cbox.

NO_CREDIT_NCB

- **Title:** NCB Credit Unavailable
- **Category:** System Bound Credits
- **Event Code:** 0x84, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending non-coherent bypass message to send and there is no NCB or VNA credit available.

NO_CREDIT_NCS

- **Title:** NCS Credit Unavailable
- **Category:** System Bound Credits
- **Event Code:** 0x83, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has a pending non-coherent standard message to send and there is no NCS or VNA credit available.

NO_CREDIT_VNA

- **Title:** VNA Credit Unavailable
- **Category:** System Bound Transmission
- **Event Code:** 0x86, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times the Sbox has exhausted its VNA credit pool.

PKTS_RCVD_DRS_FROM_R

- **Title:** DRS Packets Received from Rbox
- **Category:** Ring Bound Transmission
- **Event Code:** 0x72, **Max. Inc/Cyc:** 9,
- **Definition:** Number of data response packets the Sbox has received from the Rbox.
- **NOTE:** DRS messages are always full cacheline messages which are 9 flits. Multiply this event by 9 to derive flit traffic from the Rbox due to DRS messages.

PKTS_RCVD_NDR

- **Title:** NDR Packets Received from System
- **Category:** Ring Bound Transmission
- **Event Code:** 0x70, **Max. Inc/Cyc:** 1,
- **Definition:** Number of non-data response packets the Sbox has received from the system.

PKTS_RCVD_SNP

- **Title:** SNP Packets Received from System
- **Category:** Ring Bound Transmission
- **Event Code:** 0x71, **Max. Inc/Cyc:** 1,
- **Definition:** Number of snoop packets the Sbox has received from the system.

PKTS_SENT_DRS

- **Title:** DRS Packets Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x64, **Max. Inc/Cyc:** 1,
- **Definition:** Number of DRS packets the Sbox has transmitted to the system.
- **NOTE:** If multiple Cboxes are selected, this event counts the total data response packets sent by all the selected Cboxes. In the cases where one DRS message spawns two messages, one to the requester and one to the home, this event only counts the first DRS message. DRS messages are always full cacheline messages which are 9 flits.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	Cboxes 0 and 4
CBOX1_5	bxx1x	Cboxes 1 and 5
CBOX2_6	bx1xx	Cboxes 2 and 6
CBOX3_7	b1xxx	Cboxes 3 and 7

PKTS_SENT_HOM

- **Title:** HOM Packets Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x60, **Max. Inc/Cyc:** 1,
- **Definition:** Number of home packets the Sbox has transmitted to the System.

PKTS_SENT_NCB

- **Title:** NCB Packets Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x68, **Max. Inc/Cyc:** 11,
- **Definition:** Number of NCB packets the Sbox has transmitted to the system.
- **NOTE:** If multiple Cboxes are selected, this event counts the total non-coherent bypass packets sent by all the selected Cboxes. The only ring bound NCB message types are: NcMsgB (StartReq2, VLW), IntLogical, IntPhysical. These are all 11 flit messages.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	Cboxes 0 and 4
CBOX1_5	bxx1x	Cboxes 1 and 5
CBOX2_6	bx1xx	Cboxes 2 and 6
CBOX3_7	b1xxx	Cboxes 3 and 7

PKTS_SENT_NCS

- **Title:** NCS Packets Sent to System
- **Category:** System Bound Transmission
- **Event Code:** 0x66, **Max. Inc/Cyc:** 3,
- **Definition:** Number of NCS packets the Sbox has transmitted to the system.
- **NOTE:** If multiple Cboxes are selected, this event counts the total non-coherent standard packets sent by all the selected Cboxes. The only ring bound NCS message type is NcMsgS (StopReq1). There are always 3 flits.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	Cboxes 0 and 4
CBOX1_5	bxx1x	Cboxes 1 and 5
CBOX2_6	bx1xx	Cboxes 2 and 6
CBOX3_7	b1xxx	Cboxes 3 and 7

QEAR_EVENT

- **Title:** QEAR Event
- **Category:** Miscellaneous
- **Event Code:** 0x90, **Max. Inc/Cyc:** 1,
- **Definition:** Event was captured in the QPI EAR.

RBOX_CREDIT_CARRIERS

- **Title:** Rbox Credit Carrying Flits
- **Category:** Ring Bound Transmission
- **Event Code:** 0x76, **Max. Inc/Cyc:** 1,
- **Definition:** Number credit carrying idle flits received from the Rbox.

RBOX_CREDIT_RETURNS

- **Title:** Rbox Credit Returns
- **Category:** System Bound Transmission
- **Event Code:** 0x6A, **Max. Inc/Cyc:** 1,
- **Definition:** Number credit return idle flits sent to the Rbox.

RBOX_HOM_BYPASS

- **Title:** Rbox HOM Bypass
- **Category:** System Bound Enhancement
- **Event Code:** 0x50, **Max. Inc/Cyc:** 1,
- **Definition:** Rbox HOM Bypass optimization was utilized.

REQ_TBL_OCCUPANCY

- **Title:** Request Table Occupancy
- **Category:** Ring Bound Queue
- **Event Code:** 0x31, **Max. Inc/Cyc:** 48,
- **Definition:** Number of request table entries occupied by socket requests.
- **NOTE:** Occupancy is tracked from allocation to deallocation of each entry in the queue.

TO_RING_SNP MSGQ_OCCUPANCY

- **Title:** Ring Bound SNP Message Queue Occupancy
- **Category:** Ring Bound Queue
- **Event Code:** 0x26, **Max. Inc/Cyc:** 1,
- **Definition:** Number of entries in header buffer containing SNP messages headed for the Ring.

TO_RING_NDR_MSGQ_CYCLES_FULL

- **Title:** Cycles Ring Bound NDR Message Queue Full
- **Category:** Ring Bound Queue
- **Event Code:** 0x27, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing NDR messages on their way to the Ring, is full.

TO_RING_NDR_MSGQ_CYCLES_NE

- **Title:** Cycles Ring Bound NDR Message Queue Not Empty
- **Category:** Ring Bound Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing NDR messages on their way to the Ring, has one or more entries allocated.

TO_RING_NDR_MSGQ_OCCUPANCY

- **Title:** Ring Bound SNP Message Queue Occupancy
- **Category:** Ring Bound Queue
- **Event Code:** 0x29, **Max. Inc/Cyc:** 32,
- **Definition:** Number of entries in header buffer containing NDR messages on their way to the Ring.

TO_RING_R2S_MSGQ_CYCLES_FULL

- **Title:** Cycles Ring Bound R2S Message Queue Full
- **Category:** Ring Bound Queue
- **Event Code:** 0x2A, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing R to Sbox messages on their way to the Ring, is full.

TO_RING_R2S_MSGQ_CYCLES_NE

- **Title:** Cycles Ring Bound R2S Message Queue Not Empty
- **Category:** Ring Bound Queue
- **Event Code:** 0x2C, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing R to Sbox messages on their way to the Ring, has one or more entries allocated.

TO_RING_R2S_MSGQ_OCCUPANCY

- **Title:** Ring Bound R2S Message Queue Occupancy
- **Category:** Ring Bound Queue
- **Event Code:** 0x2E, **Max. Inc/Cyc:** 8,
- **Definition:** Number of entries in header buffer containing R to S messages on their way to the Ring.

TO_RING_SNP_MSGQ_CYCLES_FULL

- **Title:** Cycles Ring Bound SNP Message Queue Full
- **Category:** Ring Bound Queue
- **Event Code:** 0x20, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing SNP messages on their way to the Ring, is full.

TO_RING_SNP_MSGQ_CYCLES_NE

- **Title:** Cycles Ring Bound SNP Message Queue Not Empty
- **Category:** Ring Bound Queue
- **Event Code:** 0x23, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing SNP messages on their way to the Ring, has one or more entries allocated.

TO_R_DRS_MSGQ_CYCLES_FULL

- **Title:** Cycles System Bound DRS Message Queue Full.
- **Category:** System Bound Queue
- **Event Code:** 0x0E, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing DRS messages heading to a System Agent (through the Rbox), is full. Only one Cbox's DRS header buffer should be selected for the buffer full checking to be correct, else the result is undefined.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_DRS_MSGQ_CYCLES_NE

- **Title:** Cycles System Bound DRS Message Queue Not Empty
- **Category:** System Bound Queue
- **Event Code:** 0x0F, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing DRS messages heading to a System Agent (through the Rbox), has one or more entries allocated. When more than one Cbox is selected, the event is asserted when any of the selected Cbox DRS header buffers are not empty.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_DRS_MSGQ_OCCUPANCY

- **Title:** System Bound DRS Message Queue Occupancy
- **Category:** System Bound Queue
- **Event Code:** 0x10, **Max. Inc/Cyc:** 16,
- **Definition:** Number of entries in the header buffer for the selected Cbox, containing DRS messages heading to a System Agent (through the Rbox). When more than one Cbox is selected, the queue occupancy counter counts the total number of occupied entries in all selected Cbox DRS header buffers.
- **NOTE:** 1 buffer per Cbox, 4 entries each.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ALL	b1111	All Cboxes

TO_R_HOM_MSGQ_CYCLES_FULL

- **Title:** Cycles System Bound HOM Message Queue Full.
- **Category:** System Bound Queue
- **Event Code:** 0x03, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing HOM messages heading to the System is full.

TO_R_HOM_MSGQ_CYCLES_NE

- **Title:** Cycles System Bound HOM Header Not Empty
- **Category:** System Bound Queue
- **Event Code:** 0x06, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer, containing HOM messages heading to the System has one or more entries allocated.

TO_R_HOM_MSGQ_OCCUPANCY

- **Title:** System Bound HOM Message Queue Occupancy
- **Category:** System Bound Queue
- **Event Code:** 0x07, **Max. Inc/Cyc:** 64,
- **Definition:** Number of entries in the header buffer containing HOM messages heading to the System.

TO_R_NCB_MSGQ_CYCLES_FULL

- **Title:** Cycles System Bound NCB Message Queue Full.
- **Category:** System Bound Queue
- **Event Code:** 0x11, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing NCB messages heading to a System Agent (through the Rbox), is full. Only one Cbox's NCB header buffer should be selected for the buffer full checking to be correct, else the result is undefined.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_NCB_MSGQ_CYCLES_NE

- **Title:** Cycles System Bound NCB Message Queue Not Empty
- **Category:** System Bound Queue
- **Event Code:** 0x12, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing NCB messages heading to a System Agent (through the Rbox), has one or more entries allocated. When more than one Cbox is selected, the event is asserted when any of the selected Cbox DRS header buffers are not empty.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_NCB_MSGQ_OCCUPANCY

- **Title:** System Bound NCB Message Queue Occupancy
- **Category:** System Bound Queue
- **Event Code:** 0x13, **Max. Inc/Cyc:** 8,
- **Definition:** Number of entries in the header buffer for the selected Cbox, containing NCB messages heading to a System Agent (through the Rbox). When more than one Cbox is selected, the queue occupancy counter counts the total number of occupied entries in all selected Cbox NCB header buffers.
- **NOTE:** 1 buffer per Cbox, 2 entries each.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ALL	b1111	All Cboxes

TO_R_NCS_MSGQ_CYCLES_FULL

- **Title:** Cycles System Bound NCS Message Queue Full.
- **Category:** System Bound Queue
- **Event Code:** 0x14, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing NCS messages heading to a System Agent (through the Rbox), is full. Only one Cbox's NCS header buffer should be selected for the buffer full checking to be correct, else the result is undefined.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_NCS_MSGQ_CYCLES_NE

- **Title:** Cycles System Bound NCS Message Queue Not Empty
- **Category:** System Bound Queue
- **Event Code:** 0x15, **Max. Inc/Cyc:** 1,
- **Definition:** Number of cycles in which the header buffer for the selected Cbox, containing NCS messages heading to a System Agent (through the Rbox), has one or more entries allocated. When more than one Cbox is selected, the event is asserted when any of the selected Cbox NCS header buffers are not empty.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ANY	b1111	Any Cbox

TO_R_NCS_MSGQ_OCCUPANCY

- **Title:** System Bound NCS Message Queue Occupancy
- **Category:** System Bound Queue
- **Event Code:** 0x16, **Max. Inc/Cyc:** 2,
- **Definition:** Number of entries in the header buffer for the selected Cbox, containing NCS messages heading to a System Agent (through the Rbox). When more than one Cbox is selected, the queue occupancy counter counts the total number of occupied entries in all selected Cbox NCS header buffers.
- **NOTE:** 1 buffer per Cbox, 2 entries each.

Extension	umask [15:8]	Description
---	b0000	(*nothing will be counted*)
CBOX0_4	bxxx1	CBOX 0 and 4
CBOX1_5	bxx1x	CBOX 1 and 5
CBOX2_6	bx1xx	CBOX 2 and 6
CBOX3_7	b1xxx	CBOX 3 and 7
ALL	b1111	All Cboxes

TO_R_PROG_EV

- **Title:** System Bound Programmable Event
- **Category:** System Bound Queue
- **Event Code:** 0x00, **Max. Inc/Cyc:** 1,
- **Definition:** Programmable Event heading to a System Agent (through the Rbox). Match/Mask on criteria set in S_CSR_MATCH/MASK registers (Refer to Section 5.7.3.4, “Sbox Registers for Mask/Match Facility”).

5.8 Wbox Performance Monitoring

5.8.1 Overview of the Wbox

The Wbox is the primary Power Controller for the Itanium processor 9500 series.

5.8.2 Wbox Performance Monitoring Overview

The Wbox supports event monitoring through four 48-bit wide counters (W_CSR_PERF_CNT{3:0}). Each of these four counters can be programmed to count any Wbox event. The Wbox counters will increment by a maximum of 1 per cycle.

The Wbox also provides a 48-bit wide fixed counter that increments at the uncore clock frequency.

The count values of all 5 counters can be cleared by writing the W_CSR_PMON_PERF_MASTER.clr bit.

For information on how to setup a monitoring session, refer to Section 5.3, “Global Performance Monitoring Control”.

5.8.2.1 Wbox PMU - Overflow, Freeze and Unfreeze

Wbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Wbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, “Global Enable/Disable”).

Wbox PMU can be frozen due to one of three reasons:

- *Globally:* Ubox sends a disable signal (and W_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, “Global Enable/Disable”) or *local* (SW writes 0 to W_CSR_PMON_PERF_MASTER.en when W_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.

- *Locally:* The Wbox was set to local control (W_CSR_PMON_PERF_MASTER.glb_lcl = 0) and a Wbox counter overflowed.

If an overflow is detected from a Wbox performance counter, the overflow bit is set at the box level (W_CSR_PMON_GLOBAL_STATUS.ov), and forwarded up the chain towards the Sbox. That is, if a Wbox counter overflows, a notification is sent and stored in Sbox (S_CSR_PMON_SUMMARY.ov_w). Refer to Table 5-64, "S_CSR_PMON_SUMMARY Register Fields" to determine how each Wbox's overflow bit is accumulated in the attached Sbox.

The Ubox may be configured to freeze all uncore counting (refer to Table 5-65, "S_CSR_PMON_FRZ_EN Register Fields") when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field responsible for the freeze, must be cleared. Assuming all the counters have been locally enabled (.en bit in data registers meant to monitor events) and the overflow bit(s) has been cleared, the Wbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, "Enabling a New Sample Interval from Frozen Counters."), counting will resume.

5.8.3 Wbox Performance Monitors

Table 5-81. Wbox Performance Monitoring CSRs

CSR Name	Access	CSR Address	Size (bits)	Description
SAL Privilege Registers				
W_CSR_PMON_GLOBAL_STATUS	RW_RW	0x6C8	32	Wbox PMON Global Overflow Status
W_CSR_PMON_GLOBAL_CTL	RW_RW	0x6C0	32	Wbox PMON Global Control
W_CSR_PMON_FIXED_CTR_CTL	RW_RW	0x4F8	64	Wbox PMON Fixed Counter Control
W_CSR_PMON_FIXED_CTR	RW_RW	0x4E8	64	Wbox PMON Fixed Counter
W_CSR_PMON_EVT_SEL_3	RW_RW	0x4E0	64	Wbox PMON Control 3
W_CSR_PMON_EVT_SEL_2	RW_RW	0x4D8	64	Wbox PMON Control 2
W_CSR_PMON_EVT_SEL_1	RW_RW	0x4D0	64	Wbox PMON Control 1
W_CSR_PMON_EVT_SEL_0	RW_RW	0x4C8	64	Wbox PMON Control 0
W_CSR_PMON_CTR_3	RW_RW	0x4C0	64	Wbox PMON Counter 3
W_CSR_PMON_CTR_2	RW_RW	0x4B8	64	Wbox PMON Counter 2
W_CSR_PMON_CTR_1	RW_RW	0x4A8	64	Wbox PMON Counter 1
W_CSR_PMON_CTR_0	RW_RW	0x4A0	64	Wbox PMON Counter 0
NONE Privilege Registers				
W_CSR_PMON_PERF_MASTER	RW_RW	0x040	32	Wbox PMON Global Overflow Status

5.8.3.1 Wbox Box Level PMON state

The following registers represent the state governing all box-level PMUs in the Wbox.

W_CSR_PMON_PERF_MASTER controls the general characteristics of the Wbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

The _GLOBAL_CTL register contains the bits used to enable monitoring. It is necessary to set the .ctr_en bit to 1 before the corresponding data register can collect events.

If an overflow is detected from one of the Wbox PMON registers, the corresponding bit in the _GLOBAL_STATUS.ov field will be set.

Table 5-82. W_CSR_PMON_PERF_MASTER Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:3	RO_NA	0	Read zero; writes ignored. (?)
clr	2	RW_RW	0	Writing 1 clears all the Wbox counter.
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: **Global**: Enable/Disable of counters in Wbox will track U_CSR_PERF_CTL.glb_en. Local overflows will be passed on to Ubox without freezing local counters. 0: **Local**: Enable/Disable of counters in Wbox will NOT track U_CSR_PERF_CTL.glb_en. Allows SW to write the .en bit. Disables Counters on any local counter overflow.
en	0	RW_RW	0	Enable/disable Cbox PMU counters. This bit is dependent on the setting of the .glb_lcl bit. If .glb_lcl is set to 1, SW writes to this bit are ignored and only HW may affect it's state. If .glb_lcl is set to 0, SW may exert control by setting the bit. In either case, since HW may alter this bit, (due to tracking the global enable or a local overflow) SW may read it to determine the state of the Wbox counters. 1: Enable Wbox PMU counting. 0: Disable (freeze) Wbox PMU counters.

Table 5-83. W_CSR_PMON_GLOBAL_CTL Register Fields

Field	Bits	Access	HW Reset Val	Description
fixed_en	31	RW_RO	0	Enable the fixed counter
ig	30:4	RO_NA	0	Read zero; writes ignored. (?)
ctr_en	3:0	RW_RO	0	Must be set to enable each WBOX counter (bit 0 to enable ctr0, and so forth) ***Note:*** Ubox enable and per counter enable must also be set to fully enable the counter.

Table 5-84. W_CSR_PMON_GLOBAL_STATUS Register Fields

Field	Bits	Access	HW Reset Val	Description
ov_fixed	31	RO_RW	0	If an overflow is detected from the WBOX PMON fixed counter, this bit will be set.
ig	30:4	RO_NA	0	Read zero; writes ignored. (?)
ov	3:0	RO_RW	0	If an overflow is detected from the corresponding WBOX PMON register, it's overflow bit will be set.

5.8.3.2 Wbox PMON state - Counter/Control Pairs

The following table defines the layout of the Wbox performance monitor control registers. The main task of these configuration registers is to select the event to be monitored by their respective data counter. Setting the *.ev_sel* and *.umask* fields performs the event selection. The *.en* bit must be set to 1 to enable counting.

Additional control bits include:

- *.threshold* - since Wbox counters can increment by a value greater than 1, a threshold can be applied. If the *.threshold* 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. (Not present in fixed counter)

- *.invert* - Changes the .threshold test condition to '<' (Not present in fixed counter)

- *.edge_detect* - 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. (Not present in fixed counter)

Table 5-85. W_MSR_PMON_EVT_SEL_{3-0} Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63	RO_NA	0	Read zero; writes ignored. (?)
rsv	62:61	RW_NA	0	Reserved; Must write to 0 else behavior is undefined.
ig	60:51	RO_NA	0	Read zero; writes ignored. (?)
rsv	50	RW_NA	0	Reserved; Must write to 0 else behavior is undefined.
ig	49:32	RO_NA	0	Read zero; writes ignored. (?)
threshold	31:24	RW_RO	0	Threshold used for counter comparison. Set to at least 0x1 for ALL events. With a setting of 0x0, the counter will increment at the Wbox clock rate.
invert	23	RW_RO	0	Invert threshold comparison. When '0', the comparison will be thresh >= event. When '1', the comparison will be threshold < event.
en	22	RW_RO	0	Local Counter Enable. When set, the associated counter is locally enabled. NOTE: It must also be enabled in S_MSR_PMON_GLOBAL_CTL and the Ubox to be fully enabled.
ig	21	RO_NA	0	Read zero; writes ignored. (?)
rsv	20	WO_NA	0	Reserved; Must write to 0 else behavior is undefined.
ig	19	RO_NA	0	Read zero; writes ignored. (?)
edge_detect	18	RW_RO	0	Edge Detect. When bit is set, 0->1 transition of a one bit event input will cause counter to increment. When bit is 0, counter will increment for however long event is asserted.
rsv	17	WO_NA	0	Reserved; Must write to 0 else behavior is undefined.
ig	16	RO_NA	0	Read zero; writes ignored. (?)
umask	15:8	RW_RO	0	Unit Mask - select subevent of event.
ev_sel	7:0	RW_RO	0	Event Select

Table 5-86. W_MSR_PMON_FIXED_CTR_CTL Register – Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	63:3	RO_NA	0	Read zero; writes ignored. (?)
rsv	2:1	RW_RO	0	Reserved; Must write to 0 else behavior is undefined.
en	0	RW_RO	0	Counter enable

The Wbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-87. W_MSR_PMON_CTR_{3-0} Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored. (?)
event_count	47:0	RW_RW	0	48-bit performance event counter

Table 5-88. W_MSR_PMON_FIXED_CTR Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored. (?)
event_count	47:0	RW_RW	0	48-bit performance event counter

5.8.4 Wbox Performance Monitoring Events

5.8.4.1 An Overview:

The Wbox's primary offering to understanding the impact of the uncore on performance is the fixed counter. This counter, which increments at the frequency of the uncore clock, can be used to add a time element to numerous events across the uncore.

Beyond that, the Wbox provides a smattering of events that indicating when, and under what circumstances, the Wbox throttled the chip due to power constraints.

It is necessary to set the .thresh to 0x1 in order to capture any event. If .thresh is set to 0x0, the counter will increment at the Wbox clock rate.

5.8.5 Wbox Events Ordered By Code

Table 5-89 summarizes the directly-measured Wbox events.

Table 5-89. Performance Monitor Events for Wbox Events

Symbol Name	Event Code	Max Inc/Cyc	Description
S_THROTTLE_TMP	0x01	1	Socket Throttled due to Temp (Lite Throttling)
S_FORCEPR	0x02	1	FORCEPR
S_THROTTLE_PROCHOT	0x03	1	Socket Throttled due to PROCHOT (Heavy Throttling)
S_P0_STATE	0x04	1	Socket in P0 at Turbo
PSTATE_CHANGE	0x05	1	P-State Change
S_C1E_STATE	0x06	1	Socket in C1E
S_P0_REQUEST	0x07	1	P0 state requested
S_PROCHOT	0x09	1	PROCHOT

5.8.6 Wbox Performance Monitor Event List

This section enumerates Itanium processor 9500 series uncore performance monitoring events for the Wbox.

PSTATE_CHANGE

- **Title:** P-State Change
- **Category:** Wbox Events
- **Event Code:** 0x07, **Max. Inc/Cyc:** 1,
- **Definition:** Any P-State Change

Extension	Umask	Description
C0	0xfe-0x00	**(* illegal selection *)**
ALL	0xff	All Cores

RATIO_CHANGE_ABORT

- **Title:** Ratio Change Abort
- **Category:** Wbox Events
- **Event Code:** 0x08, **Max. Inc/Cyc:** 1,
- **Definition:** Selected core aborted a ratio change request.

Extension	Umask	Description
C0	0xfe-0x00	**(* illegal selection *)**
ALL	0xff	All Cores

S_P0_STATE

- **Title:** Socket in P0
- **Category:** Wbox Events
- **Event Code:** 0x04, **Max. Inc/Cyc:** 1,
- **Definition:** Pcode detects that socket is in P0 P-State

S_FORCEPR

- **Title:** FORCEPR
- **Category:** Wbox Events
- **Event Code:** 0x02, **Max. Inc/Cyc:** 1,
- **Definition:** Package is asserting the FORCEPR output.

S_PROCHOT

- **Title:** Prochot
- **Category:** Wbox Events
- **Event Code:** 0x09, **Max. Inc/Cyc:** 1,
- **Definition:** Package is asserting the PROCHOT output.

S_THROTTLE_PROCHOT

- **Title:** Socket Throttled due to PROCHOT
- **Category:** Wbox Events
- **Event Code:** 0x03, **Max. Inc/Cyc:** 1,
- **Definition:** Socket is thermally throttled due to PROCHOT condition (heavy throttling).

S_THROTTLE_TMP

- **Title:** Socket Throttled due to Temp
- **Category:** Wbox Events
- **Event Code:** 0x01, **Max. Inc/Cyc:** 1,
- **Definition:** Socket is thermally throttled due to thermal condition (lite throttling).

Extension	Umask	Description
C0	0xfe-0x00	**(* illegal selection *)**
ALL	0xff	All Cores

5.9 Zbox Performance Monitoring

5.9.1 Overview of the Zbox

The memory controller interfaces to the Intel 7500 scalable memory controller and translates read and write commands into specific Intel® Scalable Memory Interconnect operations. Intel® SMI is based on the FB-DIMM architecture, but Intel 7500 scalable memory controller is not an AMB2 device and has significant exceptions to the FB-DIMM2 architecture. The memory controller also provides a variety of RAS features, such as Intel® Double Device Data Correction, memory scrubbing, thermal throttling, mirroring, and DIMM sparing. Each socket has two independent memory controllers, and each memory controller has two Intel SMI channels that operate in lockstep.

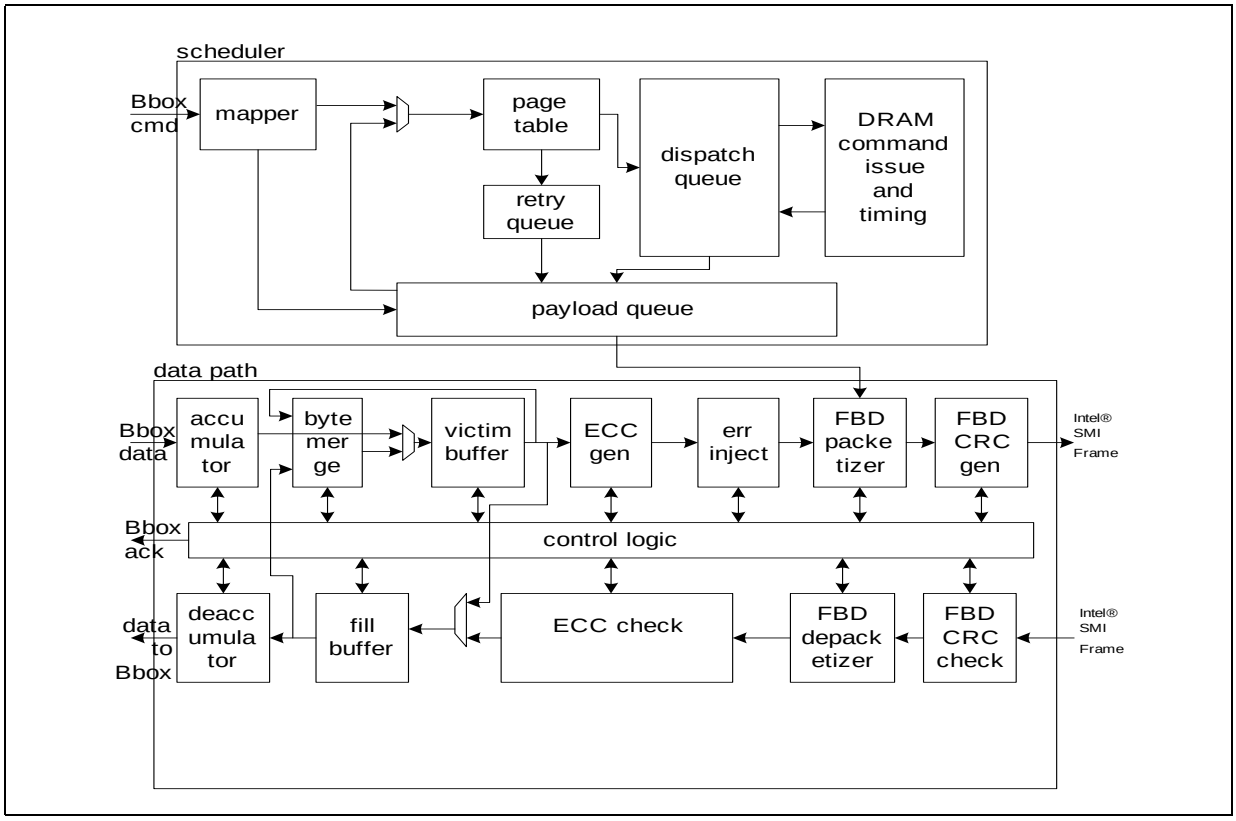
Note: If memory mirroring is enabled on the platform in the memory controller, then the Zbox Performance Monitoring registers are reserved.

5.9.2 Functional Overview

The memory controller is the interface between the home node controller (Bbox) and the the Scalable Memory Interconnect and basically translates 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. There are two memory controllers per socket, each controlling two Intel SMI channels in lockstep. Because of the data path affinity to the Bbox data path, each Bbox is paired with a memory controller, that is, Bboxes and memory controllers come in pairs.

Figure 5-4. Memory Controller Block Diagram



The memory controller interfaces to the router through the Bbox (home node coherence controller) and to the Pbox pads. The Pbox pads connect to Intel SMI memory via a synchronizer (the Jbox). This enables memory controller (which runs at system interface frequency with the home node controller and router (Bbox/Rbox)) to interface to various speeds of DIMMs.

5.9.2.1 Intel® 7500 Scalable Memory Controller

The Intel Itanium processor 9500 series supports Intel® 7500 scalable memory controllers on the Intel SMI channels.

- Intel SMI protocol and signalling includes support for the following:
  - — 4.8 Gbs, 6.4 Gbs signalling
  - — forwarded clock fail-over NB and SB.
  - — 9 data lanes plus 1 CRC lane plus 1 spare lane SB.
  - — 12 data lanes plus 1 CRC lane plus 1 spare NB.
  - — Support for integrating RDIMM thermal sensor information into Intel® SMI Status Frame.

- No support for daisy chaining (Intel 7500 scalable memory controller is the only Intel SMI device in the channel).
- No support for FB-DIMM1 protocol and signaling.

The Intel 7500 scalable memory controller provides an interface to DDR3 DIMMs and supports the following DDR3 functionality:

- DDR3 protocol and signalling, includes support for the following:
  - — Up to two RDIMMs per DDR3 bus
  - — Up to eight physical ranks per DDR3 bus (sixteen per Intel 7500 scalable memory controller)
  - — 800 MT/s or 1066 MT/s (both DDR3 buses must operate at the same frequency)
  - — Single Rank x4, Dual Rank x4, Single Rank x8, Dual Rank x8, Quad Rank x4, Quad Rank x8
  - — 1 GB, 2 GB, 4 GB, 8 GB, 16 GB DIMM
  - — DRAM device sizes: 1 Gb, 2 Gb
  - — Mixed DIMM types (no requirement that DIMMs must be the same type, except that all DIMMs attached to Intel 7500 scalable memory controller must run with a common frequency and core timings). (Host lockstep requirements may impose additional requirements on DIMMs on separate Intel SMI channels).
  - — DDR buses may contain different number of DIMMs, zero through two. (Host lockstep requirements may impose additional requirements on DIMMs on separate Intel® SMI channels).
  - — Cmd/Addr parity generation and error logging.
- No support for non-ECC DIMMs
- No support for DDR2 protocol and signaling
- Support for integrating RDIMM thermal sensor information into Intel® SMI Status Frame.

See the *Intel® 7500 Scalable Memory Buffer – External Design Specification (EDS)* for more information.

5.9.3 Zbox Perfmon Overview

Each of the Zboxes in the Itanium processor 9500 series supports event monitoring through five 48-bit wide counters (Z_CSR_PMU_CNT_{4-0}). The Zbox counters can increment by a maximum of 32 per cycle.

Although each of the Z_CSR_PMU_CNT* register can be configured to monitor any available event through its companion control register, a good chunk of the generic events defer configuration to various subcontrol registers (as detailed below). Since there are a limited number of control registers, software must pay attention to various restrictions as to what events may be counted simultaneously.

The count values of all 5 counters can be cleared by writing the Z_CSR_PMON_PERF_MASTER.clr bit.

5.9.3.1 Choosing An Event To Monitor - Example using subcontrol registers

As has been stated, monitoring a particular event often requires configuring auxiliary CSRs.

For instance, to count (in counter 0) the number of RAS DRAM commands (PLD_DRAM_EV.DRAM_CMD.RAS) that have been issued, set up is as follows:

```
Z_ZSR_PMU_CNT_CTL_0.inc_sel [9:4] = 0xa

Z_CSR_PMU_PLD.cmd [0] = 0x0

Z_CSR_PMU_PLD.dram_cmd [12:8] = 0x2
```

To count (in counter 2) the number of Victim to Fill Buffer transfers from the B to the ZBox (FVC_EV*.BBOX_CMDS.V2F), set up is as follows:

```
Z_CSR_PMU_CNT_CTL_0.inc_sel [9:4] = 0x0f
```

`Z_CSR_PMU_ZDP_CTL_FVC.evnt3 [20:18] = 0x5` (User can chose between 4 different FVC events, here the 3rd slot was arbitrarily chosen.)

```
Z_CSR_PMU_ZDP_CTL_FVC.bcmd [8:5] = 0x3
```

5.9.3.2 Zbox PMU - Overflow, Freeze and Unfreeze

Zbox PMUs support the same overflow and freeze related mechanisms that are supported by the other uncore PMUs. Users can choose to freeze just the Zbox PMUs, or all of the uncore PMUs (refer to Section 5.3.1, “Global Enable/Disable”).

Zbox PMU can be frozen due to one of three reasons:

- *Globally:* Ubox sends a disable signal (and Z_CSR_PMON_PERF_MASTER.glb_lcl is 1)
- *Manually:* SW forces a freeze either through the *global* disable (Section 5.3.1, “Global Enable/Disable”) or *local* (SW writes 0 to Z_CSR_PMON_PERF_MASTER.en when Z_CSR_PMON_PERF_MASTER.glb_lcl is 0) mechanism.
- *Locally:* The Zbox was set to local control (Z_CSR_PMON_PERF_MASTER.glb_lcl = 0) and a Zbox counter overflowed.

If an overflow is detected from a Zbox performance counter, the overflow bit is set at the box level (Z_CSR_PMU_CNT_STATUS.ov[x]), and forwarded up the chain towards the Sbox. i.e. If a Zbox counter overflows, a notification is sent and stored in Sbox (S_CSR_PMON_SUMMARY.ov_z). Refer to Table 5-90, “Zbox Performance Monitoring CSRs” to determine how each Zbox’s overflow bit is accumulated in the attached Sbox.

The Ubox may be configured to freeze all uncore counting (refer to Table 5-65, “S_CSR_PMON_FRZ_EN Register Fields”) when it receives this signal.

Once a freeze has occurred, in order to see a new freeze, the overflow field responsible for the freeze, must be cleared. Once all the overflow bit(s) have been cleared, the Zbox is prepared for a new sample interval. Once the global controls have been re-enabled (Section 5.3.4, “Enabling a New Sample Interval from Frozen Counters.”), counting will resume.

5.9.4 Zbox PerfMon Registers

5.9.4.1 Zbox Perfmon CSR Map

Table 5-90. Zbox Performance Monitoring CSRs

CSRName[1]	Addr Offset [11:0]	Priv Lvl	CSR Description
Box-Level Control			
Z_CSR_PMON_PERF_MASTER	0x0D8	None	Zbox Performance Monitoring Control
SubControl Registers			
Z_CSR_PMU_ZDP_CTL_FVC	0x0D0	None	Zbox Subcontrol for FVC events.
Z_CSR_PLD_PMU	0x0C8	None	Zbox Subcontrol for PLD events.
Z_CSR_PGT_PMU	0x0C0	None	Zbox Subcontrol for PGT events.
Z_CSR_PMU_MSC_THR	0x0B8	None	Zbox Subcontrol for THR events.
Z_CSR_ISS_PMU	0x0B0	None	Zbox Subcontrol for ISS events.
Z_CSR_DSP_PMU	0x0A8	None	Zbox Subcontrol for DSP events.
Box-Level Status			
Z_CSR_PMU_CNT_STATUS	0x0A0	None	Holds the status flags for the five performance monitor unit counters.
Generic Counter / Control			
Z_CSR_PMU_CNT_CTL_6	0x098	None	Zbox Performance Counter Control 5
Z_CSR_PMU_CNT_CTL_4	0x090	None	Zbox Performance Counter Control 4
Z_CSR_PMU_CNT_CTL_3	0x088	None	Zbox Performance Counter Control 3
Z_CSR_PMU_CNT_CTL_2	0x080	None	Zbox Performance Counter Control 2
Z_CSR_PMU_CNT_CTL_1	0x078	None	Zbox Performance Counter Control 1
Z_CSR_PMU_CNT_CTL_0	0x070	None	Zbox Performance Counter Control 0
Z_CSR_PMU_CNT_5	0x068	None	Zbox Performance Counter 5
Z_CSR_PMU_CNT_4	0x060	None	Zbox Performance Counter 4
Z_CSR_PMU_CNT_3	0x058	None	Zbox Performance Counter 3
Z_CSR_PMU_CNT_2	0x050	None	Zbox Performance Counter 2
Z_CSR_PMU_CNT_1	0x048	None	Zbox Performance Counter 1
Z_CSR_PMU_CNT_0	0x040	None	Zbox Performance Counter 0
1. The above Zbox PMU registers are reserved when memory mirroring is enabled in the memory controller.			

5.9.4.2 Zbox Box Level PMU State

The following register represents the state governing all box-level PMUs in the Zbox.

Z_CSR_PMON_PERF_MASTER controls the general characteristics of the Zbox PMU. It allows the user to freeze/unfreeze the PMU through software, clear all PMU data counters, and determine the freeze status of the PMU through SW.

If an overflow is detected from one of the Zbox PMON registers, the corresponding bit in the Z_CSR_PMU_CNT_STATUS.ov field will be set. If the counters were set to be frozen upon detection of an overflow, the *.frz* bit will be set.

SW must set *.unfrz* to 1 in order to resume counting.

Table 5-91. Z_CSR_PMON_PERF_MASTER Register – Field Definitions

Field	Bits	Type	HW Reset Val	Description
ig	63:3	RO_NA	0	Read zero; writes ignored. (?)
clr	2	RW_RW	0	Writing 1 clears all Zbox counters as well as the overflow bits found in Z_CSR_PMON_GLOBAL_STATUS.
glb_lcl	1	RW_RW	0	Used to select whether to exert local or global control. 1: **Global**: Enable/Disable of counters in Zbox will track U_CSR_PERF_CTL.glb_en. Local overflows will be passed on to Ubox without freezing local counters. 0: **Local**: Enable/Disable of counters in Zbox will NOT track U_CSR_PERF_CTL.glb_en. Allows SW to write the .en bit. Disables Counters on any local counter overflow.
en	0	RW_RW	0	Enable/disable Zbox PMU counters. This bit is dependent on the setting of the .glb_lcl bit. If .glb_lcl is set to 1, SW writes to this bit are ignored and only HW may affect it's state. If .glb_lcl is set to 0, SW may exert control by setting the bit. In either case, since HW may alter this bit, (due to tracking the global enable or a local overflow) SW may read it to determine the state of the Zbox counters. 1: Enable Zbox PMU counting. 0: Disable (freeze) Zbox PMU counters.

Table 5-92. Z_CSR_PMU_CNT_STATUS Register Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
frz	31	RW_RO	0	Write this bit to a 1 to freeze all the Zbox PMU counters. The write of a 1 generates a freeze pulse. Writing this CSR with this bit set to zero has no effect. Reading this bit will return a 1 if the Zbox PMU counters are frozen from this bit or from a counter overflowing with the frz_mode bit set to 1.
unfrz	30	RW1C_WO	0	Write this bit to a 1 to unfreeze all the Zbox PMU counters. Writing this CSR with this bet set to zero has no effect
ig	29:6	RW_NA		Reads 0; writes ignored.
ov	5:0	RW_WO	0	If an overflow is detected from the corresponding Zbox PMON register, it's overflow bit will be set.

5.9.4.3 Zbox PMON state - Counter/Control Pairs

The following table defines the layout of the Zbox performance monitor control registers. The main task of these configuration registers is to select the event to be monitored by their respective data counter. Setting the *.inc_sel* field performs the event selection. Many of the events selected may be broken into components through use of companion subcontrol registers. See Section 5.9.7, "Zbox Performance Monitor Event List" for more details.

Additional control bits include:

Table 5-93. Z_CSR_PMU_CNT_CTRL_{5-0} Register Field Definitions

Field	Bits	Access	HW Reset Val	Description
ig	31:10	RO_NA	0	Read zero; writes ignored.
inc_sel	9:4	RW_RW	0	Selects the increment input signal, the primary event select, for this counter. See Table 5-105, "Performance Monitor Events for ZBox Events" for encodings.
frz_mode	3	RW_RW	0	Counter freeze mode: 0 - On overflow, freeze only this counter 1 - On overflow, freeze all counters (if wrap_mode == 0) ***Note:*** For normal operation, it is suggested to set bits 3:2 to 11
wrap_mode	2	RW_RW	0	Counter wrap mode 0 - On overflow, stop counting 1 - On overflow, wrap:
rsv	1	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.
ig	0	RO_NA	0	Read zero; writes ignored.

The Zbox performance monitor data registers are 48b wide. A counter overflow occurs when a carry out bit from bit 47 is detected. Software can force uncore counting to freeze after N events by preloading a monitor with a count value of 2^{48} - N. Upon receipt of the masked (by S_CSR_PMON_FRZ_EN) overflow signal, the Ubox can forward the freeze signal to the other uncore boxes (Section 5.3.1, "Global Enable/ Disable"). During the interval of time between overflow and global disable, the counter value will wrap and continue to collect events.

If .*wrap_mode* in the counter's CTRL register was set to 1, during the interval of time between overflow and global freeze, the counter value will wrap and continue to collect events. In this way, software can capture the precise number of events that occurred between the time uncore counting was enabled and when it was disabled (or 'frozen') with minimal skew.

If accessible, software can continuously read the data registers without disabling event collection.

Table 5-94. Z_CSR_PMU_CNT_{5-0} Fields

Field	Bits	Access	HW Reset Val	Description
ig	63:48	RO_NA	0	Read zero; writes ignored.
cnt	47:0	RW_RW	0x0	Performance Counter Value

5.9.4.4 Zbox PMU Subcontrol Registers - Subunit descriptions

The following Tables contain information on how to program the various subcontrol registers contained within the Zbox which include the DSP, ISS, THR, PGT, PLD and FVC registers. The subcontrol registers govern events coming from subunits within the Zbox which can be roughly categorized as follows:

PLD - Payload Queue - Receives command and translated addresses from the MAP while the PGT translates MAP commands into DRAM command combinations.

Original Bbox transaction's FVID sent from DSP during subcommand execution where the appropriate subcommand information is accessed to compose the FBD command frame.

PGT - Page Table - Keeps track of open pages. Translates the read/write commands into DRAM command combinations (i.e. PRE, RAS, CASrd, CASwr). The generated command combination (e.g. PRE_RAS_CASrd) is then sent to the Dispatch Queue.

If

a. there is already a command in the DSP for a particular DIMM (rank/bank)
b. the DSP's readQ or writeQ is full
c. if a rank requires thermal throttling because the DIMM is heating up
d. or a refresh is executing to a rank.

Then the PGT will detect the conflict and place the command in the retryQ for later execution.

DSP - Dispatch Queue - receives DRAM command from PGT and stores request in a read or write subqueue. In the dispatch queue, the command combinations are broken up into subcommand kinds that are sequenced in the necessary order required to complete the read/write transaction (i.e. PRE, RAS, CAS, CASpre). All "ready to execute" subcommands stored within the various DSP queues are presented simultaneously to the issue logic.

Once the ISS returns the subcommand choice, the oldest DSP entry containing that subcommand kind (for a particular DIMM) is allowed to execute. During subcommand execution, the DSP sends the original (BBox) transaction's FVID (that was stored in the DSP entry) to the PLD. After subcommand execution, the DSP's queue entry state is updated to the next required subcommand kind (based on the original command combination) to be executed (new state).

ISS - Issue - receives "ready to execute" subcommands from the dispatch queue(s) as a bit vector that is organized with each bit representing a subcommand kind per DIMM (i.e. RAS for DIMM0, CAS for DIMM3). Having an overview of all these subcommand kinds enables the ISS to flexibly schedule/combine subcommands out-of-order. Once a subcommand kind for a particular DIMM is selected from the issue vector by the ISS, that subcommand choice is driven back to the DSP

THR - Thermal Throttling

FVC - Fill and Victim Control - drives all the control signals required by the fill datapath and victim datapath. Additionally, it handles issuing and control of the buffer maintenance commands (i.e. MRG, F2V, V2V, V2F and F2B). It also contains the logic to respond to the BBox when commands in the ZBox have completed.

The **DSP** subcontrol register contains bits to specify subevents of the DSP_FILL event, breaking it into write queue/read queue occupancy as well as DSP latency.

Table 5-95. Z_CSR_DSP_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
ig	31:18	RO_NA	0	Reads 0; writes ignored.
pgtAddr	17:11	RW_RW	0	Address used by increment select for inserts to DSP with this address(DSP_INSERT_TO_ADDRESS) and inserts to this address with conflicts(DSP_INSERT_TO_ADDR_CFLT).
dspq_empty	10	RW_RW	0	Generates DSP_FILL trigger when dispatch queue is empty.
---	9:0	RW_RW	0	(* illegal selection *)

The **ISS** subcontrol register contains bits to specify subevents for the ISS_EV (by Intel SMI frame) and PLD_DRAM_EV (DRAM commands broken down by scheduling mode in the ISS) events.

Table 5-96. Z_CSR_ISS_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
ig	31:10	RO_NA		Reads 0; writes ignored.
sched_mode_pld_trig	9:7	RW_RW	0	Selects the scheduling mode for which the number of DRAM commands is counted in ZAD_PLD. Here for implementation reasons. Uses same encodings as Z_CSR_ISSUE_MODE.iss_mode: 000: reserved 001: static tradoff 010: static rd priority 011: reserved 100: static wr priority 111-101: reserved
sched_mode	6:4	RW_RW	0	Selects the scheduling mode for which time-in-mode is counted. Uses same encodings as Z_CSR_ISSUE_MODE.iss_mode: 000: reserved 001: static tradoff 010: static rd priority 011: reserved 100: static wr priority 111-101: reserved

Table 5-96. Z_CSR_ISS_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
frm_type	3:0	RW_RW	0	Selects the frame type to be counted. 0000 - 3CMD - Count all 3-command Intel SMI frames 0001 - WDAT - Count all write data frames. 0010 - SYNC - Count all SYNC frames. 0011 - CHNL - Count all channel command frames. 0101 - 0100 - RSVD 1000 - NOP - Count all NOP frames. For post-silicon debug 1001-1011 - RSVD 1100 - 1CMD - Count all 1-command Intel SMI frames. 1101-1111 - RSVD

The **THR** subcontrol register contains bits to specify subevents for the THR_TT_TRP_UP/DN_EV events allowing a user to choose select DIMMs and whether the temperature is rising or falling.

Table 5-97. Z_CSR_PMU_MSC_THR Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
ig	31:11	RO_NA		Reads 0; writes ignored.
trp_pt_dn_cnd	10:9	RW_RW	0	Selects the condition to count for "downwards" trip point crossings. See Table 5-98 for encodings.
trp_pt_up_cnd	8:7	RW_RW	0	Selects the condition to count for "upwards" trip point crossings. See Table 5-98 for encodings.
dimm_trp_pt	6:4	RW_RW	0	Selects the DIMM for which to count the trip point crossings. Unused when all_dimms_trp_pt field is set.
all_dimms_trp_pt	3	RW_RW	0	Select all DIMMs to provide trip point crossings events instead of a single particular DIMM.
rsv	2:0	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.

Table 5-98. TRP_PT_{DN,UP}_CND Encodings

Name	Val	Description
ABOVE_TEMPMID_RISE	0b11	Above the mid temperature trip point (rising)
ABOVE_TEMPMID_FALL	0b10	Above the mid temperature trip point (falling)
ABOVE_TEMPLO	0b01	Above the low temperature trip point, but below the mid temperature trip point.
BELOW_TEMPLO	0b00	Below the low temperature trip point.

The **PGT** subcontrol register contains bits to specify subevents for ISS_CYC_SCHED_STATIC_EV (counts cycles within the specified scheduler mode) and PGT_PAGE_EV (op2cls or cls2opn transitions) as well as provide bits to further breakdown throttling events into ranks (for PGT_CNFLCT_EV).

Table 5-99. Z_CSR_PGT_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
ig	31:10	RO_NA	0	Reads 0; writes ignored.
trans_cmd_cnd	9:8	RW_RW	0	Selects the translated commands to be counted. 00 - ALL - Count all translated commands. 01 - RD - Count all translated read (read, preall, refresh, zqcal and auto-close) commands. 10 - WR - Count all translated write commands.
opncls_time	7	RW_RW	0	Selects time counting between open and closed page mode. 0 - CLS - Counts time spent in closed page mode. 1 - OPN - Counts time spent in open page mode.
rsv	6	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.
tt_rnk_cnd	5:2	RW_RW	0	Selects which rank is observed for thermal throttling events.
rnk_cnd	1	RW_RW	0	Selects how thermal throttling events are counted relative to rank. 0 - ALL - Counts thermal throttling events for all ranks. 1 - SGL - Counts thermal throttling events for the single rank selected by tt_rnk_cnd.
opn2cls_cnt	0	RW_RW	0	Counts the open/closed page policy transitions. 0 - OPN2CLS - Counts open-to-closed transactions. 1 - CLS2OPN - Counts closed-to-open transactions.

The **PLD** subcontrol register contains bits to specify subevents for PLD_DRAM_EV (by DRAM CMD type), PLD_RETRY_EV (to specify FVID).

Table 5-100. Z_CSR_PLD_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
ig	31:14	RO_NA		Reads 0; writes ignored.
addr_match1	13	RW_RW	0	Qualify trigger with address match as specified by Z_CSR_INJ_ERR_ADDR_1. Z_CSR_INJ_ERR_CTL_1.match_* and Z_CSR_INJ_ERR_CTL_1.inj_err_* fields control the match condition.
dram_cmd	12:8	RW_RW	0	The DRAM command type to be counted. 0x00 NOP 0x01 Precharge Single 0x02 RAS 0x04 CAS Read (no auto-precharge) (open page mode) 0x08 Refresh 0x09 Precharge All 0x0a Always matched 0x0c CAS read/precharge (closed page mode) 0x10 Write Trickle 0x11 SYNC 0x13 Clock Enable 0x14 CAS write (no auto-precharge) (open page mode) 0x15 Soft Reset 0x16 Write command register 0x17 Read command register 0x18 ZQCAL command 0x1c CAS write/precharge (closed page mode)

Table 5-100. Z_CSR_PLD_PMU Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
rtry_sngl_fvid	7	RW_RW	0	Controls FVID (Fill Victim Index) selection for which the number of retries is to be counted. 0 - ALL - All retries are counted, regardless of FVID 1 - FVID - Counts only the retries whose FVIDs match this CSR's fvid field.
fvid	6:1	RW_RW	0	The FVID for which the number of retries is to be counted.
cmd	0	RW_RW	0	Qualifies the DRAM commands counted by Z_CSR_ISS_PMU.sched_mode. Z_CSR_PLD_PMU.dram_cmd always needs to be matched and has no enable bit. 0 - ALL - Count all DRAM commands. 1 - SCHED - Count only the DRAM commands that come in while the ZAD_ISS section is in the scheduling mode selected by Z_CSR_ISS_PMU.sched_mode_pld_trig.

The **FVC** subcontrol register contains bits to break the FVC_EV into events observed by the Fill and Victim Control logic (i.e. BBOX commands, BBOX responses, various error conditions, etc). The FVC register can be set up to monitor four independent FVC-subevents simultaneously. However, many of the FVC-subevents depend on **additional** FVC fields which detail BBox response and commands. Therefore, only one BBox response or command may be monitored at any one time.

Table 5-101. Z_CSR_PMU_ZDP_CTL_FVC Register – Field Definitions

Field	Bits	Access	HW Reset Val	Reset Type
rsv	31:28	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.
evnt3	27:24	RW_RW	0	FVC subevent 3 selection. See Table 5-102, "Z_CSR_PMU_ZDP_CTL_FVC.evnt{3-0} Encodings"
evnt2	23:20	RW_RW	0	FVC subevent 2 selection. See Table 5-102, "Z_CSR_PMU_ZDP_CTL_FVC.evnt{3-0} Encodings"
evnt1	19:16	RW_RW	0	FVC subevent 1 selection. See Table 5-102, "Z_CSR_PMU_ZDP_CTL_FVC.evnt{3-0} Encodings"
evnt0	15:12	RW_RW	0	FVC subevent 0 selection. See Table 5-103, "Z_CSR_PMU_ZDP_CTL_FVC.RESP Encodings"
resp	11:9	RW_RW	0	Bbox response to match on. See Table 5-104, "Z_CSR_PMU_ZDP_CTL_FVC.BCMD Encodings"
bcmd	8:5	RW_RW	0	Bbox command to match on. See Table 5-104, "Z_CSR_PMU_ZDP_CTL_FVC.BCMD Encodings"
rsv	4:0	RW_RW	0	Reserved; Must write to 0 else behavior is undefined.

Table 5-102. Z_CSR_PMU_ZDP_CTL_FVC.evnt{3-0} Encodings (Sheet 1 of 2)

Name	Value	Description
ecc_corr_mapped_err	0b1110	Correctable Memory Data Check Bit Error with re-mapping (Corresponds to Z_CSR_ERR_LOG.ecc_corr_mapped_err)
ecc_corr_err	0b1101	Correctable Memory Data Check Bit Error without re-mapping (Corresponds to Z_CSR_ERR_LOG.ecc_corr_err)
fill_buf_corr_err	0b1100	Correctable Fill Buffer Data ECC Error (Corresponds to Z_CSR_ERR_LOG.fill_buf_corr_err)
status_frm_dm_err	0b1010	Status fFame Data Merge Error (Corresponds to Z_CSR_ERR_LOG.status_frm_dm_err)

Table 5-102. Z_CSR_PMU_ZDP_CTL_FVC.evnt{3-0} Encodings (Sheet 2 of 2)

Name	Value	Description
status_frm_par_err	0b1001	Status Frame Parity Error (Corresponds to Z_CSR_ERR_LOG.status_frm_par_err)
victim_buf_corr_err	0b1000	Victim Buffer Correctible Error (Corresponds to Z_CSR_ERR_LOG.victim_buf_corr_err)
smi_nb_trig	0b0111	Select Intel SMI Northbound debug event bits from the Intel SMI status frames as returned from Intel 7500 scalable memory buffers. These bits are denoted NBDE in the Intel SMI spec status frame description. An OR of all the bits over all the memory buffers is selected here as an event.
resp_match	0b0110	Use response match as programmed by Z_CSR_PMU_ZDP_CTL_FVC.resp to generate trigger.
bcmd_match	0b0101	Use Bbox command match as programmed by Z_CSR_PMU_ZDP_CTL_FVC.bcmd to generate trigger.
---	0b0100	
alrt_frm	0b0011	An alert frame was detected.
psn_txn	0b0010	The directory of a write to memory was encoded as poisoned.
mem_ecc_err	0b0001	Memory ECC error detected (that is not a link-level CRC error).
smi_crc_err	0b0000	Link level Intel SMI CRC error detected.

Table 5-103. Z_CSR_PMU_ZDP_CTL_FVC.RESP Encodings

Name	Value	Description
spr_uncor_resp	0b111	Uncorrectable response for command to misbehaving DIMM during sparing.
---	0b110	
spr_ack_resp	0b101	Positive acknowledgment for command to misbehaving DIMM during sparing. No error was detected for the transaction.
spec_ack_resp	0b100	Speculative (=early) positive acknowledgment for optimized read flow. No error was detected for the transaction.
uncor_resp	0b011	Uncorrectable response. Corrections failed.
corr_resp	0b010	Corrected (after, for example, error trials or just by a retry).
retry_resp	0b001	Retry response. Possibly a correctable error. Retries are generated until it is decided that error was either correctable or uncorrectable.
ack_resp	0b000	Positive acknowledgment. No was detected.

Table 5-104. Z_CSR_PMU_ZDP_CTL_FVC.BCMD Encodings (Sheet 1 of 2)

Name	Value	Description
megaop4	0b1111	MegaOp 4: V2V, then a write.
megaop3	0b1010	MegaOp 3: F2B, then a write.
megaop2	0b1100	MegaOp 2: F2V, then a write.
megaop1	0b1000	MegaOp 1: F2B, then a F2V, and then a write.
sprwr_bcmd	0b0111	Spare write.
f2b_bcmd	0b0110	Fill buffer read to Bbox.
f2v_bcmd	0b0101	Fill buffer to victim buffer transfer.
v2v_bcmd	0b0100	Victim buffer to victim buffer transfer.
v2f_bcmd	0b0011	Victim buffer to fill buffer transfer.

Table 5-104. Z_CSR_PMU_ZDP_CTL_FVC.BCMD Encodings (Sheet 2 of 2)

Name	Value	Description
mrg_bcmd	0b0010	Merge command from Bbox.
wr_bcmd	0b0001	Memory write command from Bbox.
rd_bcmd	0b0000	Memory read command from Bbox.

5.9.5 Zbox Performance Monitoring Events

5.9.5.1 An Overview:

The Zbox performance monitors can collect events in many of the substructures found within the Zbox including the DSP, ISS, THR, PGT, PLD and FVC (refer to Section 5.9.4.4, “Zbox PMU Subcontrol Registers - Subunit descriptions” for more detail).

A sampling of events available for monitoring in the ZBox:

- **BBox commands** - reads, writes, fill2victim, merge, etc. Can be conditioned on fvid which allows determining average latency of ZBox and memory.
- **BBox responses**. Incrementing on read command and decrementing on read response allows one to determine the number of simultaneous reads in the ZBox. A max detector can log the max number of reads the ZBox received.
- **Translated commands**: ras_caspre, ras_cas, cas, ras_cas_pre, pre, and so forth (can be filtered on r/w)
- **Memory commands**: ras, cas, pre, prefesh, preall, etc.
- **Page hits and page misses.**
- Auto page closes.
- Open-page to closed-**page policy transitions**. As well as length of time spent in each policy.
- Starvation event in scheduler, starvation state and back-pressure to BBox.
- Thermal throttling

and many more.

5.9.6 ZBox Events Ordered By Code

Table 5-80 summarizes the directly-measured ZBox events.

Table 5-105. Performance Monitor Events for ZBox Events (Sheet 1 of 3)

Symbol Name	CNT_CTLx [9:4]	SubCtl Dep	Max Inc/Cyc	Description
DSPQ_CHANGES	0x00	DSP	1	Dispatch Queue Transitions
CYC_ZFULL	0x01		1	Zbox Full Cycles
RETRY_ZFULL	0x02		1	Retry ZFull
RETRY_STARVE	0x03		1	Retry Starve
THERM_TRP_UP	0x04	THR	1	DIMM ‘Up’ Thermal Trip Points Crossed
THERM_TRP_DN	0x05	THR	1	DIMM ‘Down’ Thermal Trip Points Crossed
REFRESH	0x06		1	Refresh Commands

Table 5-105. Performance Monitor Events for ZBox Events (Sheet 2 of 3)

Symbol Name	CNT_CTLx [9:4]	SubCtl Dep	Max Inc/Cyc	Description
SCHED_MODE_CHANGES	0x08		1	Scheduling Mode Changes
FRM_TYPE	0x09	ISS	1	ISS Related Events
DRAM_CMD	0x0a	PLD,ISS	1	DRAM Commands
RETRIES	0x0b	PLD	1	Retry Events
SMI_FAST_RESETS	0x0c		1	Fast Resets
FVC_EV0	0x0d	FVC	1	FVC Event 0
FVC_EV1	0x0e	FVC	1	FVC Event 1
FVC_EV2	0x0f	FVC	1	FVC Event 2
FVC_EV3	0x10	FVC	1	FVC Event 3
CYC_PGT_STATE	0x11	PGT	1	Time in Page Table State
TRANS_CMDS	0x12	PGT	1	Translated Commands
PAGE_MISS	0x13		1	Page Table Misses
PAGE_HITS	0x14		1	Page Table Hits
PAGE_AUTOCLS_CMD	0x15		1	Page Table Autoclose Commands
PGT_PAGE_EV	0x16	PGT	1	PGT Related Page Table Events
PAGE_COLLISION	0x18		1	Refresh Commands
TT_CMD_CONFLICT	0x19	PGT	1	Thermal Throttling Command Conflicts
BBOX_CMDS_ALL	0x1a		1	All Bbox Commands
CYCLES	0x1b		1	Zbox Cycles
SCHED_INFLIGHT_CMDS	0x1c		1	Scheduler In-flight Commands
INFLIGHT_CMDS	0x1d		1	In-flight Commands
CYCLES_DSP_FILL	0x20	DSP	1	Time in DSP_FILL state
CYCLES_THROTTLE	0x21			Throttled Cycles
RETRY_OPS	0x22		1	Retry Ops
CYCLES_RETRYQ_STARVED	0x23		1	Time RetryQ Starved
CYCLES_SCHED_MODE	0x24	PGT	1	Time in SCHED_MODE state
DSP_INSERT_ADDR_CNF	0x25		1	Dispatch Queue Insert w/Addr Conflicts
DSP_INSERT_ADDR	0x26		1	Dispatch Queue Insert w/Addr
DSP_INSERT_CNF	0x27		1	Dispatch Queue Insert Conflicts
DSP_INSERT	0x28		1	Dispatch Queue Insert
CKE_CYCLES_0	0x2b		1	CKE Active Rank 0
CKE_CYCLES_1	0x2c		1	CKE Active Rank 1
CKE_CYCLES_2	0x2d		1	CKE Active Rank 2
CKE_CYCLES_3	0x2e		1	CKE Active Rank 3
CKE_CYCLES_4	0x2f		1	CKE Active Rank 4
CKE_CYCLES_5	0x30		1	CKE Active Rank 5
CKE_CYCLES_6	0x31		1	CKE Active Rank 6
CKE_CYCLES_7	0x32		1	CKE Active Rank 7
CKE_ENTERED_0	0x33		1	CKE Entered Rank 0
CKE_ENTERED_1	0x34		1	CKE Entered Rank 1
CKE_ENTERED_2	0x35		1	CKE Entered Rank 2

Table 5-105. Performance Monitor Events for ZBox Events (Sheet 3 of 3)

Symbol Name	CNT_CTLx [9:4]	SubCtl Dep	Max Inc/Cyc	Description
CKE_ENTERED_3	0x36		1	CKE Entered Rank 3
CKE_ENTERED_4	0x37		1	CKE Entered Rank 4
CKE_ENTERED_5	0x38		1	CKE Entered Rank 5
CKE_ENTERED_6	0x39		1	CKE Entered Rank 6
CKE_ENTERED_7	0x3a		1	CKE Entered Rank 7
DSPQ_RD_CNT	0x3b		1	Dispatch Queue Read Count
FVID_FIFO_COUNT	0x3c		1	FVID FIFO Count
FVID_FIFO_WRITES	0x3d		1	FVID FIFO Writes
LIVE_OPS_INFLIGHT	0x3e		1	Live Ops In-flight
DSPQ_WR_CNT	0x3f		1	Dispatch Queue Write Count

5.9.7 Zbox Performance Monitor Event List

This section enumerates Itanium processor 9500 series uncore performance monitoring events for the Zbox.

BBOX_CMDS_ALL

- **Title:** All Bbox Commands
- **Category:** Zbox Commands Received
- **Event Code:** 0x1a, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter for all new commands detected from the Bbox to the Z Box.

CKE_LOW_CYCLES_R0

- **Title:** CKE Low Active For Rank 0
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 0

CKE_LOW_CYCLES_R1

- **Title:** CKE Low Active For Rank 1
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 1

CKE_LOW_CYCLES_R2

- **Title:** CKE Low Active For Rank 2
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 2

CKE_LOW_CYCLES_R3

- **Title:** CKE Low Active For Rank 3
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 3

CKE_LOW_CYCLES_R4

- **Title:** CKE Low Active For Rank 4
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 4

CKE_LOW_CYCLES_R5

- **Title:** CKE Low Active For Rank 5
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 5

CKE_LOW_CYCLES_R6

- **Title:** CKE Low Active For Rank 6
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 6

CKE_LOW_CYCLES_R7

- **Title:** CKE Low Active For Rank 7
- **Category:** CKE Power Down
- **Event Code:** 0x2b, **Max. Inc/Cyc:** 1,
- **Definition:** Cycles CKE Low power down active for rank/rank-pair 7

CKE_LOW_SENT_R0

- **Title:** CKE Low Sent to Rank 0
- **Category:** CKE Power down
- **Event Code:** 0x33, **Max. Inc/Cyc:** 1,
- **Definition:** Number of times CKE Low power down sent to rank/rank-pair 0

CYCLES

- **Title:** Zbox Cycles
- **Category:** Cycle Events
- **Event Code:** 0x1b, **Max. Inc/Cyc:** 1,
- **Definition:** Count Zbox cycles

CYCLES_DSP_FILL

- **Title:** Time in DSP_FILL State
- **Category:** Cycle Counters
- **Event Code:** 0x20, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter each cycle that the dispatch queue meets a certain condition.

Extension	DSP Bit[10]	Description
NEMPTY	0x0	Advance counter every cycle that the dispatch queue is not empty.
EMPTY	0x1	Advance counter every cycle that the dispatch queue is empty.

CYCLES_PGT_STATE

- **Title:** Time in Page Table State
- **Category:** Cycle Counters
- **Event Code:** 0x11, **Max. Inc/Cyc:** 1,
- **Definition:** Counts cycles the Page Table is in selected mode.

Extension	PGT Bit[7]	Description
CLS_PAGE_PLCY	0x0	Advance counter every state the page table is in close page mode.
OPN_PAGE_PLCY	0x1	Advance counter every state the page table is in open page mode.

CYCLES_RETRYQ_STARVED

- **Title:** Time RetryQ Starved
- **Category:** Cycle Events
- **Event Code:** 0x23, **Max. Inc/Cyc:** 1,
- **Definition:** Counts cycles RetryQ spends in the “badly starved” state.

CYCLES_SCHED_MODE

- **Title:** Time in SCHED_MODE state
- **Category:** Cycle Counters
- **Event Code:** 0x24, **Max. Inc/Cyc:** 32,
- **Definition:** Counts cycles spent in scheduling mode specified in M_CSR_PMU_ISS.sched_mode

Extension	PGT Bits[6:4]	Description
TRDOFF	0x1	Advance counter every state that the scheduler is in static TRDOFF mode.
RDPRIO	0x2	Advance counter every state that the scheduler is in static RDPRIO mode.
WRPRIO	0x4	Advance counter every state that the scheduler is in static WRPRIO mode.

CYCLES_THROTTLE

- **Title:** Throttled Cycles
- **Category:** Cycle Events
- **Event Code:** 0x21, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter for every state that the Zbox is in the platform throttle on state

CYCLES_ZFULL

- **Title:** Zbox Full Cycles
- **Category:** Cycle Events
- **Event Code:** 0x01, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when the "zfull" state is detected.

DRAM_CMD

- **Title:** DRAM Commands
- **Category:** DRAM Commands
- **Event Code:** 0x0a, **Max. Inc/Cyc:** 1,
- **Definition:** Count PLD Related DRAM Events
- **NOTE:** Tn order to measure a non-filtered version of the .DRAM_CMD events, it is necessary to make sure the PLD Dep bits [13,7,0] are also set to 0

Extension	PLD Dep Bits	ISS Dep Bits	Description
NOP	[12:8]0x0		Count NOP DRAM commands.
PRE	[12:8]0x1		Count Precharge Single DRAM commands.
RAS	[12:8]0x2		Count RAS DRAM commands.
CAS_RD_OPN	[12:80x4		Count CAS Read (no auto-precharge, open page mode) DRAM commands.
CAS_RD_OPN.TRDOFF	[0]0x1 && [12:8]0x4	[9:7]0x1	Count CAS Read (no auto-precharge, open page mode) DRAM commands during "static trade off" scheduling mode
CAS_RD_OPN.RDPRIO	[0]0x1 && [12:8]0x4	[9:7]0x2	Count CAS Read (no auto-precharge, open page mode) DRAM commands during "static read priority" scheduling mode
CAS_RD_OPN.WRPRIO	[0]0x1 && [12:8]0x4	[9:7]0x4	Count CAS Read (no auto-precharge, open page mode) DRAM commands during "static write priority" scheduling mode
RFR	[12:8]0x8		Count Refresh DRAM commands.
PREALL	[12:8]0x9		Count Precharge All DRAM commands.
ALL	[12:8]0xa		Advance counter when a DRAM command is detected.
ALL.TRDOFF	[0]0x1 && [12:8]0xa	[9:7]0x1	Count all DRAM commands during "static trade off" scheduling mode
ALL.RDPRIO	[0]0x1 && [12:8]0xa	[9:7]0x2	Count all DRAM commands during "static read priority" scheduling mode
ALL.WRPRIO	[0]0x1 && [12:8]0xa	[9:7]0x4	Count all DRAM commands during "static write priority" scheduling mode
CAS_RD_CLS	[12:8]0xc		Count CAS Read (precharge, closed page mode) DRAM commands.
CAS_RD_CLS.TRDOFF	[0]0x1 && [12:8]0xc	[9:7]0x1	Count CAS Read (precharge, closed page mode) DRAM commands during "static trade off" scheduling mode
CAS_RD_CLS.RDPRIO	[0]0x1 && [12:8]0xc	[9:7]0x2	Count CAS Read (precharge, closed page mode) DRAM commands during "static read priority" scheduling mode
CAS_RD_CLS.WRPRIO	[0]0x1 && [12:8]0xc	[9:7]0x4	Count CAS Read (precharge, closed page mode) DRAM commands during "static write priority" scheduling mode
TRKL	[12:8]0x10		Count Write Trickle DRAM commands.
SYNC	[12:8]0x11		Count SYNC DRAM commands.
CAS_WR_OPN	[12:8]0x14		Count CAS Write (no auto-precharge, open page mode) DRAM commands.
CAS_WR_OPN.TRDOFF	[0]0x1 && [12:8]0x14	[9:7]0x1	Count CAS Write (no auto-precharge, open page mode) DRAM commands during "static trade off" scheduling mode

Extension	PLD Dep Bits	ISS Dep Bits	Description
CAS_WR_OPN.RDPRIO	[0]0x1 && [12:8]0x14	[9:7]0x2	Count CAS Write (no auto-precharge, open page mode) DRAM commands during "static read priority" scheduling mode
CAS_WR_OPN.WRPRIO	[0]0x1 && [12:8]0x14	[9:7]0x4	Count CAS Write (no auto-precharge, open page mode) DRAM commands during "static write priority" scheduling mode
WR_CFG	[12:8]0x16		Count Write Command Register DRAM commands.
RD_CFG	[12:8]0x17		Count Read Command Register DRAM commands.
ZQCAL	[12:8]0x18		Count ZQCAL DRAM commands.
CAS_WR_CLS	[12:8]0x1c		Count CAS Write (precharge, closed page mode) DRAM commands.
CAS_WR_CLS.TRDOFF	[0]0x1 && [12:8]0x1c	[9:7]0x1	Count CAS Write (precharge, closed page mode) DRAM DRAM commands during "static trade off" scheduling mode
CAS_WR_CLS.RDPRIO	[0]0x1 && [12:8]0x1c	[9:7]0x2	Count CAS Write (precharge, closed page mode) DRAM DRAM commands during "static read priority" scheduling mode
CAS_WR_CLS.WRPRIO	[0]0x1 && [12:8]0x1c	[9:7]0x4	Count CAS Write (precharge, closed page mode) DRAM DRAM commands during "static write priority" scheduling mode

DSPQ_CHANGES

- **Title:** Dispatch Queue Transitions
- **Category:** Dispatch Queue
- **Event Code:** 0x00, **Max. Inc/Cyc:** 1,
- **Definition:** Counts DispatchQ transitions to specified state.

Extension	DSP Bit[10]	Description
NEMPTY	0x0	Counts number of times DispatchQ becomes non-empty
EMPTY	0x1	Counts number of times DispatchQ becomes empty

DSPQ_INSERT

- **Title:** Dispatch Queue Insert
- **Category:** Dispatch Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Command inserted into Dispatch Queue

DSPQ_INSERT_CNF

- **Title:** Dispatch Queue Insert Conflicts
- **Category:** Dispatch Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Command inserted into Dispatch Queue with conflict (same address as existing entry)

DSPQ_INSERT_ADDR

- **Title:** Dispatch Queue Insert w/Address
- **Category:** Dispatch Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Command inserted into Dispatch Queue. Specify address in Z_CSR_DSP_PMU.pgtAddr

DSPQ_INSERT_ADDR_CNF

- **Title:** Dispatch Queue Insert w/Address Conflicts
- **Category:** Dispatch Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Command inserted into Dispatch Queue with conflict (same address as existing entry). Specify address in Z_CSR_DSP_PMU.pgtAddr.

DSPQ_INSERT_CNF

- **Title:** Dispatch Queue Insert Conflicts
- **Category:** Dispatch Queue
- **Event Code:** 0x28, **Max. Inc/Cyc:** 1,
- **Definition:** Command inserted into Dispatch Queue with conflict (same address as existing entry)

DSPQ_RD_CNT

- **Title:** Dispatch Queue Read Count
- **Category:** Running Depth Counters
- **Event Code:** 0x3b, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter by the number of read commands in the dispatch queue. The amount can be zero to 32.

DSPQ_WR_CNT

- **Title:** Dispatch Queue Write Count
- **Category:** Running Depth Counters
- **Event Code:** 0x3f, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter by the number of write commands in the dispatch queue. The amount can be zero to 32.

FRM_TYPE

- **Title:** ISS Related Events
- **Category:** DRAM Commands
- **Event Code:** 0x09, **Max. Inc/Cyc:** 1,
- **Definition:** Count ISS Related Events

Extension	ISS[3:0] Bits	Description
3CMD	0x0	Counts 3CMD (3-command) Intel SMI frames
WDAT	0x1	Counts WDAT (Write Data) Intel SMI frames
SYNC	0x2	Counts SYNC Intel SMI frames
CHNL	0x3	Counts CHNL (channel) Intel SMI frames
---	0x7-0x4	(*illegal selection*)
NOP	0x8	Counts nop Intel SMI frames

Extension	ISS[3:0] Bits	Description
---	0xb-0x9	(*illegal selection*)
1CMD	0xc	Counts all 1CMD (1-command) Intel SMI frames
---	0xf-0xd	(*illegal selection*)

FVC_EV0

- **Title:** FVC Event 0
- **Category:** FVC Events
- **Event Code:** 0x0d, **Max. Inc/Cyc:** 1,
- **Definition:** Measure an FVC related event.
- **NOTE:** It is possible to program the FVC register such that up to 4 events from the FVC can be independently monitored. However, the bcmd_match and resp_match subevents depend on the setting of **additional** bits in the FVC register (11:9 and 8:5 respectively). Therefore, only **ONE** FVC_EVx.bcmd_match event may be monitored at any given time. The same holds true for VC_EVx.resp_match

Table 5-106. Unit Masks for FVC_EV0 (Sheet 1 of 2)

Extension	FVC [15:12]	FVC [11:9]	FVC [8:5]	Description
POISON_TXN	0x2			Count poison (directory of a write to memory was encoded as poisoned) transactions
ALERT_FRAMES	0x3			Counts alert frames
---	0x4			(*nothing will be counted*)
BBOX_CMDS.READS	0x5		0x0	Reads commands to z from B
BBOX_CMDS.WRITES	0x5		0x1	Write commands from B box to Z box
BBOX_CMDS.MERGE	0x5		0x2	Merge commands from B box to Z box
BBOX_CMDS.V2F	0x5		0x3	Victim buffer to Fill buffer transfer (V2F) command from B to Z
BBOX_CMDS.V2V	0x5		0x4	Victim buffer to Victim buffer transfer (V2V) command from B to Z
BBOX_CMDS.F2V	0x5		0x5	Fill buffer to Victim buffer transfer (F2V) command from B to Z
BBOX_CMDS.F2B	0x5		0x6	Fill buffer read to Bbox (F2B) from Z
BBOX_CMDS.SPRWR	0x5		0x7	spare write commands from b to z
BBOX_CMDS.MEGAOP1	0x5		0x8	MegaOp1 commands (F2B,F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP2	0x5		0xc	MegaOp2 commands (F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP3	0x5		0xa	MegaOp3 commands (F2B and then Write) from B TO Z
BBOX_CMDS.MEGAOP4	0x5		0xf	MegaOp4 commands (V2V and then Write) from B TO Z
BBOX_RSP.ACK	0x6	0x0		Counts positive acknowledgements. No error was detected.
BBOX_RSP.RETRY	0x6	0x1		Count Retry Responses. Possibly a correctable error. Retries are generated until it is decided that the error was either correctable or uncorrectable.
BBOX_RSP.COR	0x6	0x2		Counts corrected (for example, after error trials or just by a retry)
BBOX_RSP.UNCOR	0x6	0x3		Count Uncorrectable Responses.

Table 5-106. Unit Masks for FVC_EV0 (Sheet 2 of 2)

Extension	FVC [15:12]	FVC [11:9]	FVC [8:5]	Description
BBOX_RSP.SPEC_ACK	0x6	0x4		Speculative positive acknowledgement for optimized read flow. No error was detected for the transaction.
BBOX_RSP.SPR_ACK	0x6	0x5		Count positive acknowledgements for command to misbehaving DIMM during sparing. No error was detected for the transaction.
---	0x6	0x6		(*nothing will be counted*)
BBOX_RSP.SPR_UNCOR	0x6	0x7		Counts Uncorrectable responses to Bbox as a result of commands issued to misbehaving DIMM during sparing
SMI_NB_TRIG	0x7			Select Intel SMI Northbound debug event bits from Intel SMI status frames as returned from the Intel 7500 series memory buffers. Used for Debug purposes

FVC_EV1

- **Title:** FVC Event 1
- **Category:** FVC Events
- **Event Code:** 0x0e, **Max. Inc/Cyc:** 1,
- **Definition:** Measure an FVC related event.
- **NOTE:** It is possible to program the FVC register such that up to 4 events from the FVC can be independently monitored. However, the bcmd_match and resp_match subevents depend on the setting of **additional** bits in the FVC register (11:9 and 8:5 respectively). Therefore, only **ONE** FVC_EVx.bcmd_match event may be monitored at any given time. The same holds true for VC_EVx.resp_match

Table 5-107. Unit Masks for FVC_EV1 (Sheet 1 of 2)

Extension	FVC [19:16]	FVC [11:9]	FVC [8:5]	Description
POISON_TXN	0x2			Count poison (directory of a write to memory was encoded as poisoned) transactions
ALERT_FRAMES	0x3			Counts alert frames
---	0x4			(*nothing will be counted*)
BBOX_CMDS.READS	0x5		0x0	Reads commands to z from B
BBOX_CMDS.WRITES	0x5		0x1	Write commands from B box to Z box
BBOX_CMDS.MERGE	0x5		0x2	Merge commands from B box to Z box
BBOX_CMDS.V2F	0x5		0x3	Victim buffer to Fill buffer transfer (V2F) command from B to Z
BBOX_CMDS.V2V	0x5		0x4	Victim buffer to Victim buffer transfer (V2V) command from B to Z
BBOX_CMDS.F2V	0x5		0x5	Fill buffer to Victim buffer transfer (F2V) command from B to Z
BBOX_CMDS.F2B	0x5		0x6	Fill buffer read to Bbox (F2B) from Z
BBOX_CMDS.SPRWR	0x5		0x7	spare write commands from b to z
BBOX_CMDS.MEGAOP1	0x5		0x8	MegaOp1 commands (F2B,F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP2	0x5		0xc	MegaOp2 commands (F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP3	0x5		0xa	MegaOp3 commands (F2B and then Write) from B TO Z

Table 5-107. Unit Masks for FVC_EV1 (Sheet 2 of 2)

Extension	FVC [19:16]	FVC [11:9]	FVC [8:5]	Description
BBOX_CMDS.MEGAOP4	0x5		0xf	MegaOp4 commands (V2V and then Write) from B TO Z
BBOX_RSP.ACK	0x6	0x0		Counts positive acknowledgements. No error was detected.
BBOX_RSP.RETRY	0x6	0x1		Count Retry Responses. Possibly a correctable error. Retries are generated until it is decided that the error was either correctable or uncorrectable.
BBOX_RSP.COR	0x6	0x2		Counts corrected (for example, after error trials or just by a retry)
BBOX_RSP.UNCOR	0x6	0x3		Count Uncorrectable Responses.
BBOX_RSP.SPEC_ACK	0x6	0x4		Speculative positive acknowledgement for optimized read flow. No error was detected for the transaction.
BBOX_RSP.SPR_ACK	0x6	0x5		Count positive acknowledgements for command to misbehaving DIMM during sparing. No error was detected for the transaction.
---	0x6	0x6		(*nothing will be counted*)
BBOX_RSP.SPR_UNCOR	0x6	0x7		Counts Uncorrectable responses to Bbox as a result of commands issued to misbehaving DIMM during sparing
SMI_NB_TRIG	0x7			Select Intel SMI Northbound debug event bits from Intel SMI status frames as returned from the Intel 7500 scalable memory buffers. Used for Debug purposes

FVC_EV2

- **Title:** FVC Event 2
- **Category:** FVC Events
- **Event Code:** 0x0f, **Max. Inc/Cyc:** 1,
- **Definition:** Measure an FVC related event.
- **NOTE:** It is possible to program the FVC register such that up to 4 events from the FVC can be independently monitored. However, the bcmd_match and resp_match subevents depend on the setting of **additional** bits in the FVC register (11:9 and 8:5 respectively). Therefore, only **ONE** FVC_EVx.bcmd_match event may be monitored at any given time. The same holds true for VC_EVx.resp_match

Table 5-108. Unit Masks for FVC_EV2 (Sheet 1 of 2)

Extension	FVC [23:20]	FVC [11:9]	FVC [8:5]	Description
POISON_TXN	0x2			Count poison (directory of a write to memory was encoded as poisoned) transactions
ALERT_FRAMES	0x3			Counts alert frames
---	0x4			(*nothing will be counted*)
BBOX_CMDS.READS	0x5		0x0	Reads commands to z from B
BBOX_CMDS.WRITES	0x5		0x1	Write commands from B box to Z box
BBOX_CMDS.MERGE	0x5		0x2	Merge commands from B box to Z box
BBOX_CMDS.V2F	0x5		0x3	Victim buffer to Fill buffer transfer (V2F) command from B to Z
BBOX_CMDS.V2V	0x5		0x4	Victim buffer to Victim buffer transfer (V2V) command from B to Z

Table 5-108. Unit Masks for FVC_EV2 (Sheet 2 of 2)

Extension	FVC [23:20]	FVC [11:9]	FVC [8:5]	Description
BBOX_CMDS.F2V	0x5		0x5	Fill buffer to Victim buffer transfer (F2V) command from B to Z
BBOX_CMDS.F2B	0x5		0x6	Fill buffer read to Bbox (F2B) from Z
BBOX_CMDS.SPRWR	0x5		0x7	spare write commands from b to z
BBOX_CMDS.MEGAOP1	0x5		0x8	MegaOp1 commands (F2B,F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP2	0x5		0xc	MegaOp2 commands (F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP3	0x5		0xa	MegaOp3 commands (F2B and then Write) from B TO Z
BBOX_CMDS.MEGAOP4	0x5		0xf	MegaOp4 commands (V2V and then Write) from B TO Z
BBOX_RSP.ACK	0x6	0x0		Counts positive acknowledgements. No error was detected.
BBOX_RSP.RETRY	0x6	0x1		Count Retry Responses. Possibly a correctable error. Retries are generated until it is decided that the error was either correctable or uncorrectable.
BBOX_RSP.COR	0x6	0x2		Counts corrected (for example, after error trials or just by a retry)
BBOX_RSP.UNCOR	0x6	0x3		Count Uncorrectable Responses.
BBOX_RSP.SPEC_ACK	0x6	0x4		Speculative positive acknowledgement for optimized read flow. No error was detected for the transaction.
BBOX_RSP.SPR_ACK	0x6	0x5		Count positive acknowledgements for command to misbehaving DIMM during sparing. No error was detected for the transaction.
---	0x6	0x6		(*nothing will be counted*)
BBOX_RSP.SPR_UNCOR	0x6	0x7		Counts Uncorrectable responses to Bbox as a result of commands issued to misbehaving DIMM during sparing
SMI_NB_TRIG	0x7			Select Intel SMI Northbound debug event bits from Intel SMI status frames as returned from the Intel 7500 scalable memory buffers. Used for Debug purposes
VIC_BUF_CORR_ERR	0x8			Count Victim Buffer Correctible Error (Corresponds to Z_CSR_ERR_LOG.victim_buf_corr_err)
STATUS_FRM_PAR_ERR	0x9			Count Status Frame Parity Error (Corresponds to Z_CSR_ERR_LOG.status_frm_par_err)
STATUS_FRM_DM_ERR	0xa			Count Status Frame Data Merge Error (Corresponds to Z_CSR_ERR_LOG.status_frm_dm_err)
FILL_BUF_CORR_ERR	0xc			Count Correctable Fill Buffer Data ECC Error (Corresponds to Z_CSR_ERR_LOG.fill_buf_corr_err)
ECC_CORR_ERR	0xd			Count Correctable Memory Data Check Bit Error without re-mapping (Corresponds to Z_CSR_ERR_LOG.ecc_corr_err)
ECC_CORR_ERR_MAP	0xe			Count Correctable Memory Data Check Bit Error with re-mapping (Corresponds to Z_CSR_ERR_LOG.ecc_corr_mapped_err)

FVC_EV3

- **Title:** FVC Event 3
- **Category:** FVC Events
- **Event Code:** 0x10, **Max. Inc/Cyc:** 1,
- **Definition:** Measure an FVC related event.
- **NOTE:** It is possible to program the FVC register such that up to 4 events from the FVC can be independently monitored. However, the bcmd_match and resp_match subevents depend on the setting of **additional** bits in the FVC register (11:9 and 8:5 respectively). Therefore, only **ONE** FVC_EVx.bcmd_match event may be monitored at any given time. The same holds true for VC_EVx.resp_match

Table 5-109. Unit Masks for FVC_EV3 (Sheet 1 of 2)

Extension	FVC [27:24]	FVC [11:9]	FVC [8:5]	Description
POISON_TXN	0x2			Count poison (directory of a write to memory was encoded as poisoned) transactions
ALERT_FRAMES	0x3			Counts alert frames
---	0x4			(*nothing will be counted*)
BBOX_CMDS.READS	0x5		0x0	Reads commands to z from B
BBOX_CMDS.WRITES	0x5		0x1	Write commands from B box to Z box
BBOX_CMDS.MERGE	0x5		0x2	Merge commands from B box to Z box
BBOX_CMDS.V2F	0x5		0x3	Victim buffer to Fill buffer transfer (V2F) command from B to Z
BBOX_CMDS.V2V	0x5		0x4	Victim buffer to Victim buffer transfer (V2V) command from B to Z
BBOX_CMDS.F2V	0x5		0x5	Fill buffer to Victim buffer transfer (F2V) command from B to Z
BBOX_CMDS.F2B	0x5		0x6	Fill buffer read to Bbox (F2B) from Z
BBOX_CMDS.SPRWR	0x5		0x7	spare write commands from b to z
BBOX_CMDS.MEGAOP1	0x5		0x8	MegaOp1 commands (F2B,F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP2	0x5		0xc	MegaOp2 commands (F2V and then Write) from B TO Z
BBOX_CMDS.MEGAOP3	0x5		0xa	MegaOp3 commands (F2B and then Write) from B TO Z
BBOX_CMDS.MEGAOP4	0x5		0xf	MegaOp4 commands (V2V and then Write) from B TO Z
BBOX_RSP.ACK	0x6	0x0		Counts positive acknowledgements. No error was detected.
BBOX_RSP.RETRY	0x6	0x1		Count Retry Responses. Possibly a correctable error. Retries are generated until it is decided that the error was either correctable or uncorrectable.
BBOX_RSP.COR	0x6	0x2		Counts corrected (for example, after error trials or just by a retry)
BBOX_RSP.UNCOR	0x6	0x3		Count Uncorrectable Responses.
BBOX_RSP.SPEC_ACK	0x6	0x4		Speculative positive acknowledgement for optimized read flow. No error was detected for the transaction.
BBOX_RSP.SPR_ACK	0x6	0x5		Count positive acknowledgements for command to misbehaving DIMM during sparing. No error was detected for the transaction.
---	0x6	0x6		(*nothing will be counted*)

Table 5-109. Unit Masks for FVC_EV3 (Sheet 2 of 2)

Extension	FVC [27:24]	FVC [11:9]	FVC [8:5]	Description
BBOX_RSP.SPR_UNCOR	0x6	0x7		Counts Uncorrectable responses to Bbox as a result of commands issued to misbehaving DIMM during sparing
SMI_NB_TRIG	0x7			Select Intel SMI Northbound debug event bits from Intel SMI status frames as returned from the Intel 7500 scalable memory buffers. Used for Debug purposes

FVID_FIFO_COUNT

- **Title:** FVID FIFO Count
- **Category:** Running Depth Counters
- **Event Code:** 0x3c, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter by the number of commands in the FVID FIFO. The amount can be zero to 32.

FVID_FIFO_WRITES

- **Title:** FVID FIFO Writes
- **Category:** Running Depth Counters
- **Event Code:** 0x3d, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter by the number of writes to the FVID FIFO. The amount can be zero, one or two.

INFLIGHT_CMD

- **Title:** In-flight Commands
- **Category:** Zbox Commands Received
- **Event Code:** 0x1d, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a new memory controller (read and write type) command is accepted

LIVE_OPS_INFLIGHT

- **Title:** Live Ops In-flight
- **Category:** Running Depth
- **Event Code:** 0x3e, **Max. Inc/Cyc:** 32,
- **Definition:** Advance counter by the number of pending commands in the Zbox. The amount can be zero to 32.

PAGE_AUTOCLS_CMD

- **Title:** Page Table Autoclose Commands
- **Category:** Page Table
- **Event Code:** 0x15, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when an auto page close command is detected.

PAGE_COLLISION

- **Title:** Refresh Commands
- **Category:** Page Table
- **Event Code:** 0x18, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when page collision is detected (that is, a command requires a PRE-RAS-CAS sequence).

PAGE_HITS

- **Title:** Page Table Hits
- **Category:** Page Table
- **Event Code:** 0x14, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a page hit is detected (that is, a command requires only a CAS).
- **NOTE:** Will not increment in closed-page mode.

PAGE_MISS

- **Title:** Page Table Misses
- **Category:** Page Table
- **Event Code:** 0x13, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a page miss is detected (that is, a command requires a RAS-CAS to complete).

PGT_PAGE_EV

- **Title:** PGT Related Page Table Events
- **Category:** Page Table
- **Event Code:** 0x16, **Max. Inc/Cyc:** 1,
- **Definition:** Counts PGT Related Page Table Events.

Extension	PGT Bits[0]	Description
OPN2CLS	0x1	Advance counter when an open-to-closed page transition is detected.
CLS2OPN	0x0	Advance counter when an closed-to-open page transition is detected.

REFRESH

- **Title:** Refresh Commands
- **Category:** DRAM Commands
- **Event Code:** 0x06, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a refresh command is detected.

RETRIES

- **Title:** Retry Events
- **Category:** Retry Events
- **Event Code:** 0x0b, **Max. Inc/Cyc:** 1,
- **Definition:** Count PLD Related Retry Events

Extension	PLD[7]	Description
ALL	b0	Advance counter when a retry is detected.
FVID	b1	Advance counter when a retry to a certain FVID is detected. ***Note:*** The FVID is programmed into PLD[6:1]

RETRY_OPS

- **Title:** Retry Ops
- **Category:** RetryQ
- **Event Code:** 0x22, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter for every new command that gets pushed into the retry queue.

RETRY_STARVE

- **Title:** Retry Starve
- **Category:** Retry Events
- **Event Code:** 0x03, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter every cycle when a retry is detected in the "starved" state.

RETRY_ZFULL

- **Title:** Retry ZFull
- **Category:** Retry Events
- **Event Code:** 0x02, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a retry is detected in the "zfull" state.

SCHDLR_INFLIGHT_CMDS

- **Title:** Scheduler In-flight Commands
- **Category:** Zbox Commands Received
- **Event Code:** 0x1c, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when a new scheduler command is accepted.

SCHED_MODE_CHANGES

- **Title:** Scheduling Mode Changes
- **Category:** DRAM Commands
- **Event Code:** 0x08, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when an ISS scheduling mode transition is detected.

SMI_FAST_RESETS

- **Title:** Fast Resets
- **Category:** DRAM Commands
- **Event Code:** 0x0c, **Max. Inc/Cyc:** 1,
- **Definition:** Advance counter when an Intel SMI fast reset occurs.

THERM_TRP_DN

- **Title:** DIMM ‘Down’ Thermal Trip Points Crossed
- **Category:** Thermal Throttle
- **Event Code:** 0x05, **Max. Inc/Cyc:** 1,
- **Definition:** Counts when a specified thermal trip point is crossed in the “down” direction.

Extension	THR Bits [10:9],[3]	Description
ALL.GT_MID_RISE	0x3,0x1	Advance the counter when the above mid temp thermal trip point (rising) is crossed in the "down" direction for any DIMM
ALL.GT_MID_FALL	0x2,0x1	Advance the counter when the above mid temp thermal trip point (falling) is crossed in the "down" direction for any DIMM

Extension	THR Bits [10:9],[3]	Description
ALL.GT_LO	0x1,0x1	Advance the counter when the above low temp, but below mid temp thermal trip point is crossed in the "down" direction for any DIMM.
ALL.LT_LO	0x0,0x1	Advance the counter when the below low temp thermal trip point is crossed in the "down" direction for any DIMM
DIMM{n}.GT_MID_RISE	0x3,0x0	Advance the counter when the above mid temp thermal trip point (rising) is crossed in the "down" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #
DIMM{n}.GT_MID_FALL	0x2,0x0	Advance the counter when the above mid temp thermal trip point (falling) is crossed in the "down" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #
DIMM{n}.GT_LO	0x1,0x0	Advance the counter when the above low temp, but below mid temp thermal trip point is crossed in the "down" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #
DIMM{n}.LT_LO	0x0,0x0	Advance the counter when the below low temp, but below mid temp thermal trip point is crossed in the "down" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #

THERM_TRP_UP

- **Title:** DIMM 'Up' Thermal Trip Points Crossed
- **Category:** Thermal Throttle
- **Event Code:** 0x04, **Max. Inc/Cyc:** 1,
- **Definition:** Counts when a specified thermal trip point is crossed in the "up" direction.

Extension	THR Bits [10:9],[3]	Description
ALL.GT_MID_RISE	0x3,0x1	Advance the counter when the above mid temp thermal trip point (rising) is crossed in the "up" direction for any DIMM
ALL.GT_MID_FALL	0x2,0x1	Advance the counter when the above mid temp thermal trip point (falling) is crossed in the "up" direction for any DIMM
ALL.GT_LO	0x1,0x1	Advance the counter when the above low temp, but below mid temp thermal trip point is crossed in the "up" direction for any DIMM.
ALL.LT_LO	0x0,0x1	Advance the counter when the below low temp thermal trip point is crossed in the "up" direction for any DIMM
DIMM{n}.GT_MID_RISE	0x3,0x0	Advance the counter when the above mid temp thermal trip point (rising) is crossed in the "up" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #
DIMM{n}.GT_MID_FALL	0x2,0x0	Advance the counter when the above mid temp thermal trip point (falling) is crossed in the "up" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #

Extension	THR Bits [10:9],[3]	Description
DIMM{n}.GT_LO	0x1,0x0	Advance the counter when the above low temp, but below mid temp thermal trip point is crossed in the "up" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #
DIMM{n}.LT_LO	0x0,0x0	Advance the counter when the below low temp, but below mid temp thermal trip point is crossed in the "up" direction for DIMM #? ***Note:*** THR Bits [6:4] must be programmed with the DIMM #

TRANS_CMDS

- **Title:** Translated Commands
- **Category:** Dispatch Queue
- **Event Code:** 0x12, **Max. Inc/Cyc:** 1,
- **Definition:** Counts read/write commands entered into the Dispatch Queue

Extension	PGT Bits[9:8]	Description
---	0x3	(*nothing is counted*)
INSERTS_WR	0x2	Advance counter when a translated command enters any Dispatch Queue.
INSERTS_RD	0x1	Advance counter when a translated read (includes read, preall, refresh, zqcal and auto-close) command enters any Dispatch Queue.
INSERTS_ALL	0x0	Advance counter when a translated command enters any Dispatch Queue.

TT_CMD_CONFLICT

- **Title:** Thermal Throttling Command Conflicts
- **Category:** Thermal Throttle
- **Event Code:** 0x19, **Max. Inc/Cyc:** 1,
- **Definition:** Counts command conflicts due to thermal throttling.

Extension	PGT[1] Bits	Description
TT_RANK_ALL	0x0	Advance counter when a command conflict occurs due to thermal throttling.
TT_RANK_N	0x1	Advance counter when a ranked command conflict occurs due to thermal throttling. ***Note:*** Rank can be specified by the user in PGT Bits[5:2]

5.10 Packet Matching Reference

In the B and R box, the performance monitoring infrastructure allows a user to filter packet traffic according to certain fields. A couple common fields, the Message Class/ Opcode fields, have been summarized in the following tables.

Table 5-110. Intel QuickPath Interconnect Packet Message Classes (Sheet 1 of 2)

Code	Name	Definition
b0000	HOM0	Home - Requests

Table 5-110. Intel QuickPath Interconnect Packet Message Classes (Sheet 2 of 2)

Code	Name	Definition
b0001	HOM1	Home - Responses
b0010	NDR	Non-Data Responses
b0011	SNP	Snoops
b0100	NCS	Non-Coherent Standard
---		
b1100	NCB	Non-Coherent Bypass
---		
b1110	DRS	Data Response

Table 5-111. Opcode Match by Message Class (Sheet 1 of 2)

Opc	HOM0	HOM1	SNP	DRS
0000	RdCur	RspI	SnpCur	DataC_(EIMS)
0001	RdCode	RspS	SnpCode	DataC_(EIMS)_FrcAckCnflt
0010	RdData	---	SnpData	DataC_(EIMS)_Cmp
0011	---	---	---	DataNc
0100	RdInvOwn	RspCnflt	SnpInvOwn	WbIData
0101	InvXtoI	---	SnpInvXtoI	WbSData
0110	EvctCln	---	---	WbEData
0111	---	---	---	---
1000	InvItoE	RspFwd	SnpInvItoE	WbIDataPtl
1001	---	RspFwdI	---	---
1010	---	RspFwdS	---	WbEDataPtl
1011	---	RspFwdIWb	---	---
1100	WbMtoI	RspFwdSWb	---	---
1101	WbMtoE	RspIWb	---	---
1110	WbMtoS	RspSWb	---	---
1111	AckCnflt	---	PrefetchHint	---
Opc	**NDR**	**NCB**	**NCS**	
0000	Gnt_Cmp	NcWr	NcRd	
0001	Gnt_FrcAckCnflt	WcWr	---	
0010	---	---	---	
0011	---	---	---	
0100	CmpD	---	NcRdPtl	
0101	AbortTO	---	NcCfgRd	
0110	---	---	---	
0111	---	---	NcIORd	
1000	Cmp	NcMsgB	---	
1001	FrcAckCnlft	PurgeTC	NcCfgWr	
1010	Cmp_FwdCode	IntPhysical	---	
1011	Cmp_FwdInvOwn	---	NcIOWr	
1100	Cmp_FwdInvItoE	NcWrPtl	---	
1101	---	WcWrPtl	---	

Table 5-111. Opcode Match by Message Class (Sheet 2 of 2)

Opc	HOM0	HOM1	SNP	DRS
1110	---	---	---	
1111	---	DebugData	---	

Table 5-112. Opcodes (Alphabetical Listing) (Sheet 1 of 3)

Name	Opc	MC	Rx/Tx By?	Desc
AbortTO	0101	NDR	Sr	Abort Time-out Response
AckCnflt	1111	HOM0	Br, St	Acknowledge receipt of Data_* and Cmp/FrcAckCnflt, signal a possible conflict scenario.
Cmp	1000	NDR	Brt, Sr, Urt	All snoop responses gathered, no conflicts
CmpD	0100	NDR	Sr	Completion with Data
Cmp_FwdCode	1010	NDR	Brt, Sr	Complete request, forward the line in F (or S) state to the requestor specified, invalidate local copy or leave it in S state.
Cmp_FwdInvItoE	1100	NDR	Brt, Sr	Complete request, invalidate local copy
Cmp_FwdInvOwn	1011	NDR	Brt, Sr	Complete request, forward the line in E or M state to the requestor specified, invalidate local copy
DataC_(FEIMS)	0000	DRS	Bt, Srt	Data Response in (FEIMS) state ***Note:*** Set RDS field to specify which state is to be measured. Cr for F state, Ct for I state.
DataC_(FEIMS)_Cmp	0010	DRS	Bt, Sr	Data Response in (EIS) state, Complete ***Note:*** Set RDS field to specify which state is to be measured.
DataC_(FEIMS)_FrcAck Cnflt	0001	DRS	Sr	Data Response in (EIS) state, Force Acknowledge ***Note:*** Set RDS field to specify which state is to be measured.
DataNc	0011	DRS	Bt, Sr, Ut	Non-Coherent Data
DebugData	1111	NCB	St	Debug Data.
EvctCln	0110	HOM0	Br, St	Clean cache line eviction notification to home agent.
FrcAckCnlft	1001	NDR	Sr	All snoop responses gathered, force an AckCnflt
Gnt_Cmp	0000	NDR	Brt, Sr	Signal completion and Grant E state ownership without data to an InvItoE or ‘null data’ to an InvXtoI
Gnt_FrcAckCnflt	0001	NDR	Sr	Signal FrcAckCnflt and Grant E state ownership without data to an InvItoE or ‘null data’ to an InvXtoI
PurgeTC	1001	NCB	Urt	Purge target’s page table caches
IntAck	1001	NCS	St	
IntPhysical	1010	NCB	St, Ur	Physical mode interrupt to processor
InvItoE	1000	HOM0	Br, St	Invalidate to E state requests exclusive ownership of a cache line without data.
InvXtoI	0101	HOM0	Br, St	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).
NcCfgRd	0101	NCS	St, Ur	Configuration read from configuration space
NcCfgWr	1001	NCS	St, Ur	Configuration write to configuration space
NcIORd	0111	NCS	St	Read from legacy I/O space
NcIOWr	1011	NCS	St	Write to legacy I/O space
NcMsgB	1000	NCB	St	Non-coherent Message (non-coherent bypass channel)

Table 5-112. Opcodes (Alphabetical Listing) (Sheet 2 of 3)

Name	Opc	MC	Rx/Tx By?	Desc
NcRd	0000	NCS	B*[1], St, Ur	Read from non-coherent memory mapped I/O space
NcRdPtl	0100	NCS	B*[1], St, Ur	Partial read from non-coherent memory mapped I/O space * Home Agent acting as mirroring slave receives, mirroring primary will transmit
NcWr	0000	NCB	B*[1], Ur	Write to non-coherent memory mapped I/O space * Home Agent acting as mirroring slave receives, mirroring primary will transmit
NcWrPtl	1100	NCB	B*[1], St, Ur	Partial write to non-coherent memory mapped I/O space * Home Agent acting as mirroring slave receives, mirroring primary will transmit
PrefetchHint	1111	SNP	Sr	
PurgeTC	1001	SNP	St	
RdCode	0001	HOM0	Br, St	Read cache line in F (or S, if the F state not supported)
RdCur	0000	HOM0	Br	Request a cache line in I. Typically issued by I/O proxy entities, RdCur is used to obtain a coherent snapshot of an uncached cache line.
RdData	0010	HOM0	Br, St	Read cache line in either E or F (or S, if F state not supported). The choice between F (or S) and E is determined by whether or not per caching agent has cache line in S state.
RdInvOwn	0100	HOM0	Br, St	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	0100	HOM1	Br, St	Peer is left with line in I or S state, and the peer has a conflicting outstanding request.
RspFwd	1000	HOM1	Br, St	Peer has sent data to requestor with no change in cache state
RspFwdI	1001	HOM1	Br, St	Peer has sent data to requestor and is left with line in I state
RspFwdIWb	1011	HOM1	Br, St	Peer has sent data to requestor and a WbIData to the home, and is left with line in I state
RspFwdS	1010	HOM1	Br, St	Peer has sent data to requestor and is left with line in S state
RspFwdSWb	1100	HOM1	Br, St	Peer has sent data to requestor and a WbSData to the home, and is left with line in S state
RspI	0000	HOM1	Br, St	Peer left with line in I-state
RspIWb	1101	HOM1	Br, St	Peer has evicted the data with an in-flight WbIData[Ptl] message to the home and has not sent any message to the requestor.
RspS	0001	HOM1	Br, St	Peer left with line in S-state
RspSWb	1110	HOM1	Br, St	Peer has sent a WbSData message to the home, has not sent any message to the requestor and is left with line in S-state
SnpCode	0001	SNP	Bt, Sr	Snoop Code (get data in F or S state)
SnpCur	0000	SNP	Bt, Sr	Snoop to get data in I state
SnpData	0010	SNP	Bt, Sr	Snoop Data (get data in F or S state)
SnpInvItoE	1000	SNP	Bt, Sr	Snoop Invalidate to E state. To invalidate peer caching agent, flushing any M state data to home
SnpInvOwn	0100	SNP	Bt, Sr	Snoop Invalidate Own (get data in E or M state)

Table 5-112. Opcodes (Alphabetical Listing) (Sheet 3 of 3)

Name	Opc	MC	Rx/Tx By?	Desc
SnpInvXtoI	0101	SNP	Bt, Sr	Snoop Invalidate Writeback M to I state. To invalidate peer caching agent, flushing any M state data to home.
WbEData	0110	DRS	Br	Writeback data, downgrade to E state
WbEDataPtl	1010	DRS	Br	Partial (byte-masked) writeback data, downgrade to E state
WbIData	0100	DRS	Br, St	Writeback data, downgrade to I state
WbIDataPtl	1000	DRS	Br, St	Partial (byte-masked) writeback data, downgrade to I state
WbMtoI	1100	HOM0	Br, St	Write a cache line in M state back to memory and transition its state to I.
WbMtoE	1101	HOM0	Br	Write a cache line in M state back to memory and transition its state to E.
WbMtoS	1110	HOM0	Br	Write a cache line in M state back to memory and transition its state to S.
WbSData	0101	DRS	Br, St	Writeback data, downgrade to S state
WcWr	0001	NCB	St, Ur	Write combinable write to non-coherent memory mapped I/O space
WcWrPtl	1101	NCB	St, Ur	Partial write combinable write to non-coherent memory mapped I/O space

[1] Only with memory mirroring enabled.

The ‘Rx/Tx By?’ column denotes which agents transmit (Tx) and receive (Rx) each opcode. ‘U’ refers to the UBox as the Configuration Agent. ‘B’ refers to the BBox as the Home Agent. And the ‘S’ refers to the Sbox which acts as the Caching Agent with the Cbox.

§

A Identifying Multi-Core and Multi-Threading

The dual-core Intel® Itanium® processors, the Intel® Itanium® processor 9300 series, and the Intel® Itanium® processor 9500 series support multi-threading and multi-core technologies. This chapter covers common programming questions related to these technologies. The rest of the chapter is organized into two parts: the first part describes architectural support for system software to detect multi-threading and multi-core technologies as well as cache sharing information by the logical processors, the second part highlights operating system-specific mechanisms to return such information to applications.

A.1 Architectural Support

A.1.1 Terminology

Physical Processor / Physical Processor Package – A physical processor or physical processor package can contain one or more logical processors, organized into threads and cores.

Logical Processor – A logical processor is a compute-capability-centric view of the CPU that allows the physical processor package to execute from more than one instruction stream. A physical processor package that can execute from n instruction streams has n logical processors.

Threads – Threads are logical processors that share core pipeline execution resources.

Cores – Cores are defined as a collection of hardware that implements the main execution pipeline of the processor. Multiple cores on a physical processor package do not share core pipeline resources but may share caches and bus interfaces. A core may support multiple threads of execution.

A.1.2 Detection of Intel® Hyper-Threading Technology

The PAL procedure PAL_LOGICAL_TO_PHYSICAL can be used to detect the availability of multi-threading. This PAL procedure is supported on all processor implementations that contain more than one logical processor in a physical processor package. The tpc (threads per core) field in log_overview return value indicates whether multi-threading is available on the physical processor package the procedure called was made.

Please refer to "Processor Abstraction Layer" chapter in Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_LOGICAL_TO_PHYSICAL procedure.

A.1.3 Number of Cores on a Physical Processor

The PAL procedure PAL_LOGICAL_TO_PHYSICAL returns information on logical to physical processor mappings. When this procedure is called on a logical processor, the field cpp (cores per processor) in the return value log_overview indicates the number of cores of the physical processor on which the logical processor belongs to.

Please refer to "Processor Abstraction Layer" chapter in Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_LOGICAL_TO_PHYSICAL procedure.

A.1.4 Number of Threads in a Core

The PAL procedure PAL_LOGICAL_TO_PHYSICAL returns information on logical to physical processor mappings. When this procedure is called on a logical processor, the field tpc (threads per core) in the return value log_overview indicates the number of threads in the core on which the logical processor belongs to.

Please refer to "Processor Abstraction Layer" chapter in Volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_LOGICAL_TO_PHYSICAL procedure.

A.1.5 Number of Logical Processors Enabled on a Physical Processor

The PAL procedure PAL_LOGICAL_TO_PHYSICAL returns information on logical to physical processor mappings. When this procedure is called on a logical processor, the field num_log in the return value log_overview indicates the number of logical processors enabled on the physical processor.

Please refer to "Processor Abstraction Layer" chapter in volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_LOGICAL_TO_PHYSICAL procedure.

A.1.6 Logical to Physical Translation

The PAL procedure PAL_LOGICAL_TO_PHYSICAL returns information on logical to physical processor mappings. When PAL_LOGICAL_TO_PHYSICAL procedure is called on a logical processor, the procedure returns the unique tid (thread ID), cid (core ID) and ppid (physical processor package ID) values corresponding of the logical processor.

Please refer to "Processor Abstraction Layer" chapter in volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_LOGICAL_TO_PHYSICAL procedure.

A.1.7 Number of Logical Processors Sharing a Cache

The PAL procedure PAL_CACHE_SHARED_INFO provides information on the number of logical processors sharing a certain cache level. The num_shared return value indicates the number of logical processors that share the specified processor cache level.

Please refer to "Processor Abstraction Layer" chapter in volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_CACHE_SHARED_INFO procedure.

A.1.8 Determine which Logical Processors are Sharing a Cache

The steps below can be used to identify the logical processors sharing a certain cache level within a physical processor package:

1. Call PAL_CACHE_SHARED_INFO to determine the number of threads that shares the specific cache level. The return value num_shared indicates the number of logical processors sharing the specified cache level.
2. For the specific cache level, call PAL_CACHE_SHARED_INFO with different proc_number parameter in a loop and record the tid (thread ID), cid (core ID) and la (Logical address) of each logical processor sharing that cache level.

Repeat the above steps as needed for other cache levels.

Parse the thread and core information recorded from the steps above to identify the logical processors sharing the same core. Caches can be shared at core level by multiple threads or at physical package level by multiple cores.

Please refer to "Processor Abstraction Layer" chapter in volume 2 of the *Intel® Itanium® Architecture Software Developer's Manual* for definition and usage of PAL_CACHE_SHARED_INFO procedure.

A.2 Operating System Specific Mechanisms

The following sections highlight multi-core and multi-threading support available on HP-UX*, Linux*, and Microsoft Windows*. For up-to-date information and operating systems not listed in this section, please refer to documentations from corresponding operating system vendor.

A.2.1 HP-UX*

Please refer to the following documents for design considerations, system calls and code samples on multi-core and multi-threading programming for the HP-UX operating system:

- "Dynamic logical processors for Hyper-Threading on HP-UX 11i v3
- "HP-UX Reference: Section 2: System Calls

Both documents are available at http://www.hp.com.

A.2.2 Linux*

Older versions of Linux provide topology information only through /proc/cpuinfo. Each online logical processor in the system is listed in turn with entries describing various attributes of that logical processor. Useful items are:

- "physical id" – This number will be the same for logical processors that are part of the same physical processor package.
- "siblings" – Total number of logical processors that share a physical processor package.
- "core id" – Identifies to which core on a package this logical cpu belongs.
- "thread id" – Identifies which thread on a core.

Versions of Linux newer than 2.6.15 provide more information through a number of files in /sys/devices/system/cpu/cpuN/topology/:

- "physical_package_id" – same as "physical id" in /proc/cpuinfo.
- "core_id" – same as "core id" in /proc/cpuinfo.
- "core_siblings" – Hexadecimal representation of a bitmask showing which logical cpus share a physical package. For example, 8881 means that logical cpu numbers 0, 7, 11 and 15 share a package.
- "thread_siblings" – Bitmask showing which logical cpus are threads sharing the same core. For example, 0801 means cpus 0 and 11.

Linux versions newer than 2.6.17 provide cache topology information in /sys/devices/system/cpu/cpuN/cache/indexM/*. For each cache level and type the following informational files are provided:

- "level", "type" – Cache level (1-3), and type (Instruction, Data or Unified).
- "shared_cpu_map" – bit mask of logical cpus that share this cache.

A.2.3 Microsoft Windows*

Windows-based applications can obtain processor topology and cache sharing information through GetLogicalProcessorInformation API. Please refer to Microsoft Developer Network (MSDN) for details.

§

B Example Core PMU Event Reports

B.1 Introduction

This appendix gives a number of examples of possible high level reports based on PMU event measurements. These reports are provided here both because of the usefulness of these reports and for the purpose of improving the understanding of what various PMU events actually measure. Example data is also provided to aid in understanding what these reports mean, but the data in different reports may be from different data sets.

All of the reports use the following formatting and naming conventions:

- Each report in this appendix is communicated via a pair of tables. The first table attempts to visually communicate meaning through variable names, indention, and sample data. The second table shows how each variable in the first table was calculated from measured PMU event data.
- Indention of a variable in a report implies that it is a sub-component of the variable(s) under which it is indented.
- The variable names (at least the part before any ".") attempt to be informative, but it is the calculation of the variable that defines it.
- Variables with "." in the name are calculated from measured events and possibly other calculated variables.
- Calculations sometimes take multiple lines in the tables. This should be apparent due to the lack of value and variable names on some lines.
- Variables without "." in the name are measured PMU events. Any "." in any PMU event name as defined in other sections has been converted to an "_".
- Data reference type matching has been used to measure some events. These events can be recognized because the names are of the form <pmu event name>_<data reference type>. See Section 4.2.7.1, "Primary Data Reference Types".

B.2 Retired Instruction Types

The report in this section shows for all instructions retired, how many were of various different types.

variable name	value
IA64_INST_RETIRED	1000.00000
USEFUL_INSTRS_RETIRED.calc	787.02012
LOADS_RETIRED.calc	113.40148
STORES_RETIRED.calc	35.50626
BR_PRED_DETAIL_ANY_ANY_PRED	147.17533
RETIRED_INST_FP	0.16913
OTHER_INSTRS_RETIRED.calc	490.76791
NON_USEFUL_INSTRS_RETIRED.calc	212.97988
RETIRED_INST_NOP	199.95968
RETIRED_PREDICATE_SQUASHED	13.02020

```
|                                 calculations|         value|                    variable name
|---------------------------------------------|--------------|--------------------------------
|                                             |       0.41451|DATA_REF_LOAD_FP
|                                             |     112.98697|DATA_REF_LOAD_INT
|                                             |       0.85462|DATA_REF_STORE_FP
|                                             |      34.65163|DATA_REF_STORE_INT
|                                             |    1000.00000|IA64_INST_RETIRED
|          DATA_REF_LOAD_INT +                |              |
|          DATA_REF_LOAD_FP                   |     113.40148|LOADS_RETIRED.calc
| RETIRED_INST_NOP            +               |              |
| RETIRED_PREDICATE_SQUASHED                  |     212.97988|NON_USEFUL_INSTRS_RETIRED.calc
|IA64_INST_RETIRED            -               |              |
|RETIRED_INST_NOP             -               |              |
|RETIRED_PREDICATE_SQUASHED   -               |              |
|LOADS_RETIRED.calc           -               |              |
|STORES_RETIRED.calc          -               |              |
|BR_PRED_DETAIL_ANY_ANY_PRED  -               |              |
|FPOP_RETIRED.calc                            |     490.76791|OTHER_INSTRS_RETIRED.calc
|         DATA_REF_STORE_INT +                |              |
|         DATA_REF_STORE_FP                   |      35.50626|STORES_RETIRED.calc
| IA64_INST_RETIRED           -               |              |
| RETIRED_INST_NOP            -               |              |
| RETIRED_PREDICATE_SQUASHED                  |     787.02012|USEFUL_INSTRS_RETIRED.calc
```

B.3 Back-End Cycle Accounting

The PMU cycle accounting events provide a way to account for what a the processor core is doing on each and every clock cycle. Any such accounting is somewhat subjective because in reality the core is doing many different things on any given cycle. As discussed in Section 3.2.1.4, “Cycle Accounting”, events occurring later in the pipeline take precedence over events occurring earlier in the pipeline.

In Section 4.2.4, “Back-End Cycle Accounting”, cycle accounting events are arranged according to the way they are counted in the hardware. This section attempts to group the various cycle account events in a way that will be more meaningful people who work on software performance: unstalled cycles, thread switch cycles, miscellaneous flush cycles, instruction access cycles, register hazard cycles, data access cycles, and miscellaneous cycles. These high level groupings provide a rough understanding of where the core is spending it’s time. Each of these groupings has further subdivisions that give more detail about how the time in each grouping is being spent. The instruction access group contains cycles spent waiting for instructions to be fetched, including flush cycles associated with branch mispredictions. The register hazard grouping contains cycles spent waiting on results for non-memory operations. The data access cycles grouping contains cycles spent waiting for load results to return and other cycles when the pipeline cannot make progress due waiting for a memory instruction for any reason. See the descriptions of each base PMU event in Section 4.2.4, “Back-End Cycle Accounting” for more information.

A number of abbreviations are used repeatedly in this report. Other than the overall count of all CPU_CYCLES, the base PMU events used to generate this report are the CYC_BE_* events. CYC_BE refers to an accounting of the cycles in the “Back End” of the core pipeline. EXE_REPLAY, DET_REPLAY, WB2_REPLAY refer to cycles lost due to a replay from the specified pipeline stage. IBD_STALL refers to cycles when the pipeline is not being flushed or replayed and no instructions are issued into the pipeline. Some events produce replays that are then followed by IBD stalls, and so cycles associated with such an event are split across a replay event and an IBD_STALL event. GR, FR, and PR refer to general register, floating point register, and predicate register, respectively. RSE refers to the Register Stack Engine.

The second calculations containing table of this report is fairly trivial because this report is just a fairly simple regrouping of cycle accounting events.

variable name	value
CPU_CYCLES	541.38471
UNSTALLED_EXE.calc	267.82085
TSWITCH.calc	0.00000
CYC_BE_IBD_STALL_THRSW	0.00000
CYC_BE_WB2_REPLAY_PAUSE	0.00000
MISC_FLUSH.calc	0.00648
INSTR_ACCESS.calc	204.00668
INSTR_FETCH.calc	93.27150
CYC_BE_IBD_STALL_QFULL	0.00050
CYC_BE_IBD_STALL_FEBUB	93.27100
CYC_BE_WB2_FLUSH_BRU	110.73518
REG_HAZARD.calc	18.18289
CYC_BE_EXE_REPLAY_GR_GR	0.08005
FR_FR_HAZARD.calc	6.00917
CYC_BE_IBD_STALL_FTOF	2.47294
CYC_BE_EXE_REPLAY_FR_FR	3.53623
PR_HAZARD.calc	7.05143
CYC_BE_EXE_REPLAY_FCMP	3.61864
CYC_BE_EXE_REPLAY_NOTN	0.18442
CYC_BE_EXE_REPLAY_PRED	3.24837
RSE.calc	4.86850
CYC_BE_IBD_STALL_RSE_ST	1.85476
RSE_UNDERFLOW.calc	1.81191
CYC_BE_IBD_STALL_RSE_WAIT	1.20183
MISC_REG_HAZARD.calc	0.17374
CYC_BE_EXE_REPLAY_ARCR	0.17374
CYC_BE_DET_REPLAY_DCS_HZRD	0.00000
CYC_BE_DET_REPLAY_HPW_HZRD	0.00000
CYC_BE_WB2_REPLAY_MOV_PSR_UM	0.00000
CYC_BE_EXE_REPLAY_FPSR	0.00000
DATA_ACCESS.calc	51.21450
GR_LOAD_HAZARD.calc	34.77740
CYC_BE_IBD_STALL_GR_LOAD	17.14045
CYC_BE_DET_REPLAY_GR_LOAD	13.54817
CYC_BE_EXE_REPLAY_GR_LOAD_RAW	4.00399
CYC_BE_EXE_REPLAY_GR_LOAD_WAW	0.08480
FR_LOAD_HAZARD.calc	7.96142
CYC_BE_IBD_STALL_FR_LOAD	2.96305
CYC_BE_EXE_REPLAY_FR_LOAD_RAW	4.94232
CYC_BE_EXE_REPLAY_FR_LOAD_WAW	0.05605
BACK_PRESSURE.calc	2.25033
MLD_BACK_PRESSURE.calc	2.25033
CYC_BE_IBD_STALL_OZQFULL	1.18762
CYC_BE_WB2_REPLAY_OZQ_FULL	1.06271
CYC_BE_WB2_REPLAY_DAHR_HZRD	0.00000
FLD_CONFLICT.calc	2.39922
FLD_FILL_CONFLICT.calc	0.11063
CYC_BE_DET_REPLAY_WRITE_HIT_VS_FILL	0.08636
CYC_BE_DET_REPLAY_WRITE_MISS_VS_FILL	0.02427
FLD_WRITE_CONFLICT.calc	1.93610
CYC_BE_DET_REPLAY_LOAD_AFTER_WRITE	0.55691
CYC_BE_DET_REPLAY_STORE_VS_STORE	1.37919
CYC_BE_DET_REPLAY_FLUSH_DST	0.00000
FLD_CONFLICT_OTHER.calc	0.35249
CYC_BE_IBD_STALL_FLD_DMND	0.11389
CYC_BE_IBD_STALL_MTOM	0.23841
CYC_BE_DET_REPLAY_MT1	0.00019
DATA_TLB_ACCESS.calc	3.82608
VHPT_WALKER.calc	3.82600
CYC_BE_IBD_STALL_HPW	3.04420
CYC_BE_WB2_REPLAY_BLK_HPW	0.78179
CYC_BE_WB2_REPLAY_STORE_ALIAS	0.00008
ORDERING_HAZARD.calc	0.00000
ACQUIRE_HAZARD.calc	0.00000
CYC_BE_IBD_STALL_ACQ	0.00000
CYC_BE_DET_REPLAY_LOAD_ACQ	0.00000
CYC_BE_WB2_REPLAY_LOAD_ACQ	0.00000
RELEASE_HAZARD.calc	0.00000
CYC_BE_IBD_STALL_ARCR	0.00000
CYC_BE_EXE_REPLAY_REL	0.00000
SRLZ_HAZARD.calc	0.00000
CYC_BE_IBD_STALL_SRLZ	0.00000
CYC_BE_EXE_REPLAY_SRLZ	0.00000
SPEC_PENALTIES.calc	0.00004
SPEC_FAIL_FLUSH.calc	0.00000
SPEC_HAZARD.calc	0.00004
CYC_BE_WB2_REPLAY_NAT_HZRD	0.00000
CYC_BE_WB2_REPLAY_LDC	0.00004
MISC_CYCLES.calc	0.24888
CYC_BE_IBD_STALL_DEBUG	0.00000
CYC_BE_IBD_STALL_WB2_TRAP	0.00000

CYC_BE_EXE_REPLAY_MT1_HIGH	0.24824
CYC_BE_EXE_REPLAY_MT1_LOW	0.00000
CYC_BE_WB2_REPLAY_VIRT_INT	0.00000
CYC_BE_WB2_REPLAY_SER	0.00000
CYC_BE_WB2_REPLAY_MT1	0.00064
CYC_BE_WB2_REPLAY_FP_SIR	0.00000
CYC_BE_WB2_REPLAY_FP_DEN	0.00000

calculations	value	variable name
CYC_BE_IBD_STALL_ACQ +		
CYC_BE_DET_REPLAY_LOAD_ACQ +		
CYC_BE_WB2_REPLAY_LOAD_ACQ	0.00000	ACQUIRE_HAZARD.calc
MLD_BACK_PRESSURE.calc +		
CYC_BE_WB2_REPLAY_DAHR_HZRD	2.25033	BACK_PRESSURE.calc
	0.00000	CSPEC_CHKS_FAIL_ANY
	273.56386	CYC_BE_BUBBLE_ANY
	1.81191	CYC_BE_IBD_STALL_RSE_CFLE
	0.00000	CYC_BE_IBD_STALL_RSE_LOAD
	0.00648	CYC_BE_WB2_FLUSH_XPN
GR_LOAD_HAZARD.calc +		
FR_LOAD_HAZARD.calc +		
BACK_PRESSURE.calc +		
FLD_CONFLICT.calc +		
DATA_TLB_ACCESS.calc +		
ORDERING_HAZARD.calc +		
SPEC_PENALTIES.calc	51.21450	DATA_ACCESS.calc
VHPT_WALKER.calc +		
CYC_BE_WB2_REPLAY_STORE_ALIAS	3.82608	DATA_TLB_ACCESS.calc
CYC_BE_IBD_STALL_FLD_DMND +		
CYC_BE_IBD_STALL_MTOM +		
CYC_BE_DET_REPLAY_FLUSH_STORE +		
CYC_BE_DET_REPLAY_LOAD_AFTER_WRITE +		
CYC_BE_DET_REPLAY_STORE_VS_STORE +		
CYC_BE_DET_REPLAY_WRITE_HIT_VS_FILL +		
CYC_BE_DET_REPLAY_WRITE_MISS_VS_FILL +		
CYC_BE_DET_REPLAY_MT1	2.39922	FLD_CONFLICT.calc
CYC_BE_IBD_STALL_FLD_DMND +		
CYC_BE_IBD_STALL_MTOM +		
CYC_BE_DET_REPLAY_MT1	0.35249	FLD_CONFLICT_OTHER.calc
CYC_BE_DET_REPLAY_WRITE_HIT_VS_FILL +		
CYC_BE_DET_REPLAY_WRITE_MISS_VS_FILL	0.11063	FLD_FILL_CONFLICT.calc
CYC_BE_DET_REPLAY_LOAD_AFTER_WRITE +		
CYC_BE_DET_REPLAY_STORE_VS_STORE +		
CYC_BE_DET_REPLAY_FLUSH_STORE	1.93610	FLD_WRITE_CONFLICT.calc
CYC_BE_IBD_STALL_FTOF +		
CYC_BE_EXE_REPLAY_FR_FR	6.00917	FR_FR_HAZARD.calc
CYC_BE_IBD_STALL_FR_LOAD +		
CYC_BE_EXE_REPLAY_FR_LOAD_RAW +		
CYC_BE_EXE_REPLAY_FR_LOAD_WAW	7.96142	FR_LOAD_HAZARD.calc
CYC_BE_IBD_STALL_GR_LOAD +		
CYC_BE_DET_REPLAY_GR_LOAD +		
CYC_BE_EXE_REPLAY_GR_LOAD_RAW +		
CYC_BE_EXE_REPLAY_GR_LOAD_WAW	34.77740	GR_LOAD_HAZARD.calc
CYC_BE_DET_REPLAY_HPW_HZRD	0.00000	HPW_REG_REPLAY.calc
INSTR_FETCH.calc +		
CYC_BE_WB2_FLUSH_BRU	204.00668	INSTR_ACCESS.calc
CYC_BE_IBD_STALL_QFULL +		
CYC_BE_IBD_STALL_FEBUB	93.27150	INSTR_FETCH.calc
CYC_BE_IBD_STALL_DEBUG +		
CYC_BE_IBD_STALL_WB2_TRAP +		
CYC_BE_EXE_REPLAY_MT1_HIGH +		
CYC_BE_EXE_REPLAY_MT1_LOW +		
CYC_BE_WB2_REPLAY_VIRT_INT +		
CYC_BE_WB2_REPLAY_SER +		
CYC_BE_WB2_REPLAY_MT1 +		
CYC_BE_WB2_REPLAY_FP_SIR +		
CYC_BE_WB2_REPLAY_FP_DEN	0.24888	MISC_CYCLES.calc
CYC_BE_WB2_FLUSH_XPN -		
SPEC_FAIL_FLUSH.calc	0.00648	MISC_FLUSH.calc
CYC_BE_EXE_REPLAY_ARCR +		
CYC_BE_DET_REPLAY_DCS_HZRD +		
CYC_BE_DET_REPLAY_HPW_HZRD +		
CYC_BE_WB2_REPLAY_MOV_PSR_UM +		
CYC_BE_EXE_REPLAY_FPSR	0.17374	MISC_REG_HAZARD.calc
CYC_BE_IBD_STALL_OZQFULL +		
CYC_BE_WB2_REPLAY_OZQ_FULL	2.25033	MLD_BACK_PRESSURE.calc
CYC_BE_IBD_STALL_ACQ +		
CYC_BE_IBD_STALL_ARCR +		
CYC_BE_IBD_STALL_SRLZ +		
CYC_BE_EXE_REPLAY_REL +		
CYC_BE_EXE_REPLAY_SRLZ +		
CYC_BE_DET_REPLAY_LOAD_ACQ +		
CYC_BE_WB2_REPLAY_LOAD_ACQ	0.00000	ORDERING_HAZARD.calc

CYC_BE_EXE_REPLAY_FCMP + CYC_BE_EXE_REPLAY_NOTN + CYC_BE_EXE_REPLAY_PRED	7.05143	PR_HAZARD.calc
CYC_BE_EXE_REPLAY_GR_GR + FR_FR_HAZARD.calc + PR_HAZARD.calc + RSE.calc + MISC_REG_HAZARD.calc	18.18289	REG_HAZARD.calc
CYC_BE_IBD_STALL_ARCR + CYC_BE_EXE_REPLAY_REL	0.00000	RELEASE_HAZARD.calc
CYC_BE_IBD_STALL_RSE_ST + RSE_UNDERFLOW.calc + CYC_BE_IBD_STALL_RSE_WAIT	4.86850	RSE.calc
CYC_BE_IBD_STALL_RSE_CFLE + CYC_BE_IBD_STALL_RSE_LOAD	1.81191	RSE_UNDERFLOW.calc
CSPEC_CHKS_FAIL_ANY * SPEC_FAIL_FLUSH_PENALTY.psn	0.00000	SPEC_FAIL_FLUSH.calc
	17.00000	SPEC_FAIL_FLUSH_PENALTY.psn
CYC_BE_WB2_REPLAY_NAT_HZRD + CYC_BE_WB2_REPLAY_LDC	0.00004	SPEC_HAZARD.calc
SPEC_FAIL_FLUSH.calc + SPEC_HAZARD.calc	0.00004	SPEC_PENALTIES.calc
CYC_BE_IBD_STALL_SRLZ + CYC_BE_EXE_REPLAY_SRLZ	0.00000	SRLZ_HAZARD.calc
CYC_BE_IBD_STALL_THRSW + CYC_BE_WB2_REPLAY_PAUSE	0.00000	TSWITCH.calc
CYC_BE_IBD_STALL_THRSW	0.00000	TSWITCH_STALL.calc
CPU_CYCLES - CYC_BE_BUBBLE_ANY	267.82085	UNSTALLED_EXE.calc
CYC_BE_WB2_REPLAY_BLK_HPW + CYC_BE_IBD_STALL_HPW	3.82600	VHPT_WALKER.calc

B.4 Primary Data Reference Outcomes

The reports in this section demonstrate the final outcome of all primary data references.

B.4.1 LOAD_ANY

The most obvious load outcomes are hitting in the FLD, MLD, LLC, or memory (LLC miss). Loads can also be secondary misses at the MLD (MLD_SMQ_REF_HIT_LOAD_ANY). A secondary miss is an access that at first misses a cache but waits for another access that is already outstanding to the same line and then hits. Control speculative loads can also be deferred (CSPEC_LOAD_NAT). "_LOAD_ANY" refers to data reference type matching for some of these events.

variable name	value
DATA_REF_LOAD_ANY	92.98356
CSPEC_LOAD_NAT	0.01960
FLD_LOAD_HIT	81.58444
MLD_LOAD_HIT	7.93826
MLD_SMQ_REF_HIT_LOAD_ANY	1.10795
LLC_LOAD_HIT.calc	1.98280
LLC_REF_MISS_DATA_LOAD_ANY	0.35181

calculations	value	variable name
RIL_REQ_REF_DATA_LOAD_ANY - LLC_REF_MISS_DATA_LOAD_ANY	1.98280	LLC_LOAD_HIT.calc
	2.33462	RIL_REQ_REF_DATA_LOAD_ANY
	0.35181	LLC_REF_MISS_DATA_LOAD_ANY

B.4.2 STORE_ANY

The possible store outcomes are very similar to the possible load outcomes with one exception. In a write coalescing address space, some stores can be coalesced with other writes. The number of coalesced stores (MLD_STORE_COALESCED.calc) can be determined by looking at the difference between the number of stores at the MLD (pre-coalescing) and the RIL (post-coalescing). "_STORE_ANY" refers to data reference type matching for some of these events.

variable name	value
DATA_REF_STORE_ANY	91.58478
MLD_REF_HIT_STORE_ANY	73.87507
MLD_SMQ_REF_HIT_STORE_ANY	13.01904
MLD_STORE_COALESCED.calc	0.00000
LLC_STORE_HIT.calc	4.61244
LLC_REF_MISS_DATA_STORE_ANY	0.08532

calculations	value	variable name
	0.08532	LLC_REF_MISS_DATA_STORE_ANY
RIL_REQ_REF_DATA_STORE_ANY - LLC_REF_MISS_DATA_STORE_ANY	4.61244	LLC_STORE_HIT.calc
RIL_REQ_REF_DATA_NC_WRITE_WC_MLD_STORE_ANY - RIL_REQ_REF_DATA_NC_WRITE_WC_ANY_STORE_ANY	0.00000	MLD_STORE_COALESCED.calc
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_WC_ANY_STORE_ANY
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_WC_MLD_STORE_ANY
	4.69775	RIL_REQ_REF_DATA_STORE_ANY

B.4.3 SEMAPHORE

Possible semaphore outcomes are all similar to possible load outcomes. "_SEMAPHORE" refers to data reference type matching for some of these events.

variable name	value
DATA_REF_SEMAPHORE	0.00000
MLD_REF_HIT_SEMAPHORE	0.00000
MLD_SMQ_REF_HIT_SEMAPHORE	0.00000
LLC_SEMAPHORE_HIT.calc	0.00000
LLC_REF_MISS_DATA_SEMAPHORE	0.00000

calculations	value	variable name
	0.00000	LLC_REF_MISS_DATA_SEMAPHORE
RIL_REQ_REF_DATA_SEMAPHORE - LLC_REF_MISS_DATA_SEMAPHORE	0.00000	LLC_SEMAPHORE_HIT.calc
	0.00000	RIL_REQ_REF_DATA_SEMAPHORE

B.4.4 LFETCH

Many possible lfetch outcomes are similar to possible load outcomes. However, lfetches can also be dropped in some cases when they miss the FLDTLB, hit in the FLD, miss in the FLD when there is already an outstanding request to the same line, or miss in the MLD (PREF_DROP_FLDTLB_MISS_LFETCH, PREF_DROP_FLD_HIT_LFETCH, PREF_DROP_FLD_SECONDARY_MISS_LFETCH, MLD_LFETCH_MISS_DROP.calc). "_LFETCH" refers to data reference type matching for some of these events.

variable name	value
DATA_REF_LFETCH	23.99696
PREF_DROP_FLDTLB_MISS_LFETCH	0.00000
PREF_DROP_FLD_HIT_LFETCH	1.38983
PREF_DROP_FLD_SECONDARY_MISS_LFETCH	0.03715
MLD_REF_HIT_LFETCH	7.73774
MLD_SMQ_REF_HIT_LFETCH	0.08724
MLD_LFETCH_MISS_DROP.calc	11.59322
LLC_LFETCH_HIT.calc	2.95498
LLC_REF_MISS_DATA_LFETCH	0.11996

calculations	value	variable name
RIL_REQ_REF_DATA_LFETCH - LLC_REF_MISS_DATA_LFETCH	2.95498	LLC_LFETCH_HIT.calc
	0.11996	LLC_REF_MISS_DATA_LFETCH
	0.00000	MLD_HINT_PREF_DROP_LFETCH
MLD_HINT_PREF_DROP_LFETCH + MLD_REF_SECONDARY_DROP_LFETCH + MLD_SMQ_REF_SECONDARY_DROP_LFETCH	11.59322	MLD_LFETCH_MISS_DROP.calc
	11.59196	MLD_REF_SECONDARY_DROP_LFETCH
	0.00127	MLD_SMQ_REF_SECONDARY_DROP_LFETCH
	3.07494	RIL_REQ_REF_DATA_LFETCH

B.4.5 HW_PREF

The possible hardware prefetch outcomes are similar to the possible lfetch outcomes. MLD buddy line prefetches are excluded from this hardware prefetch outcome breakdown because including them often leads to confusion. The confusion arises because MLD buddy line prefetches are inserted between MLD and LLC, and so they only appear in the LLC and memory data. The buddy line prefetches are accounted for in a separate report. Excluding the MLD buddy line prefetches from this report requires an assumption and some fairly tricky calculations. The assumption is that MLD buddy line prefetches hit in the LLC cache at the same rate as the primary accesses that trigger them. To the extent that this is not true, this report will be negatively affected.

"_HW_PREF", "_LFETCH", "LOAD_ANY", "STORE_ANY", and "SEMAPHORE" refer to data reference type matching for some of these events. The HW_PREF data reference type includes non-buddy hardware prefetches and MLD buddy line prefetches triggered by any type of data reference. The hardware enables calculation of the ratio of non-buddy hardware prefetches to buddy hardware prefetches (HWDPREF_BUDDY_FRACTION.calc). This ratio is used to figure out how many of the hardware prefetches at the LLC and memory are MLD buddy line prefetches.

variable name	value
DATA_REF_HW_PREF	18.32937
PREF_DROP_FLDTLB_MISS_HW_PREF	0.00372
PREF_DROP_FLD_HIT_HW_PREF	3.43658
PREF_DROP_FLD_SECONDARY_MISS_HW_PREF	0.54597
MLD_REF_HIT_HW_PREF	8.08746
MLD_SMQ_REF_HIT_HW_PREF	0.35239
MLD_HWDPREF_MISS_DROP.calc	4.55417
LLC_HWDPREF_NOBUDDY_HIT.calc	1.03767
MEM_HWDPREF_NOBUDDY.calc	0.31124

calculations	value	variable name
(LLC_HWDPREF.calc - MLD_REF_PRIMARY_HW_PREF)/ LLC_HWDPREF.calc	0.49846	HWDPREF_BUDDY_FRACTION.calc
RIL_REQ_REF_DATA_HW_PREF - RIL_REQ_REF_DATA_WB_MLD_BUDDY_LFETCH - RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_ANY - RIL_REQ_REF_DATA_WB_MLD_BUDDY_STORE_ANY - RIL_REQ_REF_DATA_WB_MLD_BUDDY_SEMAPHORE	2.68952	LLC_HWDPREF.calc
LLC_HWDPREF.calc * HWDPREF_BUDDY_FRACTION.calc	1.34061	LLC_HWDPREF_BUDDY.calc
LLC_HWDPREF.calc - LLC_HWDPREF_BUDDY.calc	1.34891	LLC_HWDPREF_NOBUDDY.calc
LLC_HWDPREF_NOBUDDY.calc - MEM_HWDPREF_NOBUDDY.calc	1.03767	LLC_HWDPREF_NOBUDDY_HIT.calc
	0.62056	LLC_REF_MISS_DATA_HW_PREF
LLC_REF_MISS_DATA_HW_PREF * HWDPREF_BUDDY_FRACTION.calc	0.31124	MEM_HWDPREF_BUDDY.calc
LLC_REF_MISS_DATA_HW_PREF - MEM_HWDPREF_BUDDY.calc	0.30932	MEM_HWDPREF_NOBUDDY.calc
	4.54245	MLD_HINT_PREF_DROP_HW_PREF
MLD_HINT_PREF_DROP_HW_PREF + MLD_REF_SECONDARY_DROP_HW_PREF + MLD_SMQ_REF_SECONDARY_DROP_HW_PREF	4.55417	MLD_HWDPREF_MISS_DROP.calc
	1.34891	MLD_REF_PRIMARY_HW_PREF
	0.01173	MLD_REF_SECONDARY_DROP_HW_PREF
	0.00000	MLD_SMQ_REF_SECONDARY_DROP_HW_PREF
	12.70759	RIL_REQ_REF_DATA_HW_PREF
	3.07403	RIL_REQ_REF_DATA_WB_MLD_BUDDY_LFETCH
	2.24789	RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_ANY
	0.00000	RIL_REQ_REF_DATA_WB_MLD_BUDDY_SEMAPHORE
	4.69616	RIL_REQ_REF_DATA_WB_MLD_BUDDY_STORE_ANY

B.4.6 MLD Buddy Line Prefetches

The possible MLD buddy line prefetch outcomes are simply hitting in the LLC or accessing memory. Conceptually this report is quite simple, but the calculations to produce the report are tricky because they must deal with the following issues:

- Data reference type matching works in a way that lumps together non-buddy hardware prefetches and buddy prefetches triggered by non-buddy hardware prefetches. (See Section 4.2.7.2, “Asynchronous Data References and Event Matching Constraints” for more details.) Separating these requires an assumption and the calculation of HWDPREF_BUDDY_FRACTION.calc as discussed in Section B.4.5, “HW_PREF”.
- Data reference type matching works a little differently on buddy prefetches for the RIL* than it does other events. (See Section 4.2.7.2, “Asynchronous Data References and Event Matching Constraints” for more details.)

“HW_PREF” refers to data reference type matching for some of these events. “HW_PREF_LOAD_ANY” and other combinations of data reference types refer to data reference type matching where multiple data references types are being matched.

variable name	value
LLC_BUDDY.calc	11.35868
LLC_BUDDY_HIT.calc	10.48780
LLC_HWDPREF_BUDDY_HIT.calc	1.03129
LLC_LFETCH_BUDDY_HIT.calc	2.95419
LLC_LOAD_BUDDY_HIT.calc	1.90093
LLC_STORE_BUDDY_HIT.calc	4.61038
LLC_SEMAPHORE_BUDDY_HIT.calc	0.00331
MEM_BUDDY.calc	0.85859
MEM_HWDPREF_BUDDY.calc	0.30932
MEM_LFETCH_BUDDY.calc	0.11984
MEM_LOAD_BUDDY.calc	0.34695
MEM_STORE_BUDDY.calc	0.08579
MEM_SEMAPHORE_BUDDY.calc	-0.00331

calculations	value	variable name
(LLC_HWDPREF.calc - MLD_REF_PRIMARY_HW_PREF) / LLC_HWDPREF.calc	0.49846	HWDPREF_BUDDY_FRACTION.calc
RIL_REQ_REF_DATA_HW_PREF - MLD_REF_PRIMARY_HW_PREF	11.35868	LLC_BUDDY.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY - MEM_BUDDY.calc	10.48780	LLC_BUDDY_HIT.calc
RIL_REQ_REF_DATA_HW_PREF - LLC_LFETCH_BUDDY.calc - LLC_LOAD_BUDDY.calc - LLC_STORE_BUDDY.calc - LLC_SEMAPHORE_BUDDY.calc	2.68952	LLC_HWDPREF.calc
LLC_HWDPREF.calc * HWDPREF_BUDDY_FRACTION.calc	1.34061	LLC_HWDPREF_BUDDY.calc
LLC_HWDPREF_BUDDY.io-MEM_HWDPREF_BUDDY.calc	1.03129	LLC_HWDPREF_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LFETCH - MEM_LFETCH_BUDDY.calc	2.95419	LLC_LFETCH_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_ANY - MEM_LOAD_BUDDY.calc	1.90093	LLC_LOAD_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_FP - MEM_LOAD_FP_BUDDY.calc	0.00444	LLC_LOAD_FP_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_HPW - MEM_LOAD_HPW_BUDDY.calc	0.00079	LLC_LOAD_HPW_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_INT - MEM_LOAD_INT_BUDDY.calc	1.90187	LLC_LOAD_INT_BUDDY_HIT.calc
RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_RSE - MEM_LOAD_RSE_BUDDY.calc	0.00072	LLC_LOAD_RSE_BUDDY_HIT.calc
	0.62056	LLC_REF_MISS_DATA_HW_PREF
	0.86036	LLC_REF_MISS_DATA_HW_PREF_LFETCH
	1.31932	LLC_REF_MISS_DATA_HW_PREF_LOAD_ANY
	0.61649	LLC_REF_MISS_DATA_HW_PREF_LOAD_FP
	0.62097	LLC_REF_MISS_DATA_HW_PREF_LOAD_HPW
	1.31596	LLC_REF_MISS_DATA_HW_PREF_LOAD_INT
	0.61984	LLC_REF_MISS_DATA_HW_PREF_LOAD_RSE
	0.61725	LLC_REF_MISS_DATA_HW_PREF_SEMAPHORE
	0.79166	LLC_REF_MISS_DATA_HW_PREF_STORE_ANY
	0.11996	LLC_REF_MISS_DATA_LFETCH
	0.35181	LLC_REF_MISS_DATA_LOAD_ANY

```
                                                |  0.00012|LLC_REF_MISS_DATA_LOAD_FP
                                                |  0.00121|LLC_REF_MISS_DATA_LOAD_HPW
                                                |  0.34899|LLC_REF_MISS_DATA_LOAD_INT
                                                |  0.00000|LLC_REF_MISS_DATA_LOAD_RSE
                                                |  0.00000|LLC_REF_MISS_DATA_SEMAPHORE
                                                |  0.08532|LLC_REF_MISS_DATA_STORE_ANY
  RIL_REQ_REF_DATA_WB_MLD_BUDDY_SEMAPHORE -
                    MEM_SEMAPHORE_BUDDY.calc    |  0.00331|LLC_SEMAPHORE_BUDDY_HIT.calc
  RIL_REQ_REF_DATA_WB_MLD_BUDDY_STORE_ANY -
                         MEM_STORE_BUDDY.calc   |  4.61038|LLC_STORE_BUDDY_HIT.calc
                       MEM_LFETCH_BUDDY.io +
                     MEM_HWDPREF_BUDDY.io +
                        MEM_LOAD_BUDDY.io +
                      MEM_STORE_BUDDY.io +
                    MEM_SEMAPHORE_BUDDY.calc    |  0.85859|MEM_BUDDY.calc
                LLC_REF_MISS_DATA_HW_PREF *
                 HWDPREF_BUDDY_FRACTION.calc    |  0.30932|MEM_HWDPREF_BUDDY.calc
          LLC_REF_MISS_DATA_HW_PREF_LFETCH -
                  LLC_REF_MISS_DATA_HW_PREF -
                     LLC_REF_MISS_DATA_LFETCH   |  0.11984|MEM_LFETCH_BUDDY.calc
       LLC_REF_MISS_DATA_HW_PREF_LOAD_ANY -
                  LLC_REF_MISS_DATA_HW_PREF -
                    LLC_REF_MISS_DATA_LOAD_ANY  |  0.34695|MEM_LOAD_BUDDY.calc
        LLC_REF_MISS_DATA_HW_PREF_LOAD_FP -
                  LLC_REF_MISS_DATA_HW_PREF -
                     LLC_REF_MISS_DATA_LOAD_FP  | -0.00419|MEM_LOAD_FP_BUDDY.calc
       LLC_REF_MISS_DATA_HW_PREF_LOAD_HPW -
                  LLC_REF_MISS_DATA_HW_PREF -
                    LLC_REF_MISS_DATA_LOAD_HPW  | -0.00079|MEM_LOAD_HPW_BUDDY.calc
       LLC_REF_MISS_DATA_HW_PREF_LOAD_INT -
                  LLC_REF_MISS_DATA_HW_PREF -
                    LLC_REF_MISS_DATA_LOAD_INT  |  0.34641|MEM_LOAD_INT_BUDDY.calc
       LLC_REF_MISS_DATA_HW_PREF_LOAD_RSE -
                  LLC_REF_MISS_DATA_HW_PREF -
                    LLC_REF_MISS_DATA_LOAD_RSE  | -0.00072|MEM_LOAD_RSE_BUDDY.calc
      LLC_REF_MISS_DATA_HW_PREF_SEMAPHORE -
                  LLC_REF_MISS_DATA_HW_PREF -
                   LLC_REF_MISS_DATA_SEMAPHORE  | -0.00331|MEM_SEMAPHORE_BUDDY.calc
       LLC_REF_MISS_DATA_HW_PREF_STORE_ANY -
                  LLC_REF_MISS_DATA_HW_PREF -
                    LLC_REF_MISS_DATA_STORE_ANY |  0.08579|MEM_STORE_BUDDY.calc
                                                |  1.34891|MLD_REF_PRIMARY_HW_PREF
                                                | 12.70759|RIL_REQ_REF_DATA_HW_PREF
                                                | 11.34639|RIL_REQ_REF_DATA_WB_MLD_BUDDY
                                                |  3.07403|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LFETCH
                                                |  2.24789|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_ANY
                                                |  0.00025|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_FP
                                                |  0.00000|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_HPW
                                                |  2.24828|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_INT
                                                |  0.00000|RIL_REQ_REF_DATA_WB_MLD_BUDDY_LOAD_RSE
                                                |  0.00000|RIL_REQ_REF_DATA_WB_MLD_BUDDY_SEMAPHORE
                                                |  4.69616|RIL_REQ_REF_DATA_WB_MLD_BUDDY_STORE_ANY
```

B.4.7 DREF

This data reference outcomes report combines all of the types of data references and their possible outcomes into one report. See the previous sections with reports for the individual data reference types for more information.

```
                              variable name|             value
-------------------------------------------|------------------
DATA_REF_ANY                                |226.89468
  TLB_DREF.calc                             |  0.02332
    CSPEC_LOAD_NAT                          |    0.01960
    PREF_DROP_FLDTLB_MISS                   |    0.00372
  FLD_DREF.cares                            | 86.99398
    FLD_DREAD_HIT.cares                     |   86.41085
      FLD_LOAD_HIT.cares                    |     81.58444
        FLD_LOAD_INT_HIT.calc               |       80.47364
        FLD_LOAD_RSE_HIT.calc               |       1.11080
      FLD_DPREF_HIT.calc                    |     4.82641
        PREF_DROP_FLD_HIT_LFETCH            |       1.38983
        PREF_DROP_FLD_HIT_HW_PREF           |       3.43658
    FLD_DPREF_SECONDARY.calc                |   0.58313
      PREF_DROP_FLD_SECONDARY_MISS_LFETCH   |     0.03715
      PREF_DROP_FLD_SECONDARY_MISS_HW_PREF  |     0.54597
  MLD_DREF.calc                             | 128.32017
    MLD_DREF_HIT.calc                       |   97.62069
      MLD_DREAD_HIT.calc                    |     23.74735
        MLD_LOAD_HIT                        |       7.93826
          MLD_REF_HIT_LOAD_INT              |         7.22803
          MLD_REF_HIT_LOAD_FP               |         0.30201
          MLD_LOAD_RSE_HIT.calc             |         0.18273
          MLD_REF_HIT_LOAD_HPW              |         0.20939
```

```
|        MLD_DPREF_HIT.calc                    |        15.82520
|          MLD_REF_HIT_LFETCH                  |          7.73774
|          MLD_REF_HIT_HW_PREF                 |          8.08746
|      MLD_WRITE_HIT.calc                      |       73.87334
|        MLD_REF_HIT_STORE_INT                 |         43.51066
|        MLD_REF_HIT_STORE_FP                  |         29.07523
|        MLD_REF_HIT_STORE_RSE                 |         1.28744
|        MLD_REF_HIT_SEMAPHORE                 |         0.00000
|    MLD_DREF_SECONDARY.calc                   |      14.55208
|      MLD_SMQ_REF_HIT_LOAD_ANY                |        1.10795
|        MLD_SMQ_REF_HIT_LOAD_INT              |          1.07331
|        MLD_SMQ_REF_HIT_LOAD_FP               |          0.03241
|        MLD_SMQ_REF_HIT_LOAD_RSE              |          0.00106
|        MLD_SMQ_REF_HIT_LOAD_HPW              |          0.00000
|      MLD_DPREF_SECONDARY.calc                |        0.43963
|        MLD_SMQ_REF_HIT_LFETCH                |          0.08724
|        MLD_SMQ_REF_HIT_HW_PREF               |          0.35239
|      MLD_WRITE_SECONDARY.calc                |        13.03210
|        MLD_SMQ_REF_HIT_STORE_INT             |          1.49042
|        MLD_SMQ_REF_HIT_STORE_FP              |          11.53583
|        MLD_SMQ_REF_HIT_STORE_RSE             |          0.00584
|        MLD_SMQ_REF_HIT_SEMAPHORE             |          0.00000
|    MLD_DPREF_MISS_DROP.calc                  |      16.14740
|      MLD_LFETCH_MISS_DROP.calc               |        11.59322
|      MLD_HWDPREF_MISS_DROP.calc              |        4.55417
|    MLD_WRITE_COALESCED.calc                  |      0.00000
|  LLC_DREF_HIT.calc                           |    10.59260
|    LLC_DREAD_HIT.calc                        |      5.97546
|      LLC_LOAD_HIT.calc                       |        1.98280
|        LLC_LOAD_INT_HIT.calc                 |          1.89929
|        LLC_LOAD_FP_HIT.calc                  |          0.00013
|        LLC_LOAD_RSE_HIT.calc                 |          0.00026
|        LLC_LOAD_HPW_HIT.calc                 |          0.08512
|      LLC_DPREF_HIT.calc                      |        3.99265
|        LLC_LFETCH_HIT.calc                   |          2.95498
|        LLC_HWDPREF_NOBUDDY_HIT.calc          |          1.03767
|    LLC_WRITE_HIT.calc                        |      4.61714
|      LLC_STORE_INT_HIT.calc                  |        0.12085
|      LLC_STORE_FP_HIT.calc                   |        4.49469
|      LLC_STORE_RSE_HIT.calc                  |        0.00159
|      LLC_SEMAPHORE_HIT.calc                  |        0.00000
|  MEM_DREF.calc                               |    1.17760
|    MEM_DREAD.calc                            |      1.09233
|      MEM_LOAD.calc                           |        0.35181
|        LLC_REF_MISS_DATA_LOAD_INT            |          0.34899
|        LLC_REF_MISS_DATA_LOAD_FP             |          0.00012
|        LLC_REF_MISS_DATA_LOAD_RSE            |          0.00000
|        LLC_REF_MISS_DATA_LOAD_HPW            |          0.00121
|      MEM_DPREF.calc                          |        0.43120
|        LLC_REF_MISS_DATA_LFETCH              |          0.11996
|        MEM_HWDPREF_NOBUDDY.calc              |          0.31124
|    MEM_WRITE.calc                            |      0.08527
|      LLC_REF_MISS_DATA_STORE_INT             |        0.03493
|      LLC_REF_MISS_DATA_STORE_FP              |        0.05033
|      LLC_REF_MISS_DATA_STORE_RSE             |        0.00001
|      LLC_REF_MISS_DATA_SEMAPHORE             |        0.00000
```

```
|                                 calculations|     value|                              variable name
+---------------------------------------------|----------|--------------------------------------
|                       FLD_LFETCH_HIT.calc +
|                         FLD_HWDPREF_HIT.calc|   4.82641|FLD_DPREF_HIT.calc
|                  FLD_LFETCH_SECONDARY.io +
|                  FLD_HWDPREF_SECONDARY.calc|   0.58313|FLD_DPREF_SECONDARY.calc
|                      FLD_LOAD_INT_HIT.io +
|                      FLD_LOAD_RSE_HIT.io +
|                        FLD_LFETCH_HIT.io +
|                        FLD_HWDPREF_HIT.calc|  86.41085|FLD_DREAD_HIT.calc
|                      FLD_DREAD_HIT.calc  +
|                   FLD_DPREF_SECONDARY.calc|  86.06591|FLD_DREF.calc
|                                           |  92.35311|FLD_LOAD_ANY
|                            FLD_LOAD_ANY -
|                          FLD_LOAD_MISS_ANY|  81.58444|FLD_LOAD_HIT.calc
|                                           |  91.05830|FLD_LOAD_INT
|                            FLD_LOAD_INT -
|                          FLD_LOAD_MISS_INT|  80.47364|FLD_LOAD_INT_HIT.calc
|                                           |  10.76867|FLD_LOAD_MISS_ANY
|                                           |  10.58467|FLD_LOAD_MISS_INT
|                (LLC_HWDPREF.calc -
|                 MLD_REF_PRIMARY_HW_PREF) /
|                             LLC_HWDPREF.calc|   0.49846|HWDPREF_BUDDY_FRACTION.calc
|             FLDTLB_HWDPREF_MISS_DROP.calc +
|                    FLD_HWDPREF_HIT.calc +
|               FLD_HWDPREF_SECONDARY.calc +
|                     MLD_HWDPREF_HIT.calc +
|               MLD_HWDPREF_SECONDARY.calc +
|               MLD_HWDPREF_MISS_DROP.calc +
```

```
|               LLC_HWDPREF_NOBUDDY_HIT.calc + |
|                     MEM_HWDPREF_NOBUDDY.calc |  18.32921|HWDPREF_NOBUDDY.calc
|                       LLC_LFETCH_HIT.calc +  |
|                  LLC_HWDPREF_NOBUDDY_HIT.calc|   3.99265|LLC_DPREF_HIT.calc
|                         LLC_LOAD_HIT.calc +  |
|                           LLC_DPREF_HIT.calc |   5.97546|LLC_DREAD_HIT.calc
|                        LLC_DREAD_HIT.calc +  |
|                           LLC_WRITE_HIT.calc |  10.59260|LLC_DREF_HIT.calc
|                  RIL_REQ_REF_DATA_HW_PREF -  |
|                     LLC_LFETCH_BUDDY.calc -  |
|                       LLC_LOAD_BUDDY.calc -  |
|                      LLC_STORE_BUDDY.calc -  |
|                     LLC_SEMAPHORE_BUDDY.calc |   2.68952|LLC_HWDPREF.calc
|                          LLC_HWDPREF.calc *  |
|                  HWDPREF_BUDDY_FRACTION.calc |   1.34061|LLC_HWDPREF_BUDDY.calc
|                          LLC_HWDPREF.calc -  |
|                        LLC_HWDPREF_BUDDY.calc|   1.34891|LLC_HWDPREF_NOBUDDY.calc
|                  LLC_HWDPREF_NOBUDDY.calc -  |
|                     MEM_HWDPREF_NOBUDDY.calc |   1.03767|LLC_HWDPREF_NOBUDDY_HIT.calc
|                   RIL_REQ_REF_DATA_LFETCH -  |
|                     LLC_REF_MISS_DATA_LFETCH |   2.95498|LLC_LFETCH_HIT.calc
|                  RIL_REQ_REF_DATA_LOAD_FP -  |
|                    LLC_REF_MISS_DATA_LOAD_FP |   0.00013|LLC_LOAD_FP_HIT.calc
|                 RIL_REQ_REF_DATA_LOAD_ANY -  |
|                   LLC_REF_MISS_DATA_LOAD_ANY |   1.98280|LLC_LOAD_HIT.calc
|                 RIL_REQ_REF_DATA_LOAD_HPW -  |
|                   LLC_REF_MISS_DATA_LOAD_HPW |   0.08512|LLC_LOAD_HPW_HIT.calc
|                 RIL_REQ_REF_DATA_LOAD_INT -  |
|                   LLC_REF_MISS_DATA_LOAD_INT |   1.89929|LLC_LOAD_INT_HIT.calc
|                 RIL_REQ_REF_DATA_LOAD_RSE -  |
|                   LLC_REF_MISS_DATA_LOAD_RSE |   0.00026|LLC_LOAD_RSE_HIT.calc
|                                              |   0.62056|LLC_REF_MISS_DATA_HW_PREF
|                                              |   0.11996|LLC_REF_MISS_DATA_LFETCH
|                                              |   0.35181|LLC_REF_MISS_DATA_LOAD_ANY
|                                              |   0.00012|LLC_REF_MISS_DATA_LOAD_FP
|                                              |   0.00121|LLC_REF_MISS_DATA_LOAD_HPW
|                                              |   0.34899|LLC_REF_MISS_DATA_LOAD_INT
|                                              |   0.00000|LLC_REF_MISS_DATA_LOAD_RSE
|                                              |   0.00000|LLC_REF_MISS_DATA_SEMAPHORE
|                                              |   0.05033|LLC_REF_MISS_DATA_STORE_FP
|                                              |   0.03493|LLC_REF_MISS_DATA_STORE_INT
|                                              |   0.00001|LLC_REF_MISS_DATA_STORE_RSE
|RIL_REQ_REF_DATA_NC_WRITE_ANY_SEMAPHORE +     |
|         RIL_REQ_REF_DATA_WB_RFO_SEMAPHORE    |   0.00000|LLC_SEMAPHORE.calc
|             RIL_REQ_REF_DATA_SEMAPHORE -     |
|                 LLC_REF_MISS_DATA_SEMAPHORE  |   0.00000|LLC_SEMAPHORE_HIT.calc
| RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_FP +     |
|          RIL_REQ_REF_DATA_WB_RFO_STORE_FP    |   4.54502|LLC_STORE_FP.calc
|              RIL_REQ_REF_DATA_STORE_FP -     |
|                  LLC_REF_MISS_DATA_STORE_FP  |   4.49469|LLC_STORE_FP_HIT.calc
|RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_INT +     |
|         RIL_REQ_REF_DATA_WB_RFO_STORE_INT    |   0.15579|LLC_STORE_INT.calc
|             RIL_REQ_REF_DATA_STORE_INT -     |
|                 LLC_REF_MISS_DATA_STORE_INT  |   0.12085|LLC_STORE_INT_HIT.calc
|RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_RSE +     |
|         RIL_REQ_REF_DATA_WB_RFO_STORE_RSE    |   0.00160|LLC_STORE_RSE.calc
|             RIL_REQ_REF_DATA_STORE_RSE -     |
|                 LLC_REF_MISS_DATA_STORE_RSE  |   0.00159|LLC_STORE_RSE_HIT.calc
|                       LLC_STORE_INT.calc +   |
|                        LLC_STORE_FP.calc +   |
|                       LLC_STORE_RSE.calc +   |
|                          LLC_SEMAPHORE.calc  |   4.70241|LLC_WRITE.calc
|                   LLC_STORE_INT_HIT.calc +   |
|                    LLC_STORE_FP_HIT.calc +   |
|                   LLC_STORE_RSE_HIT.calc +   |
|                      LLC_SEMAPHORE_HIT.calc  |   4.61714|LLC_WRITE_HIT.calc
|                          MEM_LFETCH.calc +   |
|                    MEM_HWDPREF_NOBUDDY.calc  |   0.43120|MEM_DPREF.calc
|                            MEM_LOAD.calc +   |
|                          MEM_LFETCH.calc +   |
|                            MEM_HWDPREF.calc  |   1.09233|MEM_DREAD.calc
|                           MEM_DREAD.calc +   |
|                              MEM_WRITE.calc  |   1.17760|MEM_DREF.calc
|                 LLC_REF_MISS_DATA_HW_PREF *  |
|                  HWDPREF_BUDDY_FRACTION.calc |   0.30932|MEM_HWDPREF_BUDDY.calc
|                 LLC_REF_MISS_DATA_HW_PREF -  |
|                       MEM_HWDPREF_BUDDY.calc |   0.31124|MEM_HWDPREF_NOBUDDY.calc
|                   LLC_REF_MISS_DATA_LOAD_ANY |   0.35181|MEM_LOAD.calc
|                       MEM_STORE_INT.calc +   |
|                        MEM_STORE_FP.calc +   |
|                       MEM_STORE_RSE.calc +   |
|                         MEM_SEMAPHORE.calc   |   0.08527|MEM_WRITE.calc
|                      MLD_LFETCH_HIT.calc +   |
|                        MLD_HWDPREF_HIT.calc  |  15.82520|MLD_DPREF_HIT.calc
|                MLD_LFETCH_MISS_DROP.calc +   |
|                  MLD_HWDPREF_MISS_DROP.calc  |  16.14740|MLD_DPREF_MISS_DROP.calc
|                MLD_LFETCH_SECONDARY.calc +   |
|                  MLD_HWDPREF_SECONDARY.calc  |   0.43963|MLD_DPREF_SECONDARY.calc
```

MLD_LOAD_INT_HIT.calc +		
MLD_LOAD_FP_HIT.calc +		
MLD_LOAD_RSE_HIT.calc +		
MLD_LOAD_HPW_HIT.calc +		
MLD_LFETCH_HIT.calc +		
MLD_HWDPREF_HIT.calc	23.74735	MLD_DREAD_HIT.calc
MLD_DREF_HIT.cares +		
MLD_SMQ_REF_HIT +		
MLD_DPREF_MISS_DROP.io +		
MLD_WRITE_COALESCED.calc	128.32017	MLD_DREF.calc
MLD_DREAD_HIT.calc +		
MLD_WRITE_HIT.calc	97.62069	MLD_DREF_HIT.calc
	4.54245	MLD_HINT_PREF_DROP_HW_PREF
	0.00000	MLD_HINT_PREF_DROP_LFETCH
MLD_HINT_PREF_DROP_HW_PREF +		
MLD_REF_SECONDARY_DROP_HW_PREF +		
MLD_SMQ_REF_SECONDARY_DROP_HW_PREF	4.55417	MLD_HWDPREF_MISS_DROP.calc
MLD_HINT_PREF_DROP_LFETCH +		
MLD_REF_SECONDARY_DROP_LFETCH +		
MLD_SMQ_REF_SECONDARY_DROP_LFETCH	11.59322	MLD_LFETCH_MISS_DROP.calc
	0.18405	MLD_LOAD_LOAD_RSE
	0.00132	MLD_LOAD_MISS_LOAD_RSE
MLD_LOAD_LOAD_RSE -		
MLD_LOAD_MISS_LOAD_RSE	0.18273	MLD_LOAD_RSE_HIT.calc
	1.34891	MLD_REF_PRIMARY_HW_PREF
	0.01173	MLD_REF_SECONDARY_DROP_HW_PREF
	11.59196	MLD_REF_SECONDARY_DROP_LFETCH
	14.55208	MLD_SMQ_REF_HIT
	0.00000	MLD_SMQ_REF_SECONDARY_DROP_HW_PREF
	0.00127	MLD_SMQ_REF_SECONDARY_DROP_LFETCH
RIL_REQ_REF_DATA_NC_WRITE_WC_MLD -		
RIL_REQ_REF_DATA_NC_WRITE_WC_ANY	0.00000	MLD_WRITE_COALESCED.calc
MLD_STORE_INT_HIT.calc +		
MLD_STORE_FP_HIT.calc +		
MLD_STORE_RSE_HIT.calc +		
MLD_SEMAPHORE_HIT.calc	73.87334	MLD_WRITE_HIT.calc
MLD_STORE_INT_SECONDARY.calc +		
MLD_STORE_FP_SECONDARY.calc +		
MLD_STORE_RSE_SECONDARY.calc +		
MLD_SEMAPHORE_SECONDARY.calc	13.03210	MLD_WRITE_SECONDARY.calc
	12.70759	RIL_REQ_REF_DATA_HW_PREF
	3.07494	RIL_REQ_REF_DATA_LFETCH
	2.33462	RIL_REQ_REF_DATA_LOAD_ANY
	0.00025	RIL_REQ_REF_DATA_LOAD_FP
	0.08633	RIL_REQ_REF_DATA_LOAD_HPW
	2.24828	RIL_REQ_REF_DATA_LOAD_INT
	0.00026	RIL_REQ_REF_DATA_LOAD_RSE
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_ANY_SEMAPHORE
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_FP
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_INT
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_ANY_STORE_RSE
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_WC_ANY
	0.00000	RIL_REQ_REF_DATA_NC_WRITE_WC_MLD
	0.00000	RIL_REQ_REF_DATA_SEMAPHORE
	4.54502	RIL_REQ_REF_DATA_STORE_FP
	0.15579	RIL_REQ_REF_DATA_STORE_INT
	0.00160	RIL_REQ_REF_DATA_STORE_RSE
	0.00000	RIL_REQ_REF_DATA_WB_RFO_SEMAPHORE
	4.54502	RIL_REQ_REF_DATA_WB_RFO_STORE_FP
	0.15579	RIL_REQ_REF_DATA_WB_RFO_STORE_INT
	0.00160	RIL_REQ_REF_DATA_WB_RFO_STORE_RSE
DTB_LOAD_NAT.calc +		
FLDTLB_DPREF_MISS_DROP.calc	0.02332	TLB_DREF.calc

B.5 Instruction Fetch Outcomes

variable name	value
FLI_READ_ANY	264.38225
FLI_IFETCH.calc	256.75359
FLI_IFETCH_HIT.calc	255.59265
FLI_IDMND_HIT.calc	254.43939
FLI_IPREF_HIT.calc	1.15326
FLI_IFETCH_SECONDARY.calc	1.16094
MLI_IFETCH_HIT.calc	7.41895
MLI_IDMND_HIT.calc	2.58748
MLI_IPREF_HIT.calc	4.83147
LLC_IFETCH_HIT.calc	0.01058
LLC_REF_MISS_INST_ANY	0.00303

calculations	value	variable name
	0.70470	FLI_FETCH_RAB_HIT_DMND
	0.45624	FLI_FETCH_RAB_HIT_PREF
FLI_READ_DMND -		

```
|       FLI_READ_MISS_DMND    | 254.43939|FLI_IDMND_HIT.calc
|FLI_IFETCH_HIT.calc       +  |
|FLI_IFETCH_SECONDARY.calc    | 256.75359|FLI_IFETCH.calc
|       FLI_IDMND_HIT.calc +  |
|       FLI_IPREF_HIT.calc    | 255.59265|FLI_IFETCH_HIT.calc
|   FLI_FETCH_RAB_HIT_DMND +  |
|   FLI_FETCH_RAB_HIT_PREF    |   1.16094|FLI_IFETCH_SECONDARY.calc
|       FLI_READ_PREF      -  |
|       FLI_READ_MISS_PREF    |   1.15326|FLI_IPREF_HIT.calc
|                             | 257.73625|FLI_READ_DMND
|                             |   3.29686|FLI_READ_MISS_DMND
|                             |   5.29934|FLI_READ_MISS_PREF
|                             |   6.45260|FLI_READ_PREF
|  RIL_REQ_REF_INST_WB_ANY -  |
|  LLC_REF_MISS_INST_ANY      |   0.01058|LLC_IFETCH_HIT.calc
|  MLI_READ_HIT_DMND_LRU   +  |
|  MLI_READ_HIT_DMND_NOLRU    |   2.58748|MLI_IDMND_HIT.calc
|       MLI_IDMND_HIT.calc +  |
|       MLI_IPREF_HIT.calc    |   7.41895|MLI_IFETCH_HIT.calc
|  MLI_READ_HIT_PREF_LRU   +  |
|  MLI_READ_HIT_PREF_NOLRU    |   4.83147|MLI_IPREF_HIT.calc
|                             |   0.01806|MLI_READ_HIT_DMND_LRU
|                             |   2.56941|MLI_READ_HIT_DMND_NOLRU
|                             |   0.02427|MLI_READ_HIT_PREF_LRU
|                             |   4.80719|MLI_READ_HIT_PREF_NOLRU
|                             |   0.01362|RIL_REQ_REF_INST_WB_ANY
```

B.6 Branch Prediction Outcomes

```
|                                      variable name|         value
|-----------------------------------------------------|--------------
|BR_PRED.calc                                        |122.18199
|  BR_PRED_CORRECT.calc                              |  116.44795
|  BR_PRED_WRONG.calc                                |  5.73404
|    BR_PRED_WRONG_PATH.calc                         |    5.15990
|      BR_PRED_DETAIL_IPREL_WRONG_PATH               |      5.14521
|      BR_PRED_DETAIL_NON_RETIND_WRONG_PATH          |      0.00051
|      BR_PRED_DETAIL_RETURN_WRONG_PATH              |      0.01418
|    BR_PRED_WRONG_TARG.calc                         |    0.57281
|      BR_PRED_DETAIL_IPREL_WRONG_TARGET             |      0.00006
|      BR_PRED_DETAIL_NON_RETIND_WRONG_TARGET        |      0.55617
|      BR_PRED_DETAIL_RETURN_WRONG_TARGET            |      0.01658
|    BR_BE_PRED_DETAIL_ANY_RETIRED                   |    0.00133
```

```
|                                 calculations|     value|               variable name
|---------------------------------------------|----------|---------------------------
|                  BR_PRED_CORRECT.calc +     |
|                  BR_PRED_WRONG.calc         | 122.18199|030_BR_PRED.calc
|         BR_PRED_DETAIL_ANY_CORR_PRED  -     |
|         BR_BE_PRED_DETAIL_ANY_RETIRED       | 116.44795|BR_PRED_CORRECT.calc
|                                             | 116.44928|BR_PRED_DETAIL_ANY_CORR_PRED
|         BR_PRED_WRONG_PATH.calc        +    |
|         BR_PRED_WRONG_TARG.calc        +    |
|         BR_BE_PRED_DETAIL_ANY_RETIRED       |   5.73404|BR_PRED_WRONG.calc
|  BR_PRED_DETAIL_IPREL_WRONG_PATH       +    |
|  BR_PRED_DETAIL_NON_RETIND_WRONG_PATH  +    |
|  BR_PRED_DETAIL_RETURN_WRONG_PATH           |   5.15990|BR_PRED_WRONG_PATH.calc
|BR_PRED_DETAIL_IPREL_WRONG_TARGET       +    |
|BR_PRED_DETAIL_NON_RETIND_WRONG_TARGET  +    |
|BR_PRED_DETAIL_RETURN_WRONG_TARGET           |   0.57281|BR_PRED_WRONG_TARG.calc
```

B.7 Latency Calculations

B.7.1 Replay Latencies

Instead of continuously replaying while waiting for the resolution of some hazard, the pipeline sometimes replays and then stalls the pipeline at the instruction issue point until the hazard is resolved. Some of the cycle accounting PMU events associated with these replays are pairs of events where one event counts the cycles associated with the initial replay and the other event counts the cycles associated with the resulting

instruction issue stall. In these cases, the average latency of the replay event can be calculated as shown in the report in this section. Note that the constants used in the calculations are the number of cycles associated with the various different initial replays.

variable name	value
AVG_BLK_HPW_REPLAY_LATENCY.calc	45.61036
AVG_FR_LOAD_REPLAY_LATENCY.calc	13.62453
AVG_GR_LOAD_REPLAY_LATENCY.calc	45.83577
AVG_LOAD_ACQ_REPLAY_LATENCY.calc	19.08641
AVG_OZQ_FULL_REPLAY_LATENCY.calc	123.77092

calculations	value	variable name
(CYC_BE_WB2_REPLAY_BLK_HPW + CYC_BE_IBD_STALL_HPW) / (CYC_BE_WB2_REPLAY_BLK_HPW/7)	45.61036	AVG_BLK_HPW_REPLAY_LATENCY.calc
(CYC_BE_IBD_STALL_FR_LOAD + CYC_BE_EXE_REPLAY_FR_LOAD_RAW + CYC_BE_EXE_REPLAY_FR_LOAD_WAW) / ((CYC_BE_EXE_REPLAY_FR_LOAD_RAW/4) + (CYC_BE_EXE_REPLAY_FR_LOAD_WAW/4))	13.62453	AVG_FR_LOAD_REPLAY_LATENCY.calc
(CYC_BE_IBD_STALL_GR_LOAD + CYC_BE_EXE_REPLAY_GR_LOAD_RAW + CYC_BE_EXE_REPLAY_GR_LOAD_WAW + CYC_BE_DET_REPLAY_GR_LOAD) / ((CYC_BE_EXE_REPLAY_GR_LOAD_RAW/4) + (CYC_BE_EXE_REPLAY_GR_LOAD_WAW/4) + (CYC_BE_DET_REPLAY_GR_LOAD/5))	45.83577	AVG_GR_LOAD_REPLAY_LATENCY.calc
(CYC_BE_IBD_STALL_ACQ + CYC_BE_WB2_REPLAY_LOAD_ACQ + CYC_BE_DET_REPLAY_LOAD_ACQ) / (CYC_BE_DET_REPLAY_LOAD_ACQ/5)	19.08641	AVG_LOAD_ACQ_REPLAY_LATENCY.calc
(CYC_BE_IBD_STALL_OZQFULL + CYC_BE_WB2_REPLAY_OZQ_FULL) / (CYC_BE_WB2_REPLAY_OZQ_FULL/7)	123.77092	AVG_OZQ_FULL_REPLAY_LATENCY.calc
	70.07033	CYC_BE_DET_REPLAY_GR_LOAD
	0.00004	CYC_BE_DET_REPLAY_LOAD_ACQ
	32.51587	CYC_BE_EXE_REPLAY_FR_LOAD_RAW
	0.15557	CYC_BE_EXE_REPLAY_FR_LOAD_WAW
	12.47120	CYC_BE_EXE_REPLAY_GR_LOAD_RAW
	0.21218	CYC_BE_EXE_REPLAY_GR_LOAD_WAW
	0.00006	CYC_BE_IBD_STALL_ACQ
	78.61186	CYC_BE_IBD_STALL_FR_LOAD
	704.92995	CYC_BE_IBD_STALL_GR_LOAD
	112.28562	CYC_BE_IBD_STALL_HPW
	0.49379	CYC_BE_IBD_STALL_OZQFULL
	20.35721	CYC_BE_WB2_REPLAY_BLK_HPW
	0.00006	CYC_BE_WB2_REPLAY_LOAD_ACQ
	0.02960	CYC_BE_WB2_REPLAY_OZQ_FULL

B.7.2 Exposed Data Access Latencies

Some of the cycle accounting events can be associated with specific data reference types. In these cases, average exposed latencies per associated access can be calculated.

variable name	value
AVG_DATA_ACCESS_TOTAL_EXPOSED.calc	0.38462
AVG_LOAD_FP_EXPOSED.calc	12.48634
AVG_LOAD_INT_EXPOSED.calc	0.36484

calculations	value	variable name
DATA_ACCESS.calc / (DATA_REF_LOAD_INT + DATA_REF_LOAD_FP + DATA_REF_STORE_INT + DATA_REF_STORE_FP + DATA_REF_SEMAPHORE)	0.38462	AVG_DATA_ACCESS_TOTAL_EXPOSED_LATENCY.calc
(CYC_BE_IBD_STALL_FR_LOAD + CYC_BE_EXE_REPLAY_FR_LOAD_RAW + CYC_BE_EXE_REPLAY_FR_LOAD_WAW) / DATA_REF_LOAD_FP	12.48634	AVG_LOAD_FP_EXPOSED_LATENCY.calc

(CYC_BE_IBD_STALL_GR_LOAD + CYC_BE_EXE_REPLAY_GR_LOAD_RAW + CYC_BE_EXE_REPLAY_GR_LOAD_WAW + CYC_BE_DET_REPLAY_GR_LOAD) / DATA_REF_LOAD_INT		
	0.36484	AVG_LOAD_INT_EXPOSED_LATENCY.calc
	13.54817	CYC_BE_DET_REPLAY_GR_LOAD
	4.94232	CYC_BE_EXE_REPLAY_FR_LOAD_RAW
	0.05605	CYC_BE_EXE_REPLAY_FR_LOAD_WAW
	4.00399	CYC_BE_EXE_REPLAY_GR_LOAD_RAW
	0.08480	CYC_BE_EXE_REPLAY_GR_LOAD_WAW
	2.96305	CYC_BE_IBD_STALL_FR_LOAD
	17.14045	CYC_BE_IBD_STALL_GR_LOAD
(see Section B.2)	51.21450	DATA_ACCESS.calc
	0.63761	DATA_REF_LOAD_FP
	95.32248	DATA_REF_LOAD_INT
	0.00000	DATA_REF_SEMAPHORE
	2.04434	DATA_REF_STORE_FP
	35.15017	DATA_REF_STORE_INT

B.7.3 Average Lifetimes in Queues

The report in this section uses Little's Law from queueing theory to calculate the average lifetime of data accesses in various queues in the cache hierarchy.

variable name	value
AVG_LIFETIME_IN_MLD_FAB.calc	54.14154
AVG_LIFETIME_IN_MLD_OZDATA.calc	7.77252
AVG_LIFETIME_IN_MLD_SMQ.calc	33.31547
AVG_LIFETIME_IN_OZQ.calc	3.48710
AVG_LIFETIME_IN_RIL_DRQ.calc	41.02823

calculations	value	variable name
	541.38471	CPU_CYCLES
	0.00000	DATA_REF_SEMAPHORE
	40.65089	DATA_REF_STORE_ANY
MLD_FAB_AVG_OCCUPANCY.mld / (MLD_REF_PRIMARY/CPU_CYCLES) ((MLD_FAB_COUNT_MSB * 2) + MLD_FAB_COUNT_LSB) / CPU_CYCLES	54.14154	AVG_LIFETIME_IN_MLD_FAB.mld
	0.06557	MLD_FAB_AVG_OCCUPANCY.mld
	18.02661	MLD_FAB_COUNT_LSB
	8.73487	MLD_FAB_COUNT_MSB
MLD_OZDATA_AVG_OCCUPANCY.calc / ((DATA_REF_STORE_ANY + DATA_REF_SEMAPHORE) / CPU_CYCLES) ((MLD_OZDATA_COUNT_MSB*4) + MLD_OZDATA_COUNT_LSB) / CPU_CYCLES	7.77252	AVG_LIFETIME_IN_MLD_OZDATA.calc
	0.58361	MLD_OZDATA_AVG_OCCUPANCY.calc
	161.46801	MLD_OZDATA_COUNT_LSB
	38.62297	MLD_OZDATA_COUNT_MSB
MLD_OZQ_AVG_OCCUPANCY.calc / (MLD_OZQ_INSERT / CPU_CYCLES) ((MLD_OZQ_COUNT_MSB*2) + MLD_OZQ_COUNT_LSB) / CPU_CYCLES	3.48710	AVG_LIFETIME_IN_OZQ.calc
	0.44833	MLD_OZQ_AVG_OCCUPANCY.calc
	106.81733	MLD_OZQ_COUNT_LSB
	67.95093	MLD_OZQ_COUNT_MSB
	69.60497	MLD_OZQ_INSERT
	0.65562	MLD_REF_PRIMARY
	0.91615	MLD_REF_SECONDARY
MLD_SMQ_AVG_OCCUPANCY.calc / (MLD_REF_SECONDARY / CPU_CYCLES) ((MLD_SMQ_COUNT_MSB*2) + MLD_SMQ_COUNT_LSB) / CPU_CYCLES	33.31547	AVG_LIFETIME_IN_MLD_SMQ.calc
	0.05638	MLD_SMQ_AVG_OCCUPANCY.calc
	5.88356	MLD_SMQ_COUNT_LSB
	12.31914	MLD_SMQ_COUNT_MSB
RIL_DRQ_AVG_OCCUPANCY.calc / ((RIL_REQ_REF_DATA_WB_ANY + RIL_REQ_REF_DATA_NC_READ_ANY + RIL_REQ_OTHER_DRQ_ANY) / CPU_CYCLES) ((RIL_DRQ_VALID_MSB*4) + RIL_DRQ_VALID_LSB) / CPU_CYCLES	41.02823	AVG_LIFETIME_IN_RIL_DRQ.calc
	0.09823	RIL_DRQ_AVG_OCCUPANCY.calc
	31.09798	RIL_DRQ_VALID_LSB
	5.52032	RIL_DRQ_VALID_MSB
	0.00000	RIL_REQ_OTHER_DRQ_ANY

```
|                                    |   0.00000|RIL_REQ_REF_DATA_NC_READ_ANY
|                                    |   1.29616|RIL_REQ_REF_DATA_WB_ANY
```

B.8 Data Prefetching

B.8.1 Data Prefetch Queue Insertions

The report in this section gives a breakdown of insertions into the data prefetch queue.

```
|                          variable name|          value
|---------------------------------------|---------------
|DPFQ_ENQ_ANY                           |34.46540
|  DPFQ_INS_LFETCH.calc                 |  23.46216
|    DFFQ_INS_LFETCH_ON_A.calc          |    22.75452
|    FLD_HWPREF_INS_OZQ_FULL_LFETCH     |     0.70764
|    DPFQ_ENQ_LFETCH_COUNT              |     0.00000
|  DPFQ_ENQ_MLD                         |   0.92920
|  DPFQ_INS_FLD_NBR.calc                |   8.46529
|    DPFQ_ENQ_FLD_BIDI                  |     5.48542
|    DPFQ_ENQ_FLD_BWD                   |     0.53471
|    DPFQ_ENQ_FLD_FWD                   |     2.44516
|  DPFQ_INS_FLD_TARGET.calc             |   1.78725
|    FLD_HWPREF_INS_ACQ_PEND            |     0.00109
|    FLD_HWPREF_INS_CANCEL_FILL         |     0.25304
|    FLD_HWPREF_INS_DTLB_MISS           |     0.00208
|    FLD_HWPREF_INS_DTLB_MISS_LFETCH    |     0.38969
|    FLD_HWPREF_INS_FLDTLB_MISS_LFETCH  |     0.07785
|    FLD_HWPREF_INS_FLDTLB_MISS         |     0.93321
|    FLD_HWPREF_INS_FLUSH_DST           |     0.12012
|    FLD_HWPREF_INS_OZQ_FULL            |     0.01015
|    FLD_HWPREF_INS_REL_OP              |     0.00000
|    FLD_HWPREF_INS_STORE_ALIAS         |     0.00002
|  DPFQ_INS_RSE.calc                    |   0.21248
|    DPFQ_ENQ_RSE_LOAD                  |     0.07291
|    DPFQ_ENQ_RSE_STORE                 |     0.13957
|    DPFQ_ENQ_MOV_BSPST                 |     0.00000
```

```
|                        calculations|    value|                        variable name
|------------------------------------|---------|-----------------------------------
|   DPFQ_ENQ_LFETCH                 -|         |
|   FLD_HWPREF_INS_OZQ_FULL_LFETCH   |  22.75452|DFFQ_INS_LFETCH_ON_A.calc
|                                    |  23.46216|DPFQ_ENQ_LFETCH
|                                    |   0.21248|DPFQ_ENQ_RSE_ANY
|                DPFQ_ENQ_FLD_BIDI +|         |
|                DPFQ_ENQ_FLD_BWD  +|         |
|                DPFQ_ENQ_FLD_FWD    |   8.46529|DPFQ_INS_FLD_NBR.calc
|FLD_HWPREF_INS_ACQ_PEND +           |         |
|FLD_HWPREF_INS_CANCEL_FILL +        |         |
|FLD_HWPREF_INS_DTLB_MISS   +        |         |
|FLD_HWPREF_INS_DTLB_MISS_LFETCH +   |         |
|FLD_HWPREF_INS_FLDTLB_MISS_LFETCH + |         |
|FLD_HWPREF_INS_FLDTLB_MISS        + |         |
|FLD_HWPREF_INS_FLUSH_DST          + |         |
|FLD_HWPREF_INS_OZQ_FULL           + |         |
|FLD_HWPREF_INS_REL_OP             + |         |
|FLD_HWPREF_INS_STORE_ALIAS          |   1.78725|DPFQ_INS_FLD_TARGET.calc
|             DPFQ_ENQ_LFETCH      + |         |
|             DPFQ_ENQ_LFETCH_COUNT  |  23.46216|DPFQ_INS_LFETCH.calc
| DPFQ_ENQ_RSE_ANY+DPFQ_ENQ_MOV_BSPST|   0.21248|DPFQ_INS_RSE.calc
```

C 'Data Fetch Software Optimization Opportunities and Examples

C.1 Transitioning from 4 M-ports to 2

Previous Intel Itanium processors had 2 load pipelines and 2 store pipelines. The processor has 2 load/store pipelines. Rescheduling for processor's pipelines can, in some cases, result in substantial speedup. The following guidelines can be used to help optimize scheduling for the memory pipelines:

C.1.1 Avoiding 4 M's in instruction group

A long stream (e.g. in a loop) of instruction groups that require 4 M pipes per group is an OK schedule for both processor and previous Intel Itanium processors:

```
loop:
   st8 [r10]=r6,8
   st8 [r11]=r7,8
   ld8 r6=[r8],8
   ld8 r7=[r9],8
   br.cloop loop;;
```

The processor will take 2 cycles to issue each instruction group where previous Intel Itanium processors will take only 1, but there isn't any re-ordering that software can do to improve this situation. However, *software should avoid scheduling instruction groups that require 4 M pipes adjacent to instruction groups that require no M pipes:*

```
loop:
   st8 [r10]=r6,8
   st8 [r11]=r7,8
   ld8 r6=[r8],8
   ld8 r7=[r9],8;;
   add r6=1,r6
   add r7=1,r7
   br.cloop loop;;
```

An iteration of the loop above would take 3 issue cycles on processor and 2 issue cycles on previous Intel Itanium processors. A much better processor schedule, when possible, would be to rearrange the instructions so that there are only 2 M pipes required by any given instruction group:

```
loop:
   add r6=1,r6
   add r7=1,r7
   st8 [r10]=r6,8
   st8 [r11]=r7,8;;
   ld8 r6=[r8],8
   ld8 r7=[r9],8
```

```
   br.cloop loop;;
```

An iteration of the loop above would take 2 issue cycles on both processor and previous Intel Itanium processors.

C.1.2 Avoiding 3 M's in instruction group

Software should avoid scheduling instruction groups that require 3 M pipes to issue:

```
loop:
   st8 [r10]=r6,8
   st8 [r11]=r6,8
   ld8 r6=[r8],8
   br.cloop loop;;
```

An iteration of the above loop would take 2 issue cycles on processor and 1 issue cycle on previous Intel Itanium processors. Unrolling and rescheduling the loop as follows would reduce the issue cycles per original iteration to 1.5 on processor and leave issue cycles unchanged on previous Intel Itanium processors:

```
loop:
   st8 [r10]=r6,16
   st8 [r11]=r6,16
   ld8 r6=[r8],16
   st8 [r12]=r7,16
   st8 [r13]=r7,16
   ld8 r7=[r9],16
   br.cloop loop;;
```

C.1.3 Scheduling lfetches on an A Pipeline

C.1.3.1 Schedule lfetch-on-A When M Not Available

On the processor, most lfetches can be issued to an A pipeline. Therefore, *software should schedule lfetches on A pipes when M pipes are not available.* The following loop will require only 2 issue cycles per iteration on processor:

```
loop:
   st8 [r2]=r1,8    // issued on M-pipe
   ld8 r4=[r3],8    // issued on M-pipe
   lfetch [r9],8    // issued on A-pipe
   st8 [r6]=r5,8    // issued on M-pipe
   ld8 r8=[r7],8    // issued on M-pipe
   br.cloop loop;;
```

C.1.3.2 Avoid Scheduling lfetch-on-A Every Cycle

Lfetches scheduled on A-ports at a rate of at least one every issue cycle are at risk for being dropped due to overflowing the DPFQ. Therefore, *if the cost of dropping some lfetches is large, software should avoid scheduling lfetches on A-ports every cycle in a loop probably even at the cost of adding an issue cycle.*

C.1.3.3 Keep lfetches from Same Stream on Either A or M Pipelines

On the processor, lfetches issued on the A pipes will be issued out of order (later) with respect to lfetches issued on the M pipes. The order of a stream of prefetches is often connected to the order in which a stream of loads or stores accesses memory. Changing the order of the prefetches with respect to the order of the dependent memory instructions can lower performance in some cases. Therefore, *when software would prefer a stream of lfetches to be issued in order, it should schedule the entire stream on either A pipes or M pipes, not both.* Assuming the lfetches and loads in the following two examples are interleaved on the same address stream, the second schedule below will sometimes outperform the first:

```
loop:
   st8 [r2]=r1,8    // issued on M-pipe
   ld8 r4=[r3],8    // issued on M-pipe
   st8 [r6]=r5,8    // issued on M-pipe
   ld8 r8=[r7],8    // issued on M-pipe
   lfetch [r9],8    // issued on A-pipe
   st8 [r12]=r11,8  // issued on M-pipe
   ld8 r14=[r13],8  // issued on M-pipe
   st8 [r16]=r15,8  // issued on M-pipe
   lfetch [r19],8   // issued on M-pipe
   br.cloop loop;;
```

```
loop:
   st8 [r2]=r1,8    // issued on M-pipe
   ld8 r4=[r3],8    // issued on M-pipe
   st8 [r6]=r5,8    // issued on M-pipe
   ld8 r8=[r7],8    // issued on M-pipe
   lfetch [r9],8    // issued on A-pipe
   st8 [r12]=r11,8  // issued on M-pipe
   ld8 r14=[r13],8  // issued on M-pipe
   lfetch [r19],8   // issued on A-pipe
   st8 [r16]=r15,8  // issued on M-pipe
   br.cloop loop;;
```

C.1.3.4 Potential Conflict Between A-port and M-port

If there is an lfetch on A0 and an lfetch or some other prefetch on M0 that needs to go into the DPFQ, only the lfetch on A0 will go into the DPFQ and the lfetch/prefetch on M0 will be dropped. The same is true for for A1 vs. M1. *Therefore, software should avoid scheduling an lfetch on A0 simultaneous with a lfetch on M0 that might need to use the DPFQ (e.g. a counted lfetch or an lfetch that is likely to miss the DTB). The same applies to A1 vs. M1.*

C.2 Data Memory Reference Clustering

C.2.1 Load Clustering

All Intel Itanium processors issue instructions to the execution pipelines in the order that software schedules them. Data memory references that miss in a cache or TLB do not block the main pipeline until the queues holding them fill up. However, a use of a register that is the target of a load that missed a cache or TLB does block the main pipeline:

```
   ld8 r4=[r8];;
```

```
add r5=1,r4
ld8 r6=[r9];;
add r7=1,r6
```

If the first load in the above example misses, the add instruction will block the main pipeline. If the second load misses, the second add will also block the pipeline. The execution time will include the sum of the 2 load miss latencies. However, if the code were rescheduled as follows, execution would include the maximum of the 2 load latencies instead of the sum:

```
ld8 r4=[r8]
ld8 r6=[r9];;
add r5=1,r4
add r7=1,r6
```

Therefore, *software should attempt to cluster loads such that there are no load uses between them in order to overlap potential cache or TLB miss latencies.* The more likely loads are to miss a cache or TLB, the more potential they have to benefit from being clustered.

C.2.2 Lfetch / Load Clustering

Just like loads, lfetches can have longer latencies due to cache or TLB misses. The latency of an lfetch can prevent it from being useful:

```
ld8 r4=[r8];;
add r5=1,r4
lfetch [r9]
ld8 r6=[r9];;
add r7=1,r6
```

It seems pretty obvious that the lfetch above isn't going to provide any value. It is scheduled in the same cycle as the corresponding load. However, it might not at first be obvious that in the following schedule can be much better:

```
ld8 r4=[r8]
lfetch [r9];;
add r5=1,r4
ld8 r6=[r9];;
add r7=1,r6
```

The lfetch is now scheduled only one issue cycle before the corresponding load. However, if both cache lines ([r8] and [r9]) are in the MLD but not the FLD, then moving the lfetch before the first add saves 8 cycles. The cycles are saved because the lfetch and the first load issue at the same time, before the add blocks the pipeline. By the time the first load is finished, so is the lfetch, and the second load hits in the FLD. Therefore, *software should attempt to cluster loads and lfetches such that there are no load uses between them in order to overlap potential cache or TLB miss latencies.* Also, *lfetch/load clustering might be something a post-compilation optimizer could safely do to improve code with poor load clustering.* Section C.5 discusses the tradeoffs between speculating loads and lfetching.

C.2.3 Store/Load Declustering

Stores cancel in-flight FLD fills:

```
ld8 r4=[r8]      // say [r8] is not in FLD, triggers FLD fill
st8 [r8]=r5;;    // this store cancels the FLD fill
add r6=r4,r3
ld8 r4=[r8]      // so this load to the same line misses FLD again
```

Therefore, *software should avoid scheduling stores that might be to the same line as a load before the first use of that load:*

```
ld8 r4=[r8];;
add r6=r4,r3     // blocks pipeline until load has filled the FLD
st8 [r8]=r5      // no in-flight FLD fill to kill
ld8 r4=r8        // so this load hits in the FLD
```

C.3 Control Speculation

The Intel Itanium instruction set provides architectural support for software based control speculation. Control speculation is defined in the architectural manual as "the execution of an operation before the branch which guards it." This architectural support allows software, by following specific rules, to reorder instructions with respect to preceding branches and still produce the same results as the original code ordering. The purpose of this capability is to allow software to expose more instruction level parallelism to the hardware and thus increase performance.

As with all speculation, control mis-speculation can incur performance penalties. For example, executing an unneeded add instruction utilizes pipeline execution resources for no benefit. The wasted usage of those resources can prevent more useful usage of the resources. The trick of speculation is to maximize the difference between the benefits of correct speculation and the costs associated with mis-speculations. The processor design contains a number of changes intended to reduce the cost of control mis-speculation.

In previous Intel Itanium processor designs, control speculation was limited by several scenarios that incurred significant control mis-speculation penalties.

- blocking DTB misses by mis-speculated control speculative loads
- hazards with the target registers of long latency mis-speculated control speculative loads
- blocking FLDTLB misses by mis-speculated control speculative loads

The processor design reduces the penalties in each of these scenarios. See the DTB, MLD, and FLDTLB sections for more details. As a result of these reductions in control mis-speculation penalties, *software should be able to do more control speculation for a given penalty, and thus, hopefully, see more performance benefit from such speculation.*

C.4 Software Data Prefetching

The Intel Itanium instruction set provides architectural support for data prefetching via an lfetch instruction. This lfetch instruction allows software to prefetch a line into various levels of the cache hierarchy for the purpose of exposing more instruction level parallelism to the hardware and thus increasing performance.

As with all speculation, data prefetching can incur performance penalties. For example, executing an unneeded lfetch instruction utilizes pipeline execution resources and cache hierarchy resources for no benefit. The wasted usage of those resources can prevent more useful usage of the resources. The trick of data prefetching is to maximize the difference between the benefits of useful prefetches and the costs associated with useless prefetches. The processor design contains a number of changes intended to reduce the cost of data prefetching via lfetch instructions.

- lfetch-on-A: reduces the cost of issuing an lfetch
- higher FLD fill bandwidth: limits the competition for FLD fill bandwidth between lfetches and other operations
- PF_DROP hints: limits the bandwidth utilization of the cache/TLB hierarchy by lfetches
- cache LOCALITY hints: limits the pollution of FLD by lfetches
- lower MLD secondary miss penalty: reduces the break-even point for scheduling an lfetch ahead of an MLD miss to about 7 cycles
- MLD queue optimizations: MLD hitting lfetches use less valuable resources in MLD than they did in previous Intel Itanium designs

As a result, *software should be able to do more lfetching for a given penalty, and thus, hopefully, see more performance benefit from such prefetching.*

C.4.1 Managing the Cost of an lfetch

The less sure software is of the value of an lfetch, the more it will want to minimize the cost of the lfetch. The more sure software is of the value of an lfetch, the higher cost it will be willing to pay to complete the prefetch and to treat it with higher priority. The processor design provides a number of features that allow software to manage the cost of an lfetch. There is a whole spectrum of cost choices that software can make. The following are the two extremes.

C.4.1.1 Minimal Cost (Lowest Priority) lfetch

Software can achieve a minimal cost lfetch with the following steps:

- minimal instruction issue cost: insert lfetches into unused M-port and A-port slots after instruction scheduling has been done
- minimal cache/TLB bandwidth utilization: DAHR.PF_DROP=ON_NON_MIN_COST
- minimal cache pollution: DAHR.FLD_LOCALITY=MARK_NRU

C.4.1.2 Highest Priority (Maximum Cost) lfetch

Software can achieve the highest priority prefetch with the following steps:

- highest priority instruction issue: schedule lfetches and any instructions they depend on first; schedule lfetches on M-ports
- highest priority cache/TLB bandwidth utilization: DAHR.PF_DROP=NONE

- highest priority cache allocation: DAHR.LLC_LOCALITY=NORMAL, DAHR.MLD_LOCALITY= =NORMAL, DAHR.FLD_LOCALITY=NORMAL

C.4.2 lfetching in Acyclic Code

Lfetching in acyclic code is difficult for at least two reasons. First, an lfetch must be separated from the first load to that corresponding cache line by at least enough cycles to hide the latency of the MLD in order to be useful. Lfetching a cache line just 2 cycles before the corresponding load does not help because the load will still miss and incur at least an MLD latency. However, *software should keep in mind that an lfetch scheduled between a load from another address and its corresponding use will have additional latency to hide behind when the load misses the FLD.* See Section C.2 on data reference clustering for more discussion of this issue.

The second reason lfetching in acyclic code is difficult is that such lfetches are often speculative and not completely free. The benefit of doing an lfetch needs to (on average) exceed the cost of doing an lfetch. *Due to the potentially lower cost of lfetches on the processor (see Section C.4.1), software should be able to do more lfetching in acyclic code regions.*

C.4.3 Prefetching Data Address Translations

On the processor, FLDTLB and DTB misses are non-blocking. This characteristic is particularly valuable in that it allow potentially unnecessary speculative accesses (e.g. lfetch, speculative load) to avoid blocking the pipeline. However, this also implies that such speculative accesses might not perform a cache line prefetch or might take longer to perform a cache line prefetch. For example, an access that misses the FLDTLB cannot perform an FLD fill. Similarly, a ld.s that misses the DTB typically is deferred and so doesn't perform a cache fill. An lfetch that misses the DTB is buffered in the data prefetch queue and may be reissued after the hardware page walk completes. Therefore, *software should consider prefetching translations into the data TLBs in a manner similar to prefetching cache lines into the data caches.* For example, *in a software pipelined loop, software should consider creating a much sparser (page size dependent) translation lfetch stream of lfetches far enough ahead of the cache line lfetch stream to hide hardware page walker latency.*

C.5 Lfetches vs. Speculative Loads

Lfetches and speculative loads are similar and might be used in similar scenarios. However, they have some differences that can lead to different usage.

C.5.1 Lfetch advantages

Lfetches have several advantages over speculative loads.

C.5.1.1 lfetches are not orderable

Lfetches are not affected by acquire and release semantics. Therefore, *software can use lfetches to improve performance of loads following operations with acquire semantics:*

```
ld8.acq r4=[r8]  // let's say [r8] and [r9] are 2 lines that miss MLD
lfetch [r9]      // this lfetch will not have to wait for ld8.acq
  ld8 r5=[r9]      // this load will hit in MLD when ld8.acq is done
```

Also, *software can use lfetches to improve the performance of stores with release semantics:*

```
lfetch.excl [r8] // let's say [r8] misses all caches
ld8 r5=[r9]      // let's say [r9] misses all caches
st.rel [r8]=r4   // if [r8] is not snooped between lfetch and st.rel,
                 //   store will hit in MLD as soon as ld8 finishes
```

C.5.1.2 lfetches can be issued to A pipes

```
br somewhere;;
ld8 r4=[r8]      // issued to M0, say misses MLD
st8 [r9]=r5      // issued to M1
lfetch [r10]     // issued to A0 - saved a cycle by not using M pipe
(p7) br.cond targ;;  // branch not taken
st8 [r12],r4     // blocks pipeline waiting for r4
ld8 r6=[r10];;
add r27=r15,r16;;
add r14=r6,r27   // use not immediately following load
                 // (see first speculative load advantage)
```

It should be noted that issuing an lfetch on an A pipe can be a disadvantage as well due to the minimum of 7 extra cycles of latency for an lfetch issued on an A pipe vs. an M pipe. Therefore, *software should use lfetches instead of speculative loads when M pipes are not available and there is some chance that the dynamic cycle distance between the lfetch and the corresponding load will be at least 15 cycles (7 + minimum MLD latency) and the cache line being prefetched is not in the FLD.* This may sound unlikely, but it happens all the time in software pipelined loops and can probably happen some in acyclic code as well.

C.5.1.3 the cost of an lfetch can be limited by DAHR.PF_DROP hints

An lfetch with an appropriate PF_DROP hint can avoid causing various kinds of memory hierarchy traffic:

```
lfetch.d1 [r8]       // say d1 contains PF_DROP=ON_DTB_MLD_MISS
                     //   and [r8] is not in any cache
(p7) br.cond targ;;  // say branch is taken
ld8 r5=[r8]
```

The lfetch in the above example will not cause any memory hierarchy traffic past the MLD. A speculative load in the same place would go all the way out to memory. Therefore, *software should use lfetches instead of speculative loads when it wants to limit memory hierarchy traffic.*

C.5.1.4 lfetches don't have target registers

Lfetches do not have target registers. Therefore, *software should use lfetches instead of speculative loads when attempting to hide data access latencies after register allocation (e.g. a late optimization pass in a compiler or a dynamic optimizer).* Also, *software should use lfetches instead of speculative loads when register pressure is likely to be too high to support the live ranges of the speculative load target registers (e.g. in software pipelined loops where the lfetch is scheduled way ahead of the corresponding uses).*

C.5.2 Speculative load advantages

C.5.2.1 speculative loads can minimize data dependence height

The use of a target of a speculative load can occur in the same cycle as the corresponding check load:

```
ld8.s r5=[r8]
(p7) br somewhere;;
ld8.c r5=[r8]
add r6=r5,r4
```

A similar sequence using an lfetch instead of a speculative load has a larger data dependence height:

```
lfetch [r8]
(p7) br somewhere;;
ld8 r5=[r8];;
add r6=r5,r4
```

Therefore, *software should use a speculative load instead of an lfetch when elimination of a single cycle of data dependence height is more important than any advantages of using an lfetch.*

C.5.2.2 speculative loads can be used for indirect prefetching

Some prefetching operations are indirect, meaning they have to first retrieve a value from a memory location to compute the address from which to prefetch:

```
ld8.s r5=[r8];;
lfetch [r5]
```

Software should use a speculative load retrieve a value used to compute a prefetch address.

C.5.2.3 a speculative load doesn't take up space in the data prefetch queue when it misses the DTB

On the processor, lfetches that miss the DTB typically are placed into the data prefetch queue where they attempt to wait until a hardware page walk can be completed and they can be re-issued. Speculative loads that miss the DTB do not take up space in the data prefetch queue. They are deferred (target register gets NATed), and any later non-speculative load will in effect re-try the load. Therefore, *software may, in some circumstances, decide to use speculative loads instead of lfetches in an attempt to reduce pressure on the data prefetch queue*.

C.6 Re-tuning ILP heuristics

The processor has many micro-architecture differences from previous Intel Itanium processors that may change the optimum settings of many experimentally determined software code optimization heuristics. Examples of such microarchitecture changes include:

- reduced and controllable cost of lfetches

- reduced and controllable cost of control speculation
- reduced cost of data speculation
- changes in instruction issue width and mix
- changes in cache/memory latency/bandwidth

Examples of software code optimization heuristics whose settings may have been determined experimentally on previous Intel Itanium processors include:

- aggressiveness of predication
- aggressiveness of control speculation
- aggressiveness of data speculation
- code scheduling algorithms
- aggressiveness of lefetching

Therefore, *all software code generation and optimization heuristics that were tuned experimentally on previous Intel Itanium processors should be re-tuned for the processor.*

C.7 Utilizing Data Access Hints

The processor has support for a variety of data access hints via the Data Access Hint Registers (DAHRs). Many memory instructions (for example, loads, stores, lfetches) point to a DAHR that contains hints from software to hardware about how to optimize such memory accesses. This section will suggest ways that software might use such hints to optimize performance.

C.7.1 Managing data access cost

All of the data access hints defined on the processor have something to do with managing the costs of data accesses. Via these hints, software can control what resources hardware will consume as a result of a data access request.

Consumption of resources is highly correlated with a reduction in the cost of future accesses to the same or related addresses. Therefore, these hints can be thought of as a communication from software to hardware of the likelihood that it will use the cache line or translation of interest during its lifetime in various structures. Software could acquire knowledge of the likelihood of future related cache line usage at a particular point in the program through some sort of general analysis related to memory disambiguation and a kind of cache line live range analysis. Software could also determine that future cache line usage characteristics are correlated with things it might know about the instruction of interest:

- instruction type: lfetch, ld, ld.s, st, ...
- data structure type: struct, array, linked list, ...
- data structure location: stack, heap, ...
- data structure locality: local, global, ...
- data structure size and alignment
- data structure access pattern: stride, hash, ...
- code region type: acyclic, cyclic, ...
- application characteristics

Data access hints can also be thought of as providing software with a dynamic quality evaluator for its speculative accesses. For instance, in acyclic code regions, lfetches are typically much more likely than not to hit in the DTB and the MLD. Therefore, a speculative lfetch that misses the DTB or the MLD is much more likely to be a useless prefetch and may be better off being dropped.

C.7.2 Using Cache Locality Hints

There are two main types of cost associated with filling a particular cache line in a cache. The first is a bandwidth cost associated with copying an entire cache line from further out in the memory hierarchy. The second is the cost of using a cache entry that could be used to store something else. The cache locality hints can be used to reduce both of these costs. They can prevent a line from being copied into a particular cache and they can reduce the lifetime of this line in a particular cache.

C.7.2.1 Using FLD Locality Hints

- FLD_NO_ALLOCATE

  General usage: This hint setting can be used to avoid the space and bandwidth costs associated with moving a cache line into the FLD.

  Useful context: Unlike non-FLD filling accesses, FLD fills use one of the 16 FLDFAB entries, require extra bandwidth in the MLD pipeline, cause replacement of an existing FLD cache line, and can cause DET replays due to conflicts in the main pipeline (see PMU events CYC_BE_DET_REPLAY.WRITE_HIT_VS_FILL and CYC_BE_DET_REPLAY.WRITE_MISS_VS_FILL). If software has pretty good reason to believe that filling the FLD will not provide benefit, then it can potentially improve performance by using this hint to eliminate the cost of the FLD fill.

  Examples of SW opportunities:

  — In a loop where loads are all scheduled at least MLD latency ahead of uses, the loads should probably use this hint.

  — This hint should probably be used for an access pattern with a stride bigger than a cache line.

  This hint should probably be used for sparse array or similar scattered access patterns.

- FLD_NRU

  General usage: This hint setting can be used to minimize pollution of the FLD when lines are allocated more speculatively than normal. This hint setting can also be used to skew the replacement policy to replace lines that software believes are less likely than normal to be accessed again within their FLD lifetime. Note that hardware initiated prefetches will not demote lines to NRU.

  Examples of SW opportunities:

  — If software is going to free some memory (or in any other way stop using some memory for an extended period of time), it could mark the last accesses to the relevant cache lines with this hint. It could also explicitly insert lfetches with this hint before freeing the memory.

  — If software is speculatively lfetching or loading a cache line, it may want to use this hint so the line will be replaced more quickly in the case the lfetch or speculative load was not needed. However, software should consider this carefully because this could cause performance degradations in the case that the cache line is already in the FLD and in use by other accesses.

- FLD_NORMAL

C.7.2.2 Using MLD Locality Hints

- MLD_NO_ALLOCATE

  General usage: This hint setting can be used to avoid the space and bandwidth costs associated with moving a cache line into the MLD.

  Examples of SW opportunities:

  — This hint should probably be used for an access pattern with a stride bigger than a cache line.

  — This hint should probably be used for sparse array or similar scattered access patterns.

  MLD_NRU

  General usage: This hint setting can be used to minimize pollution of the MLD when lines are allocated more speculatively than normal.

  Examples of SW opportunities:

  — If software is lfetching a line that is unlikely to already be in use in the MLD by some other access and there is a reasonable chance that the use targeted by this lfetch may not occur, the software should consider using this hint.

  — If the accesses to this line are likely only to be loads serviced by FLD, then using this hint will likely reduce the time to this line being freed up in the MLD.

- MLD_NORMAL

C.7.2.3 Using LLC Locality Hints

- LLC_NRU

  General usage: This hint setting can be used to minimize pollution of the LLC when lines are allocated more speculatively than normal.

  Examples of SW opportunities:

  — This hint may be useful when a cache line is only likely to accessed for a relatively small window of time following this allocation. For example, if software is striding sequentially through a cache line and then will be done with it, using this hint will cause that line to be more likely to be replaced in the LLC and thus in the MLD as well (due to LLC inclusion and MLD selection of invalid lines for replacement).

  — This hint should probably be used for an access pattern with a stride bigger than a cache line.

  — This hint should probably be used for sparse array or similar scattered access patterns.

- LLC_NORMAL

C.7.3 Using PF Hints

Data prefetches are far from free. They consume bandwidth and space in the cache hierarchy. Doing unneeded prefetches can hurt performance. Hardware data prefetchers have to guess when prefetches will be useful. The more aggressive the prefetcher, the bigger the gains in some cases and the bigger the losses in other. The aggressiveness and thus the benefits of hardware data prefetchers are often limited by a desire to minimize the performance losses on any workloads. The purpose of the HWPF hints is to allow software to use knowledge it has of the workload to communicate to hardware which instructions should not trigger hardware prefetches in the hopes that the performance gains can be maximized simultaneously with minimizing performance losses.

- PF_NORMAL

- PF_NO_FLD

  General usage: Improve multi-line FLD hardware data prefetching by filtering out prefetches that are likely to be useless.

  Examples of SW opportunities:

  — This hint should probably be used whenever FLD_LOCALITY.NO_ALLOCATE is used and none of the other HWPF.NO_FLD* hints are being used. On the processor, FLD_LOCALITY.NO_ALLOCATE actually effectively implies the HW behavior associated with this hint, but that conceivably might not be the case on future Intel Itanium processors.

- PF.NO_MLD

  General usage: Improve multi-line FLD and multi-line MLD hardware data prefetching by filtering out prefetches that are likely to be useless.

  Examples of SW opportunities:

  — This hint should probably be used whenever MLD_LOCALITY.NO_ALLOCATE is used and none of the other HWPF.NO_*MLD* hints are being used.

- PF_NONE

  General usage: Improve multi-line FLD, multi-line MLD, and buddy line prefetching by filtering out prefetches that are likely to be useless.

  Examples of SW opportunities:

  — This hint could be useful for an access pattern that is sparse when viewed at the cache line level but has some locality within a cache line. A more specific example would be a data structure such as an array of structures where the array entry access pattern is sparse, but each structure is accessed multiple times per array entry access.

  — Software should consider using this hint with speculative loads. The less likely the speculative load is to be actually used, the more likely it should use this hint.

  — See section C.7.1 for some more ideas.

C.7.4 Using PF_DROP Hints

Software data prefetches (lfetches) are far from free. They consume bandwidth and space in the cache hierarchy. Doing unneeded lfetches can hurt performance. Software has to guess when lfetches will be useful. More aggressive lfetching leads to bigger the gains in some cases and bigger the losses in other. The aggressiveness and thus the benefits of lfetching are often limited by a desire to minimize the performance losses on any workloads. The purpose of the PF_DROP hints is to allow hardware to drop lfetches based on dynamic events specified by software in the hopes that the performance gains can be maximized simultaneously with minimizing performance losses.

- PFD_NORMAL

  General usage: This setting can be used when the value of a prefetch is believed to exceed the possible cost of incurring a DTB miss, an MLD miss, or an FLDTLB miss.

  Examples of SW opportunities:

  — This hint should be used for streams of lfetches in a loop that are highly likely to be useful.

- PFD_TLB

  General usage: This setting can be used when the value of a prefetch is believed to exceed the possible cost of incurring an MLD miss or an FLDTLB miss, but not a DTB miss. Also, this setting can be used when DTB miss is considered to be a good predictor that a data prefetch is not valuable enough to continue.

  Examples of SW opportunities:

— Any lfetches that are not part of a large stream of lfetches in a loop and do not want to use a more sensitive PF_DROP hint should probably use this one.

- PFD_TLB_MLD

  General usage: This setting can be used when the value of a prefetch is believed to exceed the possible cost of incurring an FLDTLB miss, but not an MLD miss or a DTB miss. Also, this setting can be used when an MLD miss or a DTB miss is considered to be a good predictor that a data prefetch is not valuable enough to continue.

  Examples of SW opportunities:

  — FLDTLB targeting lfetches in acyclic regions that are of dubious value due to possibly being speculated from the wrong path or due to possibly being not enough cycles ahead of a load to the same line should probably use this hint.

  — FLDTLB targeting lfetches in OSes with large pages should probably use this hint because not hitting in the DTB in this case is a pretty good measure of badness and because data cache lookup and transfer costs should be minimized since the point of this lfetch is to bring a translation into the FLDTLB.

- PFD_ANY

  General usage: This setting can be used when the value of a prefetch is not believed to exceed the cost of an FLDTLB miss, an MLD miss, or a DTB miss. Also, this setting can be used when an FLDTLB miss, an MLD miss, or a DTB miss is considered to be a good predictor that a data prefetch is not valuable enough to continue.

  Examples of SW opportunities:

  — FLD targeting lfetches in acyclic regions that are of dubious value due to possibly being speculated from the wrong path or due to possibly being not enough cycles ahead of a load to the same line should probably use this hint.

C.7.5 Using PIPE Hint

- PIPE_BLOCK

  General usage: Set this bit for speculative loads used for indirect prefetching.

  Examples of SW opportunities:

  — This hint bit should be used for speculative loads used for indirect prefetching.

- PIPE_DEFER

  General usage: Unless there is a reason to use PIPE_BLOCK, this is the PIPE hint setting that should be used.

C.7.6 Using BIAS_SHARED Hint

General usage: This setting can be used when a load reads data that is never expected to be written.

C.7.7 Dynamic Optimization Opportunities

These data access hints could be quite useful to dynamic optimizers for two reasons. First, setting any DAHR to any value hints is functionally safe. Not only are all DAHR values functionally safe, but the mov-to-DAHR instruction that writes them uses an immediate source operand and so doesn’t need a general register. Given that one of the biggest obstacles to using dynamic optimizers is concern about functional correctness, these hints provide a big opportunity to build a dynamic optimizer that doesn’t have any functional correctness concerns. Second, dynamic optimizers have the opportunity

to use dynamic performance feedback to direct the setting of these hint bits. Such dynamic feedback is likely to achieve more optimal settings and thus higher performance.

C.8 Scheduling High Cache Hierarchy Bandwidth Applications

High cache hierarchy bandwidth applications can be limited by nominal the bandwidth limitations of hardware pipelines and communication channels, the ability of hardware to queue up in flight operations, or data or structural hazards of various types. The most restrictive hardware limitation will determine the bandwidth achieved in a bandwidth limited application.

C.8.1 Nominal Hardware Bandwidth Limitations

Table 5-113. Nominal Data Streaming Bandwidth Limitations

Resource	Peak Theoretical Bandwidth
main pipeline	2 M-port operation slots and 1 FLD fill / cycle
MLD pipeline	2 operation slots / cycle
Ring to MLI/MLD	32B of instr/data / cycle
MLI/MLD to Ring	1 instr/data request and 16B of instr/data / cycle
LLC to main memory	system dependent

Table 5-114. Cost of Various Operations

Operation	MLD Pipeline Slots Required
128B MLD fill[1]	9 MLD pipe slots: 1 for outgoing data request + 8 for data fill
64B MLD fill	7 MLD pipe slots: 1 for outgoing data request + 6 for data fill
load return with no MLD fill	5 MLD pipe slots: 1 for outgoing data requests + 2 for data bypass to register file[2] + 2 for dummy data fill operation
load return with 128B MLD fill	11 MLD pipe slots: 1 for outgoing data request + 2 for data bypass to register file[2] + 8 for data fill
load return with MLD fill	9 MLD pipe slots: 1 for outgoing data request + 2 for data bypass to register file[2] + 4 for data fill
lfetch-on-A	1 main pipe M-slot[3]

[1] A 128B MLD fill occurs when a 64B request triggers a MLD buddy line prefetch (the default) or when two 64B requests are made to the same 128B line. Note that if two 64B requests are made, the two outgoing data requests take 2 slots instead of one.

[2] Loads are more expensive that lfetches because of the cost in the MLD pipeline (not the main pipeline as in previous Intel Itanium processors) of returning the critical chunk of data to the register file.

[3] Lfetches that are initially issued on an A-port must eventually be issued on a M-port or be dropped. By default, software lfetches issued on the A-port are rarely dropped (need to specify when they might be). However, software can hint them so that they are dropped when the data prefetch queue starts filling up. See section C.7.4 for details on such hinting.

C.8.2 Schedule to Maximize In-flight Operations Not to Hide Latency

For high cache hierarchy bandwidth applications, the latency of an operation, and the latency exposed to the software scheduler are two very different things. Imagine a simple (blocking) pipeline that stopped processing everything else whenever it was

processing a memory operation (for example, hardware page walks on previous Intel Itanium processors). Even if the memory operation takes a thousand cycles, none of this is exposed to the software scheduler. In other words, the software scheduler can't do anything to hide this latency by doing other work in parallel. This is why caches are typically designed to be non-blocking in high performance processors. There is always, however, a limit to the number of in-flight memory operations that a processor can handle before it begins blocking the pipeline again. Thus, in a high cache hierarchy bandwidth application, the maximum amount of latency that can be exposed to and potentially hidden by the software scheduler is determined by the maximum number of in-flight memory operations a processor can handle.

The following is a list of the maximum number of in-flight memory operations of various types for the processor.

Table 5-115. Maximum Number of In-flight Memory Operations

Operation Type	Maximum Number in Flight
stores	32
MLD hits	32 M-ops when 2 M-ops / issue group; 24 M-ops when 1 M-op / issue group
MLD primary misses	16 128B lines
MLD secondary misses	16 pairs (same cache line) of secondary misses[1]

[1] Lfetches that are secondary MLD misses and do not fill the FLD are dropped, and thus do not count towards the limit of in flight secondary misses. *Therefore, when possible, the software scheduler should either avoid issuing lfetches to the same 128B line as a previous lfetch, or the software scheduler should avoid allowing such leftches to fill the FLD.*
In a high cache hierarchy bandwidth application, the software scheduler needs to work to expose as much parallelism as it can up to the above limits. Exposing more parallelism that these limits provides no benefit on the processor regardless of the actually latency of the memory operations. However, exposing more parallelism than these limits could provide benefits on future processors with higher limits.

C.8.3 Synchronous Data Hazards

This section discusses hazards between operations in the same pipeline that have address dependencies with each other. Due to the synchronous timing relationships, software schedulers may be able to schedule around these hazards. Note that operation issue from the MLD OZQ is much more in order than it was on previous Intel Itanium processors, so software schedulers may even be able to avoid synchronous MLD pipeline hazards.

C.8.3.1 Load Address Generation

Load results cannot be used as addresses until 2 cycles after they are written. Hardware handles this hazard by stalling IBL issue for a single cycle when needed to separate the address use from the load, even if one or both of the instructions are predicated off. Therefore, assuming the memory operations hit in the FLD, the following loop will iterate once every 3 cycles:

```
loop:
   ld8 r9=[r8];;
   st8 [r9]=r5            (st8 is issued stalled at the cost of a cycle)
   add r8=8,r8
   br.cloop loop;;
```

Therefore, when a software scheduler can find other work to do, it should avoid scheduling memory operations in the cycle immediately following a load that generates the address of the memory operation even when both operations will not be predicated on at the same time.

Previous Intel Itanium processors similarly stalled, except that they considered the predicate values in the stall calculation.

C.8.3.2 FLD RAW

As described in the FLD section of this document, the results of FLD writes are not available to FLD loads until two cycles later. Hardware handles this hazard by DET replaying FLD hitting loads that attempt to consume the results of FLD writes too soon. Therefore, assuming the memory operations hit in the FLD, the following loop will iterate once every 6 cycles:

```
loop:
   st8 [r8]=r4
   ld8 r5=[r8]          (ld8 is DET replayed at a cost of 5 cycles)
   add r8=64,r8
   br.cloop loop;;
```

Altering this loop as follows actually improves the iteration time to 3 cycles.

```
loop:
   st8 [r8]=r4;;
   nop 0x0;;
   ld8 r5=[r8]          (no DET replay)
   br.cloop loop;;
```

Software pipelining this loop in a way that separates the load from the store by at least two cycles, could improve the throughput of this loop to 1 (original) iteration every cycle. *Therefore, software should avoid scheduling loads in the cycle immediately following stores with which they might overlap.*

Previous Intel Itanium processors had a longer FLD write latency but lower penalties for an attempted read within the latency window.

C.8.3.3 MLD RAW

As described in the MLD section of this document, the results of MLD writes are not available to the 3 extra banks used by FLD fills until 4 cycles later. Hardware handles this hazard by stalling the MLD pipeline. Therefore, assuming the memory operations miss in the FLD and hit in the MLD, the following loop will iterate once every 6 cycles:

```
loop:
   st8 [r8]=r4
   ld8 r5=[r8]          (ld8 is stalled at a cost of 4 cycles)
   add r8=64,r8
   br.cloop loop;;
```

If there is no benefit for the load to fill the FLD, then hinting the load to not do an FLD fill could increase the throughput of the loop to an iteration every cycle:

```
   mov dahr1 = 0x2;;
```

```
loop:
   st8 [r8]=r4
   ld8.d1 r5=[r8]          (ld8 doesn't fill FLD)
   add r8=64,r8
   br.cloop loop;;
```

Alternatively, if the FLD fill is needed, software pipelining the loop in a way that separates the load from the store by at least 4 cycles could also improve the throughput of the loop to an iteration every cycle. *Therefore, when a software scheduler can find other work to do, it should avoid scheduling FLD filling operations closer than 4 cycles from a store that overlaps with a 16-byte chunk of the filling cache line other than the 16-byte chunk targeted by the FLD filling operation. Alternatively, if the FLD fill is not needed, software can tell hardware not to do the fill with a DAHR hint.*

Previous Intel Itanium processors had a much larger set of MLD data hazards that were dealt with through a mechanism called effective release. Use of the effective release mechanism incurred, at times, significant performance costs. The reduction of the number and cost of these hazards is a significant improvement on the processor. For example, streams of 1 or 2 bytes stores achieve much better throughputs on the processor.

C.8.4 Synchronous Structural Hazards

This section discusses hazards between operations in the same pipeline that are competing for the same resources. Due to the synchronous timing relationships, software schedulers may be able to schedule around these hazards. Note that operation issue from the MLD OZQ is much more in order than it was on previous Intel Itanium processors, so software schedulers may even be able to avoid synchronous MLD pipeline hazards.

C.8.4.1 FLD hitting st vs. st

As described in the FLD section of this document, the FLD cannot simultaneously handle two FLD hitting stores with the same VA[7:5] and different VA[11:8]. So, for example, the stores in a sequence of stores with a power of 2 stride from 256 to 4K bytes would all have this hazard with each other. The hardware deals with this hazard by DET replaying the second store. Therefore, assuming the stores hit in the FLD, the following loop will iterate once every 6 cycles:

```
   movl r8=0x0100000000000000
   movl r9=0x0100000000000100
loop:
   st8 [r8]=r4
   st8 [r9]=r5          (st8 is DET replayed at a cost of 5 cycles)
   add r8=8,r8
   add r9=8,r9
   br.cloop loop;;
```

Altering this loop as follows actually improves the iteration time to 2 cycles:

```
   movl r8=0x0100000000000000
   movl r9=0x0100000000000100
loop:
   st8 [r8]=r4;;
   st8 [r9]=r5          (no DET replay)
   add r8=512,r8
   add r9=512,r9
```

```
        br.cloop loop;;
```

Therefore, software should avoid scheduling FLD hitting stores with the same VA[7:5] and different VA[11:8] in the same cycle, even if it has to insert an additional stop bit to do so.

Note that previous Intel Itanium processors had a similar hazard that involved both loads and stores and different address bits.

C.8.4.2 MLD Bank Conflicts

The MLD cannot allow two operations to simultaneously access the same bank (determined by address bits 7:4). This restriction results in the following hazards:

- *RR bank* hazards occur when two memory ops flow down the MLD pipeline together that need to read the same bank, but not the same full address. In this case, the MLD pipeline will stall for one cycle. Ops issued from the OZQ will never have a RR bank hazard and thus this hazard will only occur on bypassed ops.
- *WW bank* hazards occur when two memory ops flow down the MLD pipeline together than need to write the same bank (address bits 7:4), but not the same full address. Again, the MLD pipeline will stall for one cycle. Note that if a pair of ops have both RR bank and WW bank hazards (i.e. Read-Modify-Write (RMW) stores), only one pipeline stall will occur. Ops issued from the OZQ will never have a WW bank hazard and thus this hazard will only occur on bypassed ops.
- *RW bank* hazards occur when a store that is writing a given bank is followed 4 cycles later by a memory op that is reading the same bank, but not the same full address. The pipeline will stall for one cycle. Note that the stall may cause a subsequent RW bank hazard with a following store. Because this is an inter-stage hazard, ops issued from the OZQ may have a RW bank hazard.

Strides that are multiples of 256 bytes result in every access having a bank conflict with all others. Such a stride will cut the throughput of the MLD to 1-hit per cycle. Therefore, assuming the stores miss in the FLD, the following loop will iterate once every 2 cycles:

```
      movl r8=0x0100000000000000
      movl r9=0x0100000000000100
loop:
      st8 [r8]=r4
      st8 [r9]=r5          (st8 is OZQ issue or MLD pipeline stalled for a cycle)
      add r8=512,r8
      add r9=512,r9
      br.cloop loop;;
```

Therefore, software can schedule with a throughput expectation of at most 1 MLD load hit per cycle for strides that are multiples of 256 bytes.

However, in the case where each access in a stream only has a bank conflict with either the access before it or the access after it (e.g. an 8-byte stride), the MLD stall will tend to re-align the operations that get issued together such that they are the pairs that don't have bank conflicts. For such a stream, then, the cost of bank conflicts is just a fractional (~75%) increase in effective MLD latency and no decrease in throughput. Therefore, assuming the stores miss in the FLD, the following loop will iterate once every cycle:

```
        movl r8=0x0100000000000000
```

```
   movl r9=0x0100000000000008
loop:
   st8 [r8]=r4
   st8 [r9]=r5          (st8 is OZQ issue or MLD pipeline stalled for a cycle)
   add r8=16,r8
   add r9=16,r9
   br.cloop loop;;
```

Therefore, software should either attempt to avoid MLD bank conflicts for loads or schedule uses at least 13 cycles after their respective loads.

Previous Intel Itanium processors had a more expansive definition of a MLD bank conflicts in that accesses to the same line were bank conflicts. Removing the existence of a bank conflict between two accesses to the same lines is a significant improvement on the processor. Also, the cost of hardware handling a bank conflict is less on the processor than on previous Intel Itanium processors. Previous processors retried that had a bank conflict with an earlier operation in a way that delayed them and any dependent instructions by 4 cycles.

C.8.4.3 MLD Fill Port Hazards

MLD *Fill port* hazards occur when two memory ops flow down the MLD pipeline together and both ops will perform an FLD fill. The pipeline will stall for one cycle. Note that if the ops also have a RR bank hazard, only one stall will occur. The OZQ will attempt to avoid these hazards but they are not completely prevented when issuing out of the OZQ.

C.8.5 Asynchronous Data Hazards

This section discusses hazards between operations that are NOT in the same pipeline and that have address dependencies with each other. Due to the asynchronous timing relationships, software schedulers are unlikely to be able to schedule around these hazards. These hazards are discussed to help software schedulers understand that streams that have potential for these hazards may achieve less throughput that streams without potential for these hazards. Some PMU events may provide insight into when and how often such hazards are being encountered.

C.8.5.1 FLD Fills

As described in the FLD section of this document, FLDWRs can be DET replayed if they line up close in time with FLD fills that have the same VA[13:6].

As described in the FLD section of this document, the FLD cannot handle FLDWRs and FLD fills with the same VA[13:6] that occur at about the same time. So, for example, a stream of stores and FLD filling loads with a power of 2 stride greater than or equal to 16K bytes could encounter this hazard a lot. The hardware deals with this hazard by DET replaying the FLDWR. Therefore, assuming the loads miss in the FLD and hit in the MLD, the following loop will iterate once every ? cycles:

```
   movl r8=0x0100000000000000
   movl r9=0x0100000000004000
loop:
   ld8 [r8]=r4
   st8 [r8]=r5          (st8 is DET replayed at a cost of 5 cycles)
   add r8=0x4000,r8
   br.cloop loop;;
```

C.8.6 Asynchronous Structural Hazards

This section discusses hazards between operations that are NOT in the same pipeline that are competing for the same resources. Due to the asynchronous timing relationships, software schedulers are unlikely to be able to schedule around these hazards. These hazards are discussed to help software schedulers understand the expected throughputs of various streams of operations.

C.8.6.1 Asynchronous Data Prefetches

A data prefetch that goes through the Data Prefetch Queue (DPQ) is executed asynchronously with respect to the instruction that triggered it. The execution of this data prefetch requires a variety of resources including a main pipe M-port and usually an OZQ entry. Competition with other operations for these resources can lead to data prefetches being dropped due to the DPQ overflowing or the data prefetches being held up so long that they are no longer useful.

An example where these structural hazards are relevant to streaming is when a stream of lfetches is issued on an A-port. If every issue group contains an lfetch on an A port for more than 8 cycles in a row, some of these lfetches may be dropped. *Therefore, software should avoid scheduling an lfetch on an A-port every issue cycle for more than 8 cycles in a row.*

Streams of lfetches that miss the DTB and are hinted PIPE_DEFER also get inserted into the DPQ and thus behave similarly to lfetches issued on A-ports. *Therefore, software should avoid scheduling an lfetch that misses the DTB every issue cycle for more than 8 cycles in a row.*

C.8.6.2 FLD Fills

An FLD fill can cause a DET replay (see FLD section of this document), but on average, considerably less than 5% should do so. The impact of FLD fills on the main pipeline is much less than it was on previous Intel Itanium processors.

C.8.6.3 MLD Fills

MLD *Fill-store* hazards occur when a store is followed 4 cycles later by an MLD fill operation, independent of address. The pipeline will stall for one cycle (and may cause a subsequent Fill-store hazard with a following store.

§