



---

# Power10 Processor Chip User's Manual

---

OpenPOWER

Version 1.0  
13 September 2021



© Copyright IBM Corporation 2021

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

OpenPOWER, the OpenPOWER logo, and openpowerfoundation.org are trademarks or registered trademarks of OpenPOWER Foundation, Inc., registered in many jurisdictions worldwide.

Other company, product, and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

This document is intended for the development of technology products compatible with Power Architecture®. You may use this document, for any purpose (commercial or personal) and make modifications and distribute; however, modifications to this document may violate Power Architecture and should be carefully considered. Any distribution of this document or its derivative works shall include this Notice page including but not limited to the IBM warranty disclaimer and IBM liability limitation. No other licenses (including patent licenses), expressed or implied, by estoppel or otherwise, to any intellectual property rights are granted by this document.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. IBM makes no representations or warranties, either express or implied, including but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement, or that any practice or implementation of the IBM documentation will not infringe any third party patents, copyrights, trade secrets, or other rights. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Systems and Technology Group  
2070 Route 52, Bldg. 330  
Hopewell Junction, NY 12533-6351

The IBM home page can be found at [ibm.com](http://ibm.com)®.

Version 1.0  
13 September 2021



## Contents

<b>List of Tables .....</b>	<b>15</b>
<b>List of Figures .....</b>	<b>19</b>
<b>Revision Log .....</b>	<b>21</b>
<b>About this Document .....</b>	<b>23</b>
Who Should Read this Document .....	23
Conventions Used in This Document .....	23
Representation of Numbers .....	23
Bit Significance .....	23
Other Conventions .....	23
Related Documents .....	24
<b>1. Power10 Processor Overview .....</b>	<b>25</b>
<b>2. Packages .....</b>	<b>33</b>
2.1 Power10 Single-Chip Module (68.5 x 77.5 mm) .....	33
2.1.1 Bus Features .....	33
2.2 Power10 Dual-Chip Module (74.5 x 85.75 mm) .....	36
2.2.1 Bus Features .....	36
<b>3. Power10 Processor Core .....</b>	<b>39</b>
3.1 Core Overview .....	39
3.1.1 Introduction .....	39
3.1.2 Instruction Set Architecture .....	39
3.1.3 Core Variants and Logical Partitioning (LPAR) Modes .....	39
3.1.4 Micro-Architecture Performance Improvement Summary .....	40
3.1.5 Microarchitecture Efficiency Improvement Summary .....	42
3.2 Instruction Flow .....	43
3.2.1 Fetch / Branch Prediction .....	45
3.2.2 Decode/ Microcode / Fusion .....	45
3.2.3 Dispatch and Resource Allocation .....	46
3.2.3.1 Issue Queue Entry and Slice Assignment .....	46
3.2.3.2 Register Renaming and Ordering .....	46
3.2.3.3 Load Store Queue Virtual Entry Allocation .....	47
3.2.4 Instruction Completion .....	47
3.2.5 Issues Queues .....	47
3.2.6 Compute Pipelines .....	48
3.2.6.1 Group-1 Computation Pipelines .....	48
3.2.6.2 Group-2 Computation Pipelines .....	48
3.2.6.3 Dense Math Engine .....	49
3.2.6.4 Store Address Generation, Branch, and Simple Pipelines .....	50
3.2.7 Load/Store Pipelines and Queues .....	51
3.2.8 TIQ/MMU Pipeline .....	51



3.2.8.1 Load Cache Miss .....	51
3.3 Data Handling .....	51
3.3.1 L1 Load Data Cache Access .....	51
3.3.1.1 Unaligned Loads .....	51
3.3.1.2 Load Fusion .....	52
3.3.2 Store Datapath .....	52
3.3.2.1 Unaligned Stores .....	52
3.3.2.2 Store Fusion .....	52
3.3.2.3 Store Forwarding .....	52
3.3.3 Data Prefetch .....	52
3.4 Memory Management .....	53
3.4.1 TLB .....	53
3.4.2 Data and Instruction Sharing .....	53
3.4.2.1 Alias Tables .....	53
3.4.2.2 Context Tables .....	53
3.5 Security .....	54
3.5.1 Data and Speculation Barriers .....	54
3.5.2 Speculation Controls .....	54
3.5.3 Return Oriented Programming Protection .....	54
3.6 Power Management .....	54
3.6.1 Thread Priority .....	54
3.6.2 MMA Power Down .....	54
<b>4. Thread and LPAR Management .....</b>	<b>55</b>
4.1 SMT Overview .....	55
4.1.1 SMT Modes .....	55
4.1.2 SMT Resource Partitioning .....	58
4.2 Logical Partition Specific Resources .....	58
4.2.1 Time Base Register .....	58
4.2.1.1 SMT8 Core Time Base Register Initialization .....	58
4.2.1.2 PURR Register .....	58
4.2.1.3 Region Weighted Mode Register .....	59
4.3 Core-Specific Resources .....	59
4.3.1 CTRL Register .....	59
4.3.2 SPRC/SPRD .....	60
4.3.3 SPR Mode Register .....	62
4.3.4 Recovery and Core Checkstop Handling .....	63
4.3.5 Memory Accumulation .....	63
4.4 Thread Priority Management .....	63
4.4.1 Thread Switch Control Register (TSCR) .....	64
4.4.2 Program Priority Register (PPR) .....	66
4.4.2.1 Priority NOPs .....	66
4.4.3 Thread Priority Boosting .....	67
4.4.3.1 Priority Boosting to Medium-High in User Mode .....	67
4.4.3.2 Thread Priority Boosting on Asynchronous Interrupt .....	68
4.4.4 Dynamic Thread Prioritization .....	69
4.4.4.1 Dynamic Fetch Priority .....	69
4.4.4.2 Dynamic Decode Priority .....	69
4.4.4.3 Software Set Thread Priority .....	70



4.4.4.4 Reduced Priority Modes .....	70
4.5 Thread Pipeline Management .....	70
4.5.1 Dispatch Flush .....	70
4.5.2 Decode Hold .....	70
4.5.3 Balance Flush .....	71
<b>5. Architecture Compliance .....</b>	<b>73</b>
5.1 Book I - User Instruction Set Architecture .....	73
5.1.1 Instruction Classifications .....	73
5.1.1.1 Illegal Instructions .....	73
5.1.1.2 Instructions Supported .....	73
5.1.1.3 Invalid Forms .....	73
5.1.2 Branch Processor .....	74
5.1.2.1 Instruction Fetching .....	74
5.1.2.2 Branch Prediction .....	74
5.1.2.3 Instruction Cache Block Touch Hint .....	74
5.1.2.4 Out-of-Order Execution and Instruction Flushes .....	74
5.1.2.5 Branch Processor Instructions with Undefined Results .....	75
5.1.2.6 Branch History Rolling Buffer (BHRB) .....	75
5.1.3 Fixed-Point Processor .....	75
5.1.3.1 Fixed-Point Exception Register .....	75
5.1.4 Storage Access Alignment Support Overview .....	77
5.1.4.1 Alignment Interrupts .....	77
5.1.4.2 Storage Control Attribute Caused Data Storage Interrupt or Hypervisor Data Storage Interrupts .....	79
5.1.5 Fixed-Point Load and Store Instructions .....	79
5.1.5.1 Fixed-Point Load and Store Multiple Instructions .....	79
5.1.5.2 Fixed-Point Move Assist Instructions .....	80
5.1.5.3 Integer Select Instruction .....	80
5.1.5.4 Fixed-Point Logical Instructions .....	80
5.1.5.5 Access to Performance Monitor Special Purpose Registers .....	81
5.1.5.6 Move to/from Condition Register Fields Instructions .....	81
5.2 Fixed-Point Invalid Forms and Undefined Conditions .....	81
5.3 Vector Scalar Instructions (FP, VMX, and VSX) .....	84
5.3.1 IEEE Compliance .....	84
5.3.1.1 Non-IEEE Modes .....	84
5.3.2 Floating-Point Exceptions .....	84
5.3.3 Floating-Point Load and Store Instructions .....	84
5.3.3.1 Scalar Load and Store Atomicity .....	84
5.3.3.2 Vector Load and Store Atomicity .....	84
5.3.4 Heterogeneous Precision Arithmetic .....	85
5.3.4.1 NaN Propagation .....	85
5.3.4.2 Square Root Overflow and Underflow .....	85
5.3.4.3 Hardware Behavior on Enabled Underflow and Enabled Overflow Exception .....	85
5.3.5 Handling of Denormal Single-Precision Values in Double-Precision Format .....	86
5.3.6 Vector and Floating-Point Invalid Forms and Undefined Conditions .....	86
5.4 Optional Facilities and Instructions .....	87
5.5 Little-Endian Mode .....	88



5.6 Book II - Virtual Environment Architecture .....	88
5.6.1 Cache .....	88
5.6.2 Classes of Instructions .....	89
5.6.2.1 Instruction Cache Block Touch Instruction ( <b>icbt</b> ) .....	89
5.6.2.2 Instruction Cache Block Invalidate ( <b>icbi</b> ) .....	89
5.6.2.3 Instruction Cache Synchronize ( <b>isync</b> ) .....	89
5.6.2.4 Vector Category Prefetch Instructions ( <b>dss</b> , <b>dst</b> , and <b>dstst</b> ) .....	89
5.6.2.5 Data Cache Block Touch Instructions ( <b>dcbt</b> and <b>dcbtst</b> ) .....	90
5.6.2.6 Data Cache Block Touch to a Single Block Characteristics .....	90
5.6.2.7 Data Cache Block Touch Streaming Characteristics .....	91
5.6.2.8 Data Cache Block Touch - Invalid TH Forms (TH = '00001' through TH = '00111') .....	91
5.6.2.9 Data Cache Block Touch Data Stream (TH = '01000') .....	91
5.6.2.10 Data Cache Block Touch Data Stream Descriptor (TH = '01010' through TH = '01111') .....	91
5.6.2.11 Data Cache Block Touch - Transient (TH = '10000') .....	92
5.6.2.12 Data Cache Block Touch - No Access Needed Anymore (TH = '10001') .....	92
5.6.2.13 Data Cache Block Zero ( <b>dcbz</b> ) .....	92
5.6.2.14 Data Cache Block Store ( <b>dcbst</b> ) .....	92
5.6.2.15 Data Cache Block Flush ( <b>dcbf</b> , <b>dcbfl</b> , and <b>dcbflp</b> ) .....	93
5.6.2.16 Key Aspects of Storage Control Instructions .....	93
5.6.2.17 Copy/Paste Instructions .....	93
5.6.2.18 Near Memory Instruction Support .....	94
5.6.2.19 Wait Instruction .....	94
5.6.3 Storage Model .....	94
5.6.3.1 Storage Access Ordering .....	94
5.6.3.2 Atomicity .....	94
5.6.3.3 Atomic Updates and Reservations .....	95
5.6.4 Transactional Memory (TM) .....	95
5.6.4.1 Synthetic TM (STM) .....	96
5.6.4.2 Synthetic Suspend State Details .....	96
5.6.4.3 TDOOMED .....	97
5.6.4.4 Implementation-Specific Failure Causes .....	97
5.6.5 Storage Ordering/Barrier Instructions.....	97
5.6.5.1 <b>sync</b> Instruction .....	97
5.6.5.2 <b>eieio</b> Instruction .....	98
5.6.5.3 <b>miso</b> Instruction .....	98
5.6.5.4 Return Oriented Programming (ROP) Support .....	98
5.6.6 Data Prefetch .....	98
5.6.7 Timer Facilities .....	99
5.6.8 Hypervisor Decrementer (HDEC) .....	100
5.6.9 Decrementer (DEC) .....	100
5.6.10 Book II Invalid Forms .....	100
5.7 Book III - Operating Environment Architecture .....	101
5.7.1 Classes of Instructions .....	101
5.7.1.1 Storage Control Instructions .....	101
5.7.1.2 Reserved Instructions .....	102
5.7.2 Branch Processor .....	102
5.7.2.1 SRR1 Register .....	102
5.7.2.2 HSRR1 Register .....	102
5.7.2.3 USRR1 Register .....	102



5.7.2.4 MSR Register .....	102
5.7.2.5 System Call and System Call Vectored Instructions .....	102
5.7.2.6 Support Processor Attention Instruction .....	103
5.7.2.7 Current Instruction Address Breakpoint Register (CIABR) .....	103
5.7.3 Fixed-Point Processor .....	103
5.7.3.1 Processor Version Register (PVR) .....	103
5.7.3.2 Processor ID Register (PIR) .....	104
5.7.3.3 Chip Information Register (CIR) .....	104
5.7.3.4 Thread ID Register (TIDR) .....	104
5.7.3.5 Move To/From Special Purpose Register Instructions .....	104
5.8 HID Register .....	120
5.8.1 HID Register Description .....	120
5.8.2 IMC Array Access Register .....	121
5.8.3 Performance Monitor Registers .....	121
5.8.4 Other Fixed-Point Instructions .....	121
5.9 Storage Control .....	122
5.9.1 Effective, Virtual, and Physical Address Ranges Supported .....	122
5.9.2 Control Memory Definition and Accessibility .....	122
5.9.3 Ultravisor Real Mode Addressing Using URMOR .....	122
5.9.4 Hypervisor Real Mode Addressing Using HRMOR .....	123
5.9.5 Partition Table Control Register .....	123
5.9.6 Access Segment Descriptor Register .....	123
5.9.7 Real Mode Addressing for Operating Systems .....	123
5.9.8 HRMOR Update Sequence .....	123
5.10 Translation Architecture .....	124
5.10.1 Logical Partitioning Control Register (LPCR) .....	125
5.10.2 Logical Partition Identification Register (LPIDR) .....	126
5.10.3 Process Identification Register (PIDR) .....	126
5.10.4 Translation Modes .....	126
5.10.5 <b>tlbie</b> and <b>tlbiel</b> Instruction Format and Operands .....	127
5.10.6 Radix Translation .....	130
5.10.6.1 Supported Radix Tree Configurations and Resulting Page Sizes .....	131
5.10.6.2 <b>tlbie</b> and <b>tlbiel</b> Encodings for Radix Translations .....	132
5.10.7 Changing the Process ID Register For Radix Translation .....	132
5.10.8 Switching between Radix and HPT Partitions .....	132
5.10.8.1 LPAR per Thread (Thread LPAR) Mode .....	133
5.10.8.2 LPAR per Core (1-LPAR) Mode .....	133
5.10.9 Hashed Page Table (HPT) Translation .....	134
5.10.9.1 Software Managed SLB Entries (UPRT = '0', HR = '0') .....	134
5.10.9.2 In-Memory Segment Table and Bolted SLB Entries (UPRT = '1', HR = '0') .....	134
5.10.10 Changing the Process ID Register For HPT Translation .....	134
5.10.10.1 SLB Management Instructions .....	134
5.10.10.2 Supported Segment and Page Sizes for HPT Translations .....	135
5.10.10.3 <b>tlbie</b> and <b>tlbiel</b> Usage for HPT Translations .....	136
5.10.10.4 Lockless TLBIE Support .....	137
5.10.11 Instruction, Data, and Translation Caches .....	138
5.10.11.1 Instruction Cache .....	138
5.10.11.2 Data Cache .....	139
5.10.11.3 Effective-to-Real Address Translation Cache .....	140
5.10.12 Segment Lookaside Buffer .....	142



5.10.13 Discontinued Translation Support Items .....	142
5.10.13.1 Address Space Register .....	142
5.10.14 Block Address Translation .....	142
5.10.14.1 Support for 32-Bit Operating Systems .....	143
5.10.14.2 Real Mode .....	143
5.10.15 Reference and Change Bits .....	143
5.10.16 Storage Protection .....	144
5.10.17 Ultravisor Real Mode Storage Control .....	144
5.10.18 Hypervisor Real Mode Storage Control .....	145
5.10.19 Storage Access Modes (WIMG and ATT Bits) .....	145
5.10.20 Speculative Storage Accesses .....	146
5.10.21 TLB Invalidate Entry ( <b>tlbie</b> and <b>tlbiel</b> ) Instruction .....	147
5.10.22 TLB Invalidate All ( <b>tlbia</b> ) Instruction .....	147
5.10.23 TLB Synchronize ( <b>tlbsync</b> ) Instruction .....	147
5.10.24 SLB Synchronize ( <b>slbsync</b> ) Instruction .....	147
5.10.25 Support for Store Gathering .....	148
5.10.26 Cache Coherency Paradoxes .....	148
5.10.27 Handling Parity Error, Multi-Hit, and Uncorrectable Errors .....	148
5.10.27.1 Parity Error .....	148
5.10.27.2 Multi-Hit .....	148
5.10.27.3 Both Multi-Hit and Parity Error .....	149
5.10.27.4 Uncorrectable Error Handling .....	149
5.10.27.5 TLB Parity Error and Multi-Hit Action .....	150
5.10.28 Interrupts .....	151
5.10.28.1 Interrupt Vectors .....	151
5.10.28.2 Alternate Interrupt Location .....	152
5.10.28.3 Interrupt Definitions .....	155
5.10.28.4 Synchronous Interrupts .....	156
5.10.28.5 Asynchronous Interrupt Priorities .....	156
5.10.28.6 System Reset Interrupt .....	157
5.10.28.7 Machine Check Interrupt .....	158
5.10.28.8 Hypervisor Maintenance Interrupt .....	162
5.10.28.9 External Interrupt .....	162
5.10.28.10 Alignment Interrupt .....	164
5.10.28.11 Trace Interrupt .....	164
5.10.28.12 Performance Monitor Interrupt .....	164
5.10.28.13 Facility Unavailable Interrupt .....	165
5.10.28.14 Hypervisor Emulation Assistance Interrupt .....	165
5.10.29 Strong Access Ordering Mode (SAO) .....	166
5.10.30 Graphics Data Stream Support .....	166
5.10.31 Data Address Watchpoint Debug Support .....	166
5.10.32 Performance Monitoring, Sampling, and Trace .....	166
5.10.33 Processor Compatibility Mode .....	167
<b>6. L2 Cache .....</b>	<b>169</b>
6.1 Overview .....	169
6.1.1 L2 Cache Features .....	170
6.1.2 L2 RAS Features .....	170
6.2 L2 Unit Internal Resources .....	171
6.2.1 Description of L2 Control Flow .....	172



---

6.3 Operational Flows and Bandwidths .....	173
6.4 LRU .....	174
6.4.1 LRU Modes .....	174
6.4.2 Policies .....	174
6.4.3 Line Disable .....	174
6.5 Transactional Memory Support .....	174
6.5.1 Basic Policy .....	174
<b>7. L3 Cache .....</b>	<b>175</b>
7.1 Interfaces .....	176
7.2 Features and Resources .....	176
7.3 Queues .....	177
7.3.1 Read Machines .....	177
7.3.2 Castin/Castout Machines .....	178
7.3.3 Prefetch Machines .....	179
7.3.4 Snoop Dispatch Pipes .....	179
7.3.5 Snoop Machines .....	179
7.3.6 Write Machines .....	180
<b>8. SMP Interconnect .....</b>	<b>181</b>
8.1 SMP Interconnect Features .....	181
8.1.1 General Features .....	181
8.1.2 Power10-Specific Features .....	182
8.1.3 On-Chip Features .....	182
8.1.4 Off-Chip External SMP Features .....	183
8.1.5 Power Management Features .....	183
8.1.6 RAS Features .....	183
8.2 External Power10 Fabric .....	184
8.3 Terminology .....	184
8.4 Power10 Fabric SMP Topology .....	185
8.4.1 Protocol and Data Routing in Multi-Chip Configurations .....	186
8.5 Power10 Coherency Flow .....	186
8.5.1 Broadcast Scope Definition .....	186
8.5.2 Address Definition .....	187
<b>9. NCU .....</b>	<b>189</b>
9.1 NCU Characteristics .....	190
9.1.1 Store Queue (STQ) .....	190
9.1.2 Store Modes (IG = '1X') .....	190
9.1.3 Loads .....	190
<b>10. Memory Controller .....</b>	<b>191</b>
10.1 Basic Configuration/Grouping .....	192
10.2 Atomic Memory Operations .....	193
10.3 Selective Memory Mirroring .....	195
10.4 Whole Memory Encryption .....	195
10.4.1 Modes of Operation .....	195
10.4.2 Encryption Keys .....	196



<b>11. On-Chip Accelerators .....</b>	<b>197</b>
11.1 NX Features .....	198
11.2 Using NX Coprocessors .....	200
<b>12. Virtual Accelerator Switch .....</b>	<b>201</b>
12.1 Overview .....	201
12.2 Flow for NX Invocation Through the VAS .....	201
12.3 Features .....	204
<b>13. OpenCAPI Processing in the POWERAccel Unit .....</b>	<b>205</b>
13.1 Functions Supported by the Power10 PAU Over OpenCAPI Links .....	205
13.1.1 Memory Inception .....	205
13.1.1.1 Memory Inception Mirroring .....	206
13.1.2 Integrated Networking .....	206
13.2 Features .....	206
13.3 Power10 AFU Transaction Examples .....	207
13.3.1 Read from AFU to Power10 Memory .....	207
13.3.2 Read for Shared Access from AFU to Power10 Memory .....	208
13.3.3 AFU Writes to Power10 Memory .....	208
13.3.4 AFU Posted Writes to Power10 Memory .....	210
13.3.5 Read from Power10 Chip to AFU M1 Memory .....	211
13.3.6 Write from Power10 Chip to AFU M1 Memory .....	211
<b>14. Nest MMU .....</b>	<b>213</b>
14.1 NMMU Features .....	215
14.2 Window/Process Element Context .....	216
14.2.1 Rules for Managing Address Translation Contexts .....	217
<b>15. Interrupt Controller .....</b>	<b>219</b>
15.1 External Interrupt Virtualization Engine .....	219
15.2 High-Level Block Diagram .....	220
15.3 Fabric Bus Interrupt Command .....	221
15.3.1 Message Send (Msgsend) .....	221
<b>16. PCI Express Controller .....</b>	<b>223</b>
16.1 Overview .....	223
16.1.1 Processor Bus Common Queues .....	224
16.1.2 Processor Bus AIB Interface .....	224
16.1.3 Express Transaction Unit .....	224
16.1.4 PCIe ASIC Intellectual Property .....	224
16.1.5 Physical Coding Sublayer .....	224
16.1.6 Physical Media Access .....	225
16.2 Power10 PCIe Configurations .....	225
16.3 Reliability, Availability, and Serviceability (RAS) .....	226
16.3.1 Bit-Level RAS .....	226
16.3.2 Enhanced Error Handling (EEH) .....	226
16.3.3 Freeze Mode .....	226



---

<b>17. Power Management .....</b>	<b>227</b>
17.1 Policies and Modes of Operation .....	227
17.1.1 Power Management in Linux-Based Systems (Power KVM) .....	228
17.1.2 Power Management in PowerVM-Based Systems .....	228
17.2 Architected Control Registers .....	228
17.2.1 Power Management Control Register (PMCR) .....	228
17.2.2 Power Management Idle Control Register (PMICR) .....	229
17.2.3 Power Management Status Register (PMSR) .....	229
17.2.4 Power Management Memory Activity Register (PMMAR) .....	230
17.3 Architected Idle Modes (Stop States) .....	231
17.3.1 Summary of Stop Level Categories .....	231
17.3.2 Management of Stop Entry and Exits .....	232
17.3.3 Unsupported Stop Levels .....	232
17.3.4 Auto-promote of Stop Levels .....	232
17.3.5 Recommended Stop Code Sequence to Support Lab Debug Tools .....	232
17.3.6 Power-Savings Level Status .....	233
17.3.7 Regular Wake Up Events .....	233
17.3.8 State Loss and Restoration .....	233
17.3.8.1 Timing Facilities .....	233
17.3.8.2 SPR State Loss .....	233
17.3.8.3 SPR Loss Differences From the POWER9 Chip .....	234
17.3.8.4 Ultravisor and Hypervisor StopAPI Requirement .....	234
17.3.9 Summary of Power10 Supported Stop Levels .....	235
17.3.10 Latency and Power Savings in each Stop Level .....	236
<b>18. Security .....</b>	<b>237</b>
18.1 Secure Memory Facility .....	238
18.1.1 Protected Execution Facility .....	238
18.1.2 Deviations from the SMF Architecture Specification in the Power10 Implementation .....	239
18.1.2.1 Implementation Restriction: Only URMOR[12:42] Bits are Implemented .....	239
18.1.2.2 Implementation Restriction: the UILE Bit is Not Implemented and it is a Constant Zero, Ultravisor Must Execute in Big-Endian Mode .....	239
18.1.2.3 Implementation Restriction: SMFCTRL[E] Bit is Not Implemented per Thread Only per Core .....	239
18.1.2.4 Implementation Restriction: SMFCTRL[62:63] Bits are Not Implemented .....	239
18.1.3 Secure Memory Bit in System Memory Map .....	239
18.1.4 Mandatory Software Procedures Followed by Ultravisor for Maintaining an SVM for Coarse-Grain SMF .....	240
18.1.4.1 Essential Elements of Code Sequence to Convert a Non-secure Virtual Machine into a Secure Virtual Machine .....	240
18.1.4.2 Ensure Isolation of Register State of a Secure VM from Hypervisor .....	241
18.1.4.3 Ensure Secure VM Translations for Secure Pages are Immutable by Hypervisor .....	242
18.1.4.4 Ensure Secure Memory Region Separation between Different Secure VMs .....	242
18.1.5 Code Sequence to Change Value of URMOR Register .....	243
18.1.6 Additional Restrictions .....	243
18.2 Dynamic Execution Control .....	244
18.2.1 Dynamic Execution Control Instructions .....	244
18.2.2 Dynamic Execution Control Registers .....	244



<b>19. Performance Profile .....</b>	<b>245</b>
19.1 Core .....	245
19.1.1 Microarchitecture and Pipeline Overview .....	245
19.1.2 SMT Modes and Thread Count Sensitivity .....	248
19.1.3 Instruction Fetch .....	248
19.1.3.1 Thread Priority .....	248
19.1.3.2 L1 Instruction Cache .....	249
19.1.3.3 Instruction Prefetch .....	249
19.1.3.4 Software-Initiated Instruction Prefetch .....	249
19.1.3.5 Branch Prediction .....	250
19.1.4 Instruction Decode and Dispatch Pipeline .....	255
19.1.4.1 Instruction Buffer .....	255
19.1.4.2 Effective Address Tracking .....	256
19.1.4.3 Instruction Cracking, Expansion, Fusion, and Prefixing .....	256
19.1.4.4 Instruction/IOP Completion Table .....	257
19.1.4.5 Dispatch .....	258
19.1.4.6 Register Renaming .....	260
19.1.5 Instruction Issue .....	263
19.1.5.1 Load/Store AGen Issue .....	264
19.1.5.2 Execution Pipeline Issue-to-Issue Latencies .....	264
19.1.5.3 Branches .....	267
19.1.5.4 Fusion .....	267
19.1.6 IOP Execution .....	278
19.1.6.1 Execution Pipeline Hazards .....	278
19.1.6.2 FPR Result Forwarding Restrictions .....	280
19.1.7 Load/Store Processing .....	280
19.1.7.1 Tracking Load and Store Ordering .....	280
19.1.7.2 L1 Data Cache .....	281
19.1.7.3 Effective-to-Real-Address-Translation (ERAT) .....	282
19.1.7.4 Translation Look-Aside Buffer .....	283
19.1.7.5 Store Forwarding .....	284
19.1.7.6 Out-of-Order Load/Store Execution .....	284
19.1.7.7 Load-to-Use Latency .....	285
19.1.7.8 Load/Store Throughput .....	285
19.1.7.9 Load/Store Pipeline Hazards .....	286
19.1.7.10 D-Cache Misses .....	286
19.1.7.11 Store Drain and Merge .....	287
19.1.7.12 Store Pre-Allocate .....	287
19.1.7.13 Data Prefetch .....	287
19.1.7.14 Software-Initiated Data Prefetch .....	288
19.1.7.15 Storage Priming/Zeroing Using <b>dcbz</b> .....	289
19.1.8 Special Instruction Sequences .....	289
19.1.8.1 <b>lарx/stcx</b> Instruction .....	289
19.1.8.2 Lock Critical Section Management .....	290
19.1.8.3 <b>icbi</b> Instruction .....	290
19.1.8.4 <b>isync</b> Instruction .....	291
19.1.8.5 <b>ptesync</b> Instruction .....	291
19.1.8.6 <b>sync</b> Instruction .....	291
19.1.8.7 <b>eieio</b> Instruction .....	291
19.2 Instruction Properties .....	292



---

<b>Appendix A. Instruction Properties .....</b>	<b>295</b>
<b>Appendix B. Transactional Memory Instruction Effects .....</b>	<b>381</b>
<b>Appendix C. Storage Exception Cases by Translation Type .....</b>	<b>389</b>
<b>Appendix D. tlbie(l) Encodings .....</b>	<b>421</b>
<b>Appendix E. Power10 Performance Monitor and Instrumentation .....</b>	<b>445</b>
E.1 Core Performance Monitoring Facilities .....	445
E.1.1 Essential Performance Monitor Functions .....	445
E.1.2 Definitions and Terminology .....	446
E.1.3 Essential Performance Monitor Facilities .....	447
E.1.4 Performance Monitor Special Purpose Registers and Fields .....	447
E.2 Performance Monitor Registers .....	451
E.2.1 Performance Monitor Counters (PMC1 - 6) .....	451
E.2.2 Core Monitor Mode Control Register (MMCRC) .....	452
E.2.3 Performance Monitor Control Register 0 (MMCR0) .....	453
E.2.4 Performance Monitor Mode Control Register 1 (MMCR1) .....	459
E.2.5 Performance Monitor Mode Control Register 2 (MMCR2) .....	462
E.2.6 Performance Monitor Mode Control Register 3 (MMCR3) .....	466
E.2.7 Monitor Mode Control Register A (MMCRA) .....	468
E.2.8 Sampled Instruction Event Register (SIER) .....	473
E.2.9 Sampled Instruction Event Register 2 (SIER2) .....	479
E.2.10 Sampled Instruction Event Register 3 (SIER3) .....	481
E.2.11 Sampled Instruction Address Register (SIAR) .....	482
E.2.12 Sampled Data Address Register (SDAR) .....	482
E.3 Power10 Sampling Support .....	483
E.3.1 Sampled Instruction Address Register .....	483
E.3.2 Sampled Data Address Register .....	484
E.3.3 Continuous Sampling .....	484
E.3.4 Random Instruction Sampling (RIS) .....	484
E.3.4.1 Probe NOP .....	485
E.3.5 Random Event Sampling (RES) .....	486
E.4 Thresholding .....	487
E.4.1 Floating-Point Counter .....	487
E.4.2 Thresholding Operation .....	488
E.4.3 Examples of the Ability to Change Events to Threshold .....	490
E.4.3.1 Loads .....	490
E.4.3.2 Stores .....	491
E.4.3.3 Branches .....	492
19.2.1 Threshold Event Selection .....	492
19.2.2 Threshold Start/Stop Event Selection .....	492
E.5 Core PMU Events .....	493
E.5.1 IFU Events .....	494
E.5.2 Branch Events .....	496
E.5.3 ISU Events .....	499
E.5.4 VSU Events .....	503
E.5.5 LSU Events .....	504



E.5.6 L2 Events .....	511
E.5.7 L3 Events .....	518
E.5.8 CPI Stack Events .....	525
E.5.9 Marked Events .....	531
E.5.10 MMU Events .....	536
E.5.11 PMC Events .....	542
E.5.12 Reload Source Events .....	545
E.5.13 Radix Events .....	548
E.5.14 Metrics .....	558
E.5.14.1 Power10 Events by Group .....	558
E.5.14.2 Power10 Metric Events and Formulas .....	636
E.5.15 Power10 Performance Monitor Event Selection .....	656
E.5.15.1 Select Event Code Formation .....	656
E.5.16 Power10 Events Grouping .....	658
E.6 Nest PMU Instrumentation and Performance Metrics .....	659
E.6.1 Nest Performance Monitoring Unit Instrumentation .....	659
E.6.1.1 Power10 Chip Overview .....	659
E.6.1.2 Power10 Nest PMU Instrumentation .....	660
E.6.1.3 Nest Instrumentation Counters (PMULEts) .....	660
E.6.1.4 Nest PMU Instrumentation Categories .....	660
E.6.2 Nest Units and Performance Metrics Support .....	661
E.6.2.1 Processor Fabric (SMP Interconnect) .....	661
E.6.2.2 Internal Fabric PMU Events and Performance Metrics .....	661
E.6.2.3 Memory Controller (MC – Fabric Interface) .....	663
E.6.2.4 OpenCAPI Memory Buffer (OCMB) .....	663
E.6.2.5 A/X Links – SMP Links .....	664
E.6.2.6 PAU Links - OpenCAPI Transaction Layer (OTL) .....	664
E.6.2.7 PCI Express Controller and PCIe Host Bridge .....	666
E.6.3 Nest IMC Events Grouping .....	667
<b>Appendix F. Power10 Processor Programming Model Bulletin .....</b>	<b>679</b>
<b>Appendix G. Errata .....</b>	<b>681</b>
G.1 Revision Levels Covered .....	681
G.2 Summary of Errata .....	681
<b>Glossary .....</b>	<b>683</b>



## List of Tables

Table 4-1.	Region Weighted Mode Register (RWMR) .....	59
Table 4-2.	CTRL Register .....	59
Table 4-3.	MFSPR CTRL Data Formatting .....	60
Table 4-4.	SPRC Definition for SMT8 Core Mode (1-LPAR per Core) .....	60
Table 4-5.	SPRC Definition Thread-LPAR Mode .....	61
Table 4-6.	OCC SPRC Definition .....	61
Table 4-7.	SPRC/SPRD Core Thread State .....	61
Table 4-8.	SPR Mode Register Bit Definitions .....	63
Table 4-9.	Thread Priority NOPs .....	67
Table 4-10.	Priority Boosting Asynchronous Interrupt .....	68
Table 5-1.	XER Bits and Fields .....	75
Table 5-2.	Alignment Interrupt for AMO Cases .....	78
Table 5-3.	Storage Control Instructions .....	93
Table 5-4.	System Call and System Call Vectored Invocation .....	102
Table 5-5.	PVR .....	103
Table 5-6.	PIR .....	104
Table 5-7.	SPR Table .....	106
Table 5-8.	Performance Monitor SPRs .....	116
Table 5-9.	HID Register .....	120
Table 5-10.	HRMOR Update Sequence .....	123
Table 5-11.	Description of <b>tlbie</b> Instruction Format for the Power10 Core .....	128
Table 5-12.	Description of <b>tlbiel</b> Instruction Format for the Power10 Core .....	129
Table 5-13.	Address Bit Range Checking by Hardware .....	131
Table 5-14.	Supported Radix Tree Configurations and Resulting Page Sizes .....	132
Table 5-15.	<b>tlbie(I)</b> Page Encodings for Power10 Radix (effR = '1') .....	132
Table 5-16.	PTE and STE/SLBE Correspondence for HPT Translation .....	135
Table 5-17.	Segment Size and Page Size Specifications for HPT <b>tlbie</b> and <b>tlbiel</b> .....	136
Table 5-18.	Segment Size and Page Size Specifications for HPT <b>tlbie</b> and <b>tlbiel</b> .....	136
Table 5-19.	Segment Size and Page Size Specifications for HPT <b>tlbie</b> Cluster Bombs .....	137
Table 5-20.	Supported Page Sizes Based on Value of "p" .....	141
Table 5-21.	WIMG Bits .....	146
Table 5-22.	IG Bits .....	146
Table 5-23.	Power10 Behavior on Parity Error, Multi-Hit, and Uncorrectable Error Summary .....	149
Table 5-24.	Interrupt Vectors .....	151
Table 5-25.	(H)AIL Effects on Interrupt Processing (IR = DR) .....	153
Table 5-26.	Implementation MSR and SRR1/HSRR1 Bits .....	155
Table 5-27.	HEIR Instruction Formatting for Branch-Like Instructions .....	156
Table 5-28.	System Reset Interrupt .....	157



Table 5-29.	Synchronous Machine Checks .....	160
Table 5-30.	Direct External Interrupt (LPES = '0') .....	162
Table 5-31.	Direct External Interrupt (LPES = '1') .....	163
Table 5-32.	Mediated External Interrupt (LPES = '0') .....	163
Table 5-33.	Mediated External Interrupt (LPES = '1') .....	163
Table 5-34.	Trace Interrupt .....	164
Table 6-1.	L2 Resources (Share between Threads within a Power10 Core) .....	171
Table 8-1.	Terminology .....	184
Table 8-2.	1-Hop Broadcast Scope Definition .....	186
Table 8-3.	2-Hop Broadcast Scope Definition .....	186
Table 13-1.	Read from AFU to Power10 Memory .....	207
Table 13-2.	Noncaching Read from AFU to Power10 Memory .....	208
Table 13-3.	AFU Writes to Power10 Memory .....	208
Table 13-4.	AFU Posted Writes to Power10 Memory .....	210
Table 13-5.	Power10 Read of AFU M1 Memory .....	211
Table 13-6.	Power10 Write to the AFU M1 Memory .....	211
Table 17-1.	Power Management Control Register (Version x'1') .....	228
Table 17-2.	Power Management Status Register .....	229
Table 17-3.	Supported STOP Instruction Behavior .....	235
Table 17-4.	Projected Stop State Latency and Power Savings (with POWER9 Comparison) .....	236
Table 18-1.	System Memory Map for 56-Bit System Address (8:63) .....	240
Table 18-2.	Essential Elements of Code Sequence to Launch a Secure Virtual Machine .....	240
Table 18-3.	Code Sequence to Set Value of URMOR Register .....	243
Table 19-1.	Handling of <b>bclr</b> and <b>bclrl</b> Instructions .....	253
Table 19-2.	Handling of <b>bcctr</b> and <b>bcctrl</b> Instructions .....	253
Table 19-3.	Zero-Cycle Register Move Combinations .....	261
Table 19-4.	Zero Cycle Constant Assignments .....	261
Table 19-5.	Issue-to-Issue Latency Between IOP Pipelines .....	265
Table 19-6.	Supported Fusion Sequences .....	267
Table A-1.	Instruction Properties .....	295
Table B-1.	Transactional Memory Instruction Effects .....	382
Table C-1.	Exception Cases (DSEG) .....	389
Table C-2.	Exception Cases (ISEG) .....	391
Table C-3.	Exception Cases (DSI) .....	392
Table C-4.	Exception Cases (HDSI) .....	399
Table C-5.	Exception Cases (HSI) .....	408
Table C-6.	Exception Cases (ISI) .....	416
Table C-7.	Exception Cases (Machine Check) .....	420
Table D-1.	TLBIE with GTSE = 1 .....	423



---

Table D-2.	<b>tlbiel</b> Encodings .....	433
Table E-1.	Performance Monitor Related Special Purpose Registers .....	449
Table E-2.	Performance Monitor Counter (PMC) Properties .....	450
Table E-3.	MMCR0 Freeze Logic per MSR State, FCU, FCH, FCS, FCPC, and FCP .....	458
Table E-4.	MMCR1[PMCxSEL] Selection of Direct Events versus Event Bus Events .....	461
Table E-5.	MMCR1[PMCxUNIT] Selection of Physical Event Buses .....	461
Table E-6.	Threshold Event Selection (MMCRA[45:47]) .....	470
Table E-7.	Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55]) .....	470
Table E-8.	Random Sampling Eligibility Criteria .....	472
Table E-9.	Sampled Instruction Event Register (SIER) .....	473
Table E-10.	Implementation-Dependent Extension to Data Source Encodes .....	476
Table E-11.	Implementation-Dependent Extension Bits for Data Source Encodes (SIER[EXT]) .....	478
Table E-12.	Sampled Instruction Event Register 2 (SIER2) .....	479
Table E-13.	Sampled Instruction Event Register 3 (SIER3) .....	481
Table E-14.	PMU-Related Bits in the Sampled Instruction Address Register (SIAR) .....	482
Table E-15.	PMU-Related Bits in the Sampled Data Address Register (SDAR) .....	482
Table E-16.	SDAR Modes for Continuous Sampling .....	484
Table E-17.	Random Instruction Sampling Modes .....	485
Table E-18.	Random Event Sampling Modes .....	486
Table E-19.	Floating-Point Counter Values .....	488
Table E-20.	IFU Events .....	494
Table E-21.	Branch Events .....	496
Table E-22.	ISU Events .....	499
Table E-23.	VSU Events .....	503
Table E-24.	LSU Events .....	504
Table E-25.	L2 Events .....	511
Table E-26.	L3 Events .....	518
Table E-27.	CPI Stack (Tree Format) .....	526
Table E-28.	CPI Stack Events .....	528
Table E-29.	Marked Events .....	531
Table E-30.	MMU Events .....	536
Table E-31.	PMC Events .....	542
Table E-32.	Source Encodings .....	545
Table E-33.	Reload Events .....	546
Table E-34.	Radix Events .....	548
Table E-35.	Power10 Events by Group .....	558
Table E-36.	Metric Events and Formulas .....	636
Table E-37.	Nest Unit Instrumentation and Grouping .....	661
Table E-38.	Event and Performance Metrics for Fabric Events .....	662



Table E-39. Memory Controller PMU Events and Performance Metrics .....	663
Table E-40. OCMB PMU Events and Performance Metrics .....	663
Table E-41. X Link PMU Events and Performance Metrics .....	664
Table E-42. A Link PMU Events and Performance Metrics .....	664
Table E-43. OTL PMY Events and Performance Metrics .....	665
Table E-44. PEC PMU Events .....	666
Table E-45. PHB PMU Events .....	666
Table E-46. Nest PMU Events .....	667
Table G-1. Power10 Errata .....	681



## List of Figures

Figure 1-1.	Power10 Processor Block Diagram (SMT4 Chip Variant) .....	25
Figure 1-2.	Power10 Signaling Technology .....	26
Figure 1-3.	Power10 OMI .....	27
Figure 1-4.	PowerAXON Interface .....	28
Figure 1-5.	Power10 Building Blocks .....	29
Figure 1-6.	Power10 “Racetrack On-Chip” Bandwidth Diagram .....	31
Figure 2-1.	Power10 Single-Chip Module (Logical Diagram) .....	34
Figure 2-2.	Power10 Single-Chip Module (Physical Diagram) .....	35
Figure 2-3.	Power10 Dual-Chip Module (Logical Diagram) .....	37
Figure 2-4.	Power10 Dual-Chip Module (Physical Diagram) .....	38
Figure 3-1.	Power10 Core Microarchitecture (per SMT4-Core-Resource) .....	40
Figure 3-2.	Power10 Core Instruction Pipeline Segments .....	43
Figure 3-3.	Power10 Core Instruction Pipeline Stages .....	44
Figure 3-4.	Power10 Core Issue Queues .....	47
Figure 3-5.	Dense Math Engine .....	50
Figure 4-1.	SMT4 Core in ST and SMT2 Modes .....	56
Figure 4-2.	SMT4 Core in SMT4 Mode .....	56
Figure 4-3.	SMT8 Core with Each SMT4-Core-Resource in ST or SMT2 Modes .....	57
Figure 4-4.	SMT8 Core Shown in SMT8 Mode with Each SMT4-Core-Resource in SMT4 Mode .....	57
Figure 5-1.	LPM of a POWER8/POWER9 Transaction to the Power10 Processor .....	97
Figure 5-2.	<b>tlbie</b> Instruction Format for the Power10 Core .....	127
Figure 5-3.	<b>tlbie</b> Operands for the Power10 Core .....	128
Figure 5-4.	<b>tlbiel</b> Instruction Format for the Power10 Core .....	129
Figure 5-5.	<b>tlbiel</b> Operands for the Power10 Core .....	130
Figure 6-1.	Power10 Block Diagram of Multiple SMT4 Processor Cache-Slices Interconnected via the On-Chip SMP Interconnect Fabric .....	169
Figure 6-2.	L2 Bus Bandwidths .....	173
Figure 7-1.	Block Diagram of Multiple SMT4 Core/L2/Local-L3 Regions Interconnected via the On-Chip SMP Interconnect Fabric .....	175
Figure 8-1.	1-Hop SMP Topology .....	185
Figure 8-2.	2-Hop SMP Topology .....	185
Figure 8-3.	Power10 System Real-Address Map .....	187
Figure 9-1.	NCU Block Diagram .....	189
Figure 10-1.	Power10 System Memory High-Level Diagram .....	192
Figure 11-1.	NX Block Diagram .....	197
Figure 12-1.	Flow for NX Invocation through the VAS .....	202
Figure 14-1.	Power10 Nest MMU .....	214
Figure 14-2.	Window/Process Element Context .....	217



Figure 15-1. Interrupt Presentation Interaction .....	221
Figure 16-1. PCIe Major Blocks .....	223
Figure 16-2. Power10 PCIe High-Level Diagram .....	225
Figure 19-1. High-Level Pipeline Diagram (POWER9 Core versus Power10 Core) .....	246
Figure 19-2. Execution Capability Block Diagram .....	247
Figure 19-3. Decode/Dispatch Pipeline .....	255
Figure 19-4. Decode/Dispatch Pipeline (SMT4 Mode) .....	255
Figure 19-5. Example of a Sequence with 2-Way Crack and Forced Empty Slot .....	257
Figure 19-6. Dispatch - Issue Queue Assignments for ST and SMT2 Mode .....	258
Figure 19-7. Dispatch - Issue Queue Assignments for SMT4 Mode .....	258
Figure 19-8. Dispatch (Paired Instructions) .....	259
Figure 19-9. Cycle 1 with Four Slices Full .....	259
Figure 19-10. Cycle 2 with Four Slices Full .....	260
Figure 19-11. Issue of Instructions .....	263
Figure E-1. Marked Load Events .....	491
Figure E-2. Marked Store Events .....	491
Figure E-3. Branches .....	492
Figure E-4. CPI Breakdown as a Tree .....	525
Figure E-5. Power10 Raw Event Coding .....	656
Figure E-6. Power10 Block Diagram and Corresponding Nest Units (Shown with SMT8 Core Variant) .....	659



## Revision Log

Each release of this document supersedes all previously released versions. The revision log lists all significant changes made to the document since its initial release. In the rest of the document, change bars in the margin indicate that the adjacent text was modified from the previous release of this document.

Revision Date	Description
13 September 2021	Version 1.0. Initial version.





## About this Document

This user's manual describes the IBM® Power10™ processor and provides information about the registers, facilities, initialization, and use of the Power10 processor.

This document provides information about the Power10 processor that is visible from a programming model point of view, and is intended to be a companion to the baseline architecture documentation (see *Related Documents* on page 24). While there are some programming model considerations associated with chips and subsystems outside of the Central Electronics Complex (CEC), this document focuses primarily on the microprocessor core and the storage subsystem. For information about other chips that might appear in Power10 systems, see the functional specifications for these individual chips.

## Who Should Read this Document

This manual is intended for system software and hardware developers and application programmers who want to develop products for the Power10 processor. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of reduced instruction set computer (RISC) processing, and details of the Power ISA.

## Conventions Used in This Document

This section explains numbers, bit fields, instructions, and signals that are in this document.

### Representation of Numbers

Numbers are generally shown in decimal format, unless designated as follows:

- Hexadecimal values are preceded by an “x” and enclosed in single quotation marks.  
For example: x'0A00'.
- Binary values in sentences are shown in any of the following formats.  
For example: ‘1010’ or b‘1010’ or 0b00.
- Decimal values are preceded by an “d” and enclosed in single quotation marks.  
For example: d‘80’.

**Note:** A bit value that is immaterial, which is called a “don't care” bit, is represented by an “X.”

### Bit Significance

In the Power10 documentation, the smallest bit number represents the most significant bit of a field, and the largest bit number represents the least significant bit of a field.

### Other Conventions

Instruction mnemonics are shown in lower-case, bold text. For example: **tlbie**, the I/O signal names are shown in upper case.



## Related Documents

The following documents can be helpful when reading this specification. Contact your IBM representative to obtain any documents that are not available through the [IBM Portal for OpenPOWER](#) or the [OpenPOWER foundation](#).

*Power ISA User Instruction Set Architecture - Book I (Version 3.1B)*

*Power ISA Virtual Environment Architecture - Book II (Version 3.1B)*

*Power ISA Operating Environment Architecture - Book III (Version 3.1B)*

*Power10 Processor Programming Model Bulletin (publication pending)*

*Linux on Power Architecture Platform Reference*

[\*PCI Express Base Specification\*](#), Revision 4.0

[\*PCI Express Base Specification\*](#), Revision 5.0

*IBM EnergyScale for POWER9® Processor-Based Systems*

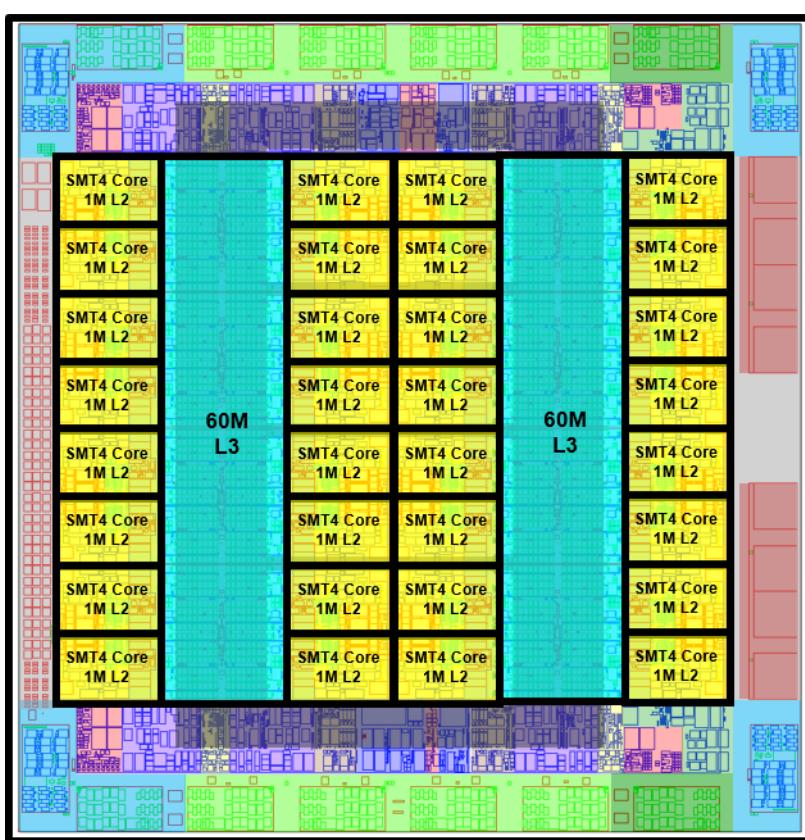
*Manual for Using WBEM CLI to Fetch Flexible Service Processor CIM Data*

*Coherent Accelerator Interface Architecture (CAIA)*

## 1. Power10 Processor Overview

The Power10 processor is a superscalar symmetric multiprocessor designed for use in servers and large-cluster systems. It uses CMOS 7 nm technology with 18 metal layers. Each processor die building block contains up to 30 high-performance, 4-threaded general-purpose cores (in the SMT4 variant), each with 1 MB of low-latency, high-bandwidth L2 cache; or 15, 8-threaded general-purpose cores (in the SMT8 variant) each with 2 MB of L2 cache. The cores share up to 120 MB of latency optimized NUCA L3 cache. Each processor die building block contains up to  $16 \times 8$  lanes of open memory interface (OMI) memory at up to 32 GTps, up to  $14 \times 9$  lanes of SMP fabric interconnect at up to 32 GTps, up to  $12 \times 8$  lanes of OpenCAPI attach at up to 32 GTps, and up to 32 lanes of PCIe Gen 5 attach at 32 GTps. Figure 1-1 provides a block diagram of the Power10 processor.

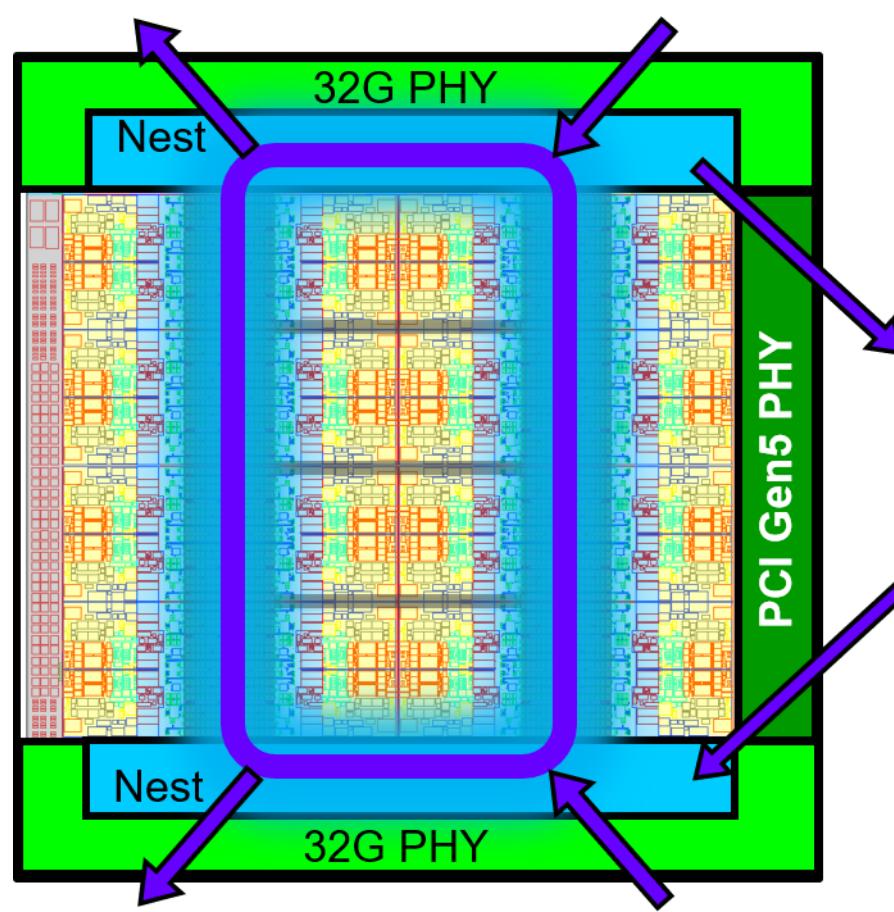
Figure 1-1. Power10 Processor Block Diagram (SMT4 Chip Variant)



Targeted for a broad range of applications, flexibility and configurability are key themes governing the Power10 design point:

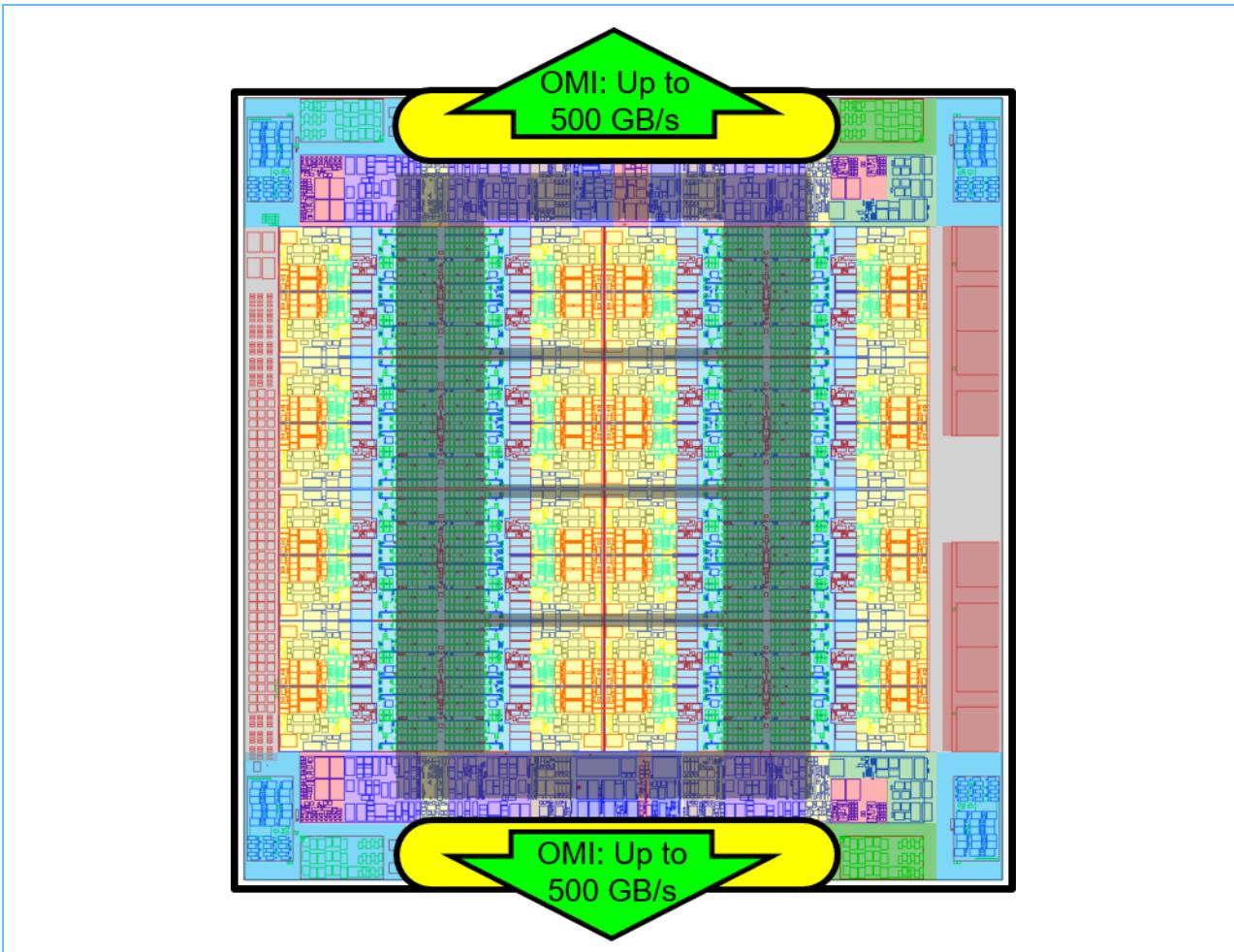
A single, highly-optimized, 32G differential signaling technology services most functional interfaces connecting the processor to other system elements (the only other functional signaling technology is the industry-standard PCIe Gen5 technology integrated into the Power10 processor). The 32G signaling technology is deployed as two interface types, both of which support a myriad of protocols and configurations. *Figure 1-2* illustrates the Power10 signaling technology.

*Figure 1-2. Power10 Signaling Technology*



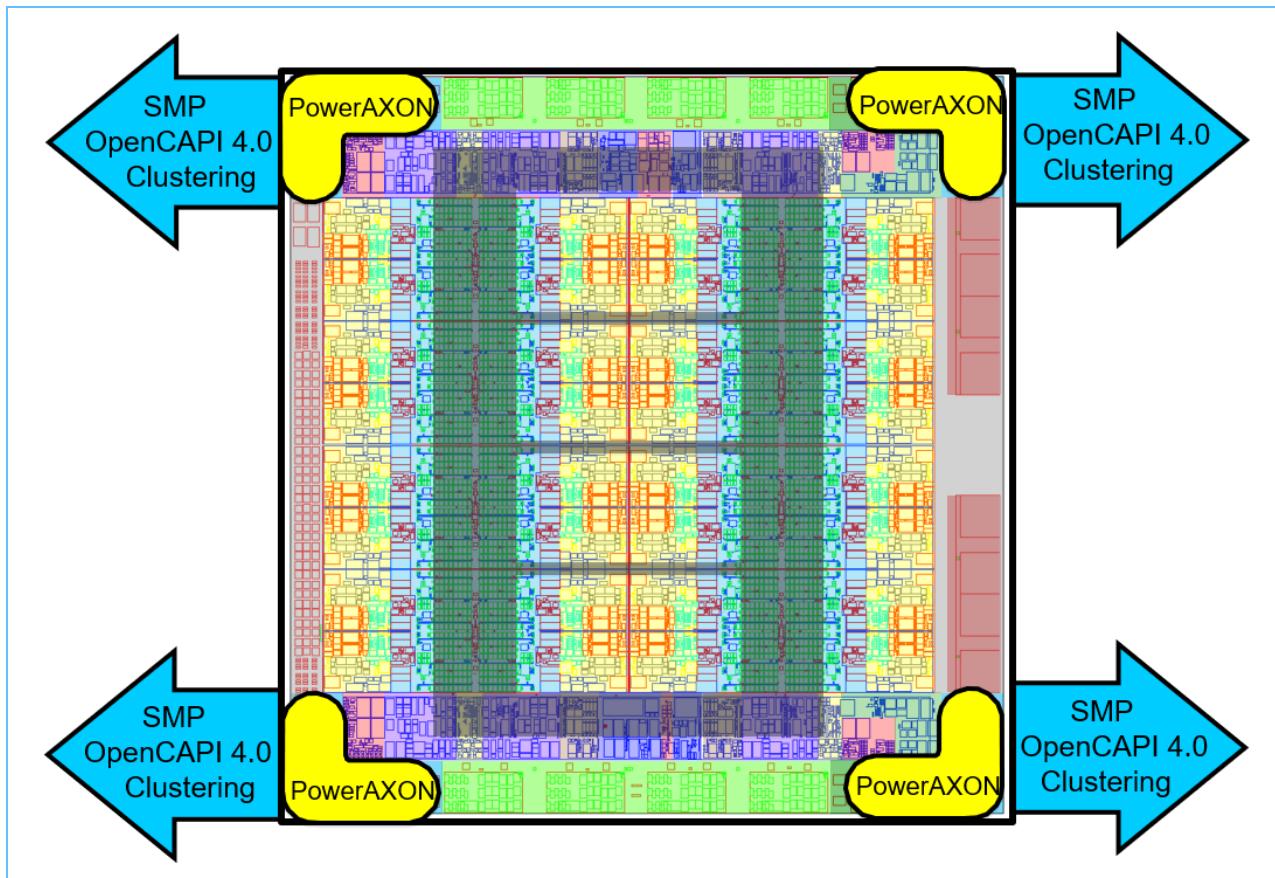
The first type of interface, the open memory interface (OMI), provides a single physical interface that enables low latency, high-bandwidth, technology-agnostic host memory semantics to attach both established and emerging memory elements to the processor. These include main tier, low-latency, commodity and enterprise-grade DDR4 and DDR5; near-tier, high-bandwidth GDDR and HBM; and storage-tier, high capacity, persistent phase-change and flash-derivative technologies. *Figure 1-3* illustrates the OMI.

*Figure 1-3. Power10 OMI*



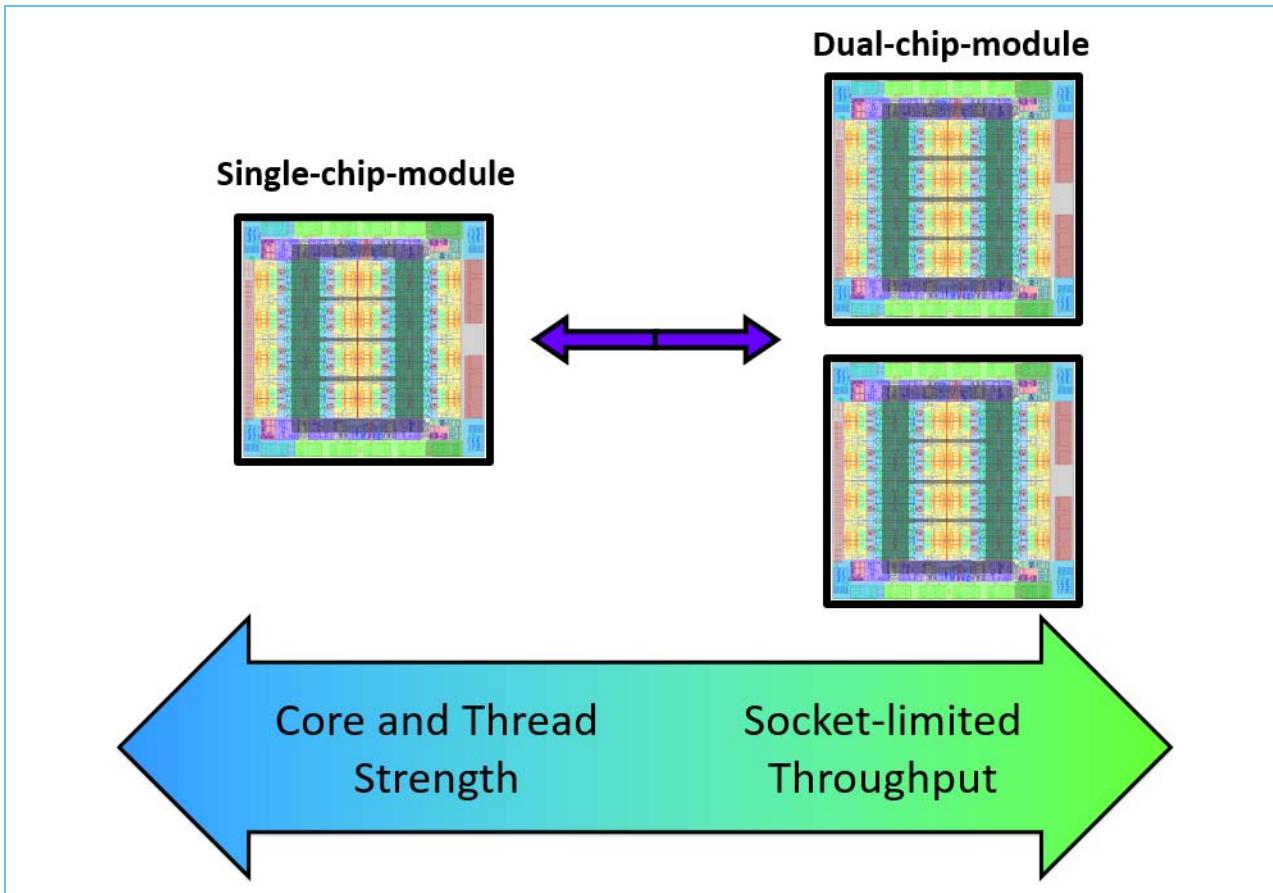
The second type of interface, the PowerAXON, provides a single physical interface that can be configured as a first- or second-tier, glueless SMP-link interface, enabling up to 16 Power10 processors to be combined into a large, robustly scalable, single-system image; or as an OpenCAPI interface, to attach the cache-coherent and I/O-coherent computational accelerators, load/store addressable host memory devices, low-latency network controllers, and intelligent storage controllers into a Power10-based system; or as a host-to-host integrated memory clustering interconnect, enabling multiple Power10 systems to directly use memory throughout the cluster. *Figure 1-4* illustrates the PowerAXON interface.

*Figure 1-4. PowerAXON Interface*



The Power10 processor chip has been designed and optimized as a building block which can be used to cover a wide variety of single die (single-chip module) and dual die (dual-chip module) per socket configurations. The high maximum frequency at maximum voltage and robust core/cache microarchitecture are foundational to single-chip module (SCM) scenarios, while the strong focus on energy efficiency and SMP link stranding technology are foundational to dual-chip module (DCM) scenarios. SCM options are best suited for cost, core/thread strength, and bandwidth-per-computation optimization; while DCM options favor balanced thread/socket performance and extreme socket performance optimization. *Figure 1-5* describes the Power10 building blocks.

*Figure 1-5. Power10 Building Blocks*



Similar to the POWER9 core, the Power10 processor core elements are comprised of modular building blocks, enabling two variants of the processor core: a smaller, 4-threaded version, targeted at the bare-metal Linux market and some segments of the KVM Linux market, and a larger, 8-threaded version, targeted at the PowerVM market, as well as some segments of the KVM Linux market. Unless otherwise specified, references to the Power10 processor core in this document will describe the SMT4 variant.



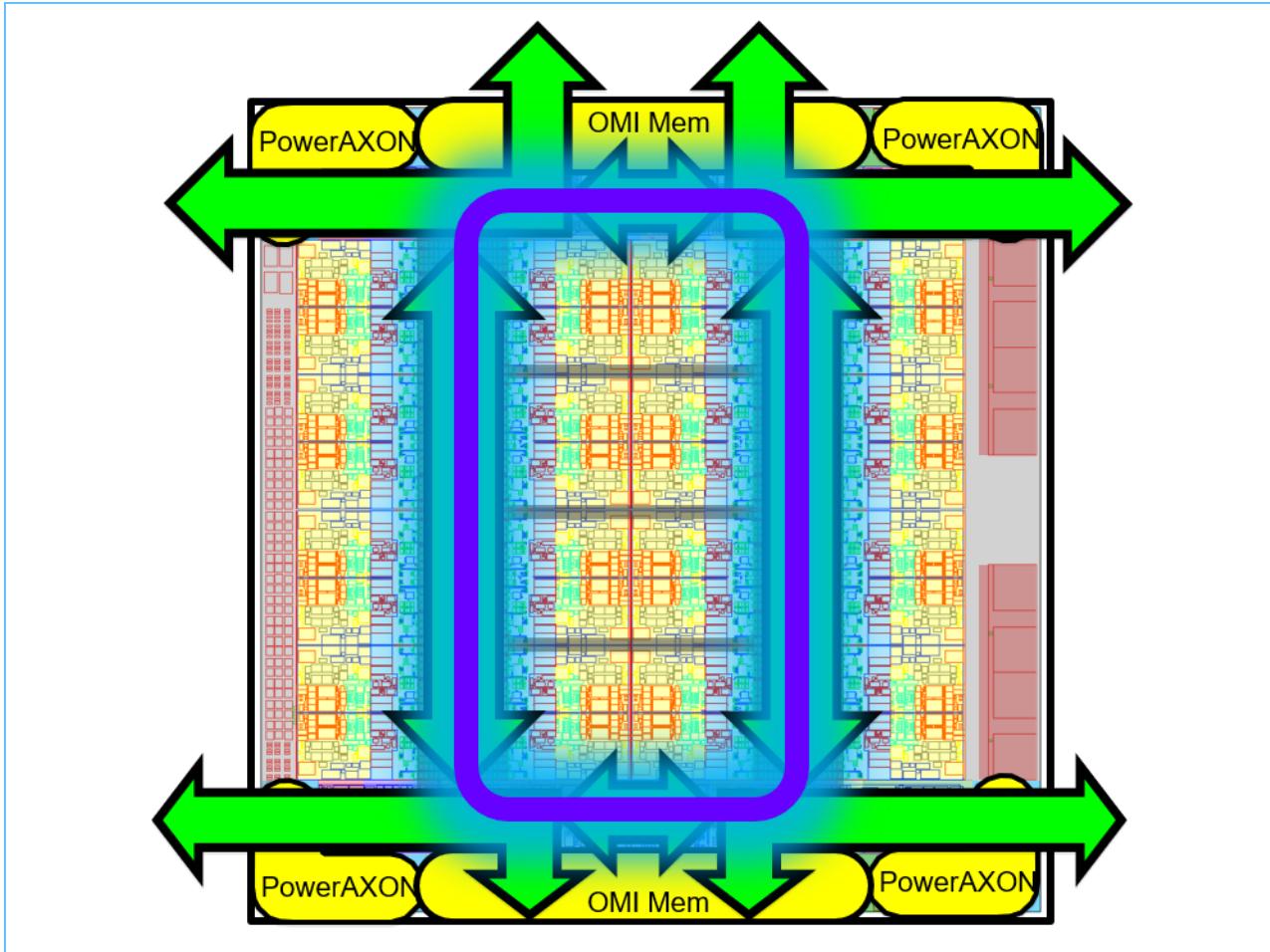
The Power10 processor offers superior performance, reliability, energy efficiency, and cost for a broad range of on-premise and cloud-focused offerings. The target application segments are:

- AIX and IBM i environments built upon PowerVM. Systems ranging from 1 - 16 processor chips, using the SMT8 variant of the processor core, and targeting the established and emerging workloads that demand mission-critical RAS, security, and scale.
- Enterprise Linux environments running on systems ranging from 1 - 16 processor chips, using the SMT8 variant of the processor core, and targeting mission-critical, large scale Linux workloads such as in-memory databases, which demand high-memory capacity, robust scaling, and high reliability.
- Cognitive and high-performance computing environments running on systems ranging from 1 - 4 sockets and typically leveraging the SMT4 variant of the processor core. The processor core provides superior SIMD performance across a wide variety of data types targeted toward both emerging AI training and inferencing workloads as well as established and emerging HPC workloads. The chip also supports high-bandwidth, off-chip accelerator attach capabilities, a robust PCIe I/O subsystem, and high-bandwidth OMI memory to address these segments.
- Hyperscale data center deployments as well as OpenPOWER partner-driven deployments. Systems will focus on one- and two-socket form factors using the SMT4 variant of the processor core. The Power10 processor has been re-engineered with a strong focus on the energy efficiency and socket performance combined with differentiated acceleration, memory, and I/O capabilities.

Given these objectives, five major architectural focus areas characterize the Power10 design point:

- **System data bandwidth:** beyond its intrinsic, general-purpose computational capabilities, the Power10 chip can be viewed as a high-bandwidth switch, using its highly-optimized, 32G signaling technology to connect PowerAXON-attached computational elements (compute accelerators, or other Power10 processors) to OMI-attached memory devices in a latency-optimized manner at unprecedented bandwidth and scale. The entire processor die layout has been re-organized to enable a “racetrack” on-chip coherence and data ring topology, with PowerAXON, OMI, and PCIe-protocol infrastructure located adjacent to 32G and PCIe signaling technology on the chip edge. The optimized PowerAXON-to-race-track-to-OMI connectivity supports up to 1 terabyte-per-second (TBps) end-to-end switching bandwidth in-and-out of the processor, making the Power10 chip an ideal hub for constructing heterogeneous, specialized data/compute architectures. *Figure 1-6* on page 31 illustrates the Power10 racetrack on-chip bandwidth.

Figure 1-6. Power10 “Racetrack On-Chip” Bandwidth Diagram



- **Cognitive-infused general-purpose compute architecture:** despite optimizations for connecting to off-chip computation-acceleration, the Power10 chip is strongly focused on providing industry-leading SIMD execution capabilities in its general-purpose, Power ISA, processor cores. The ISA has been enhanced to support the reduced-precision data types and matrix-math operations essential to high-performance inferencing and learning kernels, while maintaining focus on higher-precision data types essential to existing analytic and high-performance-computing applications. In addition to this, increased investments in robust execution prediction, address translation caching, and instruction/data caching provide strong value for workloads ranging from emerging scripting frameworks, to data analytics, transaction processing, and data-intensive enterprise applications, as well as providing robust virtualization for consolidated enterprise environments and container density for multi-tenancy cloud environments.
- **Energy-efficiency:** every aspect of the Power10 processor has been re-engineered with a strong focus on energy-efficiency. This substantial improvement, compared to prior POWER processors, combined the modular building block approach, which enables optimized dual-chip module sockets for one-, two-, and four-socket, high-density data center and cloud deployments. These optimizations also bring value to higher-power sockets found in enterprise applications, by enabling a larger number of high-clock-speed, thread-strength optimized cores per socket, and eliminates the requirement to modulate clock speed based upon workload intensity.



- **Thread-strength:** there is an inherent trade-off involved in biasing a single offering either toward thread-strength or socket-throughput. But the Power10 chip's modular SCM/DCM building block approach enables the DCM to focus on throughput, freeing silicon area for strong microarchitecture investments in thread strength. Instruction-per-cycle (IPC) improvements are provided by the more robust SIMD architecture, execution prediction, translation/instruction/data caching, internal bandwidth, and execution resource provisioning, which are enabled by providing more silicon area per core. Higher clock speeds are enabled by the strong energy-efficiency focus and the strong advances in the 7 nm semiconductor technology used by the Power10 chip. The combination of silicon area, IPC improvements, and higher clock speeds result in substantial thread strength improvement compared to prior POWER processors.
- **End-to-end security:** the Power10 processor contains numerous hardware elements, which when combined with corresponding firmware and system software elements comprise a high performance set of end-to-end security solutions. Ranging from pro-active, side-channel attack, detection/avoidance mechanisms to hardware-enforced, host-processor secure-memory isolation for secure container enablement; secure memory isolation for secure-attached accelerator support; two forms of full main memory encryption (targeted at both volatile and non-volatile memory technologies); cryptographic accelerators; application-enabled protections; and secure-trusted-boot enablement; these elements enable Power10-based systems to provide mission-critical security not only for on-premise and private cloud environments but also for public and hybrid cloud environments.

## 2. Packages

This chapter outlines the module packages for the Power10 processor.

### 2.1 Power10 Single-Chip Module (68.5 x 77.5 mm)

The Power10 SCM has the following general features:

- Body size: 68.5 mm × 77.5 mm
- Interconnect technology: Hybrid LGA socket
- 1.5 mm interstitial LGA pitch with minimum pitch of 1.06 mm and 4445 pins
- 8-2-8 organic package construction

#### 2.1.1 Bus Features

The bus features are as follows:

- 72 A-bus/X-bus/OpenCAPI lanes to the Nearstack connectors on top of the module:
  - Each OP1, OP2 runs as a  $2 \times 9$  SMP bus at 32 Gbps
  - Each OP4, OP6 runs as one of the following two modes:
    - $2 \times 9$  SMP at 32 Gbps
    - $2 \times 8$  OpenCAPI at 32 Gbps
- 72 A-bus/X-bus/OpenCAPI lanes to the bottom of the module:
  - Each OP0, OP3, OP5, OP7 runs as one of the following two modes:
    - $2 \times 9$  SMP at 32 Gbps
    - $2 \times 8$  OpenCAPI at 32 Gbps
- 128 OMI lanes to the bottom of the module:
  - Each OMI port is composed of  $2 \times 8$  lanes
  - Total  $16 \times 8$  lanes for OMI[0:7]
  - Runs at 32 Gbps
- 32 PCIe lanes to the bottom of the module:
  - Each E0, E1 runs as one of the following five modes:
    - $1 \times 16$  Gen4 at 16 Gbps
    - $2 \times 8$  Gen4 at 16 Gbps
    - $1 \times 8$  Gen4,  $2 \times 4$  Gen4 at 16 Gbps
    - $1 \times 8$  Gen5 at 32 Gbps,  $1 \times 8$  Gen4 at 16 Gbps
    - $1 \times 8$  Gen5 at 32 Gbps,  $2 \times 4$  Gen4 at 16 Gbps

Figure 2-1 shows a logical diagram of the Power10 SCM.

Figure 2-1. Power10 Single-Chip Module (Logical Diagram)

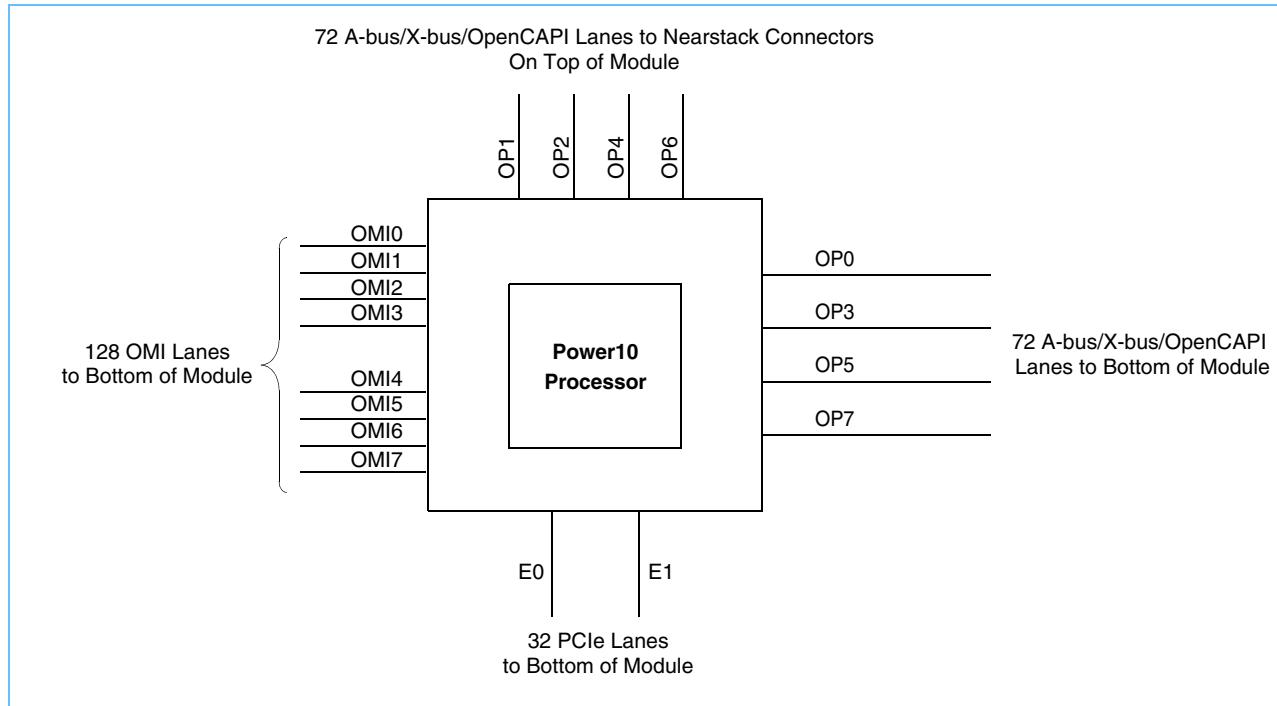
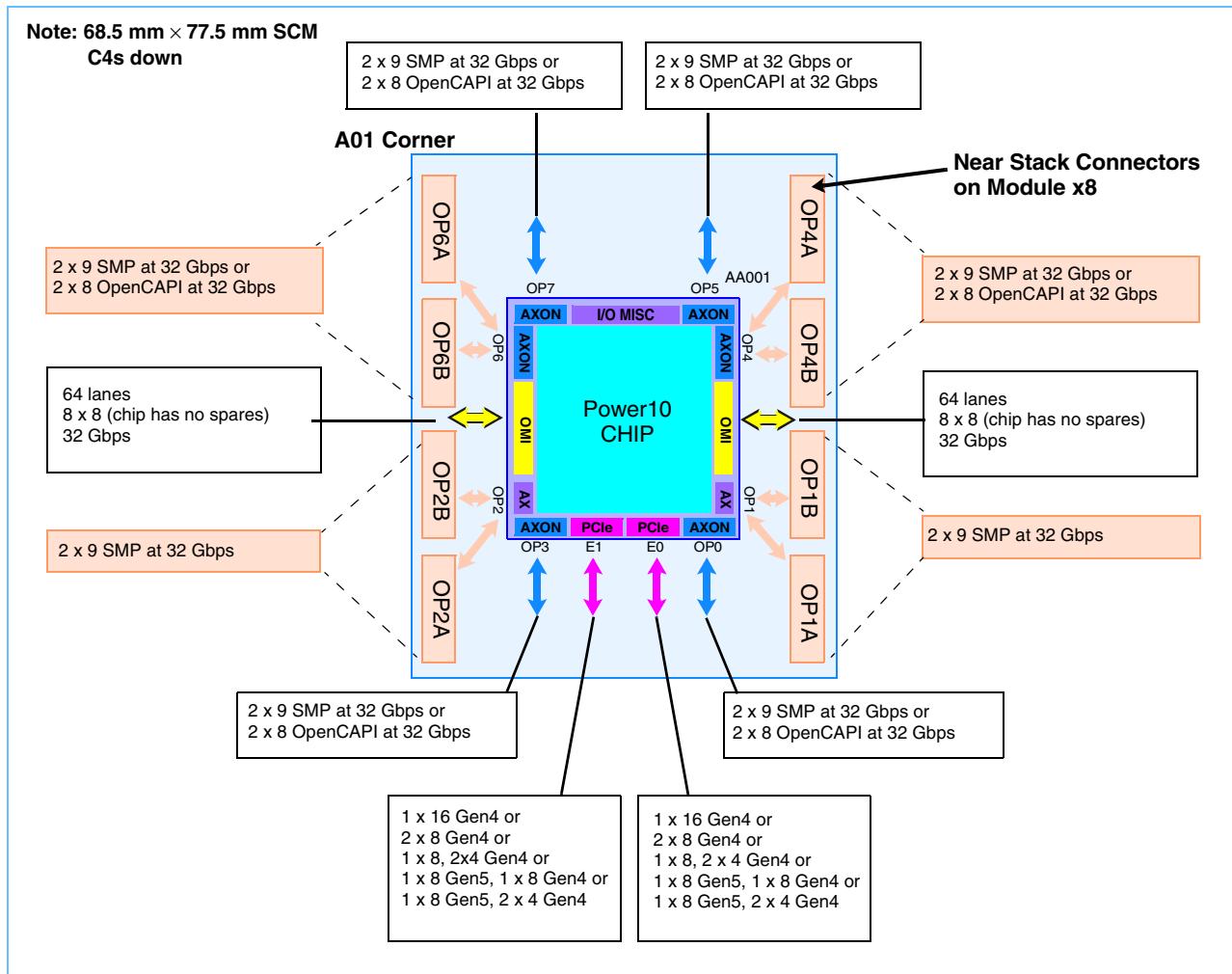


Figure 2-2 shows a physical diagram of the Power10 SCM.

Figure 2-2. Power10 Single-Chip Module (Physical Diagram)



## 2.2 Power10 Dual-Chip Module (74.5 x 85.75 mm)

The Power10 DCM has the following general features:

- Body size: 74.5 mm × 85.75 mm
- Interconnect technology: Hybrid LGA socket
- 1.5 mm interstitial LGA pitch with a minimum pitch of 1.06 mm and 5387 pins
- 8-2-8 organic package construction

### 2.2.1 Bus Features

The Power10 DCM has the following bus features:

- 36 X-bus lanes for chip-to-chip connection:
  - 2 × 9 OP2 of chip 0 connects to 2 × 9 OP1 of chip 1
  - 2 × 9 OP6 of chip 0 connects to 2 × 9 OP4 of chip 1
  - Run as an SMP bus at 32 Gbps
- 216 A-bus/X-bus/OpenCAPI lanes to the bottom of the module:
  - OP0, OP1, OP3, OP4, OP5, OP7 from chip 0
  - OP0, OP2, OP3, OP5, OP6, OP7 from chip 1
  - Each OP1, OP2 runs as a 2 × 9 SMP bus at 32 Gbps
  - Each OP0, OP3, OP4, OP5, OP6, OP7 runs one of the following two modes:
    - 2 × 9 SMP at 32 Gbps
    - 2 × 8 OpenCAPI at 32 Gbps
- 128 OMI lanes to the bottom of the module:
  - OMI [0:3] from chip 0
  - OMI [4:7] from chip 1
  - Each OMI port is composed of 2 × 8 lanes
  - Total 16 × 8 lanes for OMI[0:7]
  - Runs at 32 Gbps
- 64 PCIe lanes to the bottom of the module
  - E0, E1 from chip 0
  - E0, E1 from chip 1
  - Each E0, E1 can run as one of the following five modes:
    - 1 × 16 Gen4 at 16 Gbps
    - 2 × 8 Gen4 at 16 Gbps
    - 1 × 8 Gen4, 2 × 4 Gen4 at 16 Gbps
    - 1 × 8 Gen5 at 32 Gbps, 1 × 8 Gen4 at 16 Gbps
    - 1 × 8 Gen5 at 32 Gbps, 2 × 4 Gen4 at 16 Gbps

Figure 2-3 shows a logical diagram of the Power10 DCM.

Figure 2-3. Power10 Dual-Chip Module (Logical Diagram)

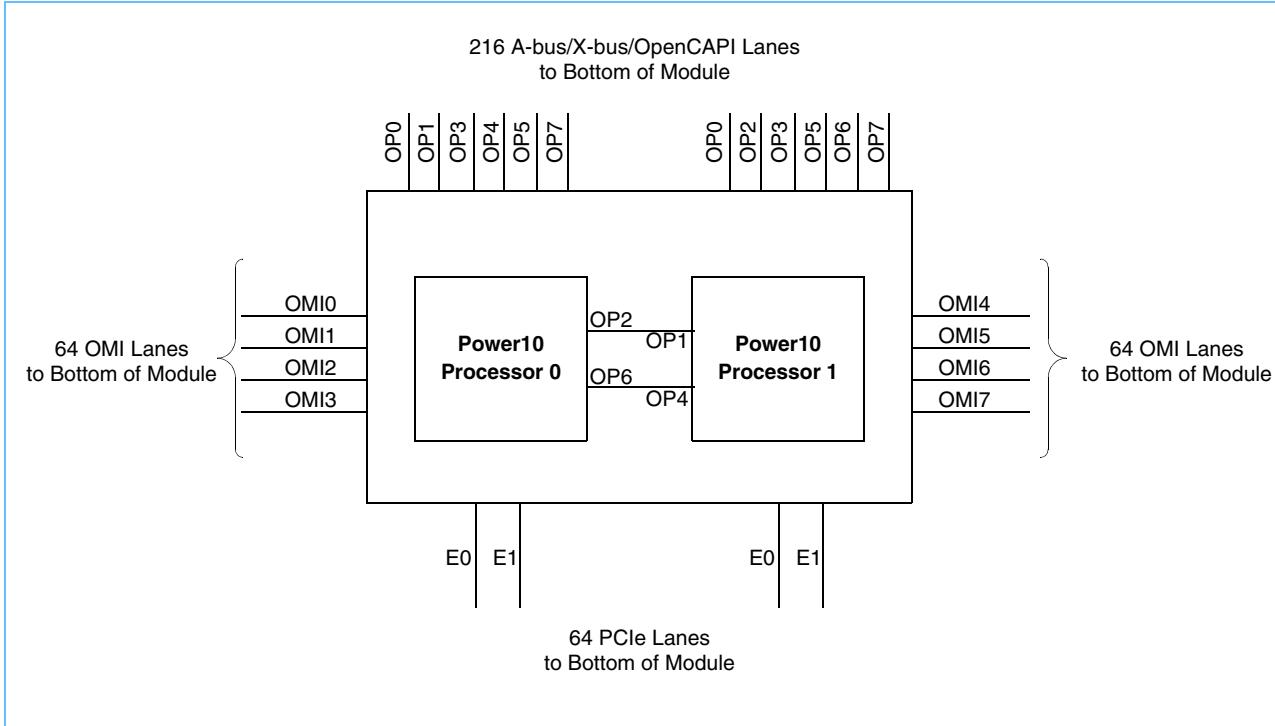
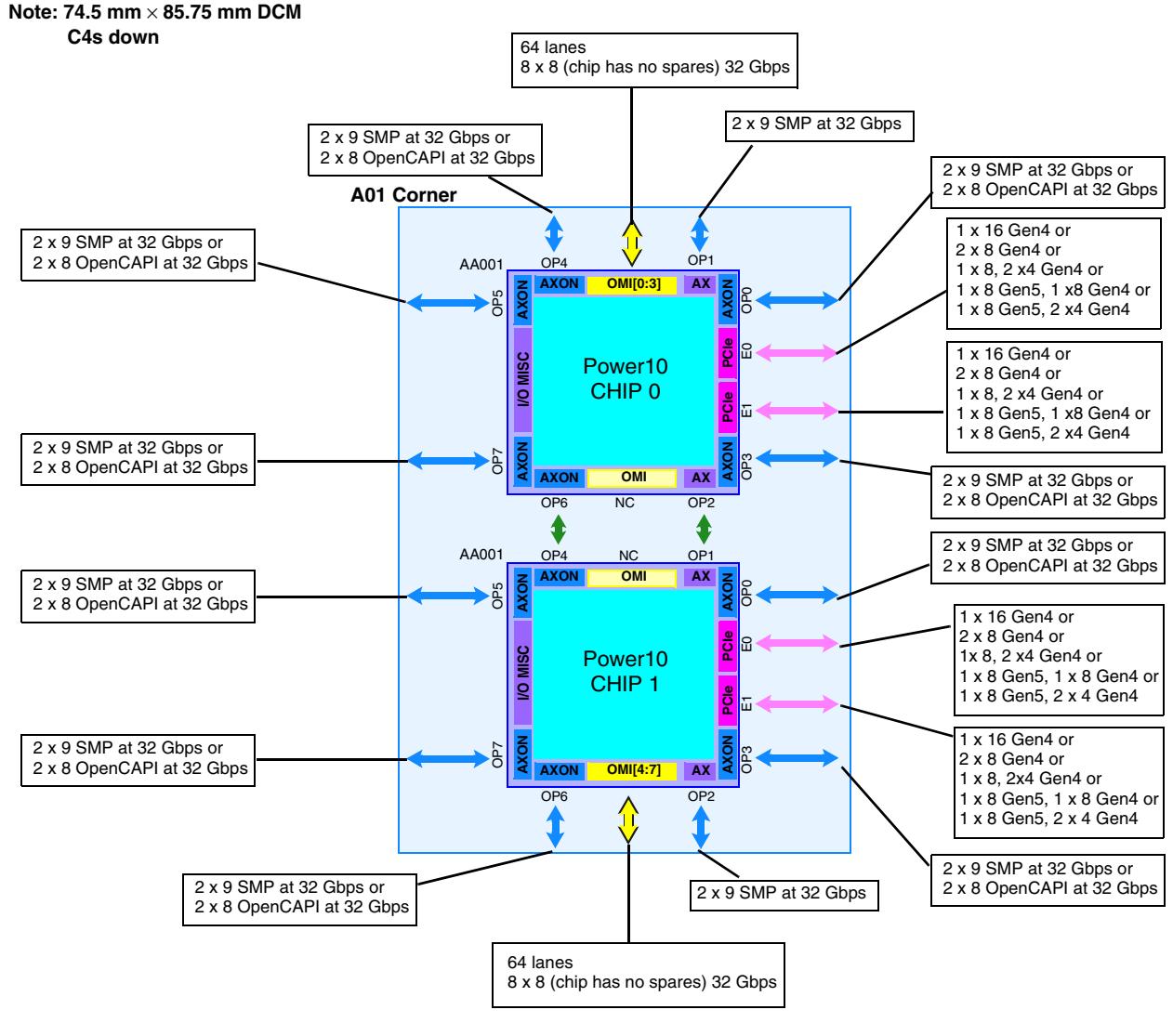


Figure 2-4 shows a physical diagram of the Power10 DCM.

Figure 2-4. Power10 Dual-Chip Module (Physical Diagram)



## 3. Power10 Processor Core

### 3.1 Core Overview

This section details the micro-architecture and major design elements of the Power10 processor core and its operation. Subsequent subsections of this chapter detail various features and capabilities in further detail.

#### 3.1.1 Introduction

The Power10 processor core is architected to retain the modular architecture from the POWER9 core while dramatically improving performance and processing efficiency. Additionally, new execution capabilities and improved cache bandwidth characteristics result in dramatic improvements in peak computational throughput with specialized engines optimized for matrix math acceleration and improved machine learning performance.

#### 3.1.2 Instruction Set Architecture

The Power10 processor core implements the *Power ISA (Version 3.1B)* and elements of the *Power10 Processor Programming Model Bulletin*. This architecture introduces the prefixed 8-byte instruction format along with hundreds of new instructions and features.

The following operating systems and modes are supported:

- AIX operating system
- IBM i operating system
- Linux operating system
- PCR modes support backward compatibility of supported facilities: POWER9 and POWER8 modes. See *Section 5.10.33 Processor Compatibility Mode* on page 167 for additional information.
- Implementation-specific and compliance details can be found in *Section 5 Architecture Compliance* on page 73.

#### 3.1.3 Core Variants and Logical Partitioning (LPAR) Modes

The modular design of the Power10 core provides for two core variants. An “SMT8 core” supports up to eight simultaneously active threads and has double the resources of an “SMT4 core” which supports up to four simultaneously active threads. Each Power10 chip is fabricated to support either SMT8 or SMT4 cores. The SMT8 cores are designed to operate with the PowerVM firmware and hypervisor.

The SMT8 cores can operate with either one partition per thread or one partition per core. The SMT4 cores can operate with one partition per thread.

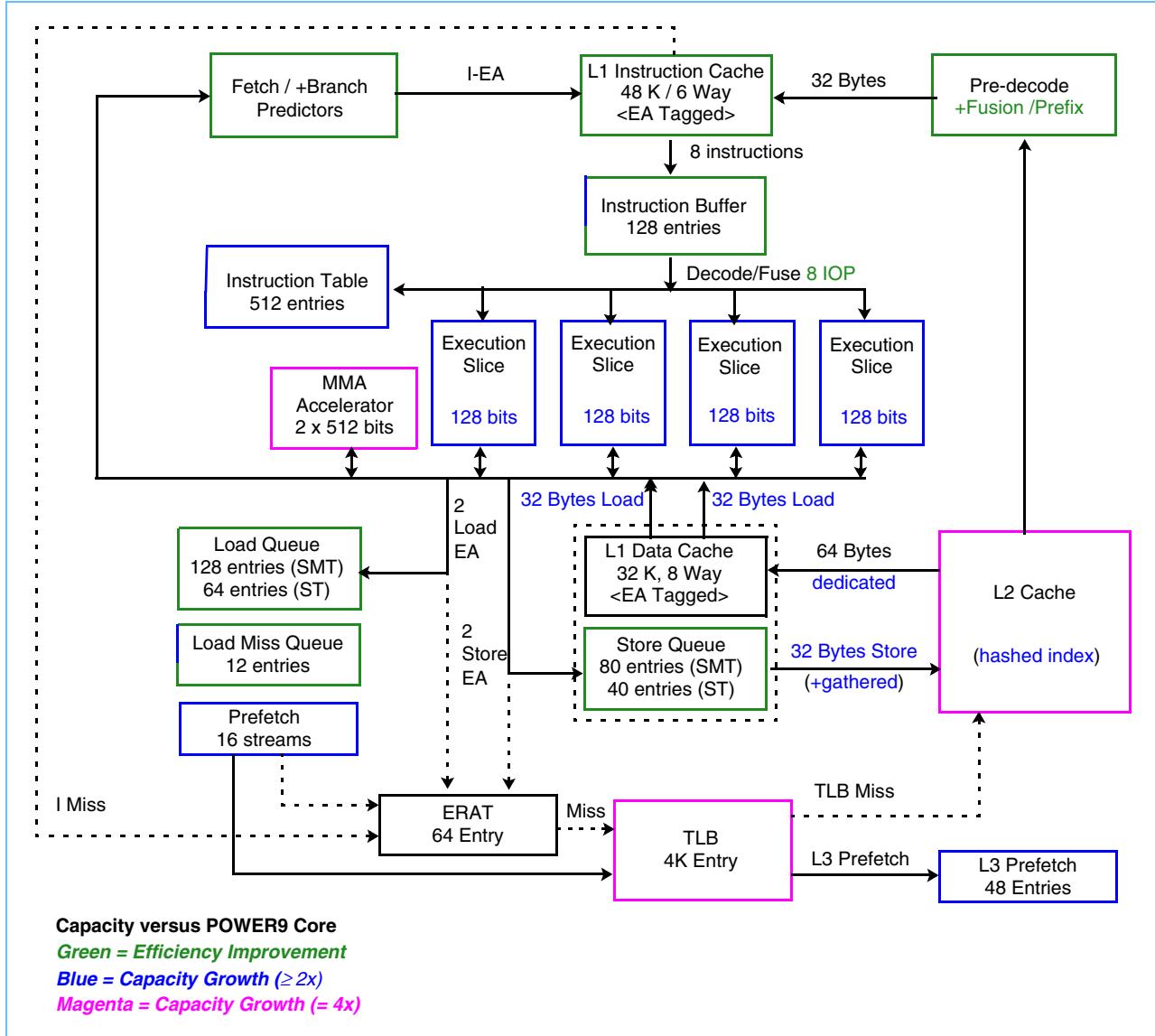
Execution resources and capability in the remainder of this chapter and subsequent chapters will refer to the resources and capability of up to four threads, herein called an “SMT4-core-resource”, of which there are two for an SMT8 core and one for an SMT4 core unless otherwise specified.

Additional details related to simultaneous multi-threading and core variants are described in *Section 4 Thread and LPAR Management* on page 55.

### 3.1.4 Micro-Architecture Performance Improvement Summary

Figure 3-1 shows a block diagram of major Power10 core organizational features. Relative sizes and capacities compared with the POWER9 core are highlighted for reference.

Figure 3-1. Power10 Core Microarchitecture (per SMT4-Core-Resource)



Some of the key features and improvements impacting performance include the following with capacities shown per SMT4-core-resource (per core for SMT4 core chip variant); actual capacities per core with the SMT8 core chip variant are double:

- Enhanced computation (see *Section 3.2.6 Compute Pipelines* on page 48 for additional information):
  - Four VSU execution slices, each supporting 64-bit scalar or 128-bit SIMD for permute, fixed-point, floating-point, and crypto (AES/SHA) operations.
  - Two VSU execution super-slices each inclusive of two of the VSU execution slices plus 64-bit scalar or 128-bit SIMD fixed-point divide, extended bit-manipulation, quad-precision fixed-point, and BCD.
  - Two units for matrix-math acceleration each capable of producing a 512-bit result per cycle.
  - One unit for quad-precision floating-point and decimal floating-point operations.
- Enhanced bandwidth (see *Section 3.3 Data Handling* on page 51 for additional information):
  - Two load issues per cycle – producing up to 32 bytes each.
  - Load/simple writeback:
    - Two 16-byte load write-back ports.
    - Two additional 16-byte load-simple write-back ports support 32-byte loads, a second load register target, branch target register or add-immediate write back.
  - 64 bytes per cycle from the L2 cache – dedicated per SMT4-core-resource.
  - 32 bytes per cycle from the L3 cache – dedicated per SMT4-core-resource.
  - Two store issues per cycle – draining up to 32 bytes per cycle to the L2 cache with store merging.
- Enhanced data prefetch (see *Section 3.3.3 Data Prefetch* on page 52 for additional information):
  - 16 active prefetch queue streams (hardware or software).
  - 48 L3 prefetch request machines servicing prefetch requests to memory.
  - Independent bandwidth for concurrent L1 and L3 prefetch launch pipelines.
  - Plus-one prefetching at the memory controller for enhanced effective prefetch depth/rate.
  - Power10 software prefetch modes support fetching blocks of data into the L3 cache.
  - Adaptive prefetch modes work in concert with the memory controller and cache to balance bandwidth consumption with prefetch aggressiveness and confidence.
- Larger working-sets:
  - L1 instruction cache: 48 KB 6-way.
  - L1 data cache: 32 KB 8-way.
  - L2 cache: 1 MB 8-way.
  - L3 cache: 4 MB 16-way.
  - L1 effective-to-real address translation (ERAT) 64 entries: data and instruction combined.
  - L2 translation lookaside buffer (TLB): 4K entries.
- Deep instruction windows:
  - 128-entry instruction fetch buffer.
  - 512-entry instruction completion buffer tracks internal operations during out-of-order execution window.



- 80-entry instruction issue queue, one per execution slice with 20 entries each.
- 128-entry load queue with operation tracking and local re-issue.
- 80-entry store queue with operation tracking and re-issue.
- Data access with reduced latencies:
  - L1 data cache access at four cycles nominal with zero penalty for store-forwarding.
  - L2 data access at 13.5 cycles nominal.
  - L3 data access at 27.5 cycles nominal.
  - TLB access at 8.5 cycles nominal for ERAT miss including for nested translation.
- Branch execution and prediction:
  - Branches integrated into main issue queues.
  - Branch target registers LNK, CNT, TAR, integrated into main register file together with GPR and VSR registers for reduced GPR to branch-target latencies.
  - Power10-specific branch predictors and enhanced predictor sizes.
- Instruction fusion:
  - ALU scalar, SIMD, load, store, and branch fusion.
  - Dependent fusion: full fusion results in zero latency between dependent operations; back-to-back fusion results in one-cycle latency; independent fusion results in parallel execution with reduced resource overhead.
- Security features:
  - Support for nested KVM on PowerVM with container isolation.
  - Enhanced hardware for speculation-based attack mitigations.
  - Dynamic Execution Control Register (DEXCR). See *Section 3.5 Security* on page 54 for additional information.
  - Return oriented programming (ROP) protection. See *Section 3.5 Security* on page 54 for additional information.

Details for software optimization related to many of the previously listed features and those described in this chapter can be found in *Section 19 Performance Profile* on page 245.

### 3.1.5 Microarchitecture Efficiency Improvement Summary

The Power10 core is designed to dramatically improve performance per watt compared with prior generations. A number of micro-architectural innovations complement physical and logic design techniques to dramatically improve energy efficiency:

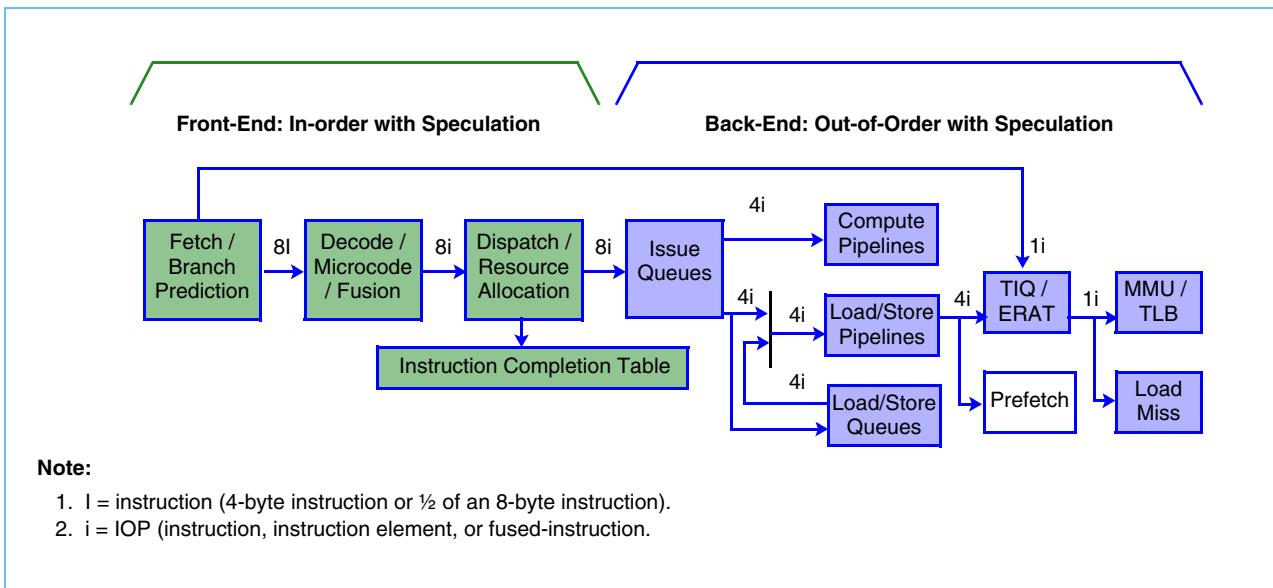
- Improved clock-gating:
  - Increased functional clock gating is employed and clock gating granularity is increased to remove wasted power on idle circuits.
- Design and micro-architecture efficiency:
  - Significant redesign focused on reducing switching capacitance and improving efficiency.  
For example, instruction pairing enables expanded operational widths while limiting tracking energy across a number of structures.

- Branch accuracy:
  - Less wasted work by reducing flush rates with improved branch prediction accuracy.
- Fusion/gather:
  - Less energy per unit of work via operation merging.
- Reduced ports and reduced access:
  - Significant port reduction for power efficiency of structures including reduced CAMs.
  - Sliced target register-file increases access to data without high count read and write port structures.
- EA-tagged L1 D-cache and L1 I-cache:
  - ERAT access only on a cache miss.
  - CAM tracking structures with cache-way/index.

## 3.2 Instruction Flow

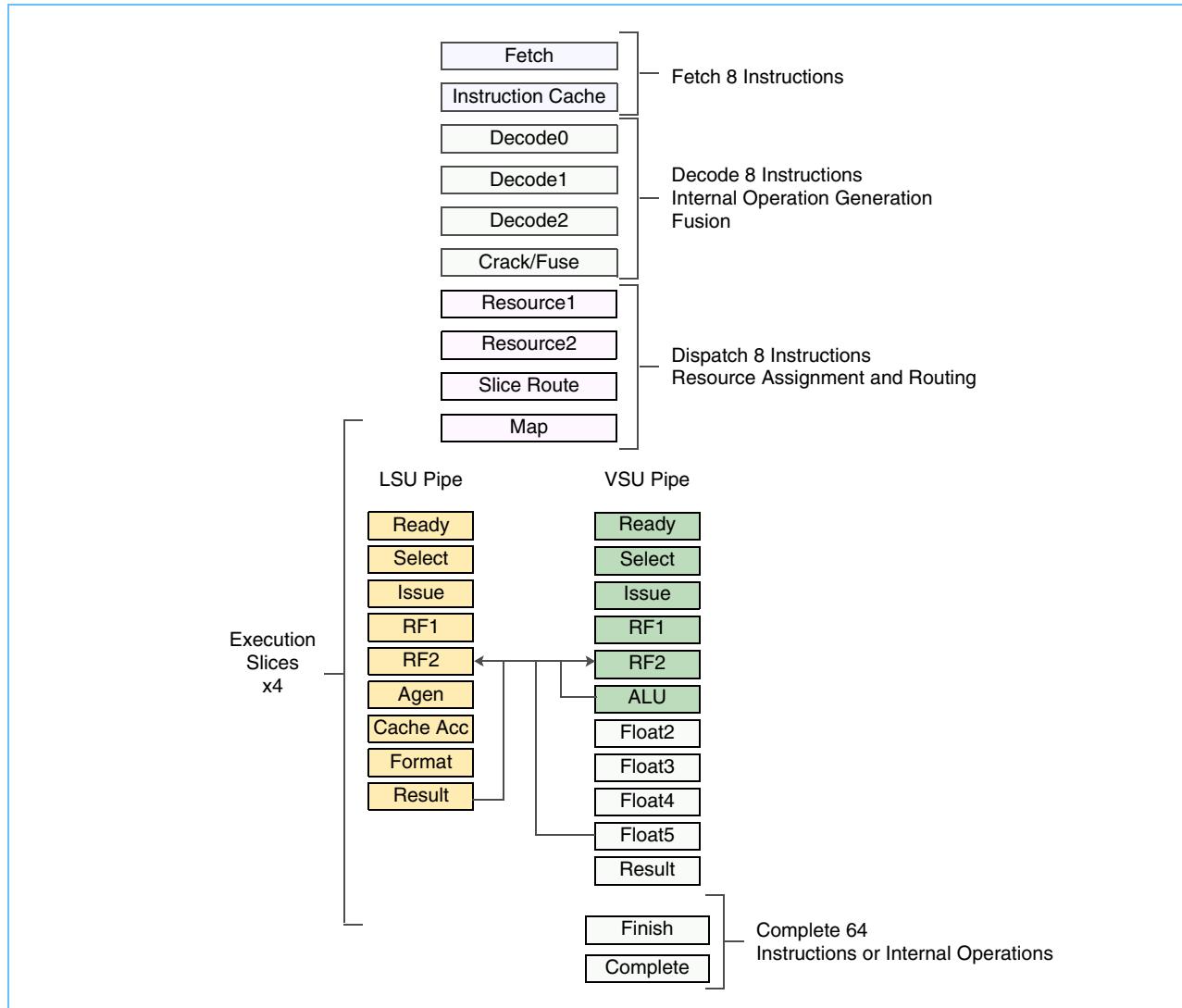
The high-level pipeline segments for instruction flow are depicted in *Figure 3-2*. The front end of the Power10 core operates with aggressive speculation and an in-order pipeline; decoding and dispatching up to eight instructions per cycle. After instruction execution resources are assigned at dispatch, internal operations (iops) are executed out-of-order in multi-slice, super-scalar compute and load/store pipelines.

*Figure 3-2. Power10 Core Instruction Pipeline Segments*



The Power10 core pipeline stages are depicted in *Figure 3-3* on page 44, showing the nominal path through the load and store unit (LSU), arithmetic and logical unit (ALU), and floating-point unit (FPU) pipelines including the shortest latency data forwarding paths.

Figure 3-3. Power10 Core Instruction Pipeline Stages





### 3.2.1 Fetch / Branch Prediction

Instructions are fetched from the L2 cache, 64 bytes to 32 bytes per cycle, then pre-decoded and stored in a 48 KB, 6-way, L1 instruction cache at a rate of up to 32 bytes per cycle. In each cycle, up to eight instructions are read from the L1 instruction cache or bypassed on write.

Fetched instructions are scanned for branches and access a set of advanced predictors for both direction and target address prediction. Predicted taken branches redirect subsequent fetches.

Directional branch predictors include:

- 16 KB, 2-bit local predictor and selector
- 8 KB, 2-bit global predictor and selector
- 4 tables by 512-entry, tagged geometric (TAGE) predictor
- 1024-entry, tagged orientation predictor (TOP)

Target branch predictors for indirect branches include:

- 256-entry, local count cache
- 512-entry, global count cache
- 256-entry, tagged indirect predictor (TIP)
- 64-entry link stack
- 128-entry BTAC

When a fetch misses the L1 cache, the request is serviced by the L2 cache after undergoing address translation. Cache misses conditionally generate up to seven cache-line prefetches based on a prefetch predictor.

Fetched instructions are bypassed into the decode pipeline when available. When decode is stalled or another thread is being selected for decode, the fetched instructions are buffered in a 128-entry instruction buffer.

### 3.2.2 Decode/ Microcode / Fusion

Instruction decode processes up to eight instructions per cycle. The 8-byte prefixed instructions each use two of the eight decode lanes. A subset of instructions are cracked into two, three, or four internal operations. A limited set of instructions are expanded with a micro-code engine that generates internal operations across multiple cycles of the decode pipeline.

Instructions flagged for fusion by pre-decode are processed at instruction decode. Fused instructions can be in one of several categories.

Dependent fusion:

- Full fusion results in zero latency between dependent operations, the second decode slot is either converted to a NOP, conveys additional information entering and exiting the issue queue together, or supports independent work and is issued independently.
- Back-to-back fusion enables a one cycle dependent latency via pipeline forwarding; the fused instructions are tracked in the issue queue together.



Independent fusion:

- Horizontal fusion results in parallel execution in a single internal operation with reduced resource overhead. Secondary operations are converted to a NOP or support independent work.

### 3.2.3 Dispatch and Resource Allocation

Up to eight internal operations or instructions are processed per cycle. Dispatch assigns execution resources as needed for each internal operation. A NOP is finished directly at dispatch and does not execute in a computational pipeline.

The following resources are allocated at dispatch and when not available can cause dispatch holds.

- Issue queue entry and slice assignment
- Register renaming and dependency ordering
- Load/store queue virtual entry

#### 3.2.3.1 Issue Queue Entry and Slice Assignment

Instructions are routed to a slice based on a dynamic policy to balance instructions across the slices considering the type of operation, operation history, and available slice resources. Each execution slice contains an issue queue, native slice execution pipelines, and access to all super-slice execution pipelines including the load/store/branch pipelines, as well as the per SMT4-core-resource execution pipelines.

A subset of instructions and fused instruction pairs use a paired issue queue entry consuming two issue queue slots.

#### 3.2.3.2 Register Renaming and Ordering

Register sources, targets, and inter-dependencies are identified at dispatch and mapped into physical register resources, avoiding most write-after-write (WAW) hazards. Register renaming occurs for the following resources:

- Main Sliced Target Register File (STF), which holds the following architected registers: GPR, FPR, VSR, LNK, CNT, TAR, HSPRG1, and SPRG2.
- XVFC STF, which holds architected fields of the following architected registers: CR, XER, FPSCR, VSCR. Multiple fields can be written in the same physical target register.

Some write-after-write (WAW) hazards and/or read-after-write (RAW) hazards are not completely handled through register renaming. The following scoreboards can cause instructions that are scoreboard checkers to be stalled at dispatch until the scoreboard setter has retired.

- Two dispatch-scoreboards are used to order SPR reads versus writes.
- FPSCR scoreboard restricts the number of writers to the FPSCR sticky bits in flight.

A subset of instructions that set constant values or move registers are executed at dispatch:

- A set of “constant setting instructions” that set values of 0, 1, or -1 into the register targets are executed at dispatch and do not enter the execution pipelines.
- Register moves between a subset of instruction and register types bypass the register dependency at dispatch if the producer move instruction and a consumer are dispatched together.

See *Section 19 Performance Profile* on page 245 for details.

### 3.2.3.3 Load Store Queue Virtual Entry Allocation

The dispatcher allocates load and store queue entries in instruction order. The allocation space at dispatch is twice that of the physical queue slots available. Instructions remain in the issue queues until the actual load or store queue entry is available.

### 3.2.4 Instruction Completion

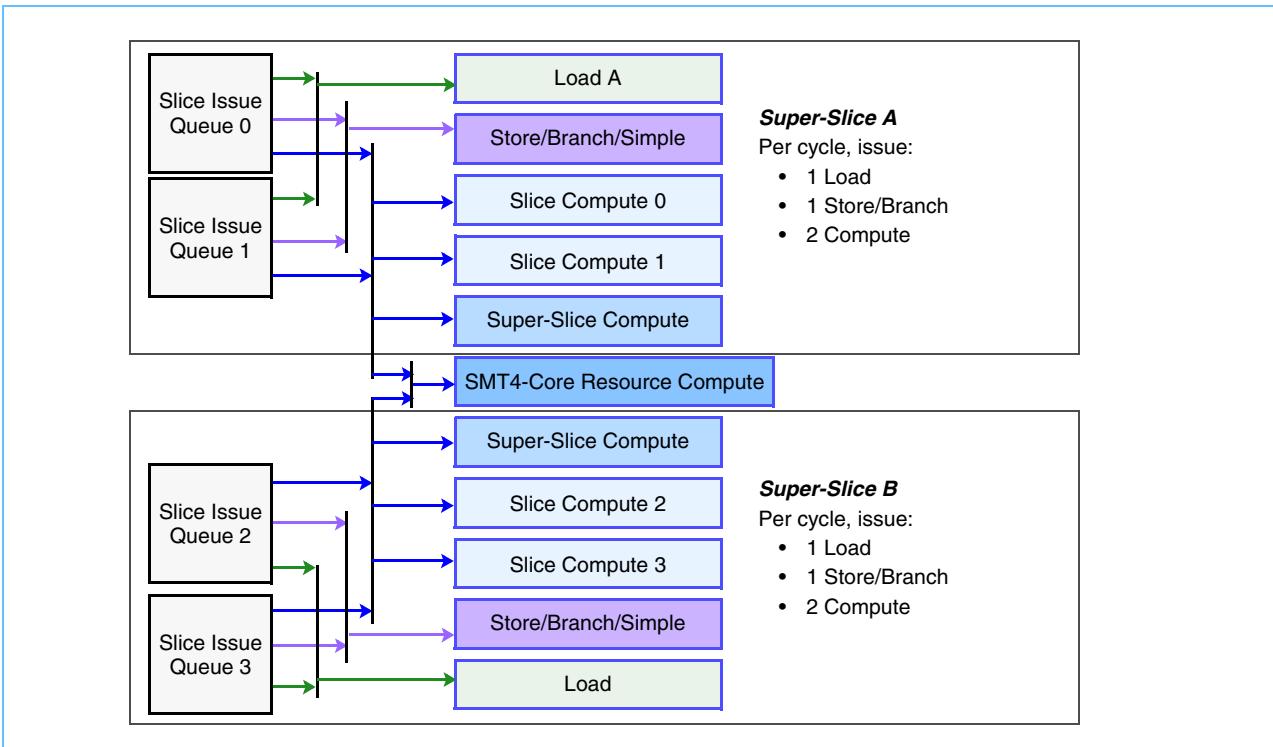
Dispatched instructions are tracked in the Instruction Completion Table (ICT) until each operation has finished execution. The ICT holds up to 512 operations per SMT4-core-resource. Operations can be marked as finished either at dispatch or as they are executed in the pipelines. Operations are completed up to 64 instructions per cycle and on the granularity of two entries, called an “instruction-pair”.

### 3.2.5 Issues Queues

Each slice has an issue queue with 20 entries. Per SMT4-core-resource, there are four issue queues supporting a total of up to 80 operations awaiting dependency resolution.

As depicted in *Figure 3-4*, each issue queue is associated with a native computation slice and paired with a second slice to form a super-slice. The super-slice contains an additional set of computation pipelines as well as a load port and a store/branch port. Each cycle, the issue queue in each slice can issue three instructions: a computational operation, a load, and a store/branch/simple operation. Each super-slice selects a single load and a single store/branch/simple operation for execution.

*Figure 3-4. Power10 Core Issue Queues*



### 3.2.6 Compute Pipelines

The Power10 core includes compute pipelines in two groups:

- Group-1 pipelines are able to consume load results with a nominal latency of 4 cycles.
- Group-2 pipelines are able to consume load results with a nominal latency of 5 cycles.

Dependent operation forwarding between the two groups requires an additional cycle of latency beyond the nominal pipeline latency. See *Appendix A Instruction Properties* on page 295 for details on specific instructions.

#### 3.2.6.1 Group-1 Computation Pipelines

Per slice compute:

- FXU pipe:
  - Scalar or vector: single-cycle ALU, rotate, CR logical, and trivial float operations
  - Nominal 2-cycle dependent latency
  - Supports 1-cycle latency for same pipeline
  - Supports fused operations including 0-cycle dependent latency
- FXU-2 pipe:
  - Scalar or vector: 2-cycle ALU and rotate operations
  - Nominal 3-cycle dependent latency
  - Supports 2-cycle latency for same pipeline
  - Supports fused operations including 0 cycle dependent latency
- Permute pipe:
  - Vector: permute operations
  - Nominal 3-cycle dependent latency

Per super-slice compute:

- Store data:
  - Scalar or vector: provide store data, up to 128 bits per unit per cycle

#### 3.2.6.2 Group-2 Computation Pipelines

Per slice compute:

- Multiply-add pipe:
  - Scalar or vector: binary floating-point, integer, integer-multiply, complex, floating-point divide operations
  - Nominal 4-cycle dependent fixed-point multiply latency
  - Nominal 5-cycle dependent FMA latency
  - Nominal 6+ cycle dependent latency for other operations

- Crypto pipe:
  - Vector: AES and carryless-multiply operations
  - Nominal 4-cycle dependent latency to other crypto operations and 6 cycles to other pipelines in Group-2

Per super-slice compute:

- Fixed-divide pipe:
  - Scalar or vector: fixed-point divide and modulo operations
  - Nominal 9+ cycle dependent latency
  - Scalar: binary-coded-decimal assist operations
  - Nominal 4-cycle dependent latency
- Extended BMI pipe:
  - Scalar and vector: subset of bit-manipulation operations
  - Nominal 4-cycle dependent latency
- Decimal fixed-point pipe:
  - Scalar (using vector registers): 128-bit fixed-point binary and decimal operations
  - Nominal 4-cycle dependent latency
- MMA pipe:
  - Vector input, matrix operations: see *Section 3.2.6.3 Dense Math Engine*
  - Nominal 4-cycle dependent latency

Per SMT4-core-resource compute:

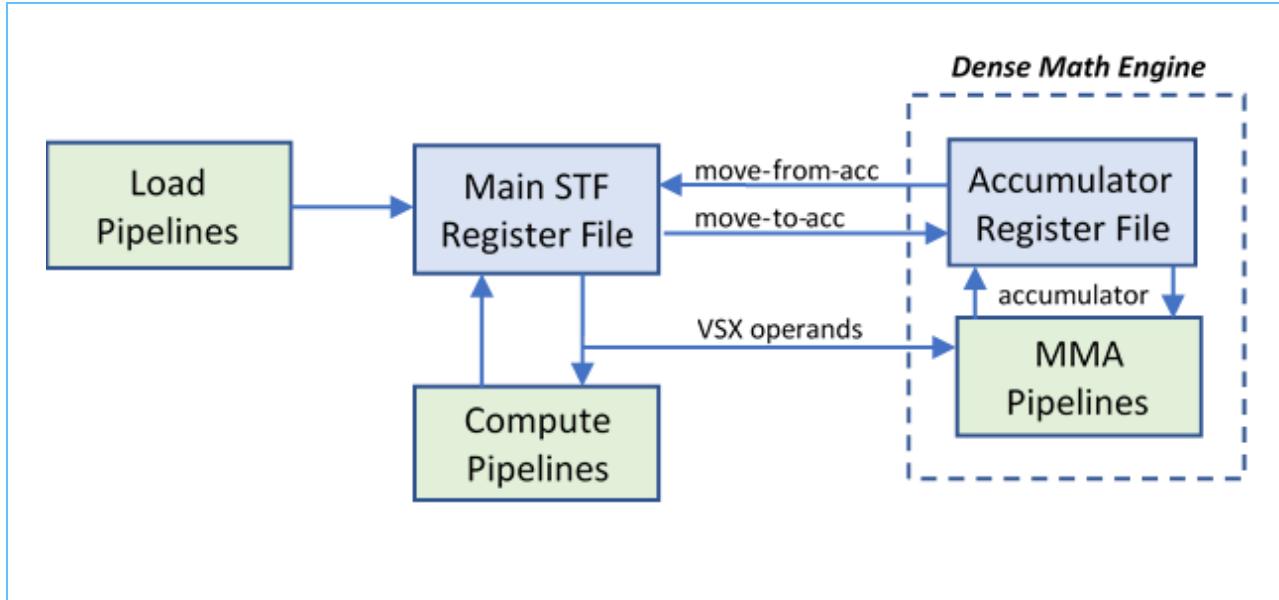
- QFP/DFU pipe:
  - Scalar or vector: quad-precision floating point operations, a subset of fixed-point operations, decimal floating-point operations, simon-cipher operations, and wide-integer divide operations
  - Nominal 13+ cycle dependent latency

### **3.2.6.3 Dense Math Engine**

To efficiently implement MMA operations, such as matrix-multiply-assist instructions, the core implements a dense math engine (DME) microarchitecture. The DME encapsulates compute efficient pipelines, a physical register file, and associated data-flow that keep resulting accumulator data local to the compute units. Each MMA pipeline performs outer-product matrix operations, reading from and writing back a 512-bit accumulator register. Each MMA operation also receives VSX SIMD operand inputs from the main STF register file.

The Power10 core implements the MMA accumulator architecture without adding an additional architected state. Each architected 512-bit accumulator register is backed by four 128-bit VSX registers. Physically, the accumulators are implemented independently from the VSX registers. Software uses the implicit accumulator value, move-to-accumulator, and move-from-accumulator instructions to load and unload the accumulators. Once an accumulator register is loaded, and until it has been unloaded, software should not reference the associated VSX registers.

Figure 3-5. Dense Math Engine



#### 3.2.6.4 Store Address Generation, Branch, and Simple Pipelines

##### Branch Pipeline

Branches are issued from an issue port shared with both store address generation and simple addition operations. Each SMT4-core-resource can resolve one branch per cycle.

Branches are issued after the branch target-address source register (LNK, CNT, or TAR), if any, is ready; even when a condition register (CR) source is still awaiting resolution. These partially executed branches awaiting CR results are held in the branch condition queues (BCQ). This enables target register dependent branches to resolve target register dependencies and extends the effective capacity of the main issue queues.

Move-to and move-from operations between the target registers and GPRs are optimized for latency. The nominal latency of these operations has been reduced by sharing the physical register file between the LNK, CNT, TAR, and the GPRs. Further optimizations include dependency bypass at dispatch to completely eliminate the dependent latency between a target producing instruction and the consuming branch in some scenarios. See *Section 19 Performance Profile* on page 245 for more details.

##### Simple Pipeline

Add immediate instructions, such as those used for address manipulation, are supported on either the main ALU pipelines or share the simple pipeline used for some branch instructions by issuing to the store/branch/simple issue port. These operations can use either of the two simple ports per SMT4-core-resource to produce a result with a nominal 2-cycle latency. A dynamic policy steers the add immediate instructions to the simple pipelines or the main ALU pipelines.

### 3.2.7 Load/Store Pipelines and Queues

Load and store instructions are issued and released from the slice issue queues once operand dependencies are met and once the assigned entry is available in the load or store queue respectively. Store instructions issue with the address generation first and subsequently issue store data.

Load and store pipeline hazards that require re-entry into the pipeline are handled locally by the load and store pipelines and queues. A load hazard such as a read-write cache bank conflict can be accommodated by a single cycle pipeline delay. Other hazards that require pipeline re-entry are managed by the load and store queues and are fully pipelined with operations from the main issue queues.

### 3.2.8 TIQ/MMU Pipeline

All load and store cache misses, as well as instruction cache misses per SMT4-core-resource, are arbitrated and resolved through a single miss handling pipeline; the translation issue queue (TIQ). The TIQ ensures that cache misses have a valid address translation and then that the miss is processed successfully.

The TIQ pipeline first checks the 64-entry effective-to-real address translation (ERAT) for a hit. When the ERAT access is a miss, the L1 translation lookaside buffer (TLB) is accessed in a pipelined fashion with a nominal net latency of 8.5 cycles impact on load-to-use.

#### 3.2.8.1 Load Cache Miss

Once address translation is resolved, load cache misses are tracked by the load miss queues (LMQ). The LMQ holds up to 12 outstanding cache line misses. An arbitrary number of load misses may map to a given LMQ entry as long as they are for the same thread.

## 3.3 Data Handling

The 32 KB L1 data cache uses an effective address tagged directory to process load and store instructions. Load misses from each L1 data cache are serviced at up to 64 bytes per cycle from the L2 cache and up to 32 bytes per cycle from the L3 cache. The L1 data cache is store-through. Stores that hit the cache write back at up to 32 bytes per cycle; while stores that miss the cache only temporarily allocate a cache entry during execution and write their data back to the L2 cache.

### 3.3.1 L1 Load Data Cache Access

The L1 data cache can return two results per cycle at a size of up to 32 bytes for each result. Each of the two load pipelines has access to two 16-byte write-back ports that provide data to the register files and compute pipelines. One of the two 16-byte ports is shared and arbitrated with write backs from the branch/simple execution pipelines. When a single load's data is less than or equal to 16 bytes, it is returned using a single port; whereas, loads that write either 32 bytes or a pair of target registers based on instruction or fusion type use both 16-byte ports. Each load queue entry tracks up to a 32-byte load result.

#### 3.3.1.1 Unaligned Loads

A load is considered unaligned if it either crosses a 128-byte cache line, or if it is a 32-byte load that is not word (4-byte) aligned. An unaligned load makes two passes through the load pipeline to return the data associated with each cache line or a portion of the data granule on either side of the alignment boundary.



### 3.3.1.2 Load Fusion

Certain load sequences support fusion. See *Section 19 Performance Profile* on page 245 for details.

### 3.3.2 Store Datapath

Stores drain up to 32 bytes per cycle, up to four store instructions, into the L2 cache and conditionally into the L1 data cache per cycle after they are completed. The store drain pipeline supports store queue entry merging enabling up to two consecutive store queue entries to drain per cycle. Each store queue entry can hold up to two store instructions if they are fused. Subsequent store-gathering within a cache line takes place prior to writing the L2 cache.

Store data is provided up to 16 bytes per data issue and written into the entries at a rate of up to 32 bytes per cycle in the store queue. Each store queue entry holds up to 16 bytes of store data; a 32-byte store uses two entries.

#### 3.3.2.1 Unaligned Stores

A store is considered unaligned if it crosses a 128-byte cache line. An unaligned store makes two passes through the store address generation pipeline.

#### 3.3.2.2 Store Fusion

Certain store sequences support fusion. See *Section 19 Performance Profile* on page 245 for details.

#### 3.3.2.3 Store Forwarding

Load pipeline operations that are dependent on older stores that are still resident in the store queue can forward from up to one store entry with no change to the load-to-use latency.

### 3.3.3 Data Prefetch

Cache misses are snooped by the 16-entry prefetch queues (PRQ). Each PRQ entry can manage an independent prefetch stream generated by either hardware or software. Prefetches are generated for either the L3 cache, to hide memory latency, or for the L1 cache to hide L2 and L3 cache latencies.

Full software prefetch capabilities are supported, including both those specified in the *Power ISA (Version 3.1B)*, as well as an L3 block prefetch capability supporting a subset of the architecture specified in the *Power10 Processor Programming Model Bulletin*. Block prefetch enables software to specify blocks of memory up to 128 KiB to fetch into the L3 cache.

The Power10 core uses an adaptive prefetch mechanism, which employs L3 cache and nest feedback and long-term averaging to automatically reduce prefetch aggressiveness and increase performance in areas where prefetch consumption or memory bandwidth is low.

The aggressiveness of prefetch requests is automatically and dynamically controlled based on policies that optimize for bandwidth utilization and prefetch effectiveness together with specifications in the DSCR for urgency (URG). These policies can vary by machine type; so programs that are able to set the DSCR optimally might benefit significantly from an improved performance; mainly by indicating the prefetch friendliness



---

of the program via the DSCR(URG) field. Additional benefits can also be obtained by setting other fields such as store-stream prefetch enable (SSE), prefetch depth (DPFD) or others. See *Section 19.1.7.13 Data Prefetch* on page 287 for details.

## 3.4 Memory Management

Address translation is preserved within the ERAT, TLB, SLB, and L1 instruction and data caches. Address translation invalidations are processed across each of these structures and invalidate impacted entries.

### 3.4.1 TLB

The TLB has a parent-child structure, holding up to 4096 mappings from virtual-to-real pages in the child table as long as each entry has a valid parent table entry. Nested radix translations only use a single entry. The parent table supports up to 512 1 GB regions. For radix translation, the child table also supports L3 and L2 page-walk cache (PWC) entries and the parent table also supports L1 PWC entries.

When a TLB miss occurs, up to four page-walk requests can be processed concurrently.

Once address translation is resolved, cache hits are processed when operations are re-issued into the load/store pipeline.

### 3.4.2 Data and Instruction Sharing

There are multiple mechanisms that enable the sharing of instructions and data between threads:

- Alias table: enables sharing between threads for the same effective-to-real address mapping.
- Context table: for radix translation, enables sharing of data between different privilege levels and threads with the same process ID as specified in the LPIDR and PIDR Registers.
- Alias prefetcher: enables automatic sequential caches line alias resolution to avoid latency penalties associated with alias identification.

#### 3.4.2.1 Alias Tables

Alias tables enable cache sharing across a similarly mapped region of memory that shares the same effective and real address. There are four alias regions supported for data, each supporting up to 64 KB each. There are eight alias regions supported for instructions, each supporting either 4 KB or 64 KB each. Alias regions are assigned dynamically and enable sharing active cache lines within the address region. Alias table based sharing in the instruction cache applies to HPT translated addresses only.

#### 3.4.2.2 Context Tables

Context tables enable cache sharing between threads with the same radix translation. There are eight context table entries assigned dynamically that are shared for both instructions and data. Each context table entry supports an arbitrary amount of sharing; for example, multiple pages.



## 3.5 Security

### 3.5.1 Data and Speculation Barriers

The Power10 core provides enhanced handling of automatic thread isolation from speculation-based attacks. This reduces the scenarios under which supervisor levels of software must assist with isolation measures and reduces the cost of these measures when using the preferred mechanisms.

### 3.5.2 Speculation Controls

The Dynamic Execution Control Registers ([H]DEXCR) provide a per thread mechanism for software to control various features that have security or performance trade-offs and can assist software in managing speculation.

### 3.5.3 Return Oriented Programming Protection

Two units per SMT4-core-resource support the new Simon Cipher-based ROP mitigation instructions, hash check, and hash store, for protecting the integrity of the return stack, as well as for other uses. These instructions operate under the control of the [H]DEXCR Register as specified in the *Power ISA (Version 3.1B)*. See *Section 5.6.5.4 Return Oriented Programming (ROP) Support* on page 98.

## 3.6 Power Management

### 3.6.1 Thread Priority

To enable software management of a thread's relative rate of execution based on low-priority states and events, the effective thread-priority encodings specified by PPR, PRR of 0b001 very low, and 0b010 low are enforced independent of the state of other threads on the processor, resulting in a respectively significant or moderate slow down in the instruction decode rate. See *Section 4.4.4.4 Reduced Priority Modes* on page 70 for additional details.

### 3.6.2 MMA Power Down

The MMA pipeline and associated circuitry is automatically powered off when unused; to save power and assist with frequency boosting. Once powered off, MMA instructions might incur a significant latency delay when the unit is used again; on the order of up to 10 µs. The Power10 core supports the MMA power-on hint instruction **xxmfacc AS = 0** as a nonblocking indication that the MMA should be kept active. Programs or operating systems supporting programs that are sensitive to the power-on latency of the MMA should issue these hint instructions periodically.



## 4. Thread and LPAR Management

This chapter summarizes the thread modes and thread interaction controls provided by the Power10 processor core variants including SMT and LPAR controls.

### 4.1 SMT Overview

Multiple SMT modes are supported by each Power10 processor core and are dynamically managed by the hardware based on the set of active threads. Each Power10 SMT8 processor core supports up to eight threads, while each SMT4 processor core supports up to four threads. Resources of each core are shared and divided based on the number of active threads, each division of resources is called an SMT mode. The SMT modes are managed by hardware independently for each SMT4-core-resource.

With SMT8 cores, SMT4-core-resource\_0 supports the even logical threads (0, 2, 4, 6) and SMT4-core-resource\_1 supports the odd logical threads (1, 3, 5, 7). All external interrupt lines and internal interrupts, such as decrementer, hypervisor, and door-bell interrupts, are steered to the correct resources to wake up the correct logical threads.

#### 4.1.1 SMT Modes

Within an SMT4-core-resource, the following SMT modes are supported:

- Single-thread (ST) mode: one thread active.
- SMT2 mode: two threads active.
- SMT4 mode: three or four threads active.

When the number of active threads per SMT4-core-resource moves between SMT2 and SMT4 modes, an SMT mode change procedure is executed by the hardware to re-balance resources. Threads become active via enabled interrupts and are de-activated by the stop instruction.

Figure 4-1. SMT4 Core in ST and SMT2 Modes

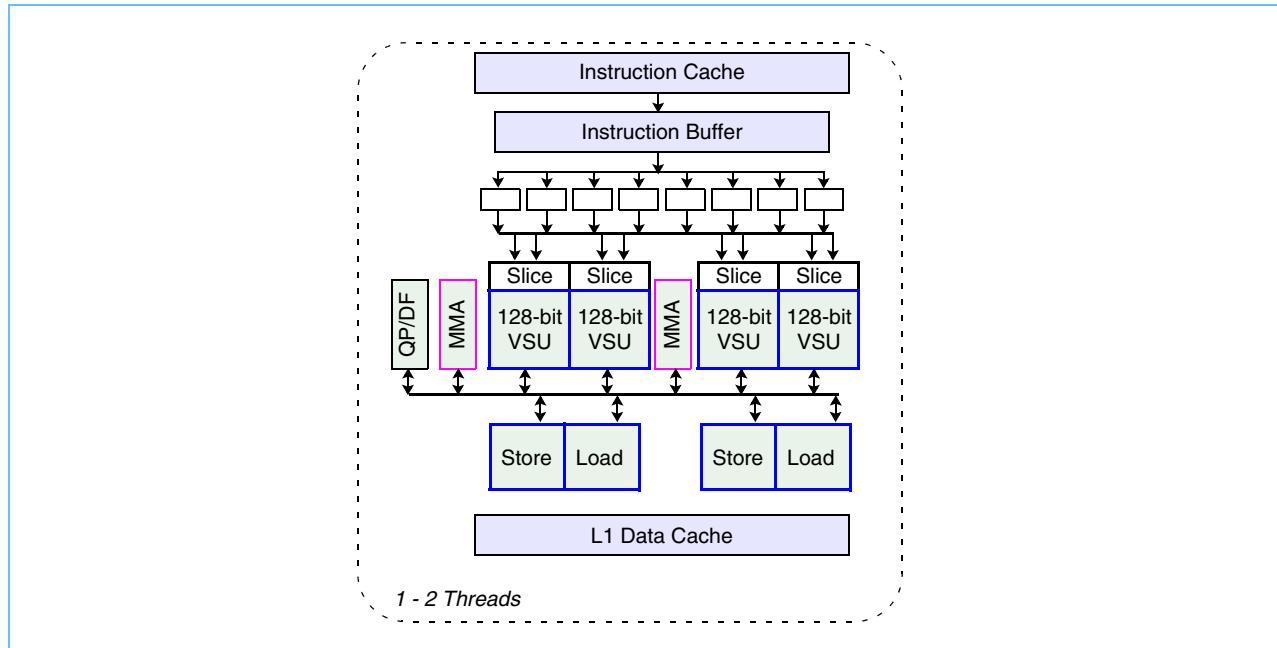


Figure 4-2. SMT4 Core in SMT4 Mode

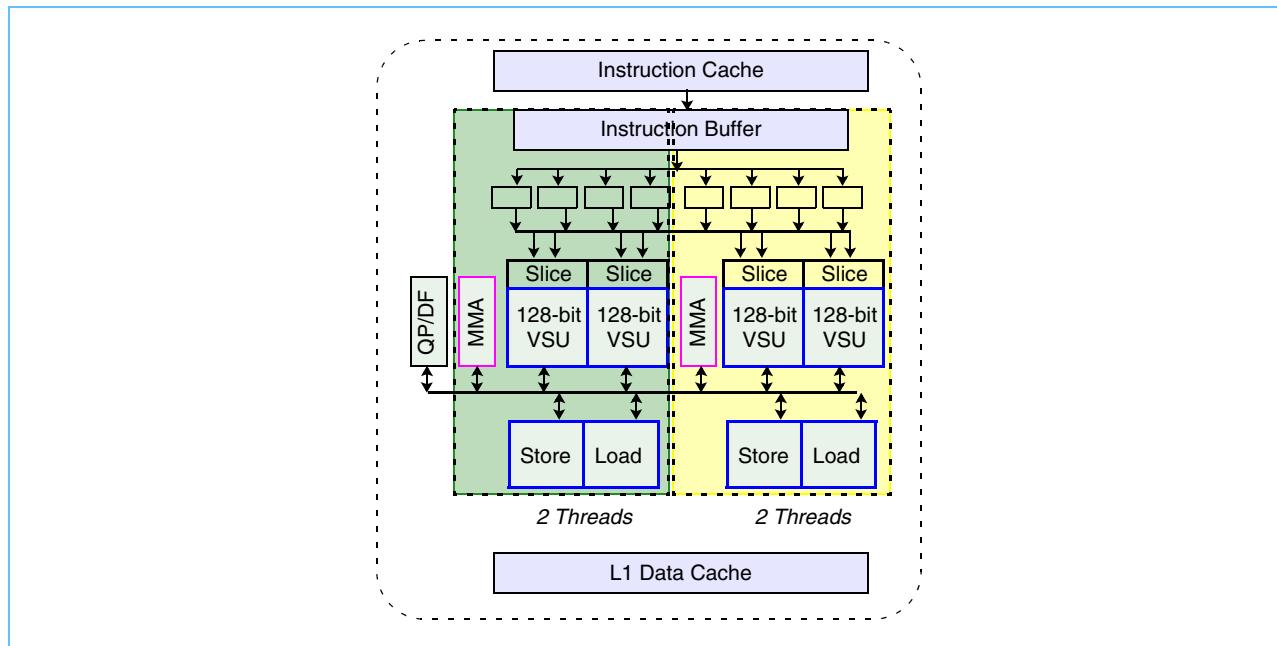


Figure 4-3. SMT8 Core with Each SMT4-Core-Resource in ST or SMT2 Modes

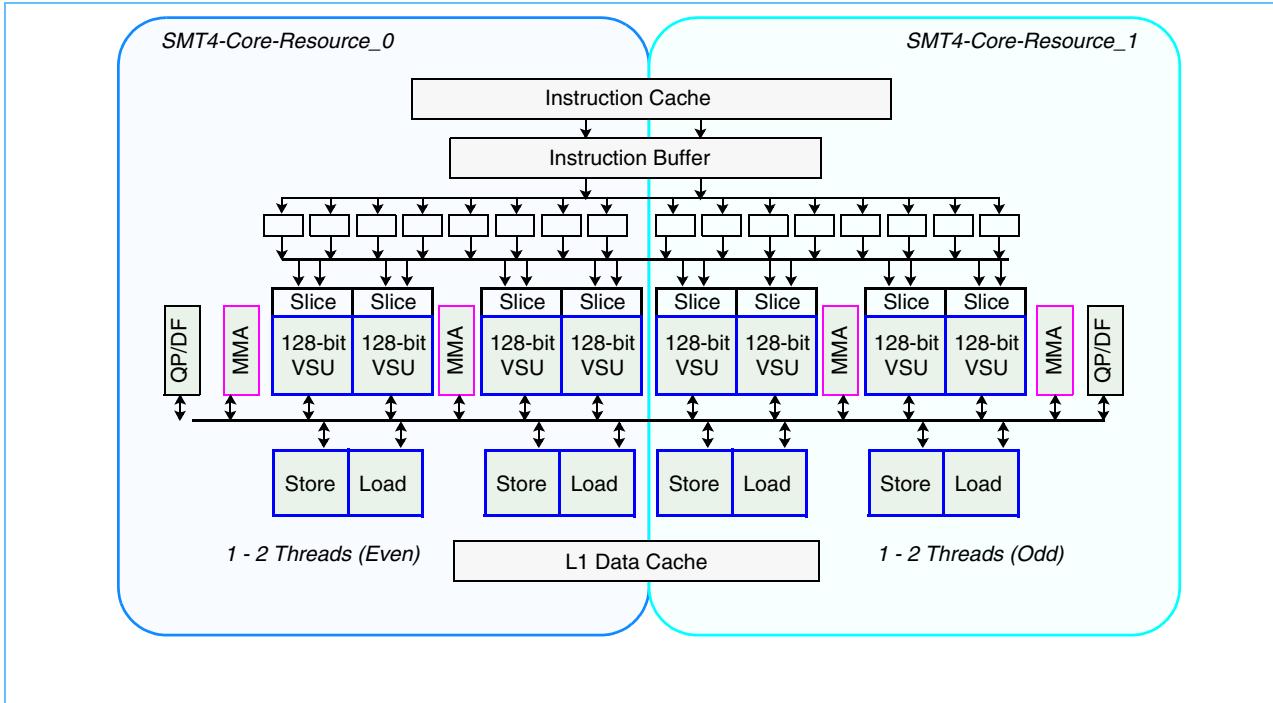
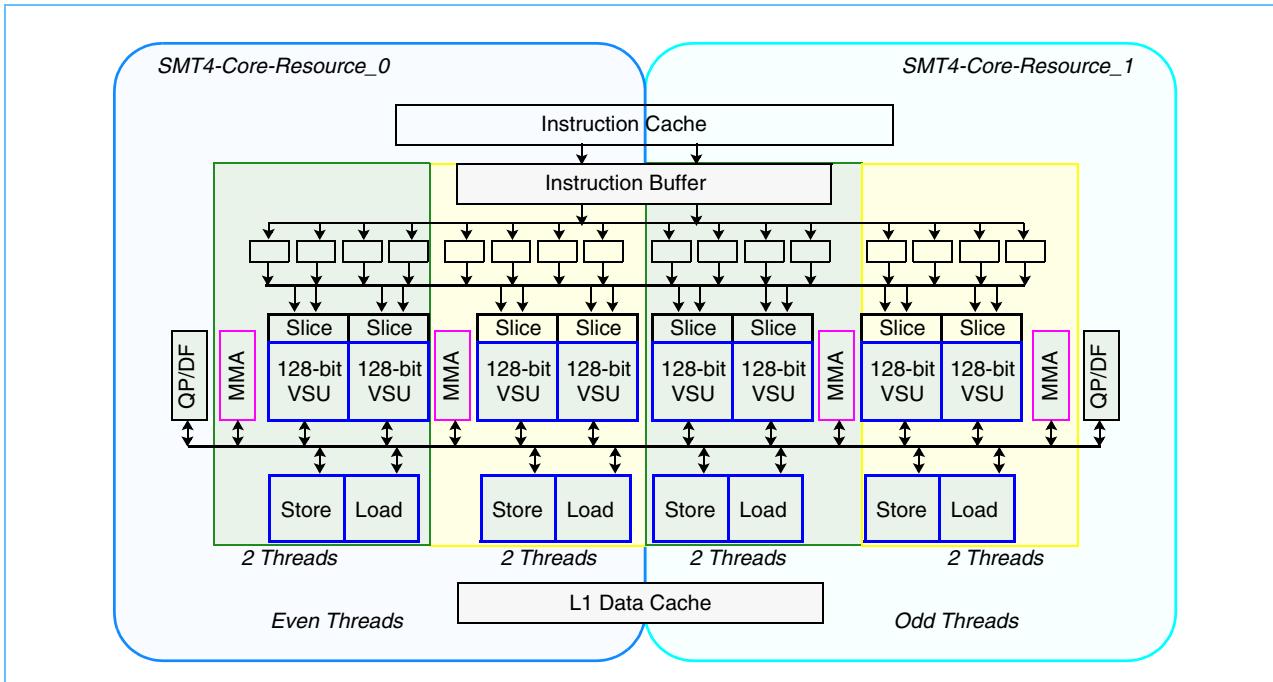


Figure 4-4. SMT8 Core Shown in SMT8 Mode with Each SMT4-Core-Resource in SMT4 Mode





#### 4.1.2 SMT Resource Partitioning

As loosely depicted in *Figure 4-1*, *Figure 4-2*, *Figure 4-3*, and *Figure 4-4*, a subset of the core resources are dynamically partitioned between threads based on the SMT mode. A number of resources are shared between pairs of active threads (thread-pair). A summary of thread partitioning follows:

- Fetch is toggled between active threads while honoring dynamic thread priority.
- In SMT4 mode, decode and dispatch are split to support up to four instructions per thread-pair per cycle. Each four-instruction pipe flows independently from the others except in the case of the microcode engine, which is shared within an SMT4-core-resource.
- In SMT4 mode, issue queues, execution units, load and store execution units are divided by thread-pair, with each thread-pair using a single super-slice.
- The load miss queue and L1 caches are shared dynamically by threads within an SMT4-core-resource.
- Prefetch queues are shared dynamically for hardware detected streams and are statically partitioned based on the active thread count for software-initiated streams.

### 4.2 Logical Partition Specific Resources

As summarized in *Section 3 Power10 Processor Core* on page 39, SMT8 cores can operate with one partition per core, herein termed “1-LPAR” mode. SMT8 and SMT4 cores can operate with one partition per thread, herein termed “Thread-LPAR” mode.

The SMT8 cores can be reconfigured from 1-LPAR mode to Thread-LPAR mode and vice-versa via scan re-initialization of the core performed after a stop level 11 deep power loss state. The new configuration is available after the core is powered back on. The Thread-LPAR mode for the SMT8 core supports the Enterprise Protected Container (EPC) capability.

The LPAR-specific SPRs are: TB, HDEC, PCR, AMOR, VTB, and LDBAR. In Thread-LPAR mode, these SPRs can only be accessed by the logical thread they are assigned to.

In 1-LPAR mode, the TB and HDEC are set based on a **mtspr** from any of the active core threads. In 1-LPAR mode, the AMOR, PCR, and LDBAR are implemented per SMT4-core-resource and require the first active thread on that resource to set the copy for that resource; for example, first even and first odd thread.

#### 4.2.1 Time Base Register

##### 4.2.1.1 SMT8 Core Time Base Register Initialization

There are two TB/TOD state machines that must be synchronized for an SMT8 core. The TOD\_SYNC SCOM (X‘84A3’) must be initiated on SMT4-core-resource\_0.

##### 4.2.1.2 PURR Register

The charge rate for PURRs is calculated and distributed with an SMT4-core-resource. In 1-LPAR mode, each SMT4-core-resource receives one-half of the timebase.

#### 4.2.1.3 Region Weighted Mode Register

Table 4-1 describes the bits in the per SMT4-core-resource RWMR Register.

Table 4-1. Region Weighted Mode Register (RWMR)

Bit	Field	Description
0	DK	Dispatch kicker.
1:5	IW1	Idle weight, ST, 1 thread running on both super slices.
6:10	IW2	Idle weight, SMT2, 1 thread running on both super slices.
11:15	IW3	Idle weight, SMT2, 2 threads running on both super slices.
16:20	IW4	Idle weight, SMT4, 1 thread running on both super slices.
21:25	IW5	Idle weight, SMT4, 2 threads running on one super slice.
26:30	IW6	Idle weight, SMT4, 2 threads running, one on each super slice.
31:35	IW7	Idle weight, SMT4, 3 threads running, two on one super slice and one on the other super slice.
36:63	Reserved	Reserved.

### 4.3 Core-Specific Resources

The RWMR, CTRL, DPDES, TFMR, HMEER, PMCR, and PMSR SPRs are defined per core and a write from any thread will set the SPR value for either an SMT8 core or SMT4 core.

The HID, PTCR, HRMOR, MMCRC, TSCR, TTR, and RPR are defined per SMT4-core-resource. For an SMT8 core, it is required that the first active thread on each SMT4-core-resource sets the copy for that resource; for example, first-even and first-odd thread.

#### 4.3.1 CTRL Register

CTRL is an architected 64-bit register. The bit assignment for the thread control bits in the CTRL Register supports up to eight threads. Table 4-2 displays the bit assignments of the CTRL Register.

Table 4-2. CTRL Register

Bit	Description
0:47	Reserved.
48:55	Thread run latches (indirectly set by writing to bit 63 and decode of thread-id).
56:62	Reserved.
63	Run latch for thread doing CTRL read/write.

All bits except bit 63 are read only. Bits 48:55 are set indirectly based on a write to bit 63 and the thread that is doing the writing. For example, a write to bit 63 from thread 0 sets the Run Latch for thread 0, which can be read from bit 48.

The data read on an **mfsp**(CTRL) is formatted differently based on MSR[PR] and MSR[HV]. Bit 63 is always the Run Latch of the thread executing the **mfsp**.



CTRL bits 48:55 are formatted as shown in Table 5-2, where R0 equals Run Latch for thread 0, RT equals Run Latch of the thread executing the mfspr. *Table 4-3*, where R0 equals Run Latch for thread 0, RT equals Run Latch of the thread executing the **mfspr**.

*Table 4-3. MFSPR CTRL Data Formatting*

Core Mode	MSR[HV], MSR[PR]	LPAR Mode	Bits 48:55
SMT8 core	*1-Problem	Core	RT,0,0,0,0,0,0
SMT8 core	*0 - Hypervisor/Privileged	Core	R0,R1,R2,R3,R4,R5,R6.R7
SMT8 core	0/*1 - Privileged/Problem	Thread	RT,0,0,0,0,0,0
SMT8 core	10 - Hypervisor	Thread	R0,R1,R2,R3,R4,R5,R6.R7
SMT4 core	0/*1 - Privileged/Problem	Thread	RT,0,0,0,0,0,0
SMT4 core	10 - Hypervisor	Thread	R0,R1,R2,R3,0,0,0,0

#### 4.3.2 SPRC/SPRD

SPRC (d'276') and SPRD (d'277') are used as a pair to provide a programmable interface into micro-architected SPRs in the pervasive design using indirect addressing. The address of the SPR to access is written into SPRC. A following read or write to SPRD will read or write the SPR addressed by SPRC. The SPRC and SPRD can both be accessed via an **mtspr** or **mfspr** instruction or SCOM. An SPRC/SPRD pair is dedicated per thread for SPR access and per SMT4-core-resource for SCOM access. Only 8 bits of the SPRC register [54:60] are implemented, where bits [61:63] are reserved for future use and should be set to '000'. SPRD is a full 64-bit register whose meaning changes based on the contents of SPRC. This register is an alias to micro-architected hypervisor resources.

For **mtspr** SPRC access, the thread determines which thread-specific SPRC to set. For SPRD access, each thread-specific SPRC can be used to access the sprs. In 1-LPAR mode, SPRC/SPRD can access all eight logical threads of the core, as shown in *Table 4-4*. In Thread-LPAR mode, SPRC/SPRD can only access a limited set of registers as shown in *Table 4-5* on page 61. The thread-specific SPRC can also be set via SCOM using the SPR\_MODE\_REG.

For SMT8 cores only, PowerVM SCOM accesses to a specific SMT4-core-resource SPRC can be written by bits [54:60] by SCOM SPRC (x'8480'). The SCOM SPRD (x'8481') then uses this SPRC address to select which SPR to access.

For OCC SCOM access, a dedicated SPRC/SPRD only accesses the activity counters and auto-increments as shown in *Table 4-6* on page 61. SCOM (x'8410') is used to read the data addressed by this SPRC (x'8411').

*Table 4-4. SPRC Definition for SMT8 Core Mode (1-LPAR per Core) (Sheet 1 of 2)*

SPRC[54:60]								SPRD Selection	Notes
0	0	0	0	S	S	S	SCRATCH0 - 7		1,3
0	0	0	1	T	T	T	TFMR (Logical Thread)		1,3
0	0	1	0	T	T	T	PURR (Logical Thread)		1,3
0	0	1	1	T	T	T	SPURR (Logical Thread)		1,3

1. Registers can be accessed by both **mt/mfspr** and SCOM access. Only core TFMR bits [0:56] can be accessed via SCOM access.
2. Registers can only be accessed via **mt/mfspr** access.
3. S = scratch register number; T = Thread; for SMT8 core mode.



Table 4-4. SPRC Definition for SMT8 Core Mode (1-LPAR per Core) (Sheet 2 of 2)

SPRC[54:60]								SPRD Selection	Notes
0	1	0	0	T	T	T		DEC (Logical Thread)	1,3
0	1	1	1	0	0	0		SPR_MODEREG SPR	2
0	1	1	1	0	0	1		AVP OUTPUT PIN-	2
0	1	1	1	0	1	0		Core Checkstop	2
0	1	1	1	0	1	1		SPATTN	2
0	1	1	1	1	0	0		Core-thread state	1

1. Registers can be accessed by both **mt/mfspr** and SCOM access. Only core TFMR bits [0:56] can be accessed via SCOM access.  
2. Registers can only be accessed via **mt/mfspr** access.  
3. S = scratch register number; T = Thread; for SMT8 core mode.

Table 4-5. SPRC Definition Thread-LPAR Mode

SPRC[54:60]								SPRD Selection	Notes
0	0	0	0	S	S	S		SCRATCH0 - 7	1, 3
0	1	1	1	0	0	0		SPR_MODEREG SPR	2
0	1	1	1	0	0	1		AVP output pin	2
0	1	1	1	0	1	0		Core checkstop	2
0	1	1	1	0	1	1		SPATTN	2
0	1	1	1	1	0	0		Core-thread state	1

1. Registers can be accessed by both **mt/mfspr** and SCOM access. Only core TFMR bits [0:56] can be accessed via SCOM access.  
2. Registers can only be accessed via **mt/mfspr** access.  
3. S = scratch register number; T = Thread; for SMT8 core mode.

Table 4-6. OCC SPRC Definition

SPRC[54:60]								SPRD Selection	Notes
0	0	0	e	e	e	e		Empath counters. Always do auto-increment	1

1. e = Empath counter number;

Table 4-7. SPRC/SPRD Core Thread State (Sheet 1 of 2)

Bits	Description
0:31	Zeros
32:35	Virtual Thread 0 PSSCR(IL)
36:39	Virtual Thread 1 PSSCR(IL)
40:43	Virtual Thread 2 PSSCR(IL)
44:47	Virtual Thread 3 PSSCR(IL)
48:55	Zeros
56	Virtual Thread 0 quiesced or inactive due to STOP
57	Virtual Thread 1 quiesced or inactive due to STOP



Table 4-7. SPRC/SPRD Core Thread State (Sheet 2 of 2)

Bits	Description
58	Virtual Thread 2 quiesced or inactive due to STOP
59	Virtual Thread 3 quiesced or inactive due to STOP
60:62	Zeros
63	SMT8 core mode

#### 4.3.3 SPR Mode Register

This single SPR mode register (spr\_modereg) is shared by all threads in the core. It contains special modes (and several status bits) that are deemed potentially useful for host firmware for either hardware “bug” work-arounds or other special situations. This register can be accessed with SPR (SPRC/SPRD) or SCOM (x‘8484’).

#### 4.3.4 Recovery and Core Checkstop Handling

Table 4-8. SPR Mode Register Bit Definitions

Bit	Field	Description
0:9	–	Reserved
10:15	Timefac Error Inject	A write to these bits causes an error to be injected into the selected register. These bits automatically clear the next cycle (causing a pulse). 00000 No errors enabled xxx001 TFMR xxx010 HDEC xxx011 TB TTT100 PUR TTT101 SPUR TTT110 DEC <b>Note:</b> Where TTT = thread ID.
16:19	–	Reserved.
20	sprct0_select	Allows SPRC SCOM write of bits 54:60 of T0 SPRC (T0_SPRC).
21	sprct1_select	Allows SPRC SCOM write of bits 54:60 of T1 SPRC (T1_SPRC).
22	sprct2_select	Allows SPRC SCOM write of bits 54:60 of T2 SPRC (T2_SPRC).
23	sprct3_select	Allows SPRC SCOM write of bits 54:60 of T3 SPRC (T3_SPRC).
24	sprct4_select	Allows SPRC SCOM write of bits 54:60 of T4 SPRC (T4_SPRC).
25	sprct5_select	Allows SPRC SCOM write of bits 54:60 of T5 SPRC (T5_SPRC).
26	sprct6_select	Allows SPRC SCOM write of bits 54:60 of T6 SPRC (T6_SPRC).
27	sprct7_select	Allows SPRC SCOM write of bits 54:60 of T7 SPRC (T7_SPRC).
28:63	–	Reserved.

There is a FIR per SMT4-core-resource to isolate faults.

When an error recovery is initiated, threads on the assigned SMT4-core-resource receive a hardware maintenance interrupt (HMI).

For an SMT8 core, FIR[56] is set in the other SMT4-core-resource and can be configured to initiate recovery actions. Both FIR registers should be read to get the complete core status.

#### 4.3.5 Memory Accumulation

Events recorded to memory through continuous monitoring are written to a location specified in the LDBAR. The LDBAR must be set per SMT4-core-resource.

### 4.4 Thread Priority Management

Various mechanisms are in place to allow multiple threads to achieve optimal priority to boost efficiency and meet quality of service requirements. The Power10 core priority mechanisms include both hardware based dynamic priority with a subset of features controllable by the software, as well as support for software specified priority. Software specified priority is used to determine the relative priority of threads and to enter reduced resource consumption low-priority modes of operation for a given thread.

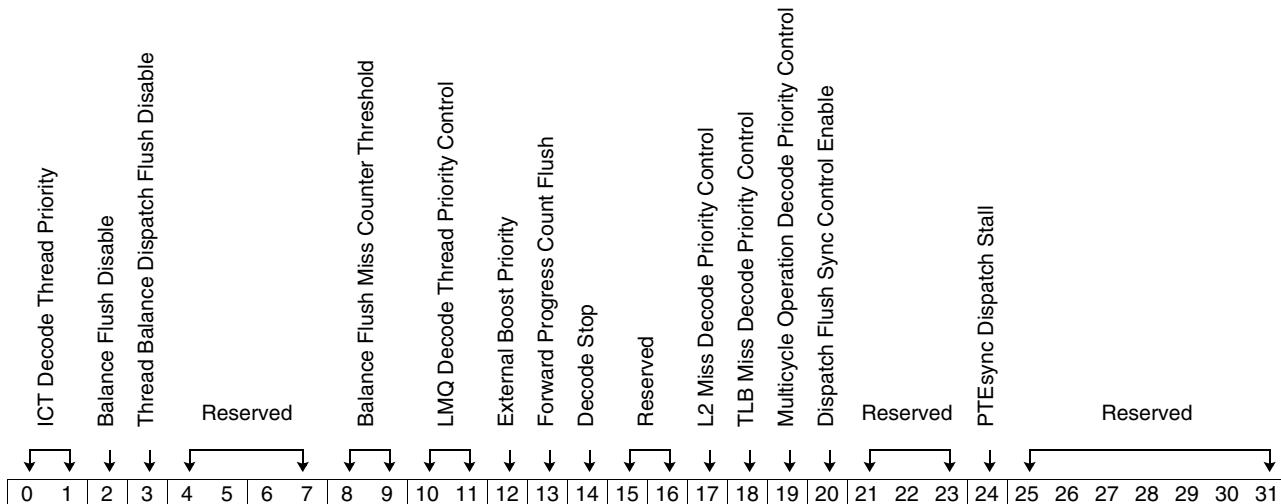
#### 4.4.1 Thread Switch Control Register (TSCR)

Thread priority, control, and status registers enable software to do the following:

- Give a large percentage of execution resources to critical tasks.
- Reduce the amount of resources and power used by low-priority work.
- Read foreground and background thread priority and status.
- Save and restore priority during interrupts.
- Provide a controlled way to enable supervisor/user code to change priority.

Thread priority controls are programmable. All bits are read/write. There is one Thread Switch Control Register (TSCR) per core. The TSCR can be accessed with the **mtspr** and **mfsp** instructions using SPR 921. An SCOM latch is added to make this register read-only for debug purposes.

TSCR is initialized to x'0000\_0000' at power-on and software should set the preferred value before program execution. The recommended setting is: x'8028\_7880'.



Bits	Field Name	Description
0:1	ICT Decode Thread Priority	ICT Decode Thread Priority. If all threads are at the same software-thread priority, decrease the priority when a thread uses more than the following number of ICT entries: 00 Function disabled. 01 SMT2: 80; SMT4: 40. 10 SMT2: 96; SMT4: 48. 11 SMT2: 112; SMT4: 56.
2	Balance Flush Disable	Balance Flush Disable. 0 Enable NTCP1 balance flushes. 1 Disable NTCP1 balance flushes.
3	Thread Balance Dispatch Flush Disable	Thread Balance Dispatch Flush Disable. 0 Enable dispatch flush for the thread that was chosen for a balance flush if that thread is stalled at dispatch. A dispatch flush is a lower-latency flush than a balance flush. 1 Disable. <b>Note:</b> Conditions for dispatch flush are the same as a balance flush.
4:7	Reserved	Reserved.

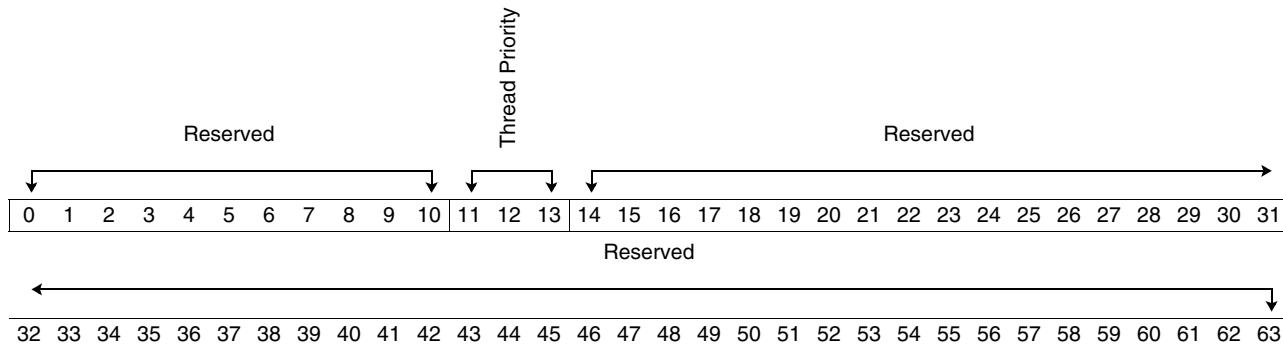


Bits	Field Name	Description
8:9	Balance Flush Miss Counter Threshold	<p>Balance Flush Miss Counter Threshold. If a balanced flush occurs, apply a CLB hold until the counter threshold is released or the miss is resolved.</p> <p>00 Function is disabled, CLB hold is applied until the miss is resolved. 01 Scan-only latch, value is programmable, 10-bit LFSR. POR default is: 256 cycles (x'3C1' LFSR). 10 Scan-only latch, value is programmable, 10-bit LFSR. POR default is: 384 cycles (x'0E4' LFSR). 11 Scan-only latch value programmable, 10-bit LFSR. POR default is: 512 cycles (x'20F' LFSR).</p>
10:11	LMQ Decode Thread Priority Control	<p>LMQ Decode Priority Control. When the threads have the same software-set decode priority, decrease the thread's priority if a thread has more misses outstanding than described as follows:</p> <p>00 Function disabled. 01 SMT2: 4; SMT4: 2. 10 SMT2: 5; SMT4: 3. 11 SMT2: 6; SMT4: 4.</p>
12	External Boost Priority	<p>External Boost Priority. If '1' and an external interrupt request is active and the corresponding threads' priority is less than normal priority, set the threads' priority to normal.</p> <p><b>Note:</b> This does not change the value in PPR[11:13] for the affected thread.</p>
13	Enable Forward Progress Count Flush	<p>Enable Forward Progress Count Flush. <b>Note:</b> This bit only enables/disables the flush from occurring. The forward progress timer does not stop decrementing when set to '0'. SMT2 and higher: If one thread is not making progress, enable flushing the other active threads.</p>
14	Decode Stop	<p>Decode Stop. When set to '0', the forward progress timer (in PPR) is decremented even when the current thread is in decode stop state. When set to '1', the forward progress timer is not decremented when the current thread is in decode stop state.</p>
15:16	Reserved	Reserved.
17	L2 Miss Decode Priority Control	<p>L2 Miss Decode Priority Control. If all threads are at the same software set priority, then:</p> <p>0 L2 miss is disabled for use in adjusting decode priority. 1 L2 miss is enabled for use in adjusting decode priority.</p>
18	TLB Miss Decode Priority Control	<p>TLB Miss Decode Priority Control. If all threads are at the same software set priority, then:</p> <p>0 TLB miss is disabled for use in adjusting decode priority. 1 TLB miss is enabled for use in adjusting decode priority.</p>
19	Multicycle Operation Decode Priority Control	<p>If all threads are at the same software-set priority, then:</p> <p>0 Multicycle operations are disabled for use in adjusting decode priority. 1 Multicycle operations are enabled for use in adjusting decode priority.</p>
20	Dispatch Flush Sync Control Enable	<p>Dispatch Flush Sync Control Enable. Stop decode if the mode for the thread with <b>sync</b> instruction is outstanding, (always on in shipping mode). Applies only to SMT2 and higher.</p>
21:23	Reserved	Reserved.
24	PTEsync Dispatch Stall	<p>Ptesync dispatch stall. Set to '1' to enable the following function for <b>ptesync</b> instruction. Hold the <b>ptesync</b> instruction at dispatch until LMQ is empty, <b>SRQ</b> is empty, no instruction-side (l-side) table walk pending, and wait for a minimum of 15 cycles. After these conditions are satisfied, dispatch the <b>ptesync</b>. Afterwards, wait for LMQ empty and SRQ empty to continue with the dispatch of further instructions.</p>
25:31	Reserved	Reserved.

#### 4.4.2 Program Priority Register (PPR)

Each thread has a 64-bit status register associated with it. Some bits are read-only, while other bits are read/write. There is one PPR per thread.

The local PPR can be accessed with the **mtspr** or **mfsp** instructions using SPR 896. The PPR for each thread is initialized to x'0010\_0000\_0000\_0000' at power-on.



Bits	Fields Name	Description
0:10	Reserved	Reserved (not implemented).
11:13	Thread Priority	Thread Priority. 000 Not allowed 001 Very low 010 Low 011 Medium low 100 Normal 101 Medium high 110 High 111 Extra high Set to '100' on system reset interrupt.
14:63	Reserved	Reserved (not implemented).

##### 4.4.2.1 Priority NOPs

The thread switch priority can be read or written by software using the **mfsp** and **mtspr** instruction to the Thread Status Register. Thread priority can also be altered by executing special forms of the or x,x,x NOP. The priority is changed upon completion of the operation, provided the function is enabled for the current privilege level. Thread priority can be set as follows:

- On the Power10 core, problem-state programs can set their priority from very-low to medium priority.
- Supervisor programs can set their thread priority from very-low to high priority.
- Hypervisor code can set all levels.

Table 4-9 on page 67 describes how to set the thread priority NOPs.

Table 4-9. Thread Priority NOPs

Priority NOP/mtSPR	PPR[11:13]	Thread Priority	Required Privilege Level to Set Given Thread Priority Value
or 31,31,31 / <b>mtPPR[11:13]</b>	'001'	Very Low	Hypervisor, Supervisor, Problem
or 1,1,1 / <b>mtPPR[11:13]</b>	'010'	Low	Hypervisor, Supervisor, Problem
or 6,6,6 / <b>mtPPR[11:13]</b>	'011'	Medium Low	Hypervisor, Supervisor, Problem
or 2,2,2 / <b>mtPPR[11:13]</b>	'100'	Medium (Normal)	Hypervisor, Supervisor, Problem
or 5,5,5 / <b>mtPPR[11:13]</b>	'101'	Medium High	Hypervisor, Supervisor, Problem <sup>1</sup>
or 3,3,3 / <b>mtPPR[11:13]</b>	'110'	High	Hypervisor, Supervisor
or 7,7,7 / <b>mtPPR[11:13]</b>	'111'	Extra High	Hypervisor

1. See [Section 4.4.3.1 Priority Boosting to Medium-High in User Mode](#).

#### 4.4.3 Thread Priority Boosting

The PPR is updated by software when an **mtPPR** or one of the priority changing NOP instructions is committed. Problem-state programs can change the thread priority value to medium-high ('5'), or a lower priority depending on the contents of the Problem-State Priority Boost Register (PSPBR). The problem-state boosting changes the contents of PPR[11:13].

The thread priority can be boosted internally by the hardware (in a software invisible manner) in certain cases (as described in [Section 4.4.3.2 Thread Priority Boosting on Asynchronous Interrupt](#) on page 68) to medium priority ('4'). The boosting of thread priority for pending asynchronous interrupts does not affect the actual architected thread priority value in the PPR. Therefore, if the software does an **mfPPR** at any time during the asynchronous boosting, it always gets the last priority value that is explicitly set by the software for that thread.

##### 4.4.3.1 Priority Boosting to Medium-High in User Mode

The Power10 core allows a problem-state program that executes on a thread to temporarily change the PPR thread priority value to medium high ('5') by executing an **mtPPR** or priority NOP. The temporary thread priority boost is controlled by a 32-bit privileged Problem-State Priority Boost (PSPB) Register. There is one PSPB per thread, which is set by a move-to PSPB.

A problem state program can set the program priority to medium-high only when the PSPB of the thread contains a nonzero value. The maximum value to which the PSPB can be set must be a power of 2 minus 1. Bits that are not required to represent this maximum value must return '0's when read, regardless of what was written to them.

When the PSPB of the thread is set to a value less than its maximum value but greater than '0', its contents decrease monotonically at the same rate as the SPURR until its contents minus the amount it is to be decreased are '0' or less. The PSPB contents can decrease to less than zero when a problem state program is executing on the thread at a priority of medium high.

When the contents of the PSPB minus the amount it is to be decreased are '0' or less, its contents are replaced by '0'. When the PSPB is set to its maximum value or '0', its contents do not change until it is set to a different value.



Whenever the priority of a thread is medium high and either of the following conditions exist, hardware changes the priority to medium:

- PSPB counts down to '0'
- PSPB = '0' and the privilege state of the thread is changed to problem state (MSR[PR] = '1')

While in problem state at medium-high priority, there can be the potential of the PSPBR reaching '0' at the same time a priority NOP or **mtPPR** is trying to lower the thread priority to a value less than medium. If the attempted write to the PPR occurs in the same cycle, the priority NOP or **mtPPR** will update the PPR with its thread priority instead of allowing the PSPB reset to set the PPR to medium priority.

#### 4.4.3.2 Thread Priority Boosting on Asynchronous Interrupt

If a thread has a priority less than medium ('4'), the priority of the thread is boosted on a pending asynchronous interrupt. This allows the interrupt to be serviced faster for a thread that may be waiting for the interrupt at a low-priority state. TSCR[12] is used to enable or disable priority boosting for any pending asynchronous interrupt. The boosting of thread priority does not affect the actual architected thread priority value in the PPR. Therefore, if the software does a move from PPR (**mfPPR**) at any time during asynchronous boosting, it always gets the last priority value explicitly set by the software for that thread.

Thread priority is boosted internally by the hardware on an asynchronous interrupt based on *Table 4-10*. After the priority is boosted, the hardware continues to treat the thread at medium ('4') priority, until there is a move to PPR (**mtPPR**) or priority NOP instruction that changes the thread priority.

*Table 4-10. Priority Boosting Asynchronous Interrupt*

Interrupt	MSR Bits
Hmaintenance	EE = 1 or HV = 0 or PR = 1
Directed External	(EE = 1 and not (HV = 1 and PR = 0 and HEIC = 1)) or (lpes0 = 0 and (HV = 0 or PR = 1))
Mediated External	EE = 1 and (HV = 0 or PR = 1)
Directed Privileged Doorbell	EE = 1
Directed Hypervisor Doorbell	EE = 1 or HV = 0 or PR = 1
Hypervisor Virtualization	(EE = 1 or HV = 0 or PR = 1) and HVICE = 1
Perfmon	EE = 1
HDEC	(EE = 1 or HV = 0 or PR = 1) and HDICE = 1
DEC	EE = 1
System Reset	Always (ignores TSCR[12])



#### 4.4.4 Dynamic Thread Prioritization

##### 4.4.4.1 Dynamic Fetch Priority

The thread priority is used to apportion fetch cycles. Fetch cycles are apportioned within each SMT4-core-resource independently. For example, if two threads have a priority weighting that is different, the ratio of those two weights determines the relative number of cycles those two threads will be given access to the I-cache. If all threads have equal priority, the threads are accessed in a round-robin manner.

##### 4.4.4.2 Dynamic Decode Priority

The decode and dispatch pipes in SMT2 mode cycle share their resources. In SMT4 mode, they are divided into two parallel groups. Each group has two threads that cycle share that half of their resource. This drives the need for a pair of decode priority engines. In SMT2 mode, one engine controls two threads. In SMT4 mode, both engines run in parallel: one for the threads assigned to thread set 0 and one for the threads assigned to thread set 1. In SMT modes, each thread set operates in its own half of the dispatch group, independent of the other thread set.

In SMT mode, decode cycles are given to a thread based on the following ordered criteria:

1. Thread must be active. For example, no slots are given to a stopped thread, CTRL[12:15].
2. Micro-architected decode hold. See *Section 4.5.2 Decode Hold* on page 70.
3. Instruction availability in the Instruction Buffer. Used only if the priority in PPR[11:13] is equal for all enabled threads.
4. Software-set thread priority, controlled in Thread Status Register (PPR[11:13]). See *Section 4.4.4.3 Software Set Thread Priority* on page 70.
5. Dynamically changing thread decode priority. See *Section 4.4.4.3 Software Set Thread Priority* on page 70.

##### *Relative Decode Priority*

If all enabled threads have the same thread priority value, a dynamic thread priority algorithm based on the state of the eligible threads determines which will get the next decode cycle. This algorithm uses a scoring system built on the resources occupied by each thread, whether the thread has any outstanding L2 or TLB misses, or if there is an active multicycle operation active for a thread. This algorithm is implemented per thread set managing 1 - 2 threads, but always toggles between two threads if in SMT4 mode to ensure fairness when there are three threads active.

##### *Microcode Fairness*

In Thread-LPAR mode, the shared micro-code engine per SMT4-core-resource can cause an LPAR to consume multiple consecutive cycles of decode. To compensate, the Power10 core gives the other LPAR sharing the dispatch bandwidth extra decode cycles to compensate for the loss of decode cycles.



#### 4.4.4.3 Software Set Thread Priority

Software-set priority is used to determine the thread to receive the next decode cycle if at least one of the enabled threads has a different Thread Priority value in PPR[11:13] than the other enabled threads. The priority algorithm divides decode cycles according to the relative values of the thread priority values.

The hypervisor defines the mappings between the software-set priority and the effective priority for each of the seven priority levels. A 64-bit architected SPR, relative priority register (RPR), is provided to set any 6-bit value (0 - 63) for each of the seven priority levels (very low, low, medium low, normal, medium high, high, extra high). Then, PPR[11:13] for each active thread determines which value to read from the RPR.

Each active thread receives a number of decode cycles, relative to the other threads, equal to their priority values. For example, within thread set 0, T0 has a relative priority of 17 (as defined by PPR[11:13] and the RPR), T3 has a relative priority of 6. Within a window of  $17 + 6 = 23$  decode cycles, T0 gets 17 cycles and T3 gets 6 cycles.

#### 4.4.4.4 Reduced Priority Modes

The Power10 core slows the rate of decode for a thread to reduce power and save execution resources whenever any enabled thread has an effective priority of lowest ('010') or low ('001'). For the lowest priority, decode is limited to one of every 128 cycles. For the low priority, decode is limited to one of every 16 cycles.

### 4.5 Thread Pipeline Management

The ability to control the relative flow of instructions in an SMT processor is important for optimal performance. When one thread is not making good progress due to reasons such as an L2 miss, TLB miss, sync, or other long-latency operations, the other threads can be given additional machine resources. The following features are built into the Power10 core for controlling SMT instruction flow.

#### 4.5.1 Dispatch Flush

A dispatch flush is a low-latency flush that flushes the decode pipeline to free it up for the alternate thread to make progress. General guidance on dispatch flush policy is as follows:

1. Dispatch flush occurs only if a thread shares a decode set with another thread.
2. A **ptesync** instruction causes a dispatch flush.
3. A **tlbie** instruction causes a dispatch flush when the ICT is not empty.
4. A dispatch flush is generated if a dispatch scoreboard is set requiring a serialization such as an **mfsp** that encounters an **mtspr** using the same scoreboard which is in flight.
5. A dispatch flush is generated if TSCR[3] is set and a thread is selected for a balanced flush.

#### 4.5.2 Decode Hold

A thread can also be held at decode to promote overall progress when that thread has a long latency event pending. For example, as a response to a dispatch flush and until progress conditions are met, or after a balanced flush and until the originating miss is resolved, or until the counter reaches the threshold value as determined by TSCR[8:9].



#### 4.5.3 Balance Flush

A balance flush is a pipeline flush for a thread encountering a long latency operation to promote overall progress for other threads in the core. A balanced flush occurs only when there is an L3 data or translation miss, and another thread is being held at dispatch due to resource starvation.

Balance flushes can be disabled using TSCR[2]. If the thread that is stalled at dispatch is also the thread that was chosen to be balanced flushed, then a dispatch flush on that thread is also performed if TSCR[3] is set.





## 5. Architecture Compliance

The following sections are intended to be read with their respective companion documents. Throughout these sections, it is assumed that the reader is familiar with the following architecture documents:

- *Power ISA User Instruction Set Architecture - Book I (version 3.1B)*
- *Power ISA Virtual Environment Architecture - Book II (version 3.1B)*
- *Power ISA Operating Environment Architecture - Book III (version 3.1B)*

### 5.1 Book I - User Instruction Set Architecture

This section of the document identifies version 3.1 architectural implications of the Power10 design point as they relate to the User Instruction Set Architecture (UISA). This is accomplished by walking through each of the relevant sections of Book I and highlighting the Power10 solution to the architectural flexibility provided by the *Power ISA (Version 3.1B)*.

#### 5.1.1 Instruction Classifications

The Power10 core implements all Book I instructions specified in the *Power ISA (Version 3.1B)*. Introduced in the Power10 processor is the support for instruction prefixing as defined in the *Power ISA (Version 3.1B)*. Noteworthy restrictions permitted by the architecture, which the Power10 takes advantage of, include taking an alignment interrupt for prefixed instructions which span a 64-byte boundary and taking an (hypervisor) instruction storage interrupt (HISI/ISI) when an attempt is made to fetch a prefixed instruction from a caching-inhibited page.

##### 5.1.1.1 Illegal Instructions

An attempt to execute an illegal instruction as defined in the *Illegal Instructions* appendix in the *Power ISA (Version 3.1B)* results in a hypervisor emulation assistance interrupt.

##### 5.1.1.2 Instructions Supported

The Power10 core supports all of the instructions described in the *Power ISA User Instruction Set Architecture - Book I (version 3.1B)* and the *Power10 Processor Programming Model Bulletin*. Furthermore, it supports the Service Processor “Attention” described in the *Reserved Instructions* appendix of the *Power ISA (Version 3.1B)*. This instruction is conditionally enabled by HID[3] = ‘1’. When enabled, this instruction is a user-level instruction.

##### 5.1.1.3 Invalid Forms

In general, the Power10 core handles *invalid forms* of instructions in the manner that is most convenient for the particular case (within the scope of meeting the boundedly-undefined definition described in the *Power ISA (Version 3.1B)*). This document specifies the behavior for some of these cases. However, it is not recommended that software or other system facilities make use of the Power10 behavior in these cases because such behavior might be different in another processor that implements the Power ISA.

The Power10 core ignores the state of reserved bits in the instructions (denoted by “//” in the instruction definition) and executes the instruction normally. Software should set these bits to ‘0’ per the Power ISA.



## 5.1.2 Branch Processor

### 5.1.2.1 Instruction Fetching

In an effort to increase performance, the Power10 core does instruction prefetching before it determines whether or not particular instructions will actually execute. This prefetching follows all of the architectural constraints relative to instruction alignment as well as caching-inhibited and guarded regions of storage. More information on instruction fetching is available in *Section 19.1.3 Instruction Fetch* on page 248.

### 5.1.2.2 Branch Prediction

The Power10 core uses several dynamic branch prediction mechanisms to improve performance. See *Section 19.1.3.5 Branch Prediction* on page 250. When enabled, when the effective value of [H]DEXCR aspect SBHE is '1' or dynamically set by hardware, software can override the hardware mechanisms for branch prediction by using the architected BO field "a" and "t" hint bits in the instruction itself as described in the *Power ISA (Version 3.1B)*.

In addition, for **bclr** instructions, a link stack (or call-return stack) is used to predict the target address of the branch as long as the effective value of [H]DEXCR aspect SRAPD is '0'. Similarly, for **bcctr** instructions, various predictors are used to predict the target address for this type of branch as long as the effective value of [H]DEXCR aspect IBRTPD is '0'. To improve the efficiency of these address predictors, the Power10 core uses the architected BH-field hints associated with several of the branch instructions.

### 5.1.2.3 Instruction Cache Block Touch Hint

The Power10 core supports the instruction cache block touch (**icbt**) instruction. However, instead of bringing the instructions into the level 1 (L1) cache as described in the Power ISA, it prefetches the instructions into the level 3 (L3) cache. Thus, **icbt** is implemented internally as a data cache block touch for store (**dcbtst**) hint instruction.

### 5.1.2.4 Out-of-Order Execution and Instruction Flushes

The Power10 processor uses out-of-order instruction execution. Instructions can be speculative on a predicted branch direction, or simply speculative beyond an instruction that might cause an interrupt condition. In the event of a misprediction or an interrupt, instructions from the mispredicted path and the results produced by those instructions are discarded, presenting the effect of sequentially executed instructions down the appropriate branch paths and precise exceptions as required by the *Power ISA (Version 3.1B)*. For details and exceptions about the rules for obeying the sequential execution model, see the following sections: *Instruction Execution Order* in the *Power ISA User Instruction Set Architecture - Book I (version 3.1B)*, *Definitions* in the *Power ISA Virtual Environment Architecture - Book II (version 3.1B)*, and sections *Dynamic Execution Control* and *Definitions and Notation* in the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*.

### 5.1.2.5 Branch Processor Instructions with Undefined Results

The results of executing an invalid form of a branch instruction or an instance of a branch instruction for which the architecture specifies that some results are undefined are described as follows. Only results that differ from those specified by the architecture are described in the following list.

- Instructions with reserved fields:  
Bits in reserved fields including the z-bits in the BO field are ignored. The results of executing an instruction in which one or more of these bits are '1' is the same as if the bits were '0'.
- bcctr** and **bcctrl** instructions:  
If BO[2] = '0', the contents of CTR (before any update) are used as the target address. The contents of the CTR are then decremented and written back to the CTR. The updated contents of the CTR are used to determine whether the branch is taken or not.

### 5.1.2.6 Branch History Rolling Buffer (BHRB)

The Power10 core implements the Branch History Rolling Buffer (BHRB) mechanism as defined in the *Power ISA (Version 3.1B)* with one notable exception: in either branch trace or single-step trace mode, in cases where both the source address (that is, the effective address of the branch instruction itself) and the branch target address should be written into the BHRB, the Power10 core will only update a single BHRB entry and that one entry will contain the source effective address of the branch itself.

## 5.1.3 Fixed-Point Processor

### 5.1.3.1 Fixed-Point Exception Register

The *Power ISA (Version 3.1B)* defines the Fixed-Point Exception Register (XER) bits: XER[0:15], XER[35:43] and XER[46:56] as reserved. An **mfker** instruction returns the values as shown in *Table 5-1*.

Table 5-1. XER Bits and Fields (Sheet 1 of 2)

XER Bits	Name	Field	Read/Write Behavior
0:15	Reserved	Unimplemented	Returns zeros on <b>mfker</b> .
16:31	Reserved		Returns zeros on <b>mfker</b> .
32	SO	SO	Set to '1' whenever OV = '1', except when <b>mtker</b> sets SO = '0' and OV = '1'. This bit can be set to '0' or '1' by <b>mtker</b> . An <b>mfker</b> reads the bit contents. This bit is implemented inside the mapper, but it is only updated at completion time.
33	OV	F1	Set to '0' or '1' by various fixed-point instructions with OE = '1' or by <b>mtker</b> . An <b>mfker</b> reads the bit contents.
34	CA	F2	Set to '0' or '1' by add-carrying, subtract-from carrying, shift-right algebraic-type instructions, and by <b>mtker</b> . An <b>mfker</b> reads the bit contents.
35:43	Reserved	F2	Returns zeros on <b>mfker</b> .

Table 5-1. XER Bits and Fields (Sheet 2 of 2)

XER Bits	Name	Field	Read/Write Behavior
44	OV32/Reserved	F2	<p>There are two bits representing bit 44, an OV32 bit and a reserved bit.</p> <ul style="list-style-type: none"> <li>The <b>mfker</b> reads the OV32 bit if PCR[v2.07] = '0' and reads the reserved bit if PCR[v2.07] = '1'; this applies independent of privilege state and is the only PCR effect on bit 44.</li> <li>The <b>mtxer</b> instruction writes both bits.</li> <li>The <b>mcrxrx</b> instruction reads only the OV32 bit because it is an invalid instruction independent of the PR bit when PCR[v2.07] = '1'.</li> <li>The instructions that implicitly set the OV bit also set the OV32 bit as described in the <i>Power ISA (Version 3.1B)</i>, and do not modify the reserved bit.</li> </ul>
45	CA32/Reserved	F1	<p>There are two bits representing bit 45, a CA32 bit and a reserved bit.</p> <ul style="list-style-type: none"> <li>The <b>mfker</b> reads the CA32 bit if PCR[v2.07] = '0' and reads the reserved bit if PCR[v2.07] = '1'; this applies independent of privilege state and is the only PCR effect on bit 45.</li> <li>The <b>mtxer</b> writes both bits.</li> <li>The <b>mcrxrx</b> reads only the CA32 bit because it is an invalid instruction independent of the PR bit when PCR[v2.07] = '1'.</li> <li>The instructions that implicitly set CA also set the CA32 bit as described in the <i>Power ISA (Version 3.1B)</i> and do not modify the reserved bit.</li> </ul>
46:56	Reserved	F5	Written by <b>mtxer</b> . An <b>mfker</b> reads the bit contents.
57:63	String length	F6	String length field used <b>lswx</b> and <b>stswx</b> . Written by <b>mtxer</b> . An <b>mfker</b> reads the bit contents.



## 5.1.4 Storage Access Alignment Support Overview

Most storage accesses are performed without software intervention (such as, an alignment interrupt). For more information on misaligned cases that do not result in an interrupt, see *Section 19 Performance Profile* on page 245. The storage accesses that result in an interrupt condition are described in the following sections.

### 5.1.4.1 Alignment Interrupts

The LSU reports an alignment interrupt for the following conditions:

- Storage operand is not on a natural alignment boundary.
  - Halfword boundary:
    - **Iharx**
    - **sthcx**.
  - Word boundary:
    - **Iwarx**, **Iwat**
    - **stwcx**, **stwat**
  - Doubleword boundary:
    - **Idarx**, **Idat**
    - **stdcx**, **stdat**
  - Quadword boundary:
    - **lfdp**, **lfdpx**, **lq**, **stfdp**, **stfdpx**, **stq**, **stqcx**, **lqarx**
  - Cache-line (128-byte) boundary:
    - **copy**, **paste**.
- Atomic memory operations take an alignment interrupt for cases identified by **ALI** in *Table 5-2* on page 78. Some cases only take an alignment interrupt for specific function code (FC) values as indicated. If no FC is indicated but ALI is indicated, the alignment interrupt occurs for any FC value. The columns indicate the 32-byte aligned address offset (the five LSBs of the EA).



Table 5-2. Alignment Interrupt for AMO Cases

AMO Instruction	x'00'	x'04'	x'08'	x'0C'	x'10'	x'14'	x'18'	x'1C'
<b>Iwat</b>	ALI FC '11100'							ALI FC '11000'
								ALI FC '11001'
<b>Idat</b>	ALI FC '11100'	ALI		ALI		ALI	ALI FC '11000'	ALI
							ALI FC '11001'	
<b>stwat</b>								ALI FC '11000'
<b>stdat</b>		ALI		ALI		ALI	ALI FC '11000'	ALI

Note: ALI = alignment interrupt; FC = function code

- Little-endian mode
  - **Imw, Iswi, Iswx**
  - **stmw, stswi, stswx**
- Caching-inhibited storage
 

The following instructions take an alignment interrupt regardless of their alignment when accessing a caching inhibited page/address.

  - **Ifdp, Ifdp, Imw, Iswi, Iswx,**
  - **dcbz, stfdp, stfdpx, stmw, stswi, stswx,**
  - Any load or store not on a natural alignment boundary:
    - Halfword boundary:
      - **Iha, Ihau, Ihaux, Ihax, Ihbrx, Ihz, Ihzcix, Ihzu, Ihzux, Ihzx, Ixsihx, Ixvrhx**
      - **sthbrx, sth, sthcix, sthu, sthux, sthx, stxsihx, stxvrhx**
    - Word boundary:
      - **Ifiwax, Ifiwzx, Ifs, Ifsu, Ifsux, Ifsx, Iwa, Iwaux, Iwax, Iwbrx, Iwz, Iwzcix, Iwzu, Iwzux, Iwzx, Ixssp, Ixvwsx, Ixsiwax, Ixsiwzx, Ixsspx, Ixvrwx**
      - **stfiwx, stfs, stfsu, stfsux, stfsx, stwbrx, stw, stwcix, stwu, stwux, stwx, stxssp, stxsiwx, stxsspx, stvrwx**
    - Doubleword boundary:
      - **Id, Idbrx, Idcix, Idu, Idux, Idx, Ifd, Ifdu, Ifdux, Ifdx, Ixsd, Ixsd, Ixvdsx, Ixvrdx**
      - **std, stdbrx, stdcix, stdu, stdux, stdx, stfd, stfd, stfdx, stxsd, stxsd, stxvrdx**
    - Quadword boundary:
      - **Ixvd2x, Ixvw4x, Ixvh8x, Ixvb16s**
      - **stxvd2x, stxvw4x, stxvh8x, stxvb16x**

**Note:** Unlike the POWER9 processor, the Power10 processor does not take an alignment interrupt when the following instructions access caching-inhibited storage when they are unaligned:

**Ixvl, Ixvll, Ixv, Ixvx, Ixvp, Ixvpx, plxvp, plxv, stxvl, stxvll, stxv, stvxv, stxvp, stxvpx, pstxvp, and pstxv.**



#### 5.1.4.2 Storage Control Attribute Caused Data Storage Interrupt or Hypervisor Data Storage Interrupts

The LSU reports either a data storage interrupt (DSI) or hypervisor data storage interrupt (HDSI) for the following conditions as permitted by the *Power ISA (Version 3.1B)*:

- The effective address specified by a **lq**, **stq**, **Iwat**, **Idat**, **Ibarx**, **Iharx**, **Iwarx**, **Idarx**, **Iqarx**, **stwat**, **stdat**, **stbcx.**, **sthcx.**, **stwcx.**, **stdcx.**, **stqcx.**, **copy** or **paste** instruction refers to storage that is caching inhibited; or the effective address specified by a **Iwat**, **Idat**, **stwat**, or **stdat** instruction refers to storage that is guarded. Note that unlike their non-prefixed counterparts, **plq** and **pstq** do not take a DSI or HDSI if the operand is quadword aligned and accesses storage that is caching inhibited.
- An attempt is made to execute one of the hypervisor accessible (Book IIIS) **Ibzcix**, **Ihzcix**, **Iwzcix**, **Idcix**, **stbcix**, **sthcix**, **stwcix**, or **stdcix** instructions with MSR[DR] = '1' or specifying a storage location (page) that was previously accessed as non-guarded using the *Hypervisor Real Mode Storage Control* facility.

#### 5.1.5 Fixed-Point Load and Store Instructions

The Power10 core implements the fixed-point load and store instructions per *Power ISA (Version 3.1B)*. Aside from the octword load and store operations, the Power ISA specifies that fixed-point loads and stores to storage, which are neither caching inhibited nor write-through and are aligned on their operand size boundary, are performed atomically.

There are some cases where the Power ISA states that some portion of the results of the instructions are undefined or some forms of the instructions are invalid. See *Section 5.2 Fixed-Point Invalid Forms and Undefined Conditions* on page 81 for details.

##### 5.1.5.1 Fixed-Point Load and Store Multiple Instructions

**Note:** These instructions are provided for compatibility with legacy software. Software should use a sequence of load or store instructions for optimal performance.

The **Imw** and **stmw** instructions, regardless of operand alignment, are executed in hardware, even when they cross page and segment boundaries. These instructions are not considered atomic. However, the individual storage accesses associated within the instructions are atomic. For example, if an **stmw** crosses a page boundary, and the second page translation signals an exception condition, then after the interrupt is taken, it appears as though none, some, or all of the accesses to the first page have occurred, and none of the accesses to the second page have occurred. On the other hand, for the **Imw** instruction that cross a page boundary where the second page translation signals an exception condition, some of the target registers might not be updated. Similarly, partially executed results might be observed for other conditions; therefore, the user should always expect nonatomic behavior.

An attempt to execute a non-word-aligned **Imw** or **stmw** instruction does not cause an alignment interrupt.

An attempt to execute an **Imw** or **stmw** instruction to storage-marked cache inhibited causes an alignment interrupt.

See *Section 5.1.4 Storage Access Alignment Support Overview* on page 77 for details.

The architecture allows these instructions to be interrupted by certain types of asynchronous interrupts (external interrupts, decrementer interrupts, machine check interrupts, and system reset interrupts). However, the Power10 core does not process an asynchronous interrupt in the middle of one of these instructions.



#### 5.1.5.2 Fixed-Point Move Assist Instructions

**Note:** These instructions are provided for compatibility with legacy software. Software should use a sequence of load or store instructions for optimal performance.

The **lswi**, **lswx**, **stswi**, **stswx** instructions, when aligned on a word boundary, are executed in hardware, even when they cross page and segment boundaries. These instructions are not considered atomic. Furthermore, the individual storage accesses associated within the instructions are not atomic. For example, if a store string operand crosses a page boundary, and the second page translation signals an exception condition, then after the interrupt is taken, it appears as though none, some, or all of the accesses to the first page have occurred, and none of the accesses to the second page have occurred. On the other hand, for a load string operand that crosses a page boundary where the second page translation signals an exception condition, some of the target registers might not be updated. Similarly, partially executed results might be observed for other conditions; therefore, the user should always expect non-atomic behavior.

An attempt to execute a non-word-aligned **lswi**, **lswx**, **stswi**, or **stswx** instruction causes an alignment interrupt.

An attempt to execute an **lswi**, **lswx**, **stswi**, or **stswx** instruction to storage marked cache inhibited causes an alignment interrupt.

See *Section 5.1.4 Storage Access Alignment Support Overview* on page 77 for details.

The architecture allows these instructions to be interrupted by certain types of asynchronous interrupts (external interrupts, decrementer interrupts, machine check interrupts, and system reset interrupts). However, the Power10 core does not process an asynchronous interrupt in the middle of one of these instructions.

#### 5.1.5.3 Integer Select Instruction

The Power10 core implements the integer select (**isel**) instruction as defined in the *Power ISA (Version 3.1B)*.

#### 5.1.5.4 Fixed-Point Logical Instructions

The architecture defines the *preferred NOP* to be '**ori 0,0,0**'. In the Power10 processor, this NOP form is recognized by the hardware and allowed to complete without taking any execution resources. This makes the instruction valuable for padding other instructions to achieve better alignment or better separation.

#### 5.1.5.5 Access to Performance Monitor Special Purpose Registers

The Power10 core supports the following performance monitor unit (PMU) special purpose registers (SPRs) as specified in the *Power ISA (Version 3.1B)*:

- PMC1 - Performance Monitor Counter 1
- PMC2 - Performance Monitor Counter 2
- PMC3 - Performance Monitor Counter 3
- PMC4 - Performance Monitor Counter 4
- PMC5 - Performance Monitor Counter 5
- PMC6 - Performance Monitor Counter 6
- MMCR0 - Monitor Mode Control Register 0
- MMCR1 - Monitor Mode Control Register 1
- MMCR2 - Monitor Mode Control Register 2
- MMCR3 - Monitor Mode Control Register 3
- MMCRA - Monitor Mode Control Register A
- MMCRC - Monitor Mode Control Register C
- SDAR - Sampled Data Address Register
- SIAR - Sampled Instruction Address Register
- SIER - Sampled Instruction Event Register
- SIER2 - Sampled Instruction Event Register 2
- SIER3 - Sampled Instruction Event Register 3

#### 5.1.5.6 Move to/from Condition Register Fields Instructions

The architecture warns that updating a subset of the **CR** fields on an **mtrcf** instruction might have poorer performance than updating all of the fields. For best performance in the Power10 processor, software should use the single-field variants (**mtocrf** and **mfocrf**) of these instructions as described in the *Power ISA (Version 3.1B)*.

## 5.2 Fixed-Point Invalid Forms and Undefined Conditions

The results of executing an invalid form of a fixed-point instruction or an instance of a fixed-point instruction when the architecture specifies that some results are undefined are described in the following list (for the cases when executing an instruction does not cause an exception).

- Instruction with Reserved Fields  
Bits in reserved fields are ignored; the results of executing an instruction in which one or more reserved bits are '1' is the same as if the bits were '0'.
- Load with Update Instructions (RA = 0)  
EA is placed into R0.
- Load with Update Instructions (RA = RT)  
The storage operand addressed by EA is accessed. The displacement field is added to the data returned by the load and placed into RT.
- Load Quadword Instruction (RTp is odd or RTp = RA)  
The Power10 processor always takes a hypervisor emulation assistance interrupt anytime RTp is an odd register, RTp = RA (including when RA = 0) or RTp = RB for **Iq**.



- Load Quadword and Reserve Indexed Instruction (RTp is odd, RTp = RA, RTp = RB)  
The Power10 processor always takes a hypervisor emulation assistance interrupt anytime RTp is an odd register, RTp = RA (including when RA = 0) or RTp = RB for **Iqarx**.
- Prefixed Load VSX Vector Paired 8LS (**plxvp**) (if R is equal to '1' and RA is not equal to '0')  
Treated the same as if RA is equal to zero (ignore the RA field)
- Prefixed Store VSX Vector Paired MLS: (**pstxvp**) (if R is equal to '1' and RA is not equal to '0')  
Treated the same as if RA is equal to zero (ignore the RA field)
- Prefixed [Masked] Load VSX Vector (**plxv0**, **plxv1**, **pmlxv0**, and **pmlxv1**) (if R is equal to '1' and RA is not equal to '0')  
Treated the same as if RA is equal to zero (ignore the RA field)
- Prefixed [Masked] Store VSX Vector (**pstxv0**, **pstxv1**, **pmstxv0**, and **pmstxv1**) (if R is equal to '1' and RA is not equal to '0')  
Treated the same as if RA is equal to zero (ignore the RA field)
- Load Multiple Instructions (RA in the range of registers to be loaded)  
If an exception (for example, data storage or external) causes the execution of the instruction to be interrupted, the instruction is restarted, RA has been altered by the previous partial execution of the instruction, and RA is less than or greater than '0', the new contents of RA are used to compute EA.
- Load Multiple Instructions (causing a misaligned access)  
For a Load Multiple Word instruction, if the storage operand specified by EA is not a multiple of 4, the access is performed anyway (that is, naturally). No alignment interrupt is taken.
- Load String Instructions (zero length string)  
RT is not altered.
- Load String Instructions (RA and/or RB in the range of registers to be loaded)  
If RA and/or RB is in the range of registers to be loaded, the results are as follows.
  - Indexed Form: If RA = 0, let Rx be RB; otherwise, let Rx be the register specified by the smaller of the two values in instruction fields RA and RB.  
If RT = Rx, no registers are loaded; otherwise, registers RT through RX - 1 are loaded as specified in the architecture (that is, only part of the storage operand is loaded).
  - Immediate Form: If RA = 0, the instruction is executed as if it were a valid form.  
If RA = RT, no registers are loaded; otherwise, registers RT through RA - 1 are loaded as if the instruction were a valid form but specifying a shorter operand length.

- Store with Update Instructions (RA = 0)  
EA is placed into R0.
- Store Quadword and Store Quadword Conditional Instruction (RSp is odd)  
For the **stq** and **stqcx** instructions, the contents of RSp are stored into the words of storage addressed by EA and EA + 4 respectively. If RSp is odd, the low-order bit of the register number is considered to be '0' such that RSp - 1 and RSp are stored into the words in storage addressed by EA and EA + 4 respectively.
- **subfic**, **subfc**, and **subfco** Instructions and their Rc = 1 Forms  
If RA[0:15] = x'0000', XER[CA] reflects the carry-out of bit 16; otherwise, it reflects the carry-out of bit 40.
- **divw**, **divwo**, **divwu**, and **divwuo** Instructions  
RT[0:31] is set to x'00000000'.
- **mulhw** and **mulhwu** Instructions  
RT[0:31] contains the same result as RT[32:63].



- Divide Instructions (divide by zero)  
If the divisor is 0, RT is set to zero. If Rc = '1' also, CR0 is set to '0010'.
- Move To/From Special Purpose Register Instructions  
*Table 5-7 SPR Table* on page 106 describes the read/write **mtspr** behavior for an spr value that is not defined for the implementation.
- Move From Time Base Instruction  
The **mftb** instruction is treated as an alias for the "mfspr Rx, 268" instruction. The results are the same as when executing an "mfspr Rx, 268" instruction.
- Move From Condition Register Instruction  
The entire CR is copied into RT[32:63]. RT[0:31] is set to zero.
- Move From One Condition Register Field Instruction (only 1 bit of FXM set to '1')  
Let n be the bit set to '1' in the FXM field. The CR field *n* is copied to RT[(4 × n + 32):(4 × n + 35)]. The remaining bits are set to zero.
- Move From One Condition Register Field Instruction (multiple bits of FXM set to '1')  
Let n be the first bit (from left to right) set to '1' in the FXM field. The CR field *n* is copied to RT[(4 × n + 32):(4 × n + 35)]. The remaining bits are set to zero.
- Shift Right Algebraic (**srad**, **sradi**, **sraw**, **rawi**)  
These instructions write OC = '0', and have no dependency on CAOC (read-modify-write CAOC).



## 5.3 Vector Scalar Instructions (FP, VMX, and VSX)

The Power10 VSU supports the full architecture for the FP, VMX and VSX instruction set. The pipelines are loosely described in *Section 3.2.6 Compute Pipelines* on page 48 and details for specific instructions can be found in *Appendix A Instruction Properties* on page 295.

### 5.3.1 IEEE Compliance

The Power10 implementation of binary floating-point (BFP), decimal floating-point (DFP), and vector-scalar floating-point (VSX) architecture complies with the IEEE P754-2008 standard.

#### 5.3.1.1 Non-IEEE Modes

If FPSCR[NI] is set, the architecture allows a change in the behavior of the binary floating-point unit (BFU) instructions and the VSX floating-point instructions. See the sections Floating-Point Facility and Vector-Scalar Floating-Point Instructions in the *Power ISA (Version 3.1B)*. In the Power10 core, setting FPSCR[NI] has no effect.

The architecture requires implementation of VSCR[NJ]. This alters the behavior of the VMX floating-point instructions [see the section Vector Facility in the *Power ISA (Version 3.1B)*] by replacing denormal operands and results with the value '0'. There is no performance difference.

### 5.3.2 Floating-Point Exceptions

Precise floating-point exceptions are provided whenever either of the floating-point enabled exception mode bits (MSR[FE0], MSR[FE1]) are set. In all cases, the floating-point instructions are executed out-of-order, and any resulting exceptions are resolved at completion time.

### 5.3.3 Floating-Point Load and Store Instructions

Most forms of unaligned floating-point storage accesses are executed entirely in hardware. See *Section 5.1.4 Storage Access Alignment Support Overview* on page 77 for details.

#### 5.3.3.1 Scalar Load and Store Atomicity

The *Power ISA (Version 3.1B)* requires binary floating-point and VSX scalar load and store accesses be treated as atomic provided they are aligned on an operand boundary and access storage that is not caching inhibited. The Power10 core complies with the Power ISA in this regard. Furthermore, binary floating-point and VSX scalar load and store accesses, which do not cross a doubleword boundary and access storage that is not caching inhibited, are also atomic.

#### 5.3.3.2 Vector Load and Store Atomicity

The *Power ISA (Version 3.1B)* requires each doubleword of vector (both VMX and VSX) load and store accesses be treated as atomic provided they are doubleword aligned and access storage that is not caching inhibited. The Power10 core complies with the Power ISA in this regard.



### 5.3.4 Heterogeneous Precision Arithmetic

The following instructions are referred to as scalar single-precision arithmetic instructions. These instructions perform single-precision arithmetic operations using a double-precision representation in the register file.

- **fadds[.], xsaddsp, fsubs[.], xssubsp, fmuls[.], xsmulsp**
- **fmadds[.], xsmadd[am]sp, fmsubs[.], xsmsub[am]sp**
- **fnmadds[.], xsnmadd[am]sp, fnmsubs[.], xsnmsub[am]sp**
- **fsqrts[.], xssqrtsp, fdivs[.], xsdivsp**
- **fres[.], xsresp, frsqrtes[.], xsrsqrtesp**

#### 5.3.4.1 NaN Propagation

If a single-precision arithmetic instruction propagates a not-a-number (NaN) where any of the fraction bits [24:52] is nonzero, the resulting quiet not-a-number (QNaN) has all of the fraction bits [24:52] cleared to zero.

#### 5.3.4.2 Square Root Overflow and Underflow

Due to the compacting nature of the square-root operation, the instructions **fsqrts**, **xssqrtsp**, **frsqrtes**, and **xsrsqrtesp** cannot underflow or overflow if their operands are representable in single-precision format. However, if the operand is not representable in single-precision format, an underflow or overflow can occur. This result is recorded in the FPSCR.

#### 5.3.4.3 Hardware Behavior on Enabled Underflow and Enabled Overflow Exception

If FPSCR[UE] is enabled and an underflow occurs, the contents of the result register and FPSCR status are not defined for scalar single-precision (SP) instructions. The hardware takes the following actions:

1. Underflow exception is set, FPSCR[UX] = '1'.
2. The exponent of the normalized intermediate result is adjusted by adding 192.
3. The double-precision bias of 1023 is added to the exponent.
4. The biased exponent is reduced to 11 bits by ANDing with x'7FF'.
5. The adjusted rounded result is placed into the target FPR.
6. FPSCR[FPRF] is set to indicate a normalized number.

If FPSCR[OE] is enabled and an overflow occurs, the contents of the result register and the FPSCR status are not defined for scalar SP instructions. The hardware takes the following actions:

1. Overflow exception is set, FPSCR[OX] = '1'.
2. The exponent of the normalized intermediate result is adjusted by subtracting 192.
3. The double-precision bias of 1023 is added to the exponent.
4. The biased exponent is reduced to 11 bits by ANDing with x'7FF'.
5. The adjusted rounded result is placed into the target FPR.
6. FPSCR[FPRF] is set to indicate normalized number.



### 5.3.5 Handling of Denormal Single-Precision Values in Double-Precision Format

Like the POWER9 processor, the Power10 processor is capable of handling denormal single-precision values as inputs for all subsequent instructions.

### 5.3.6 Vector and Floating-Point Invalid Forms and Undefined Conditions

The results of executing an invalid form of a floating-point instruction or an instance of a floating-point instruction when the architecture specifies that some results are undefined are described in the following list (for the cases when executing an instruction does not cause an exception). Software should avoid becoming dependent on the behaviors as they are subject to change in future generation POWER processors.

- Scalar single-precision instructions with operands not representable in single-precision format.  
See *Section 5.3.4 Heterogeneous Precision Arithmetic* on page 85.
- Instructions with reserved fields.  
Bits in reserved fields are ignored. The results of executing an instruction when one or more reserved bits are '1' is the same as if the bits were '0'.
- Load or store floating-point with update instructions (RA = 0).  
EA is placed into R0.
- Floating-point store single instructions (exponent < 874 and FRS[9:31] less than or greater than '0').  
The value placed in storage is a '0' with the same sign as the value in the register.
- Because the binary floating-point registers (FPRs) are mapped to the vector-scalar registers 0 - 31 in the *Power ISA (Version 3.1B)*, the rightmost doubleword is updated with zero whenever a binary or decimal floating-point instruction writes the target FPR. This behavior applies to any binary or decimal floating-point instruction that writes an FPR, not just loads.
- Scalar convert to integer word instructions (**xscvdpxws**, **xscvdpsxws**, **fctiwuz**, **fctiwz**, **ctiwu**, **fctiw**).  
VSR[0:31] is set to VSR[32:63].
- Scalar convert to integer instructions.  
FPSCR[FPRF] is set to '00000'.
- VSX vector convert from double-precision to single-precision (**xvcvdpsp**).  
VSX vector convert double-precision to integer word (**xvcvdpsxws**, **xvcvdpxws**).  
VSX vector convert from integer doubleword to single-precision (**xvcvsxdsp**, **xvcvuxdsp**).
  - VSR[32:63] is set to VSR[0:31].
  - VSR[96:127] is set to VSR[64:95].
- Move from FPSCR instruction.  
FRT[0:63] is set to FPSCR[0:63] with the first 29 bits set to zero.
- Scalar reciprocal estimate instructions: **fre**, **fres**, **xsredp**, **xsresp**, **frsqrte**, **frsqrtes**, **xrsqrtep**, **xsrsqrtesp**.  
FPSCR[FR] and FPSCR[FI] are set to '0' and FPSCR[XX] is unchanged, even if an overflow exception occurs.
- VSX vector floating-point reciprocal estimate instructions: **xvredp**, **xvresp**, **xvrsqrtep**, **xvrsqrtesp**.  
FPSCR[XX] is unchanged, even if an overflow exception occurs.
- Disabled overflow exception (OX = '1', OE = '0').  
For divide, square root, and all 128-bit binary FP instructions (**xs\*qp**), FPSCR[FR] is set to '1' if the result is rounded to  $\pm\infty$ , and set to '0' if the result is rounded to the largest representable number.

For scalar reciprocal estimate instructions, FPSCR[FR] is set to '0'.

For all other instructions, FPSCR[FR] is set to '1' if a disabled overflow exception occurs.

- Decimal floating-point quad instructions, where an odd target or source register is specified, are considered invalid forms. The Power10 core always ignores the low-order VSR bit number and addresses the even-odd resulting register pair. This applies to both the source and target register pairs.
- For cacheable ( $I = 0$ ) memory accesses, the Power10 core implements the load vector element instructions the same as **lvx** (same as the POWER8 and POWER9 cores). Thus, those bytes in the target VSR that are undefined in the Power ISA for the load vector element instructions contain the value that would be written there if an **lvx** instruction were executed instead. For non-cacheable memory accesses, the load vector element instructions are implemented as described in the *Power ISA (Version 3.1B)*. Note that in terms of Data Address Watchpoint Register (DAWR) match criteria, a match only occurs for the bytes specified as written to the target VSR by the Power ISA regardless of whether the access is to cacheable or non-cacheable storage.
- VSX load and store vector with length (**lxvl**, **stxvl**, **lxvll**, **stxvll**) specify the number of bytes to load in RB[0:7]. The architecture requires RB[8:63] to be equal to zero. For these instructions, for effective address calculation purposes, the hardware discards the upper 8 bits of RB in computing the effective address EA; cycle time does not permit for zeroing out RB[8:63]. Therefore, the result of the address generation is: EA = RA[0:63] + RB[8:63].
- Vector shift right, vector shift left (**vsl**, **vsr**)  
The **vsr** and **vsl** instructions behave the same as **vsrv** and **vslv**, respectively. This defines their behavior if for any byte element in register VR[VRB], the low-order 3 bits are not equal to the shift amount.
- Vector extract/insert  
In case the index is too large, RFC2609 does not define instruction behavior for the following instructions:  
**vextractuh**, **vextractuw**, **vextractd**, **vinserth**, **vinsertw**, **vinsertd**, **vextuhlx**, **vextuhrx**, **vinshlx**,  
**vinshrx**, **vextuwlx**, **vextuwrx**, **vinswlx**, **vinswrx**, **vinsw**, **vextduhlx**, **vextduhrx**, **vinshvlx**, **vinshvrx**,  
**vextduwvlx**, **vextduwvrx**, **vinswvlx**, **vinswvrx**, **vextddvlx**, **vextddvrx**, **vinsdlx**, **vinsdrx**, **vinsd**
- Load Special Value Quadword (**lxvkq**)  
For **lxvkq**, reserved values of UIM return zero.
- Vector Binary Floating Point Record-Form Compare Instructions:  
**xvcmpqdp\_rc**, **xvcmpqsp\_rc**, **xvcmpgedp\_rc**, **xvcmpgesp\_rc**, **xvcmpgtdp\_rc**, **xvcmpgtsp\_rc**  
If a trap-enabled invalid exception occurs in any element of the vector (suppress condition), CR[6] is written as if the exception was not enabled.

## 5.4 Optional Facilities and Instructions

The Power10 processor implements all optional features as defined in the *Power ISA (Version 3.1B)* except the "not Memory Coherence Required" ( $M = 0$ ) and the "Write Through Required" ( $W = 1$ ) storage control attribute values.



## 5.5 Little-Endian Mode

The Power10 core supports true little-endian mode. Byte swapping is performed before each instruction is written to the I-cache and before data is fetched into the execution units; that is, between the D-cache and the execution units (for example, GPR, FPR/VR/VSR). Prefixed instructions are treated as two independent words meaning the byte ordering of each word is swapped before each word is written into the I-cache. However, the order of the words is not swapped from the order in which they appear in big-endian mode.

The load and store multiple instructions and the move-assist instructions are not supported in little-endian mode. Attempting to execute any of these instructions in little-endian mode causes the system alignment error handler to be invoked.

## 5.6 Book II - Virtual Environment Architecture

### 5.6.1 Cache

The Power10 core supports a coherence block size of 128-bytes that is commonly referred to as a cache line.

The Power10 chip contains three levels of cache hierarchy. All the caches (L1 I-cache, L1 D-cache, L2 cache, and L3 cache) are dynamically shared among all the threads on an SMT4-core-resource. For the L1 I-cache and L1 D-cache, the contents are located using an effective address and are in general, owned by whichever thread installed that cache line. The exceptions to this rule and other details of these effective addressed caches are described in *Section 5.10.11.1 Instruction Cache* on page 138 and *Section 5.10.11.2 Data Cache* on page 139. For the L2 and L3 caches, the contents are real address indexed and are fully shareable, such that a cache line can be installed by one thread and used by the other threads. See *Section 6 L2 Cache* on page 169 for details on the L2 cache and *Section 7 L3 Cache* on page 175 for details on the L3 cache.

The Power10 chip automatically maintains the coherency of all data cached in these caches. The L1 cache employs Harvard-cache organization, with separate L1 I-cache and L1 D-cache. L2 and L3 caches are unified. Because some levels of the cache hierarchy contain both instructions and data, when an instruction cache reload request is serviced by the L2 or the L3 cache, it is done so in a coherent manner.

The processor keeps the instruction storage consistent with the data storage. All cache lines in the L1 I-cache and L1 D-cache are also present in the L2 cache (inclusive property maintained).

The L1 I-cache is 48 KB, 6-way set associative that is tagged using the effective address (EA). See *Section 5.10.11.1 Instruction Cache* for more details.

The L1 D-cache is 32 KB, 8-way set associative that is tagged using the EA. See *Section 5.10.11.2 Data Cache* for more details.

In addition to maintaining caches, the Power10 chip also includes several types of queues that act as logical predecessors and extensions to the caches. In particular, the machine contains store queues for holding store data above the caches, cast-out queues for holding modified data that has been pushed out of the caches (by the replacement algorithm, cache control instructions, or snoop requests), and others. All of these queues are maintained coherent by the hardware to the extent required by the architecture. In general, their presence should not be observable by either software or system hardware.

## 5.6.2 Classes of Instructions

The Power10 core implements all of the Book II instructions as described in the following subsections.

### 5.6.2.1 Instruction Cache Block Touch Instruction (**icbt**)

The Power10 core supports the instruction cache block touch (**icbt**) instruction by mapping it to **dcbtst** to prefetch a 128-byte line into the L3 cache.

### 5.6.2.2 Instruction Cache Block Invalidate (**icbi**)

The Power10 core implements a split instruction/data (I/D) L1 cache where both caches are kept coherent with the L2 cache. Whenever any modification is made to the cache lines contained in the L2 cache, the L2 cache invalidates the copies in the L1 I-caches. Because of this, after an **icbi** instruction is translated, the processor core converts it to a NOP and does not broadcast the cache line targeted by the **icbi** instruction as the architecture stipulates. However, it is still recommended that software follows the required instruction sequence for both same-thread and cross-thread code modifications in the event that future processors implement the **icbi** instruction differently.

### 5.6.2.3 Instruction Cache Synchronize (**isync**)

As a performance optimization, the Power10 core internally tracks and scoreboards specific instructions that are required to be synchronized by the **isync** instruction per the *Power ISA (Version 3.1B)*. When the **isync** instruction is executed, this scoreboard bit is checked to see whether or not the machine must *flush and refetch* the instructions following the **isync**. The set of instructions that require synchronization by the **isync** instruction are as follows:

- Store instruction that updates the I-cache
- The move-to SPR scoreboard operations which require a subsequent CSI (see *Power ISA Operating Environment Architecture - Book III (version 3.1B)* for CSI registers)
- **icbi**, **ptesync**, **tlbie**, **tlbiel**, **slbie**, **slbia**, and **slbmte**. Note that **slbieg** (when UPRT = '1') does not set the scoreboard because the Power10 core does not support UPRT = '1' for HPT.

### 5.6.2.4 Vector Category Prefetch Instructions (**dss**, **dst**, and **dstst**)

The vector category data stream instructions **dss**, **dst**, and **dstst** are implemented as NOPs.



#### 5.6.2.5 Data Cache Block Touch Instructions (**dcbt** and **dcbtst**)

The data cache block size for **dcbt** and **dcbtst** on the Power10 core is 128 bytes.

With the exception of trace interrupts, these instructions do not take interrupts. That means, if they miss either the SLB (for HPT translation) or the page table, or they encounter some other type of translation-related exception condition, no interrupt will be reported. This property applies regardless of which TH value is specified.

In general, the **dcbt** and **dcbtst** instructions check the state of the cache hierarchy, and if the addressed block is not present, it initiates a reload. Note that this might result in the reload of any of the L1 D-cache, L2 cache, or the L3 cache with the addressed block if it is not already present in these caches. If the address block is already present in a cache, the cache content is not altered. If the **dcbt** or **dcbtst** instruction does reload the addressed blocks, it affects the state of the cache replacement algorithm bits. The cache level targeted by the processor and other details of a single data block prefetch can be found in *Section 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics* on page 90. For streaming prefetch operations, which include streams defined by a sequence of **dcbt** or **dcbtst** instructions, the cache level targeted by the processor and other details of streaming prefetch operations can found in *Section 5.6.2.7 Data Cache Block Touch Streaming Characteristics* on page 91.

The **dcbt** and **dcbtst** instructions perform their intended function as described in the *Power ISA (Version 3.1B)*. Reference bit updates vary by TH value as described in the various following sections. Change bits are never set in PTEs by either **dcbt** or **dcbtst** instructions regardless of the TH value.

The Power10 core implements all the TH values defined in the *Power ISA (Version 3.1B)*. Additionally, the Power10 core implements the TH = '01110' form of **dcbt** and **dcbtst** consistent with the specification in the *Power10 Processor Programming Model Bulletin* and as further described in *Section 5.6.2.10 Data Cache Block Touch Data Stream Descriptor (TH = '01010' through TH = '01111')* on page 91. Fields specified in the *Power10 Processor Programming Model Bulletin* that appear in red text should be set to '0' as they are not implemented. The fields not implemented for TH = '01010' are: CNR, RM, CD, GNR. The fields not implemented for TH = '01110' are: Stride, Offset, Go, S, CR, RM, CD, and GR.

Details of **dcbt** and **dcbtst** that are unique to specific TH values are documented in the following subsequent sections.

#### 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics

The **dcbt** and **dcbtst** instructions with TH = '00000', TH = '10000', or TH = '10001' access a single block of data (128-bytes). These instructions act as a touch for the cache hierarchy, ERAT, and the TLB.

For HPT translation, if data translation is enabled (MSR[DR] = '1') and an SLB miss results, the instruction is treated as a NOP. Otherwise, the page table reference bits will be set and if there is a TLB miss, the instruction will reload the TLB and continue to reload the cache.

For radix translation, if there is a TLB miss, the TLB will be reloaded only if the reference bit is already set.

Once past translation, if the page protection attributes prohibit access, the page is marked cache inhibited or the page is marked guarded, the instruction is finished as a NOP and it does not reload the cache.

The cache targeted for potential reload is the L1 D-cache for **dcbt** and the L3 cache for the **dcbtst**.

When MSR[SE] = '1', a trace interrupt for TH = '00000' will result in SRR1[35] = '1' and SDAR will be updated containing the EA of the data access.



### 5.6.2.7 Data Cache Block Touch Streaming Characteristics

The **dcbt** and **dcbtst** instructions initiated by a GO have streaming characteristics. These include TH = '01000' operations specifying GO = '1' and include stream specification sequences with TH = '01010', TH = '01011', and TH = '01110'. These instructions can initiate a sequence of touch operations for the cache hierarchy, ERAT, and the TLB.

For HPT translation, if data translation is enabled (MSR[DR] = '1') and an SLB miss results from a touch, the touch behavior is treated as a NOP. Otherwise, the page table reference bits will be set and if there is a TLB miss as a result of a touch, it will reload the TLB, but individual touches may act as a NOP for the cache hierarchy.

For radix translation, if there is a TLB miss as a result of a touch, the TLB will be reloaded only if the reference bit is already set, but individual touches may act as a NOP for the cache hierarchy.

Once past translation, if the page protection attributes prohibit access, the page is marked cache inhibited or the page is marked guarded, touches are treated as NOP operations.

The cache targeted for potential reload is always the L3 for TH = '01110' initiated streaming prefetches, and is otherwise the L1 D-cache for **dcbt** and the L3 cache for the **dcbtst** initiated streams (Note: in the Power10 processor, the opcode of the TH = '01000' dictates the cache target for these cases).

### 5.6.2.8 Data Cache Block Touch - Invalid TH Forms (TH = '00001' through TH = '00111')

The Power10 core treats **dcbt** and **dcbtst** for the invalid TH values of '00001' through '00111' the same as TH = '00000'. See *Section 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics* on page 90 for additional characteristics.

### 5.6.2.9 Data Cache Block Touch Data Stream (TH = '01000')

The Power10 core treats **dcbt** and **dcbtst** with a TH value of '01000' as described in the *Power ISA (Version 3.1B)*. When MSR[SE] = '1', a trace interrupt for one of these TH values will result in SRR1[35] = '1' and SDAR will be updated containing the EA of the data access. See *Section 5.6.2.7 Data Cache Block Touch Streaming Characteristics* on page 91 for additional characteristics.

### 5.6.2.10 Data Cache Block Touch Data Stream Descriptor (TH = '01010' through TH = '01111')

The Power10 core treats **dcbt** and **dcbtst** with a TH value of '01010' through TH = '01111' as described in the *Power ISA (Version 3.1B)* and inclusive of the *Power10 Processor Programming Model Bulletin*. For these values of TH, there is no associated address; therefore, there are no reference or change bits to be set in the PTE; however, references can be generated as the streaming touches are generated, as described in *Section 5.6.2.7 Data Cache Block Touch Streaming Characteristics* on page 91. When MSR[SE] = '1', a trace interrupt for one of these TH values will result in SRR1[35] = '0' and SDAR will not be updated. As described in the *Power10 Processor Programming Model Bulletin*, a **dcbt** and **dcbtst** with a TH value of '011110' initiates a sequence of prefetches to a block of storage when U = '0' and when preceded by a TH = '01000' that describes the starting address of the data block. In this case, the block of storage is fetched into the L3 cache and can be used to avoid the latency associated with fetches from distant caches or from memory. The maximum block size that is prefetched without first requiring subsequent stream accesses by the program is



128 KB. See *Section 5.6.2.7 Data Cache Block Touch Streaming Characteristics* on page 91 for additional characteristics. Also see *Section 19.1.7.14 Software-Initiated Data Prefetch* on page 288 for additional information about utilizing streaming prefetch instructions effectively.

#### **5.6.2.11 Data Cache Block Touch - Transient (TH = '10000')**

The Power10 core implements the load and store version of the following transient touch instructions: **dcbtct**, **dcbtds**, **dcbtt**, **dcbtstct**, and **dcbtsttt**. The transient property of a cache line is retained in the L3 cache for both the load and store version of the transient touch instructions. The transient property of a cache line is retained in the L2 cache for the load version of the transient touch instruction for the case that the line is loaded but not modified. In this transient state, the transient line becomes the most likely cache line in its congruence class to be replaced next, thus preserving the other cache lines in that congruence class. This behavior is useful if it is known that a set of lines will be loaded or stored with a low probability for temporal cache reuse and it is desirable that they be as minimally intrusive to the cache as possible (for example, displacing as few lines in the cache as possible).

See *Section 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics* on page 90 for additional characteristics.

#### **5.6.2.12 Data Cache Block Touch - No Access Needed Anymore (TH = '10001')**

The Power10 core implements the load version of the **dcbna** instruction (previously called Data Cache Block Discard). This operation sets the cache replacement state in the L2 or L3 cache to indicate that it should be replaced next within its congruence class. See *Section 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics* on page 90 for additional characteristics.

#### **5.6.2.13 Data Cache Block Zero (dcbz)**

The data cache block size for **dcbz** on the Power10 core is 128 bytes.

The function of **dcbz** is performed in the L2 cache. As a result, if the block addressed by the **dcbz** is present in the L1 D-cache, the block is invalidated before the operation is sent to the L2 cache logic for execution. The L2 cache gains exclusive access to the block (without actually reading the old data) and performs the zeroing function in a broadside manner.

If the cache block specified by the **dcbz** instruction contains an error (even one that is not correctable with ECC), the contents of all locations within the block are set to zeros in the L2 cache. If the specified block in the L2 cache does not contain a hard fault, a subsequent load from any location within the cache block returns zeros and does not cause a machine-check interrupt.

If the block addressed by the **dcbz** instruction is in a memory region marked *cache inhibited*, the instruction causes an alignment interrupt to occur.

#### **5.6.2.14 Data Cache Block Store (dcbst)**

The data cache block size for **dcbst** on the Power10 core is 128 bytes.

The **dcbst** instruction has no direct effect on the L1 D-cache (because it is store-through and it never contains modified data). The **dcbst** instruction also has no direct effect on the L2 cache or L3 cache (both of these are kept coherent with memory and I/O, so that nothing special must be done). As a result, the instruc-

tion simply goes through address translation, reports any errors, and is completed. The instruction is not sent to the storage subsystem, and consequently it does not broadcast any transactions onto the inter-processor SMP interconnect.

#### 5.6.2.15 Data Cache Block Flush (**dcbf**, **dcbfl**, and **dcbflp**)

The data cache block size for **dcbf**, **dcbfl**, and **dcbflp** on the Power10 core is 128 bytes.

The Power10 core supports **dcbf** (L = 0), **dcbfl** (**dcbf** with L = 1), **dcbflp** (**dcbf** with L = 3), **dcbfps** (**dcbf** with L = 4) and **dcbstps** (**dcbf** with L = 6) as specified in the *Power ISA (Version 3.1B)*. Reserved L-values of 2, 5, and 7 behave the same as L = 0.

#### 5.6.2.16 Key Aspects of Storage Control Instructions

In the Power10 core, all cache control instructions operate on aligned 128-byte sections of storage. *Table 5-3* summarizes many of the key aspects of the storage control instructions.

*Table 5-3. Storage Control Instructions*

Aspect	Book II Cache Instructions					
	<b>icbi</b>	<b>dcbt</b>	<b>icbt/dcbtst</b>	<b>dcbz</b>	<b>dcbst</b>	<b>dcbf/dcbfl</b>
Granularity	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes
Semantic checking	load (DSI on storage exception)	load (NOP on storage exception)	load (NOP on storage exception)	store (DSI on storage exception)	load (DSI on storage exception)	load (DSI on storage exception)
“r” bit update	yes	yes	yes	yes	yes	yes
“c” bit update	no	no	no	yes	no	no
L1 I-cache effect	see <i>Section 5.6.2.3</i> on page 89	none	none	none	none	none
L1 D-cache effect	none	see <i>Section 5.6.5</i> on page 97	see <i>Section 5.6.5</i> on page 97	as define in architecture	NOP	as define in architecture
L2 cache effect	none	see <i>Section 5.6.5</i> on page 97	see <i>Section 5.6.5</i> on page 97	see <i>Section 5.6.2.13</i> on page 92	see <i>Section 5.6.2.14</i> on page 92	see <i>Section 5.6.2.15</i> on page 93
L3 cache effect	none	see <i>Section 5.6.5</i> on page 97	see <i>Section 5.6.5</i> on page 97	see <i>Section 5.6.2.13</i> on page 92	see <i>Section 5.6.2.14</i> on page 92	see <i>Section 5.6.2.15</i> on page 93
TLB effect	reload as required	reload as required	reload as required	reload as required	reload as required	reload as required
<u>SLB</u> effect	reload as required	None (NOP if miss)	None (NOP if miss)	reload as required	reload as required	reload as required

#### 5.6.2.17 Copy/Paste Instructions

As stated in the *Power ISA (Version 3.1B)*, the **copy** instruction is only permitted to read from non-control memory address space. The **paste** instruction is permitted to write to either control memory or non-control memory addresses. See *Section 5.9.2 Control Memory Definition and Accessibility* on page 122 for the definition of control memory versus non-control memory.



### 5.6.2.18 Near Memory Instruction Support

The Power10 processor supports the atomic memory operation (AMO) instructions as described in the *Power ISA (Version 3.1B)*. For details regarding when AMOs take alignment interrupts, see *Section 5.1.4 Storage Access Alignment Support Overview* on page 77.

### 5.6.2.19 Wait Instruction

The **wait** instruction is implemented as described in the *Power ISA (Version 3.1B)* with the exception that for  $WC = 1$ , instead of waking up due to the loss of a reservation granule, the Power10 processor instead wakes up based on a fixed unit of time, similar to a pause-short, or when an update to the L1 data cache is detected based on traffic from other processing cores. Additionally, the Power10 processor implements the **pause-short** variant of the **wait** instruction ( $WC = 2$ ).

## 5.6.3 Storage Model

### 5.6.3.1 Storage Access Ordering

The architecture defines a weakly ordered storage model. The Power10 processor takes advantage of this relaxed requirement to achieve better performance through out-of-order instruction execution and out-of-order bus transactions. As a result, if strongly-ordered storage accesses are required, software must use the appropriate synchronizing instruction (**hwsync**, **ptesync**, **eieio**, **lwsync**, **phwsync**, or **plwsync**) to enforce order explicitly. Unlike the POWER8 and POWER9 processors, the Power10 processor does not support the Strong Access Ordering storage attribute, because this support was removed in the *Power ISA (Version 3.1B)*.

In hypervisor real mode, the Power10 core employs the page-based Real Mode Storage Control (RMSC) facility described in *Power ISA Operating Environment Architecture - Book III (version 3.1B)*.

Stores to storage marked as *non-guarded* can be observed as being performed out-of-program-order by other threads unless they are ordered by one of the above mentioned synchronizing instructions. Stores to storage marked as *guarded* and *caching inhibited* cannot be performed out-of-order (that is, they must be observed by other threads as always being performed in program order).

### 5.6.3.2 Atomicity

The Power10 core is fully compliant with the architectural requirement for single-copy atomicity on naturally aligned cacheable storage accesses. This includes the quadword data atomicity associated with the **lq**, **lqarx**, **stq**, and **stqcx** instructions. Additional information regarding which instruction accesses are performed atomically are described in *Section 5.1.5.1 Fixed-Point Load and Store Multiple Instructions* on page 79 and *Section 5.3.3 Floating-Point Load and Store Instructions* on page 84.

The remaining support for transactional memory accesses in the Power10 processor adds nothing to further in the way of atomic memory access functionality. See *Section 5.6.4 Transactional Memory (TM)* on page 95.



### 5.6.3.3 Atomic Updates and Reservations

Atomic accesses can be performed using the load and reserved and store conditional family of instructions. In the *Power ISA Virtual Environment Architecture - Book II (version 3.1B)*, the load and reserve instructions are: **Ibarx**, **Iharx**, **Iwarx**, **Idarx**, and **Iqarx**. The store conditional instructions are: **stbcx.**, **sthcx.**, **stwcx.**, **stdcx.**, and **stqcx**. While these instructions differ in the size of the operand read or written, in regards to the establishment or clearing of a reservation, all of the instructions operate on the same size reservation granule. The reservation granule size in the Power10 core is 128 bytes. There is at most one reservation per thread at any point in time.

**Note:** The load and reserve instructions and store conditional instructions are generically referred to as **lарx** and **stcx** in the remainder of this document.

### 5.6.4 Transactional Memory (TM)

The *Power ISA (Version 3.1B)* removes support for the transactional memory (TM) facility which was implemented in the POWER8 and POWER9 processors. As such, the Power10 processor also removes support for *Power ISA (Version 3.1B)* compliant workloads to use the transactional memory facility. That being said, the Power10 processor provides a means for software written for the POWER8 and POWER9 families of processors which implemented the *Power ISA (Versions 2.07B and 3.0C)* respectively to run unaltered thereby maintaining binary compatibility. The mechanisms to accomplish this are as follows:

- *Power ISA (Version 3.1B)* compliant software: The hypervisor software will make the transactional memory facility unavailable by setting the Hypervisor Facility Unavailable Register (HFSCR) bit 58 to '0'. Attempts to execute any TM instructions will result in behavior consistent with what is described in the *Illegal Instructions* appendix in the *Power ISA (Version 3.1B)*.
- *Power ISA (Versions 2.07B and 3.0C)* compliant software: The hypervisor software will make the transactional memory facility available by setting the HFSCR bit 58 to '1' and a new lightweight version of TM, referred to as Synthetic Transactional Memory (STM), is employed.
- A reduction in the transactional states supported to only non-transactional and synthetic suspend states. The non-transactional state is defined in the *Power ISA (Version 2.07B and 3.0C)* as the MSR[TS] = '00' state while suspend state is defined in the same documents as the MSR[TS] = '01' state. The transactional state defined as MSR[TS] = '10' is not supported on the Power10 processor and the hardware prevents privileged and problem state code from entering this mode. Hypervisor and ultravisor software must avoid executing any instruction or sequence of instructions that will result in the processor entering the MSR[TS] = '10' state.



#### 5.6.4.1 Synthetic TM (STM)

Synthetic TM (STM) consists of two main components for handling different situations that can occur when legacy transactional memory software attempts to execute on the Power10 processor:

- Software that attempts to start a new transaction on the Power10 processor by executing the **tbegin** instruction will immediately observe transaction failure recording and handling without ever entering the transactional state. When this occurs, execution will continue at the address pointed to by the Transaction Failure Handler Address Register (TFHAR) with the Transaction Execution and Status Register (TEXASR) indicating an implementation-specific failure cause (that is, TEXASR[15] = '1') with the persistent bit set (that is, TEXASR[7] = '1') in the non-transactional state (MSR[TS] = '00'). The Transaction Failure Instruction Address Register (TFIAR) points to the same address as the TFHAR. Note that the previously described TM instructions, registers, and terminology are otherwise implemented as described in the *Power ISA (Versions 2.07B and 3.0C)*. If software attempts to execute a **tbegin** instruction in the non-transactional state when the Power10 processor is in single-step trace mode (that is, MSR[TE] = '10'), the **tbegin** instruction will not take a trace interrupt and the transaction immediately fails without incurring a transactional state change (that is, the thread will remain in a non-transactional mode).
- Software that is migrated from a POWER8 or POWER9 processor to a Power10 processor as part of a live partition mobility (LPM) migration while in the suspended state (as defined in the *Power ISA [Versions 2.07B and 3.0C]* respectively), will initiate execution on the Power10 processor in what is referred to as the “synthetic suspended” state with TDOOMED set to ‘1’, indicating that the transaction will immediately fail once software attempts to resume that transaction on the Power10 processor. See details regarding synthetic suspend state in *Section 5.6.4.2 Synthetic Suspend State Details*.

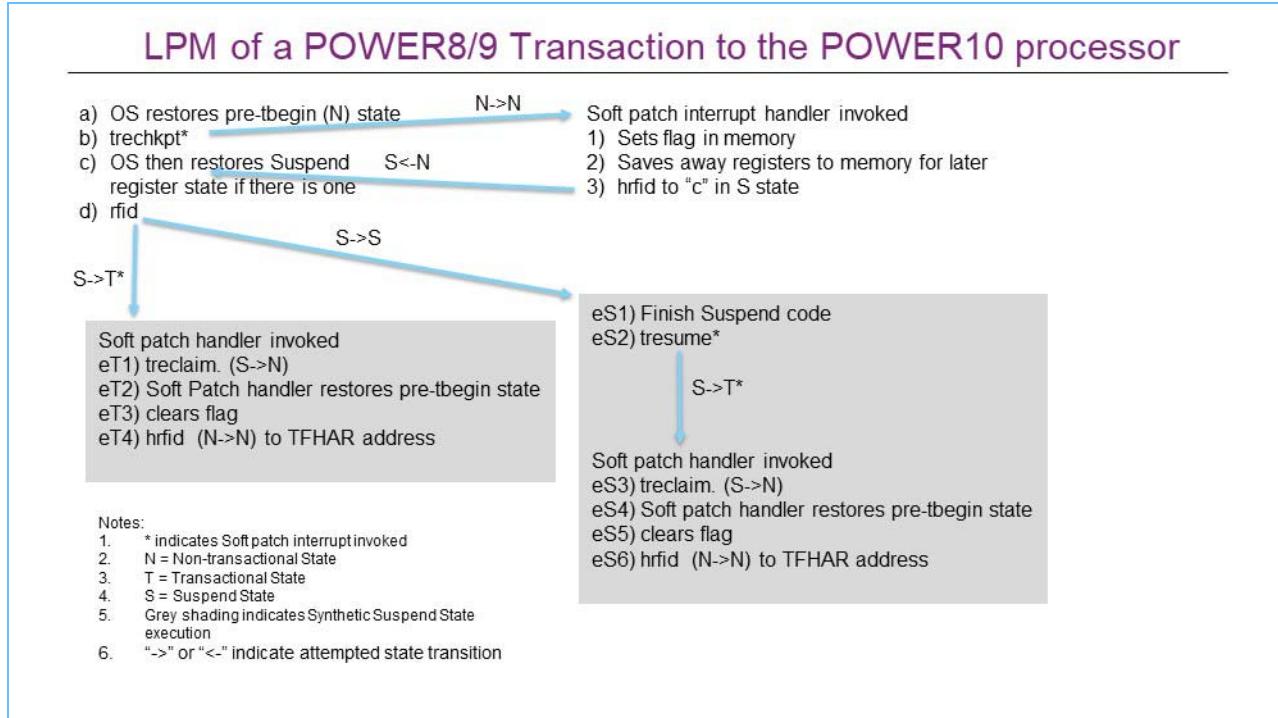
#### 5.6.4.2 Synthetic Suspend State Details

To applications and operating systems, the Power10 processor supports the suspended state as defined in the *Power ISA (Versions 2.07B and 3.0C)*. However, to hypervisor software, a number of deviations are implemented in the Power10 processor. These deviations involve:

- Trapping via a soft-patch interrupt on certain instructions when executed in either problem state or privileged state. In the soft patch interrupt handler, the hypervisor or ultravisor software decides whether to execute the trapped instruction normally or to perform an alternative code sequence.
- Modifications to the **hrfid** and **urfid** instructions to enable hypervisor or ultravisor software respectively to enter the suspended state directly from the non-transactional state. When a suspended state is entered via the **hrfid** and **urfid** instructions in this way, the thread is defined as being in the “synthetic suspended” state.

Except as described in the per-instruction hardware deviations in *Table B Transactional Memory Instruction Effects* on page 381, synthetic suspended state semantics are the same as described in the *Power ISA (Version 2.07B)* and *Power ISA (Version 3.0C)* sections on “suspend mode”. See *Figure 5-1* on page 97 for a high-level illustration of migrating a live partition involving an active or suspended transaction from a POWER8 or POWER9 processor to the Power10 processor.

Figure 5-1. LPM of a POWER8/POWER9 Transaction to the Power10 Processor



#### 5.6.4.3 TDOOMED

The *Power ISA (Version 3.0C)* defines the TDOOMED bit to be an indication as to whether or not a given transaction has failed. The value of the TDOOMED bit is determined by executing the **tcheck** instruction. The Power10 core always returns a value of TDOOMED = '1'.

#### 5.6.4.4 Implementation-Specific Failure Causes

The Transaction Exception and Status Register (TEXASR) is used to record various failure conditions that are described in the *Power ISA (Versions 2.07B and 3.0C)*. In addition, TEXASR[15] is used to specify various implementation-specific transaction-failure causes that are not architected. The Power10 processor sets TEXASR[15] = '1' for the following reason (implementation-specific transactional failure causes):

- The thread attempted to execute a **tbegin** instruction while in the non-transactional state.

The persistent bit is set to '1' to indicate the transaction should not be retried.

#### 5.6.5 Storage Ordering/Barrier Instructions

##### 5.6.5.1 sync Instruction

The Power10 design achieves high performance by exploiting speculative out-of-order instruction execution. The heavyweight sync (**hwsync**) instruction, as defined in the architecture, acts as a serious barrier to this type of aggressive execution and therefore, can have a dramatic effect on performance. Although the Power10 core has optimized the performance of **hwsync**, care should be exercised in the indiscriminate use



of this instruction. As a performance consideration, software should attempt to use the lightweight version of **sync** (often referred to as **lwsync** in this document) whenever possible. Unless otherwise stated, **sync** refers to **hwsync**.

The Power10 core implements the **ptesync** for use in synchronizing both segment and page table updates as described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*.

For ordering accesses to persistent storage, the Power10 core implements the persistent storage forms of **hwsync** and **lwsync**, respectively **phwsync** and **plwsync**, as described in the *Power ISA (Version 3.1B)*.

See the *Power ISA Virtual Environment Architecture - Book II (version 3.1B)* and *Power ISA Operating Environment Architecture - Book III (version 3.1B)* for a complete description of the different forms of the **sync** instruction.

#### 5.6.5.2 **eieio** Instruction

The Power10 core implements the **eieio** instruction as described in the *Power ISA (Version 3.1B)*.

In the Power10 nest logic, the store queues above the L2 cache attempt to gather sequential cacheable and caching-inhibited store operations to improve bandwidth. If this behavior is not desired, software must insert either an **eieio** (preferable for performance) or a **sync** to prevent it.

#### 5.6.5.3 **miso** Instruction

The Power10 core implements the **miso** instruction by closing gathering for all previous stores.

#### 5.6.5.4 Return Oriented Programming (ROP) Support

The Power10 core supports both the non-privileged and privileged forms of the hash-based pointer authentication instructions as described in the *Power ISA (Version 3.1B)*. Supported instructions include the following: **hashst[p]** and **hashchk[p]**.

### 5.6.6 Data Prefetch

The Power10 core provides a hardware-based, data-prefetching engine that is designed to work well for stride-one and stride-N access patterns with up to 16 streams depending on the demands of other active threads in the SMT4-core-resource. The 16 streams can be dynamically shared among all of the threads. The Power10 core implements **dcbt** and **dcbtst** instructions as described in *Section 5.6.2.5 Data Cache Block Touch Instructions (dcbt and dcbtst)* on page 90, where each thread can employ up to 16 software-initiated streams in ST mode, eight in SMT2 mode, and four in SMT4 mode.

The Power10 core supports instruction cache block touch (**icbt**) by mapping it to **dcbtst** to prefetch into the L3 cache.

The Power10 core also allows problem state access to DSCR[58], which turns off hardware load-stream prefetching.

## 5.6.7 Timer Facilities

### *Time Base*

Time base is designed to tick at the rate of time-of-day (TOD). In other words, bit 59 of the Time-Base Register increments at the 32 MHz clock. There is one time base per LPAR that is shared by all the threads, running on a core.

### *Time Facility Management Register*

The Time Facility Management Register (TFMR) is an SPR that is accessible only in the hypervisor state. Executing a move to or move from TFMR in a nonhypervisor state causes a privileged interrupt. There is one TFMR per processor core that is shared among the threads. The TFMR is used as both a status and control register.

### *Power10 Time-Base Mode*

The time-base function uses an external time-of-day (TOD) clock, which is independent of the processor frequency. This is required to support dynamic frequency variation for power management. The external TOD oscillator operates at 32 MHz. The external TOD oscillator is sampled to provide a 32 MHz step signal, which is distributed to all processors in the system.

Bits 0:59 of the TB are incremented at the 32 MHz frequency as provided by the distributed step signal. Bits 60:63 of the TB are incremented at a fixed frequency of 500 MHz. If the value of bits 60:63 is '1111' (saturated), it is held until the 32 MHz step signal causes bit 59 to change. At that time, bits 60:63 are allowed to change to '0000'.

To support multi-node configurations across multiple oscillator domains, error detection and recovery, and concurrent maintenance, the Power10 core uses the following means of synchronizing the time bases across all processors. Each multicore processor chip contains a Time-of-Day (TOD) Register. The chip TOD registers are first synchronized across all the processor chips. Then, the time-base registers in each processor core are synchronized to the chip TOD. Also encoded on the step signal is a synchronization pulse that is used for synchronization and error checking. The synchronization mechanism requires system operations to complete within a synchronization interval. The synchronization interval can be set via the TFMR bits to be, for example, 1 μs, 2 μs (default, corresponds to TB bit 53), 4 μs, or 8 μs.

Error checking includes parity checks on all registers, and functional checking such as missing step signal detection and synchronization errors. The step signal rate is defined in the Power10 mode to be 32 MHz, and the logic checks for the correct number of steps for each synchronization signal (which is selected by TFMR). After the TB is operational, the hardware also detects a missing step signal, which requires specifying in the TFMR the maximum number of processor cycles allowed without seeing a 32 MHz step signal, for the fastest allowable operating frequency. The TFMR maximum cycle step timeout should be specified as  $(2 \times 31.25 \text{ ns}) / (\text{minimum processor cycle time in ns} \times 4)$ .

The initial synchronization requires some software sequencing, which is performed by writing values to the TFMR (via **mtspr**). The TFMR also indicates the status of the various time facilities. The status bits in the TFMR are read-only, not modified by **mtspr** to the TFMR. The time facility logic implements error detection for hardware and also for invalid software sequencing. Because synchronized time is critical to a system, writes to the time base or the TFMR that would break synchronization cause the logic to enter an error state and trigger a hypervisor maintenance interrupt.



To initially set the time and synchronize the time-base values, software must synchronize all processors in the system and choose one processor to perform updates to the TFMR via a read-modify-write operation to preserve the other bits. This sequence assumes the external TOD oscillator distribution is already running.

After the chip TOD is running on all chips and the TB is running on the processor that drove this sequence, software must then release the remaining processors to synchronize their TB registers to their corresponding chip TOD.

### 5.6.8 Hypervisor Decrementer (HDEC)

There is one Hypervisor Decrementer Register per LPAR. HDEC decrements every time TB bit 63 is incremented. The *Power ISA (Version 3.1B)* defines the HDEC as a 64-bit register architecturally but states that a given implementation can implement fewer than 64 bits. The Power10 core implements 56 bits (bits 8:63) of the HDEC register. The number of bits is not changeable by the LPCR[LD] value.

### 5.6.9 Decrementer (DEC)

There is one Decrementer Register per thread. The DEC decrements every time TB bit 63 is incremented. The *Power ISA (Version 3.1B)* defines the DEC as a 64-bit register architecturally but states that a given implementation can implement fewer than 64 bits. The Power10 core implements 56 bits (bits 8:63) of the DEC Register. The number of bits used in determining when a decrementer interrupt occurs is 56 bits when the LPCR[LD] = '1' but only 32 bits when LPCR[LD] = '0'.

### 5.6.10 Book II Invalid Forms

The results of executing an invalid form of an instruction in Book II or an instance of such an instruction for which the architecture specifies that some results are undefined, are described here for the cases when executing an instruction does not cause an exception. Only results that differ from those specified by the architecture are described in the following list. Software should avoid these forms since their behavior is subject to possible change on future Power processor generations.

- Instruction with reserved fields  
Bits in reserved fields are ignored; the results of executing an instruction in which one or more reserved bits are '1' is the same as if the bits were '0'.
- Transactional memory instructions and store conditional instructions (bit 31 is ignored)  
Bit 31 of **tbegin.**, **tend.**, **tabort.**, **tabortwc.**, **tabortdci.**, **tabortwci.**, **tabortdci.**, **treclaim.**, **trechkpt.**, **tsr.**, **stbcx.**, **sthcx.**, **stwcx.**, **stdcx.** and **stqcx.** is ignored. Because these x-form instructions do not have a non-record form variant, ignoring bit 31 is an acceptable way to handle this invalid form.
- **mftb** instruction  
This instruction produces the same result as the **mfspcr** instruction.

## 5.7 Book III - Operating Environment Architecture

### 5.7.1 Classes of Instructions

#### 5.7.1.1 Storage Control Instructions

The Power10 core provides support for the following instructions:

- **tlbie** - TLB invalidate entry
- **tlbiel** - Processor local form of TLB invalidate entry
- **tlbsync** - TLB synchronize
- **slbmte** - Segment lookaside buffer move to entry
- **slbmfv** - Segment lookaside buffer move from entry VSID
- **slbmfe** - Segment lookaside buffer move from entry ESID
- **slbfee.** - Segment lookaside buffer find entry ESID
- **slbie** - SLB invalidate entry
- **slbieg** - SLB invalidate global
- **slbsync** - SLB synchronize
- **slbia** - SLB invalidate all
- **slbiag** - SLB invalidate global
- **mtmsr** - Move to Machine State Register (32-bit)
- **mtmsrd** - Move to Machine State Register (64-bit)
- **sc** - System call
- **scv** - System call vectored
- **rfscv** - Return from system call vectored
- **rfid** - Return from interrupt doubleword
- **hrfid** - Hypervisor return from interrupt doubleword
- **urfid** - Ultravisor return from interrupt doubleword

The Power10 core does not provide support for the following optional or obsolete instructions (attempted use of these results in a hypervisor emulation assistance interrupt):

- **tlbia** - TLB invalidate all
- **tlbiex** - TLB invalidate entry by index (obsolete)
- **slbiex** - SLB invalidate entry by index (obsolete)
- **dcba** - Data cache block allocate (Book II; obsolete)
- **dcbi** - Data cache block invalidate (obsolete)
- **rfi** - Return from interrupt (32-bit; obsolete)

The following instruction variants are implemented:

- **ptesync** - Page table synchronize
- **hwsync** - Heavyweight synchronize
- **lwsync** - Lightweight synchronize
- **phwsync** - Persistent heavyweight synchronize
- **plwsync** - Persistent lightweight synchronize

### 5.7.1.2 Reserved Instructions

The architecture breaks the reserved instruction class down into several categories as described in the *Reserved Instructions* appendix of the *Power ISA (Version 3.1B)*. The Power10 core treats three of these cases as resulting in a hypervisor emulation assistance interrupt:

- The primary opcode of zero is treated as an illegal instruction.
- The Power Architecture® instructions not supported in the *Power ISA (Version 3.1B)*. See a complete list in the *Power ISA (Version 3.1B)* appendices.
- The service processor “Attention” instruction is treated as an illegal instruction unless HID[en\_attn] = ‘1’.

In addition, there are several implementation-specific registers available for access through the **mtspr** and **mfsp** instructions. These are described in *Section 5.7.3.5 Move To/From Special Purpose Register Instructions* on page 104.

## 5.7.2 Branch Processor

### 5.7.2.1 SRR1 Register

In the Power10 processor core, the SRR1 is implemented per the *Power ISA (Version 3.1B)*.

### 5.7.2.2 HSRR1 Register

In the Power10 processor core, the HSRR1 is implemented per the *Power ISA (Version 3.1B)*.

### 5.7.2.3 USRR1 Register

In the Power10 core, the USRR1 is implemented per the *Power ISA (Version 3.1B)*.

### 5.7.2.4 MSR Register

In the Power10 core, the MSR is implemented per the *Power ISA (Version 3.1B)*. All reserved bits should be set to ‘0’ by software.

### 5.7.2.5 System Call and System Call Vectored Instructions

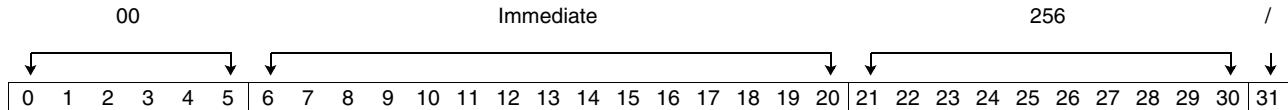
In the Power10 core, the system call (**sc**) and system call vectored (**scv**) instructions are implemented as described in *Table 5-4* using primary opcode 17. For that primary opcode, the opcode bits 30:31 determine whether the instruction is a system call or system call vectored instruction as described in *Table 5-4*.

*Table 5-4. System Call and System Call Vectored Invocation*

Opcode Bits [30:31]	Description
‘00’	<b>sc</b> instruction
‘01’	<b>scv</b> instruction
‘10’	<b>sc</b> instruction
‘11’	<b>sc</b> instruction

### 5.7.2.6 Support Processor Attention Instruction

The Power10 processor core supports a special, implementation-dependent instruction for signaling an attention to the support processor.



The immediate field has no effect on the operation of this instruction in the Power10 processor core.

If HID[3] = '1' (support-processor attention enable bit is set), this instruction causes all preceding instructions to run to completion, the machine to quiesce, and the assertion of the support processor attention signal. Instruction execution does not resume until the support processor signals it to do so. When setting HID[3] = '1', the I-cache must be flushed by setting HID[2] so that all Attention instructions in the I-cache see the effect of enabling the Attention Enable bit.

If HID[3] = '0' (support-processor attention enable not set), this instruction causes a hypervisor emulation assistance interrupt. Note that due to some design features unique to the Power10 core, when the hypervisor emulation assistance interrupt occurs, the instruction opcode saved in the Hypervisor Emulation Assistance Interrupt Register (HEIR) is slightly modified and appears as x'001E0200' when read.

### 5.7.2.7 Current Instruction Address Breakpoint Register (CIABR)

The Power10 processor core supports the CIAB Register as implemented per the *Power ISA (Version 3.1B)*.

## 5.7.3 Fixed-Point Processor

### 5.7.3.1 Processor Version Register (PVR)

The Process Version Register (PVR) is a 32-bit register that contains the version and revision level information for the Power10 core. *Table 5-6* summarizes how to interpret the PVR.

Table 5-5. PVR

Bits	Field Name	Description
32:47	Version	Processor version number. The processor version number for the Power10 core is x'0080'.
48:50	Reserved	<b>Note:</b> Read as zeros.
51	Chip Type	Power10 chip scaling factor. 0      8-threaded cores 1      4-threaded cores
52:55	Revision (major)	Major revision level. The major processor revision level starts at x'1', indicating major revision '1'. Subsequent major revisions will be x'2', x'3', and so on.
56:59	Revision (reserved)	Read as zero.
60:63	Revision (minor)	Minor revision level. Minor revisions are indicated in PVR[60:63]. Each major revision will reset the minor revision field to x'00' and each minor revision will increment PVR[60:63] by x'01'.

Example: The PVR value for the 12-core Power10 chip for design revision level 1.0 is: x'00800100'.



### 5.7.3.2 Processor ID Register (PIR)

The Processor Identification Register (PIR) is a 64-bit register that holds a processor identification tag in the least-significant bits. This tag is used for tagging bus transactions and for processor differentiation in multiprocessor systems.

Table 5-6 shows how to interpret the PIR values.

Table 5-6. PIR

Bits	Field Name	Description
0:48	Reserved	Read as zeros
49:51	Spare ID	Read as zeros.
52:55	Topology ID	Topology ID
56	Spare Quad ID	Spare Quad ID
57:59	Quad ID	Quad ID
60	Core Chiplet Pair ID	Core Chiplet Pair ID.
61:63	Thread/Core Chiplet ID t0 - t2	Thread/Core Chiplet ID t0 - t2

The PIR is a read-only register. During power-on reset, PIR is set to a unique value for each processor in a multiprocessor system.

### 5.7.3.3 Chip Information Register (CIR)

The Power10 chip does not support the CIR, which was described in earlier versions of the Power ISA. It was not being used; therefore, it was removed in the Power10 chip and the *Power ISA (Version 3.1B)*.

### 5.7.3.4 Thread ID Register (TIDR)

The Power10 core does not support the TIDR, which was described in earlier versions of the Power ISA.

### 5.7.3.5 Move To/From Special Purpose Register Instructions

The Power10 core supports the SPRs listed in Table 5-7 on page 106. Many of these SPRs are only accessible in hypervisor or privileged modes. Additionally, some SPRs are only accessible in ultravisor privileged mode when the optional Secure Memory Facility (SMF) is supported and enabled. See Section 18.1 *Secure Memory Facility* on page 238. A handful of these registers (for example, DSCR) are also user-mode accessible through a second SPR number.

To support multithreading, some of the SPRs are replicated on a per thread or per LPAR basis in the Power10 core, while others are shared on a per core basis, as shown in the Thread/LPAR Replica column in Table 5-7. See Section 4.2 *Logical Partition Specific Resources* on page 58 and Section 4.3 *Core-Specific Resources* on page 59 for additional information.

In the table column headers:

- Prob indicates problem state ( $S = x$ ,  $HV = x$ ,  $PR = 1$ )
- Priv indicates privileged state ( $S = x$ ,  $HV = 0$ ,  $PR = 0$ )



- Hyp indicates hypervisor state ( $S = 0$ ,  $HV = 1$ ,  $PR = 0$ )
- UV indicates ultrvisor state ( $S = 1$ ,  $HV = 1$ ,  $PR = 0$ ).

In the SPR-specific rows:

- Priv indicates that a privileged instruction type program interrupt will occur in that state for the attempted read or write of the SPR.
- Illegal indicates a hypervisor emulation assistance interrupt will occur.
- NOP indicates the instruction will be treated as a NOP.
- A blank under each column indicates the access will be performed normally.

In the “Affected by PCR Bit” column, the value specified in that column indicate which PCR bit affects that SPR access. See *Section 5.10.33 Processor Compatibility Mode* on page 167 for details on the PCR definition.



**Table 5-7. SPR Table (Sheet 1 of 9)**

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
FPSCR			Per Thread											
VSCR			Per Thread											
MSR			Per Thread											
Special SPR 0	00000 00000	0			HEA	HEA	HEA	HEA	HEA	HEA	HEA	HEA		
XER	00000 00001	1	Per Thread	64										
Reserved (MTMSR)	00000 00010	2												
DSCR_RU (HFSCR[61] = 0)	00000 00011	3	Per Thread	25	HFAC Unavail				HFAC Unavail					
DSCR_RU (HFSCR[61]=1 and FSCR[61] = 0)	00000 00011	3	Per Thread	25	FAC Unavail				FAC Unavail					
DSCR_RU (FSCR[61] = 1)	00000 00011	3	Per Thread	25										
Special SPR 4	00000 00100	4			HEA	HEA	HEA	HEA	HEA	HEA	HEA	HEA		
Special SPR 5	00000 00101	5			HEA	HEA	HEA	HEA	HEA	HEA	HEA	HEA		
Special SPR 6	00000 00110	6			HEA	HEA	HEA	HEA	HEA	HEA	HEA	HEA		
LR	00000 01000	8	Per Thread	64										
CTR	00000 01001	9	Per Thread	64										
AMR (also known as UAMR)	00000 01101	13	Per Thread	64										
DSCR (HFSCR[61] = 0)	00000 10001	17	Per Thread	25	Priv	FAC Unavail			Priv	FAC Unavail				
DSCR (HFSCR[61] = 1)	00000 10001	17	Per Thread	25	Priv				Priv					
DSISR	00000 10010	18	Per Thread	32	Priv				Priv					
DAR	00000 10011	19	Per Thread	64	Priv				Priv					
DEC	00000 10110	22	Per Thread	32	Priv				Priv					



Table 5-7. SPR Table (Sheet 2 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
SRR0	00000 11010	26	Per Thread	64	Priv				Priv					
SRR1	00000 11011	27	Per Thread	64	Priv				Priv					
CFAR	00000 11100	28	Per Thread	64	Priv				Priv					
AMR	00000 11101	29	Per Thread	64	Priv				Priv					
PIDR	00001 10000	48	Per Thread	32	Priv				Priv					
IAMR	00001 11101	61	Per Thread	32	Priv				Priv					
Reserved (BHRB)	00010 00000 00010 11111	64-95												
TFHAR	00100 00000	128	Per Thread	64										
TFIAR	00100 00001	129	Per Thread	64										
TEXASR	00100 00010	130	Per Thread	64										
TEXASRU	00100 00011	131	Per Thread	32										
CTRL_RU	00100 01000	136	Per Core	32	Return Bit 63 Only				HEA	Nop_ev	Nop_ev	Nop_ev		
CTRL	00100 11000	152	Per Core	32	Priv	Nop_ev	Nop_ev	Nop_ev	Priv					
FSCR	00100 11001	153	Per Thread	64	Priv				Priv					
UAMOR	00100 11101	157	Per Thread	64	Priv				Priv					
PSPB	00100 11111	159	Per Thread	32	Priv				Priv					
DPDES	00101 10000	176	Per Core	8	Priv				Priv	Priv_ev				
DAWR0	00101 10100	180	Per Thread	64	Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1		Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1			



Table 5-7. SPR Table (Sheet 3 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
DAWR1	00101 10101	181	Per Thread	64	Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1		Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1			
RPR	00101 11010	186	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				Y
CIABR	00101 11011	187	Per Thread	64	Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1		Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1			
DAWRX0	00101 11100	188	Per Thread	32	Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1		Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1			
DAWRX1	00101 11101	189	Per Thread	32	Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1		Priv	Priv_ev	HEA w HSSR1[45] SMFCTRL[D] = 1			
HFSCR	00101 11110	190	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
VRSAVE	01000 00000	256	Per Thread	32										
SPRG3_RU	01000 00011	259	Per Thread	64					HEA	Nop_ev	Nop_ev	Nop_ev		
TB	01000 01100	268	Per LPAR	64					HEA	Nop_ev	Nop_ev	Nop_ev		
TBU_RU	01000 01101	269	Per LPAR	32					HEA	Nop_ev	Nop_ev	Nop_ev		
SPRG0	01000 10000	272	Per Thread	64	Priv				Priv					
SPRG1	01000 10001	273	Per Thread	64	Priv				Priv					
SPRG2	01000 10010	274	Per Thread	64	Priv				Priv					
SPRG3	01000 10011	275	Per Thread	64	Priv				Priv					
SPRC	01000 10100	276	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
SPRD	01000 10101	277	n/a (physical target controlled by SPRC)	64	Priv	Priv_ev			Priv	Priv_ev				
TBL	01000 11100	284	Per LPAR	32	Priv	Nop_ev	Nop_ev	Nop_ev	Priv	Priv_ev				



Table 5-7. SPR Table (Sheet 4 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read (mfspr) (see note 1)				Write (mtspr) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
TBU	01000 11101	285	Per LPAR	32	Priv	Nop_ev	Nop_ev	Nop_ev	Priv	Priv_ev				
TBU40	01000 11110	286	Per LPAR	64	Priv	Nop_ev	Nop_ev	Nop_ev	Priv	Priv_ev				
PVR	01000 11111	287	Per Core	32	Priv				Priv	Nop_ev	Nop_ev	Nop_ev		
HSPRG0	01001 10000	304	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HSPRG1	01001 10001	305	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HDSISR	01001 10010	306	Per Thread	32	Priv	Priv_ev			Priv	Priv_ev				
HDAR	01001 10011	307	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
SPURR	01001 10100	308	Per Thread	64	Priv				Priv	Priv_ev				
PURR	01001 10101	309	Per Thread	64	Priv				Priv	Priv_ev				
HDEC	01001 10110	310	Per LPAR	32	Priv	Priv_ev			Priv	Priv_ev				
HRMOR	01001 11001	313	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				Y
HSRR0	01001 11010	314	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HSRR1	01001 11011	315	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
TFMR	01001 11101	317	Per Core/ LPAR bits 26 and 45 replicated	64	Priv	Priv_ev			Priv	Priv_ev				
LPCR	01001 11110	318	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
LPIDR	01001 11111	319	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HMER	01010 10000	336	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HMEER	01010 10001	337	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				
PCR	01010 10010	338	Per LPAR	64	Priv	Priv_ev			Priv	Priv_ev				Y



**Table 5-7. SPR Table (Sheet 5 of 9)**

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
HEIR	01010 10011	339	Per Thread	32	Priv	Priv_ev			Priv	Priv_ev				
AMOR	01010 11101	349	Per LPAR	64	Priv	Priv_ev			Priv	Priv_ev				Y
TIR	01101 11110	446	Per Thread	8	Priv				Priv	Nop_ev	Nop_ev	Nop_ev		
HDEXCR_RU	01110 00111	455	Per Thread	32					HEA	Nop_ev	Nop_ev	Nop_ev	v3.0	
PTCR	01110 10000	464	Per Core	64	Priv	Priv_ev			Priv	Priv_ev	HEA with HSSR1[45] SMFCTRL[E] = 1			Y
HASHKEY	01110 10100	468	Per Thread	64	Priv				Priv					
HASHPKEY	01110 10101	469	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
HDEXCR	01110 10111	471	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
Reserved	01110 11100	476												
Reserved	01110 11101	477												
Reserved	01110 11110	478												
Reserved	01110 11111	479												
USPRG0	01111 10000	496	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
USPRG1	01111 10001	497	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
URMOR	01111 11001	505	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
USRRO	01111 11010	506	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
USRRI	01111 11011	507	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
SMFCTRL	01111 11111	511	Per Thread	64	Priv	Priv	Priv		Priv	Priv	Priv			
SIER2_RU	10111 00000	736	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop	v3.0	



Table 5-7. SPR Table (Sheet 6 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfsp</b> r) (see note 1)				Write ( <b>mtsp</b> r) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
SIER3_RU	10111 00001	737	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop	v3.0	
MMCR3_RU	10111 00010	738	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop	v3.0	
SIER2	10111 10000	752	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
SIER3	10111 10001	753	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
MMCR3	10111 10010	754	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
SIER_RU	11000 00000	768	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop		
MMCR2_RU	11000 00001	769	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
MMCRA_RU	11000 00010	770	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC1_RU	11000 00011	771	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC2_RU	11000 00100	772	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC3_RU	11000 00101	773	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC4_RU	11000 00110	774	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC5_RU	11000 00111	775	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
PMC6_RU	11000 01000	776	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
MMCR0_RU	11000 01011	779	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8				
SIAR_RU	11000 01100	780	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop		
SDAR_RU	11000 01101	781	Per Thread	64	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop		
MMCR1_RU	11000 01110	782	Per Thread	32	See Table 5-8	See Table 5-8			See Table 5-8	See Table 5-8	Nop	Nop		
SIER	11000 10000	784	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
MMCR2	11000 10001	785	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				



**Table 5-7. SPR Table (Sheet 7 of 9)**

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
MMCRA	11000 10010	786	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
PMC1	11000 10011	787	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
PMC2	11000 10100	788	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
PMC3	11000 10101	789	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
PMC4	11000 10110	790	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
PMC5	11000 10111	791	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
PMC6	11000 11000	792	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
MMCR0	11000 11011	795	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
SIAR	11000 11100	796	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
SDAR	11000 11101	797	Per Thread	64	Priv	See Table 5-8			Priv	See Table 5-8				
MMCR1	11000 11110	798	Per Thread	32	Priv	See Table 5-8			Priv	See Table 5-8				
IMC	11000 11111	799	Per Core	64	Priv	Priv_ev			Priv	Priv_ev	HEA with HSSR[45] SMFCTRL[E] = 1			Y
BESCRS	11001 00000	800	Per Thread	64										
BESCRSU	11001 00001	801	Per Thread	32										
BESCRR	11001 00010	802	Per Thread	64										
BESCRRU	11001 00011	803	Per Thread	32										
EBBHR	11001 00100	804	Per Thread	64										
EBBRR	11001 00101	805	Per Thread	64										
BESCR	11001 00110	806	Per Thread	64										



Table 5-7. SPR Table (Sheet 8 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read (mfspr) (see note 1)				Write (mtspr) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
Reserved	11001 01000	808	NA		Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	
Reserved	11001 01001	809	NA		Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	
Reserved	11001 01010	810	NA		Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	
Reserved	11001 01011	811	NA		Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	Nop	
DEXCR_RU	11001 01100	812	Per Thread	32					HEA	Nop_ev	Nop_ev	Nop_ev	v3.0	
TAR	11001 01111	815	Per Thread	64										
ASDR	11001 10000	816	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
PSSCR_SU	11001 10111	823	Per Thread	64	Priv				Priv					
Reserved (MTXER)	11001 11001	825												
Reserved (MFNIA)	11001 11010	826												
DEXCR	11001 11100	828	Per Thread	64	Priv				Priv					
IC	11010 10000	848	Per Thread	64	Priv				Priv	Priv_ev				
VTB	11010 10001	849	Per LPAR	64	Priv				Priv	Priv_ev				
LDBAR	11010 10010	850	Per LPAR	64	Priv	Priv_ev			Priv	Priv_ev	HEA with HSSR1[45] SMFCTRL[E] = 1		Y	
MMCRC	11010 10011	851	Per Core	32	Priv	Priv_ev			Priv	Priv_ev				Y
PMSR	11010 10101	853	Per Core	32	Priv	Priv_ev			Priv	Nop_ev	Nop_ev	Nop_ev	Nop_ev	
PSSCR	11010 10111	855	Per Thread	64	Priv	Priv_ev			Priv	Priv_ev				
L2QOSR	11010 11101	861	Per Core		Priv	Nop_ev	Nop_ev	Nop_ev	Priv	Priv_ev				
TRIG1	11011 10001	881	Per Thread	64	Priv	Nop_ev	Nop_ev	Nop_ev	Priv					

Table 5-7. SPR Table (Sheet 9 of 9)

SPR Name	Binary SPR Code	Decimal SPR Code	Replicated	Length (bits)	Read ( <b>mfspr</b> ) (see note 1)				Write ( <b>mtspr</b> ) (see note 2)				Affected by PCR Bit	For an SMT8 core, must set from each core chiplet
					Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0	Prob S = x HV = x PR = 1	Priv S = x HV = 0 PR = 0	Hyp S = 0 HV = 1 PR = 0	UV S = 1 HV = 1 PR = 0		
TRIG2	11011 10010	882	Per Thread	64	Priv	Nop_ev	Nop_ev	Nop_ev	Priv					
PMCR	11011 10100	884	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				
RWMR	11011 10101	885	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				
WORT	11011 11111	895	Per Thread	18	Priv				Priv					
PPR	11100 00000	896	Per Thread	64										
PPR32	11100 00010	898	Per Thread	32										
TSCR	11100 11001	921	Per Core	32	Priv	Priv_ev			Priv	Priv_ev				Y
TTR	11100 11010	922	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				Y
TRACE	11111 01110	1006	Per Core	64	HEA	Nop_ev	Nop_ev	Nop_ev						
HID	11111 10000	1008	Per Core	64	Priv	Priv_ev			Priv	Priv_ev				Y
PIR	11111 11111	1023	Per Thread	32	Priv				Priv	Nop_ev	Nop_ev	Nop_ev		
Unsupported SPRs with SPR(0) = 0	xxxxx 0xxxx				HEA	Nop_ev	Nop_ev	Nop_ev	HEA	Nop_ev	Nop_ev	Nop_ev		
Unsupported SPRs with SPR(0) = 1	xxxxx 1xxxx				Priv	Nop_ev	Nop_ev	Nop_ev	Priv	Nop_ev	Nop_ev	Nop_ev		





**Notes:**

1. Read (**mfsp**)

HEA = Hypervisor Emulation Assist (E40)

PRIV = Program (700)

Fac Unavail = (F60)

H Fac Unavail = (F80)

Nop\_ev = IF(EVIRT = 0), then nop;  
IF(EVIRT = 1), then HEA\_evirt.

Priv\_ev = IF(EVIRT = 0), then priv;

IF(EVIRT = 1), then HEA\_evirt.

**Note:** HSRR1[45] = 1, IF HEA\_evirt occurs and any of the following conditions are met:

- Resource is hypervisor/ultravisor controlled and accessed from HV = 0, PR = 0.
- Resource is ultravisor controlled and accessed from HV = 1, PR = 0, S = 0.

2. Write (**mtsp**)

HEA = Hypervisor Emulation Assist (E40)

PRIV = Program (700)

Fac Unavail = (F60)

H Fac Unavail = (F80)

Nop\_ev = IF(EVIRT = 0), then nop;  
IF(EVIRT = 1), then HEA\_evirt.

Priv\_ev = IF(EVIRT = 0) ,then priv;

IF(EVIRT = 1 ), then HEA\_evirt.

**Note:** HSRR1[45] = 1, if HEA\_evirt occurs and any of the following conditions are met:

- Resource is hypervisor/ultravisor controlled and accessed from HV = 0, PR = 0.
- Resource is ultravisor controlled and accessed from HV = 1, PR = 0, S = 0.

Table 5-8 lists the performance monitor sprs. It assumes access is allowed by PCR version. Interrupts in the following table are abbreviated as follows: facility unavailable interrupt (Fac Unavail), hypervisor emulation assistance interrupt (HEA), hypervisor facility unavailable interrupt (H Fac Unavail).

Table 5-8. Performance Monitor SPRs (Sheet 1 of 4)

Group	Performance Monitor SPRs	Affected by PCR Bit	Problem State										Problem State		Priv		Priv	
			HFSCR(60) = 1										HFSCR(60) = 0		HFSCR(60) = 1		HFSCR(60) = 0	
			READ					WRITE					READ	WRITE	READ	WRITE	READ	WRITE
PMCC = 00 and MMCR0(54) and !PCR (3.0)	PMCC = 00 and !MMCR0(54) and PCR (3.0)	PMCC = 00 and PCR (3.0)	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = 00	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	
Group A	spr_mmcr2_769		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_mmcr4_770		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Fac Unavail	Fac Unavail	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc1_771		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc2_772		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc3_773		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc4_774		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc5_775		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Fac Unavail	HEA	Fac Unavail	Allowed	Fac Unavail	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc6_776		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Fac Unavail	HEA	Fac Unavail	Allowed	Fac Unavail	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_mmcr0_779		Allowed	Allowed	Allowed	Fac Unavail	Allowed	Allowed	HEA	Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail	Allowed	Allowed	H Fac Unavail	H Fac Unavail





Table 5-8. Performance Monitor SPRs (Sheet 2 of 4)

Group	Performance Monitor SPRs	Affected by PCR Bit	Problem State										Problem State		Priv		Priv	
			HFSCR(60) = 1										HFSCR(60) = 0		HFSCR(60) = 1		HFSCR(60) = 0	
			READ					WRITE					READ	WRITE	READ	WRITE	READ	WRITE
			PMCC = 00 and MMCR0(54) and !PCR (3.0)	PMCC = 00 and !MMCR0(54) and !PCR (3.0)	PMCC = 00 and PCR (3.0)	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = 00	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX
Group B	spr_sier2_736	v3.0	Fac Unavail	Allowed	HEA	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_sier3_737	v3.0	Fac Unavail	Allowed	HEA	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_m_mcr3_738	v3.0	Fac Unavail	Allowed	HEA	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_sier_768		Fac Unavail	Allowed	Allowed	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_siар_780		Fac Unavail	Allowed	Allowed	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_sdar_781		Fac Unavail	Allowed	Allowed	Fac Unavail	Allowed	Allowed	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP
	spr_mmcr1_782		Fac Unavail	Allowed	Allowed	Fac Unavail	Fac Unavail	Fac Unavail	Priv				H Fac Unavail	Priv	Allowed	NOP	H Fac Unavail	NOP

Table 5-8. Performance Monitor SPRs (Sheet 3 of 4)

Group	Performance Monitor SPRs	Affected by PCR Bit	Problem State										Problem State		Priv		Priv		
			HFSCR(60) = 1										HFSCR(60) = 0		HFSCR(60) = 1		HFSCR(60) = 0		
			READ					WRITE					READ	WRITE	READ	WRITE	READ	WRITE	
			PMCC = 00 and MMCRO(54) and !PCR (3.0)	PMCC = 00 and !MMCRO(54) and !PCR (3.0)	PMCC = 00 and PCR (3.0)	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = 00	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	
			These are not affected by PMCC					These are not affected by PMCC											
Group A	spr_mmcr2_785			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_mmcr4_786			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc1_787			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc2_788			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc3_789			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc4_790			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc5_791			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_pmc6_792			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail
	spr_mmcr0_795			Priv					Priv					Priv	Priv	Allowed	Allowed	H Fac Unavail	H Fac Unavail





Table 5-8. Performance Monitor SPRs (Sheet 4 of 4)

Group	Performance Monitor SPRs	Affected by PCR Bit	Problem State								Problem State		Priv		Priv		
			HFSCR(60) = 1								HFSCR(60) = 0		HFSCR(60) = 1		HFSCR(60) = 0		
			READ				WRITE				READ	WRITE	READ	WRITE	READ	WRITE	
			PMCC = 00 and MMCR0(54) and !PCR (3.0)	PMCC = 00 and !MMCR0(54) and !PCR (3.0)	PMCC = 00 and PCR (3.0)	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = 00	PMCC = 01	PMCC = 10	PMCC = 11	PMCC = XX	PMCC = XX	PMCC = XX	PMCC = XX	
Group B	spr_sier2_752					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_sier3_753					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_mmcr3_754					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_sier_784					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_siар_796					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_sdar_797					Priv				Priv				Priv	Priv	Allowed	Allowed
	spr_mmcr1_798					Priv				Priv				Priv	Priv	Allowed	Allowed



## 5.8 HID Register

The Power10 processor core includes several implementation-dependent mode bits that allow various features of the chip to be enabled and disabled. These bits are included in the Hardware Implementation Dependent Register (HID). In general, the HID Register controls high-level functions of the Power10 core and is only accessible in hypervisor mode. Reserved bits in the HID Register should not be set by software and can return either a zero or one value depending on the bit if set. Attempts to set some of these bits might enable functions that are no longer supported and thus could cause unpredictable behavior. Two values of the contents of the register are shown in the descriptions.

Initial state:

This is the state of the register after a normal scan-based power-on-reset (POR). The actual and full POR sequence can set bits beyond the scan-based POR.

Preferred state:

This is the preferred state of the register for optimal performance and function.

The following sequence must be used when modifying the HID Register:

```
sync  
mtspr HID,Rx  
isync
```

### 5.8.1 HID Register Description

Initial state: x'0400\_0000\_0000\_0000'

Preferred state: x'0000\_0000\_0000\_0000'

Table 5-9 describes the HID Register.

Table 5-9. HID Register (Sheet 1 of 2)

Bits	Field Name	Description
0	one_ppc	One (Power ISA) instruction is sent out of the IBuffers and decoded at a time. The IFU waits for ICT empty to let the next instruction go. Multi-thread mode uses a round-robin method through the enabled threads.
1	en_instruc_trace	Enable the enhanced trace facility, which requires special hardware initializations.
2	flush_ic	Flush the instruction cache and the instruction <a href="#">EADIR</a> on a transition from '0' to '1'.
3	en_attn	Enable the support-processor attention instruction. This bit is used to enable the <a href="#">attn</a> instruction to quiesce the thread. <b>Note:</b> The instruction cache must be flushed after changing the value of this bit.
4	HILE	The contents of this bit are copied into the MSR[LE] by interrupts that result in MSR[S,HV] = '01'.
5	dis_recovery	Disable the processor recovery mechanism.
6	megamouth	Order stores for the Megamouth adapter when IG = '10'.
7	prefetch_reset	Clear all streams from the prefetch unit and restart from an idle state.
8	Reserved	This bit was used to configure the TLB in the POWER9 processor core. The bit has no function in the Power10 core or later designs. However, it must not be assigned a new function for Linux/KVM backward compatibility requirements. The bit can be set, but its value will be ignored. The <a href="#">mfhid</a> instruction will return the last value written to the bit.

Table 5-9. HID Register (Sheet 2 of 2)

Bits	Field Name	Description
9	Reserved	This bit was used in the POWER9 core to control how the D-cache was partitioned by the thread in a balanced method, such that each active thread can use an equal portion of the cache. This bit is not supported in the Power10 core. However, it must not be assigned a new function for Linux/KVM backward compatibility requirements. The bit can be set, but its value will be ignored. The <b>mfhid</b> instruction will return the last value written to the bit.
10	Reserved	This bit was used in the POWER9 core to control how the I-cache was partitioned by the thread in a balanced method, such that each active thread can use an equal portion of the cache. Not supported in the Power10 core. However, it must not be assigned a new function for Linux/KVM backward compatibility requirements. The bit can be set but its value will be ignored. The <b>mfhid</b> instruction will return the last value written to the bit.
11	Reserved	Reserved. In the POWER9 processor, this was the Enable speculative execution mode bit. The bit has no function in the Power10 core or later designs. However, it must not be assigned a new function for Linux/KVM backward compatibility requirements. The bit can be set, but its value will be ignored. The <b>mfhid</b> instruction will return the last value written to the bit.
12	spare	The bit can be set, but its value will be ignored. The <b>mfhid</b> instruction will return the last value written to the bit.
13	spare	The bit can be set, but its value will be ignored. The <b>mfhid</b> instruction will return the last value written to the bit.
14	spare	Reserved. Software must set this bit to '0'.
15	TLB_64K_opt_mode	Enables the core to be optimized for 64 KB pages over 4 KB pages. The 4 KB pages are still functional in 64K optimization mode, but they might suffer a performance loss. The 64K optimization mode should only be set for operating systems that use predominately the 64 KB page size versus the 4 KB page size. When changing this bit, the hypervisor must ensure all threads on a given core are in hypervisor real mode and subsequently, that the entire TLB is purged for that core before exiting hypervisor real mode. See <i>Section 5.10.22 TLB Invalidate All (tlbia) Instruction</i> on page 147 for details on purging the TLB.
16	spare	Reserved. Software must set this bit to '0'.
17:63	Reserved	Reserved unimplemented bits. The <b>mfhid</b> instruction will return zeros.

### 5.8.2 IMC Array Access Register

The Instruction Match CAM (IMC) array facility is used for performance monitoring instrumentation and for the soft patch of instructions. (This latter use is restricted for the support processor and is not available through the SPR access to this register array.) The array has hypervisor write and read access via this SPR. Writes to the register array are used to configure the IMC, and reads return information about the availability of registers within the facility.

### 5.8.3 Performance Monitor Registers

The performance monitor counter registers (PMC1 - PMC6), the performance monitor control registers (MMCR0, MMCR1, MMCRA), and the sampled address registers (SIAR, SDAR) are supported in the Power10 processor core. The performance monitor counter registers PMC7 and PMC8 are not implemented in the Power10 core (an operation for these two performance counter registers is treated as a NOP).

### 5.8.4 Other Fixed-Point Instructions

The Power10 core supports both the 32-bit **mtmsr** instruction and the 64-bit **mtmsrd** instruction.



The Power10 core optimizes the **mtmsr** and **mtmsrd** instructions by helping to speed up the cases where little or no synchronization is required (such as, updates to the EE and RI bits). To exploit this capability, software should set the L-bit of the desired instruction to '1' as described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*.

Software must avoid placing **mtmsr** and **mtmsrd** instructions that change the SF bit at address x'00000000FFFFFC' or x'FFFFFFFFFFFFFFFC'.

## 5.9 Storage Control

### 5.9.1 Effective, Virtual, and Physical Address Ranges Supported

The Power10 core supports a 64-bit effective address (EA), 68-bit virtual address (VA), and a 53-bit host real (physical) address (RA). The 53-bits of host real address correspond to the range of bits specified by RA bits 11:63. RA bits 0:10 are unimplemented in the Power10 processor core. See *Section 5.10.4 Translation Modes* on page 126 for details specific to various address translation modes. With the exception of RA(11) = '1', host real addresses in this 53-bit range are considered to be system-wide coherent. See *Section 5.9.2 Control Memory Definition and Accessibility* for a description of the meaning of RA(11).

### 5.9.2 Control Memory Definition and Accessibility

The Power10 core considers host real addresses with a nonzero value in RA(11) as control memory address space, accessible only by the **paste** instruction. By specifying a host real address in this range for the **paste** instruction, a **copy** and **paste** instruction pair can be used to invoke the Nest accelerator via the Virtual Accelerator Switchboard (VAS). For more details on invoking the Nest accelerators, see *Section 11.1 NX Features* on page 198. A host real address containing a zero value in RA(11) is referred to as non-control memory. Any reference to memory, memory ranges, or memory address space without the control or non-control qualifier implies non-control memory.

Attempts to access host real pages in the control memory addressing range by an instruction fetch or any other data access (that is, other than a **paste** instruction) results in a machine check interrupt per the Power ISA.

As described in the *Power ISA (Version 3.1B)*, the **copy** instruction can copy a 128-byte cache line (block) from a non-control memory address to a per thread non-coherent buffer. Similarly, a **paste** instruction reads the non-coherent buffer and writes the contents of the buffer to either a control memory or a non-control memory address. For more details on the **copy** and **paste** instructions, see the *Power ISA (Version 3.1B)*.

### 5.9.3 Ultravisor Real Mode Addressing Using URMOR

The Power10 processor supports the Ultravisor Real Mode Offset Register (URMOR) as described in the *Power ISA (Version 3.1B)* for the purpose of accessing real memory when the processor thread is operating in Ultravisor Real Mode. As described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*, when EA(0) = '1', the URMOR is bypassed. When EA(0) = '0', the URMOR value is logically OR'ed with the EA to produce the real address (RA). The ISA defines the number of implemented bits in the URMOR as implementation dependent. The Power10 core implements URMOR[12:42]. All other URMOR bits are reserved and return zero when read.

### 5.9.4 Hypervisor Real Mode Addressing Using HRMOR

The Power10 processor supports the Hypervisor Real Mode Offset Register (HRMOR) as described in the *Power ISA (Version 3.1B)* for the purpose of accessing real memory when the processor thread is operating in Hypervisor Real Mode. As described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*, when EA(0) = '1', the HRMOR is bypassed. When EA(0) = '0', the HRMOR value is logically OR'ed with the EA to produce the real address (RA). The ISA defines the number of implemented bits in the HRMOR as implementation dependent. The Power10 processor implements HRMOR[13:42]. All other HRMOR bits are reserved and return zero when read.

### 5.9.5 Partition Table Control Register

The Power10 core supports the Partition Table Control Register (PTCR) mostly as described in the *Power ISA (Version 3.1B)* for the purpose of accessing virtual memory either in virtual real mode (HPT) and guest real mode (nested Radix) or when translation is enabled for either HPT or Radix. The PTCR implements bits 12:51 for the Partition Table Base (PATB) field and bits 59:63 for the Partition Table Size (PATS) field. All other PTCR bits are reserved and return zero when read. However, the Power10 core ignores the value in the PATS field and only supports a 64 KB partition table size.

### 5.9.6 Access Segment Descriptor Register

The Power10 core supports the Access Segment Descriptor Register (ASDR) as described in the *Power ISA (Version 3.1B)*. See *Appendix C Storage Exception Cases by Translation Type* on page 389 for details on the contents of the ASDR for each exception type.

### 5.9.7 Real Mode Addressing for Operating Systems

The Power10 core does not implement the real mode offset (R MOR/R MLS) mechanism described in the *Power ISA (version 2.07)*. Instead, it implements the virtual real mode addressing mechanism for HPT translation and guest real mode for radix translation as described in the *Power ISA (Version 3.1B)*.

### 5.9.8 HRMOR Update Sequence

*Table 5-10* describes a sequence that the hypervisor privileged software might use to update the HRMOR.

*Table 5-10. HRMOR Update Sequence*

Primary	Secondary
Thread sync up point 1	Thread sync up point 1
EA[0] = 1	EA[0] = 1
Thread sync up point 2	Thread sync up point 2
Change HRMOR	
Thread sync up point 3	Thread sync up point 3
isync	isync
slbia IH = x'7'	slbia IH = x'7'
isync	isync
Thread sync up point 4	Thread sync up point 4



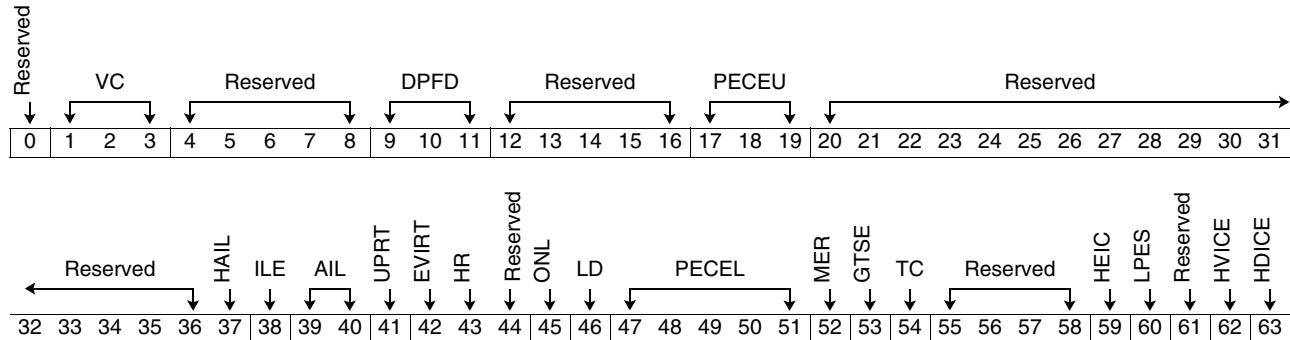
## 5.10 Translation Architecture

The Power10 core supports the various translation mechanisms described in the *Power ISA (Version 3.1B)*. When the Power10 core is not operating in either ultrervisor or hypervisor real mode, the Logical Partition Control Register (LPCR) determines how address translation is performed. Noteworthy details are described as follows:

- Reserved bits in the LPCR are unimplemented in the hardware and return zero when read.
- The Power10 processor uses the Partition Table Control Register (PTCR) to find the partition table in memory. The Host Radix (HR) bit in the LPCR should be set by software to the same value as the HR bit in the partition table entry indexed by the Logical Partition Identification (LPID) by the time the partition table entry is referenced.
- Because LPIDR = 0 is reserved for use by the hypervisor on radix systems, software should avoid using LPIDR = 0 for HPT (HR = '0') partitions (that is, MSR[HV] = '0').
- When LPCR[GTSE] = '0', **slbiag**, **slbieg**, **slbsync**, and **tlbsync** are hypervisor privileged only and take a privileged instruction interrupt when HV = '0' (independent of PR); or when HV = '1' and PR = '1'.
- When LPCR[GTSE] = '1', **slbiag**, **slbieg**, **slbsync**, and **tlbsync** are legal instructions when PR = '0' and take a privileged instruction interrupt when PR = '1'.
- The **tlbie** instruction is privileged except when LPCR[GTSE] = '0' or when PRS = '0' and HR = '1'; making it hypervisor privileged. Note that the POWER9 processor used the "R" bit in the instruction instead of the "HR" bit in the partition-table entry, as described in the *Power ISA (Version 3.1B)* for determining the instruction's privilege level.
- The **tlbiel** instruction is privileged except when PRS = '0' and HR = '1', making it hypervisor privileged. Note that the POWER9 processor used the "R" bit in the instruction instead of the "HR" bit in the partition-table entry as described in the *Power ISA (Version 3.1B)* for determining the instruction's privilege level. When the congruence class (SET) ID is set to zero, the hardware loops through and invalidates all TLB entries, ERAT, and L1-cache entries satisfying the match criteria (for example, PID or LPID). When the congruence class (SET) ID is set to a non-zero value, the instruction is treated as a NOP. This behavior is in agreement with the function described in *Power ISA (Version 3.1B)*.

### 5.10.1 Logical Partitioning Control Register (LPCR)

The Power10 LPCR Register is illustrated as follows:



Bits	Field Name	Description
0	Reserved	Reserved.
1:3	VC	Virtualization control.
4:8	Reserved	Reserved.
9:11	DPFD	Default prefetch depth.
12:16	Reserved	Reserved.
17:19	PECEU	Power-saving mode exit causes enable upper section.
20:36	Reserved	Reserved.
37	HAIL	Hypervisor alternate interrupt location.
38	ILE	Interrupt little-endian mode.
39:40	AIL	Alternate interrupt location.
41	UPRT	Use the process table. Software should set this bit to '0' when the thread is performing HPT translation and set to '1' when the thread is performing radix translation.
42	EVIRT	Enhanced virtualization enable.
43	HR	Host radix.
44	Reserved	Reserved.
45	ONL	Online (PURR/SPURR incrementing control).
46	LD	Large decrementer.
47:51	PECEL	Power-saving mode exit causes enable lower section.
52	MER	Mediated external exception request.
53	GTSE	Guest translation shootdown enable.
54	TC	Translation control secondary PTEG is not searched if TC = '1'.
55:58	Reserved	Reserved.
59	HEIC	Hypervisor external interrupt control.
60	LPES	Logical partitioning environment selector.
61	Reserved	Reserved.
62	HVICE	Hypervisor virtualization interrupt conditionally enable.
63	HDICE	Hypervisor decrementer interrupt conditionally enable.



**Note:** All fields except the AIL, EVIRT, ONL, HDICE, MER, PECE, HEIC, and HVICE fields must be set by software to the same value by all subprocessors with the same LPIDR value.

### 5.10.2 Logical Partition Identification Register (LPIDR)

The LPIDR is a 32-bit register [defined in the *Power ISA (Version 3.1B)*] as bits 32:63], which implements 12 bits of LPID in LPIDR(52:63). Unimplemented bits in LPIDR(32:51) are ignored and always return zeros when read.

Unlike the POWER9 core, the Power10 core does not always perform a full invalidation of implementation-specific translation caches as part of the **mtlpidr** instruction. Rather, the Power10 processor only performs a partial implicit invalidation of the pertinent entries in the ERAT plus the L1 instruction and data caches when architecturally required. When the implicit invalidation is performed, it is analogous to that which is performed by software, explicitly executing an **sibia** IH = x'6'.

### 5.10.3 Process Identification Register (PIDR)

The PIDR is a 32-bit register [defined in the *Power ISA (Version 3.1B)*] as bits 32:63] which implements 20 bits of PID in PIDR(44:63). Unimplemented bits in PIDR(32:43) are ignored and always return zeros when read.

Unlike the POWER9 core, the Power10 core does not always perform a full invalidation of implementation-specific translation caches as part of the **mtpidr** instruction. Rather, the Power10 core only performs a partial implicit invalidation of the pertinent entries in the ERAT plus the L1 instruction and data caches when architecturally required. When the implicit invalidation is performed, it is analogous to that which is performed by software explicitly executing an **sibia** IH = x'3'.

### 5.10.4 Translation Modes

Within the translation architecture described in the *Power ISA (Version 3.1B)*, there are two types of address translation:

- Radix: The Host Radix (HR) bit in both the LPCR and the partition table entry is set to '1'. A radix guest (operating system) running on top of a radix host (hypervisor) is also commonly referred to as nested radix or multi-level radix. When there is no guest operating system (that is, when the operating system is also the hypervisor), this is referred to as single-level radix.
- Hashed Page Table (HPT): The Host Radix (HR) bit in both the LPCR and the partition table entry is set to '0'. HPT is also sometimes referred to as paravirtualized translation. For HPT translation, the processor core supports only the behavior specified when the LPCR Use Process Table (UPRT) bit is set to '0'. Note that when the LPCR[UPRT] = '0', there must be an in-memory segment table for use by the nest memory management unit (NMMU) for the purpose of address translation by devices (for example, accelerators, GPUs, and so on) outside of the Power10 core.

In these modes, the Partition Table Control Register (PTCR) contains the host real address of the partition-table base and the size of the table itself. In general, the partition table is indexed by the logical partition ID (LPID) value specified in the LPIDR. The value of the HR bit in the partition table entry must match the HR value in the LPCR for the translation access being performed. If not, the hardware considers this to be a hypervisor programming error and the results are undefined.

When the partition-table entry is read, the HR bit determines which translation type is used by the hardware to convert an effective address to a host real address. When either single-level or nested radix is used for translation, HR = '1'. When HPT translation is used, HR = '0'. For either of the these translation types, there exists a partition-scoped page table that translates a host virtual address (hVA) (referred to as a guest real address or gRA for radix) to a host real address (hRA).

For either of the translation types, the in-memory translation related tables managed by the operating system (guest) are translated as though they reside in "normal" memory (such as, ATT = '00' or WIMG = '0010'), regardless of the storage attributes specified in the partition scoped page-table entries used to translate those guest tables. These guest-managed translation tables include the process table for either radix or HPT, the guest radix tree for radix, and the segment table for HPT (for the in-memory segment table supported by the Nest).

### 5.10.5 **tlbie** and **tlbiel** Instruction Format and Operands

When software changes a translation path that involves either a radix or HPT page table, either the **tlbie** or **tlbiel** instruction must be used in a manner as specified in the *Power ISA (Version 3.1B)*. The ISA provides the format for both the **tlbie** and **tlbiel** instructions, but the AP and L/LP encodings are implementation specific. The format and operands are shown in *Figure 5-2* for the **tlbie** instruction and *Figure 5-4* on page 129 for the **tlbiel** instruction. Note that for both the **tlbie** instruction and **tlbiel** instruction, the term "effective R" or "effR" is defined as follows:

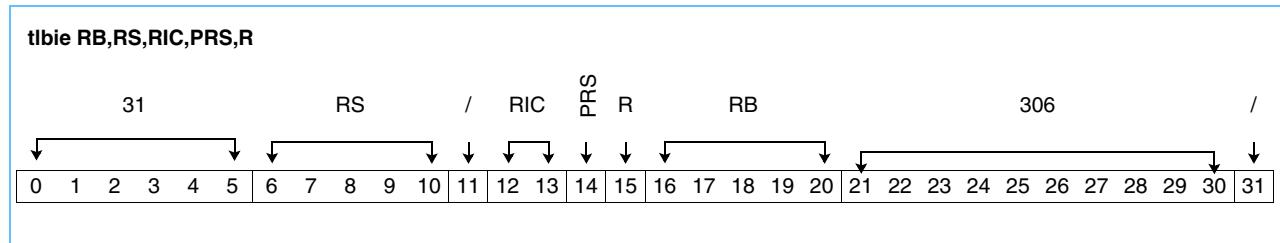
- When HV = 1, the effR is the value of the R-bit taken from the **tlbie** or **tlbiel** instruction itself.
- When HV = 0, the effR is the value of the LPCR[HR] bit. The R-bit specified in the **tlbie** or **tlbiel** instruction is ignored by the hardware.

Note that for invalidations of an SMT8 core, the **tlbiel** instruction applies only to the local SMT4-core-resource. The previously described behavior allows the hypervisor to invalidate translations for partitions other than the current partition pointed to by the LPIDR while preventing non-hypervisor privileged software from invalidating partition-scoped translations.

The effR serves two additional functional purposes. The first is to specify the instruction operand format and the second is used to tell the hardware how to interpret the address associated with an IS = 0 form of the **tlbie** or **tlbiel** instruction.

**Important note:** In the POWER9 processor, the effR value was always the R bit from the instruction regardless of the value of HV. This deviation is described in the *POWER9 Processor User's Manual*.

*Figure 5-2. tlbie Instruction Format for the Power10 Core*



*Table 5-11. Description of **tlbie** Instruction Format for the Power10 Core*

Bits	Description
RIC	Radix invalidation control. 0 Only invalidate the TLB and ERATs. 1 Invalidate only the page-walk cache (PWC). 2 Invalidate TLB, ERAT, PWC, and any caching of partition and process table entries. 3 Invalidate a series of consecutive translations (only in TLB/ERATs, cluster bomb).
PRS	Process scoped. 0 Invalidate partition-scoped translations. 1 Invalidate process-scoped translations.
R	Radix. 0 Invalidate HPT translations when HV = 1. LPCR[HR] is used instead of R when HV = 0. 1 Invalidate Radix tree translations when HV = 1. LPCR[HR] is used instead of R when HV = 0.
IS	Invalidation selector (specified in the RB Register). 0 Invalidate only the target VA for HPT or EA/gRA for Radix for matching PID and LPID. 1 Invalidate matching PID (and matching LPID). 2 Invalidate matching LPID. 3 If MSR[HV] = '1', invalidate all entries; otherwise, invalidate matching LPID.

The format and operands for the **tlbie** instructions are indicated in *Figure 5-3*. The effR, IS, and RIC values determine the format of the RB operand.

*Figure 5-3. **tlbie** Operands for the Power10 Core*

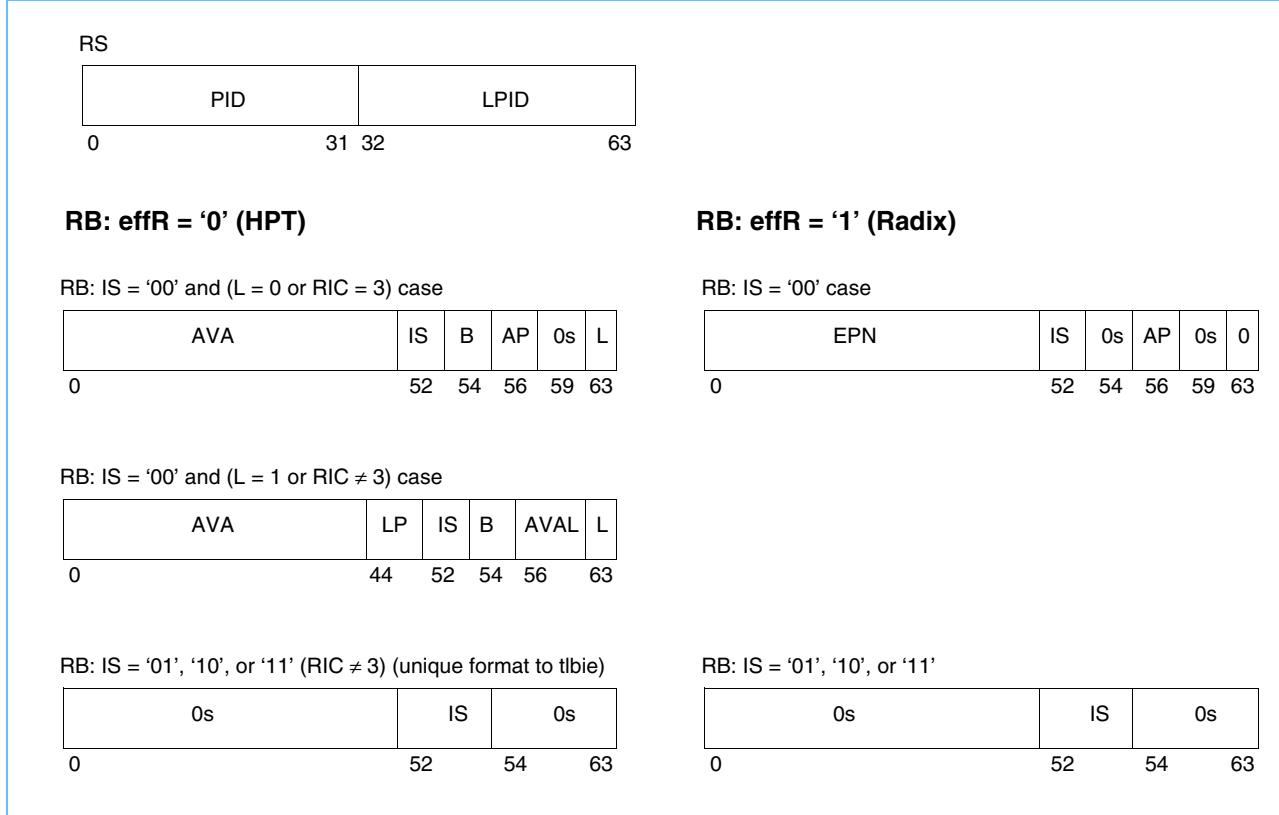


Figure 5-4. ***tlbiel*** Instruction Format for the Power10 Core

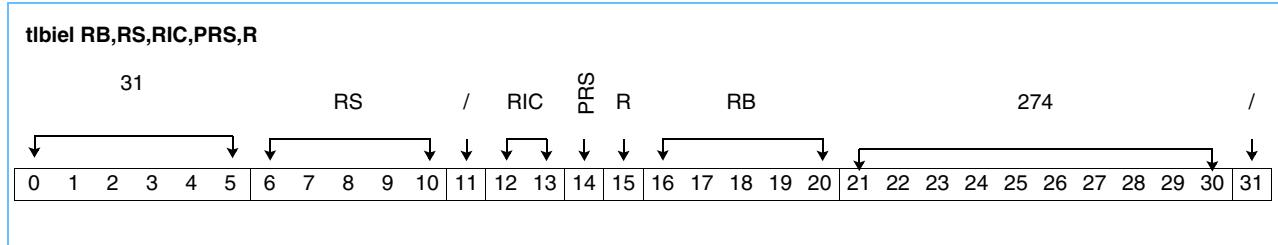


Table 5-12. Description of ***tlbiel*** Instruction Format for the Power10 Core

Bits	Description
RIC	Radix invalidation control. 0 Only invalidate TLB and ERATs. 1 Invalidate only page-walk cache (PWC). 2 Invalidate TLB/ERATs, PWC, and any caching of partition and process table entries. 3 Not supported.
PRS	Process scoped. 0 Invalidate partition-scoped translations. 1 Invalidate process-scoped translations.
R	Radix. 0 Invalidate HPT translations when HV = 1. LPCR[HR] is used instead of R when HV = 0. 1 Invalidate radix tree translations when HV = 1. LPCR[HR] is used instead of R when HV = 0.
IS	Invalidation selector (specified in the RB Register). 0 Invalidate only the target VA for matching PID and LPID. 1 Invalidate all entries in specified TLB congruence class (SET) with a matching PID (and matching LPID). 2 Invalidate all entries in specified TLB congruence class (SET) with a matching LPID. 3 If MSR[HV] = '1', invalidate all entries in specified TLB congruence class (SET); otherwise, invalidate all entries in specified TLB congruence class with a matching LPID.

The format and operands for the **tlbiel** instructions are indicated in *Figure 5-5*. The effR and IS values determine the format of the RB operand.

*Figure 5-5. tlbiel Operands for the Power10 Core*

RS																																					
	PID	///																																			
0	31 32	63																																			
<b>RB: effR = '0' (HPT)</b>		<b>RB: effR = '1' (Radix)</b>																																			
RB: IS = '00' and (L = 0) case		RB: IS = '00' case																																			
<table border="1"> <tr> <td>AVA</td> <td>IS</td> <td>B</td> <td>AP</td> <td>0s</td> <td>L</td> </tr> <tr> <td>0</td> <td>52</td> <td>54</td> <td>56</td> <td>59</td> <td>63</td> </tr> </table>		AVA	IS	B	AP	0s	L	0	52	54	56	59	63	<table border="1"> <tr> <td>EPN</td> <td>IS</td> <td>0s</td> <td>AP</td> <td>0s</td> <td>0</td> </tr> <tr> <td>0</td> <td>52</td> <td>54</td> <td>56</td> <td>59</td> <td>63</td> </tr> </table>								EPN	IS	0s	AP	0s	0	0	52	54	56	59	63				
AVA	IS	B	AP	0s	L																																
0	52	54	56	59	63																																
EPN	IS	0s	AP	0s	0																																
0	52	54	56	59	63																																
RB: IS = '00' and (L = 1) case		RB: IS = '01', '10', or '11' (unique format to tlbiel)																																			
<table border="1"> <tr> <td>AVA</td> <td>LP</td> <td>IS</td> <td>B</td> <td>AVAL</td> <td>L</td> </tr> <tr> <td>0</td> <td>44</td> <td>52</td> <td>54</td> <td>56</td> <td>63</td> </tr> </table>		AVA	LP	IS	B	AVAL	L	0	44	52	54	56	63	<table border="1"> <tr> <td>0s</td> <td>SET</td> <td>IS</td> <td>0s</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0</td> <td>40</td> <td>52</td> <td>54</td> <td>63</td> <td></td> <td></td> <td></td> </tr> </table>								0s	SET	IS	0s					0	40	52	54	63			
AVA	LP	IS	B	AVAL	L																																
0	44	52	54	56	63																																
0s	SET	IS	0s																																		
0	40	52	54	63																																	

## 5.10.6 Radix Translation

When the partition-table entry has HR = '1', the translation mechanism is referred to as radix translation. Radix translation can be either single-level or nested.

For nested radix, two sets of radix trees exist:

- One set of radix trees that is managed by the guest operating system, is determined by indexing a process table using the process table ID (PID) value.
- A second set of radix trees is managed by the host (hypervisor) software.

For a radix host, the LPIDR value can be overridden using EA(0:1) to indicate which EA quadrant the thread is accessing as described in the ISA. The term effective LPID (effLPID) is more commonly used to discuss how the partition table is indexed. Analogous to how the effLPID is used to index the partition table for host translation, an effective PID (effPID) is the commonly used term for indexing the process table. See the *Power ISA (Version 3.1B)* for more details on EA quadrants, effLPID, and effPID. When the effLPID = '0', there exists only a single level of Radix translation. This translation is considered to be process-scoped (that is, indexed by the effPID) and used to translate the EA directly to an hRA.

For radix translation, the EA range is 64 bits, but as stated previously, EA(0:1) indicate which EA quadrant the thread is in. When EA(2:11) bits are nonzero, a segment interrupt results. Thus, the usable EA space (when the thread is not in hypervisor real mode) is limited to 52 bits. In addition, the gRA is limited to 52 bits; otherwise, a segment interrupt occurs. *Table 5-13* shows the details of the address bit range checking by hardware.

*Table 5-13. Address Bit Range Checking by Hardware*

Mode	Instruction or Data EA(0:1)	EA(2:11) (Nonzero)	Instruction or Data gRA(0:11) (Nonzero)	Guest PDE RPN(4:11) (Nonzero)	Guest PTE RPN(7:11) (Nonzero)	Host PDE RPN(4:10) (Nonzero)	RPN(11) Host PDE or Host PTE <sup>1</sup> (Nonzero)	Host PTE RPN(7:10) (Nonzero)	Host PTE RPN(11) (Nonzero) <sup>2</sup>
Guest Real Mode (IR/DR off)	Ignore EA(0:1); always treat as '00'.	Segment interrupt (gEA = gRA)	Segment interrupt	N/A	N/A	Ignore <sup>3</sup>	Machine check	Ignore <sup>3</sup>	<b>paste.</b> data access: ignore; otherwise, machine check
Nested Radix with IR/DR on	Quadrant bits	Segment interrupt	Ignore <sup>3</sup>	Ignore <sup>3</sup>	Ignore <sup>3</sup>	Ignore <sup>3</sup>	Machine check	Ignore <sup>3</sup>	<b>paste.</b> data access: ignore; otherwise, machine check
Process scoped single-level Radix (effLPID = 0)	Quadrant bits	Segment interrupt	N/A	N/A	N/A	Ignore <sup>3</sup>	Machine check	Ignore <sup>3</sup>	<b>paste.</b> data access: ignore; otherwise, machine check
HPT	Only part of EA	Only part of EA	N/A	N/A	N/A	Ignore <sup>3</sup>	Machine check	Ignore <sup>3</sup>	<b>paste.</b> data access: ignore; otherwise, machine check

1. RPN(11) nonzero in any host PDE (that maps a guest table or instruction/data access) or in any host PTE that maps a guest table.  
 2. Host PTE for instruction/data access where RPN(11) is nonzero.  
 3. Future processors might generate an instruction or data segment interrupt.

### 5.10.6.1 Supported Radix Tree Configurations and Resulting Page Sizes

The *Power ISA (Version 3.1B)* provides the general architecture to support implementations that might implement a variety of page sizes. The Power10 processor supports only the page sizes specified in *Table 5-14* on page 132 when performing radix translation. The values in the Level columns indicate the supported size of each level of the radix tree for each resulting page size. All other radix tree configurations are unsupported and result in the hardware generating an unsupported radix tree type of DS1, HDSI, ISI, or HISI.



*Table 5-14. Supported Radix Tree Configurations and Resulting Page Sizes*

Page Size	Level 1 Size	Level 2 Size	Level 3 Size	Level 4 Size
4 KB	64 KB	4 KB	4 KB	4 KB
64 KB	64 KB	4 KB	4 KB	256 B
2 MB	64 KB	4 KB	4 KB	
1 GB	64 KB	4 KB		

#### 5.10.6.2 **tlbie** and **tlbiel** Encodings for Radix Translations

When software must invalidate the translation caches for one of these page sizes, it should execute the appropriate **tlbie** or **tlbiel** instruction sequence as specified in the *Power ISA (Version 3.1B)* with effR = '1' and AP set to the corresponding value as indicated in *Table 5-15*, and a RIC value of either '0' or '2' for the required outcome.

*Table 5-15. **tlbie(I)** Page Encodings for Power10 Radix (effR = '1') Only RIC ≠ 3 is supported*

RB(32:51)	RIC	RB(56:58) AP <sup>1</sup>	Actual Page Size to be Invalidated <sup>2</sup>
vvv vvvvvv vvvvvv	0, 2	000	4 KB
vvv vvvvvv vvvxxx	0, 2	101	64 KB
vvv vvvvvv xxxxxx	0, 2	001	2 MB
vxx xxxxxxxx xxxxxxx	0, 2	010	1 GB

1. All other values of AP should not be used when effR = 1 and results in a machine check interrupt.  
2. 2 MB and 1 GB page sizes are only supported for radix.

Alternatively, software can use the **tlbiel** instruction with IS ≠ 0 per the *Power ISA (Version 3.1B)* to perform a congruence class invalidation of the TLB on the processor that executes the **tlbiel** instruction. When this option is chosen, RB(45:51), select the congruence class of the TLB and/or PWC to be invalidated.

See *Appendix D tlbie(I) Encodings* on page 421.

#### 5.10.7 Changing the Process ID Register For Radix Translation

As described in *Section 5.10.3 Process Identification Register (PIDR)* on page 126, the Power10 core implements 20-bits of process ID in the Process ID Register (PIDR). When using radix translation and software must change the PIDR value, software should do so only when the PIDR value is not being used for the current translation to avoid implicit branch conditions. This means either instruction translation is disabled (MSR[IR] = '0') or instruction translation is enabled (MSR[IR] = '1'); and the effPID value is zero (effPID = '0').

#### 5.10.8 Switching between Radix and HPT Partitions

The details for switching between radix and HPT partitions depend on the LPAR mode in which the core is operating. See *Section 4.2 Logical Partition Specific Resources* on page 58 for a description of thread LPAR (LPAR per thread) mode and 1-LPAR (LPAR per core) mode.



### 5.10.8.1 LPAR per Thread (Thread LPAR) Mode

The generic sequence to switch a thread from running an HPT partition to a radix partition when the core is operating in “Thread LPAR” mode (also known as, “thread per LPAR” mode) is as follows:

1. Start in an HPT partition with translation on.
2. Switch the thread to hypervisor real mode.
3. Change LPIDR to point to a radix partition via an **mtlpidr** instruction.
4. Change LPCR so that the following conditions are true: UPRT = ‘1’, HR = ‘1’, VC = ‘000’ via an **mtlpcr** instruction.
5. Execute an **rfid** to pass control to the target radix partition.
6. Change PIDR without causing an implicit branch.

To switch a thread from a radix partition to an HPT partition, the following sequence must be observed:

1. Start out in a radix partition with translation on.
2. Switch the thread to hypervisor real mode.
3. Change LPIDR to point to an HPT partition via an **mtlpidr** instruction.
4. Change LPCR so that the following conditions are true: UPRT = ‘0’, HR = ‘0’, and set VC accordingly for the target HPT partition via an **mtlpcr** instruction.
5. Execute an **rfid** to pass control to the target HPT partition.

### 5.10.8.2 LPAR per Core (1-LPAR) Mode

The generic sequence to switch a core between running a radix partition and an HPT partition when the core is operating in “1-LPAR” mode (also known as “LPAR per Core” mode) is as follows:

1. Start in an HPT partition with translation on.
2. Switch all active threads on the core to hypervisor real mode.
3. On each thread, change LPIDR to point to the target radix partition via an **mtlpidr** instruction.
4. On each thread, change LPCR so that the following conditions are true: UPRT = ‘1’, HR = ‘1’, VC = ‘000’ via an **mtlpcr** instruction.
5. On each thread, execute an **rfid** to pass control to the target radix partition.
6. On each thread, change PIDR without causing an implicit branch.

To switch from a radix partition to a HPT partition, the following sequence must be observed:

1. Start out in a radix partition with translation on.
2. Switch all active threads on the core to hypervisor real mode.
3. On each thread, change LPIDR to point to the target HPT partition via an **mtlpidr** instruction.
4. Change LPCR so that the following conditions are true: UPRT = ‘0’, HR = ‘0’, and set VC accordingly for the target partition via a **mtlpcr** instruction.
5. Execute an **rfid** to pass control to the target HPT partition.



### 5.10.9 Hashed Page Table (HPT) Translation

When the partition table entry and LPCR have HR = '0', the translation mechanism is referred to as either paravirtualized or HPT. In HPT mode, there is no concept of an effPID or an effLPID, only PIDs and LPIDs. In other words, only the values found in the PIDR and LPIDR, respectively, are used to index the appropriate translation table. This translation mode is most similar to the legacy translation architecture supported on past processors such as the POWER8 and POWER9 processors. In HPT mode, the effective address space is 64 bits (0:63), the virtual address space is implemented as 68 bits (10:77) of the 78-bit architected maximum virtual address space, and the real address space is 51 bits (13:63).

#### 5.10.9.1 Software Managed SLB Entries (UPRT = '0', HR = '0')

As stated in *Section 5.10.4 Translation Modes*, the Power10 processor core only supports the LPCR[UPRT] = '0' submode of the HPT translation architecture. In this mode, the Power10 core supports 32 software-managed SLB entries (the same number as supported by the POWER8 and POWER9 processors).

#### 5.10.9.2 In-Memory Segment Table and Bolted SLB Entries (UPRT = '1', HR = '0')

The Power10 core does not support LPCR[UPRT] = '1' mode for HPT (HR = '0') translation mode. Support for hardware tablewalk of an in-memory segment table is implemented only in the NMMU hardware. It is up to the hypervisor software to set UPRT = '0' whenever HR = '0'.

### 5.10.10 Changing the Process ID Register For HPT Translation

As described in *Section 5.10.3 Process Identification Register (PIDR)*, the Power10 processor implements 20-bits of process ID in the Process ID Register (PIDR). When using HPT translation and software needs to change the PIDR value, software should do so only when the PIDR value is not being used for the current translation to avoid implicit branch conditions. Per the *Power ISA (Version 3.1B)*, a subsequent **slbia** IH = x'3' instruction is required when changing the PIDR for HPT translation.

#### 5.10.10.1 SLB Management Instructions

The Power10 core implements the SLB management instructions as defined in the *Power ISA (Version 3.1B)*. Specifically, the following instruction details are noteworthy:

- The **slbmfee** and **slbmfev** instructions can read any SLB entry when UPRT = '1', if the L-bit in the instruction image is set to a '1'. This is an implementation-specific feature that will only be used in the future if and when the Power10 core supports UPRT = '1' for HPT translation.
- The **slbfee.** instruction writes '0' to CR field 0 whenever UPRT = '1'.

**Note:** Always set UPRT to '1' (per the ISA) whenever radix translation is being performed (that is, when LPCR[HR] = '1').

- The **slbia** instruction with IH = '101' (which is a reserved value) is treated the same as IH = '111'.

### 5.10.10.2 Supported Segment and Page Sizes for HPT Translations

The Power10 core supports two segment sizes for HPT translation: 256 MB and 1 TB. The Power10 core also provides support for 4 KB, 64 KB, 16 MB, and 16 GB page sizes for HPT translation. Translation information for all these page sizes is kept in the TLB. Page sizes of 4 KB, 64 KB, and 16 MB are stored natively in the TLB. However, 16 GB pages are stored in the TLB as a series of 1 GB pages which are each installed on-demand.

If a given virtual address is mapped by two overlapping pages of different sizes (for example, a small page is mapped initially but is later re-mapped into a larger page without invalidating the original TLB entry), a machine check interrupt can result with an indication that either a parity error or a multi-hit occurred when the TLB was accessed to translate the address. The error condition can be corrected by invalidating the entire TLB and SLB.

The Power10 core also supports multiple page sizes per segment (MPSS) as described in the *Power ISA (Version 3.1B)*. Specifically, the Power10 core supports mixing page sizes in a single segment with the following combinations of base and actual page sizes:

- 4 KB base: 4 KB actual, 64 KB actual, or 16 MB actual
- 64 KB base: 64 KB actual or 16 MB actual

*Table 5-16* shows the correspondence between PTE[L, LP] values and STE[L, LP]/SLBE[L, LP] values. The supported segment table and SLB entry sizes and page sizes are also shown in *Table 5-16*. These same page sizes and their associated encodings are also used in the Partition Table Entry “PS” field.

*Table 5-16. PTE and STE/SLBE Correspondence for HPT Translation* **Note:** Unimplemented STE/SLBE page size encodings are treated the same as the ‘000’ case.

Entry Number	PTE			STE/SLBE		Base Page Size	Actual Virtual Page Size	Notes
	L	LP		L	LP			
1	0	rrrr	rrrr	0	00	4 KB	4 KB	1, 4
2	1	0000	0000	1	00	16 MB	16 MB	
3	1	rrrr	0001	1	01	64 KB	64 KB	2, 4
4	1	0000	0011	1	10	16 GB	16 GB	3
5	1	rrrr	0111	0	00	4 KB	64 KB	1, 4
6	1	0000	1000	1	01	64 KB	16 MB	2
7	1	0011	1000	0	00	4 KB	16 MB	1

1. Entries 1, 5, and 7 all use STE/SLBE[L, LP] = ‘000’ encoding for base page size 4 KB but have unique PTE[L, LP] encodings for actual page size.
2. Entries 3 and 6 both use STE/SLBE[L, LP] = ‘101’ encoding for base page size of 64 KB but have unique PTE[L, LP] encodings for actual page size.
3. If the STE/SLBE page size is ‘110’ (16 GB) and the segment size is small (256 MB), hardware treats the STE/SLBE page size the same as the ‘000’ case.
4. The ‘r’ bits are part of the real page number. They can be any value.



### 5.10.10.3 **tlbie** and **tlbiel** Usage for HPT Translations

For HPT translation, *Table 5-17* on page 136, *Table 5-18* on page 136, and *Table 5-19* on page 137 show the **tlbie** and **tlbiel** specifications for various page sizes supported by the Power10 core.

*Table 5-17* shows the legal segment size and page size specifications for **tlbie** and **tlbiel** for the Power10 HPT translations (effR = '0') when L = '0', and RIC ≠ '3'.

*Table 5-17. Segment Size and Page Size Specifications for HPT **tlbie** and **tlbiel** (effR = '0', L = '0', and RIC ≠ '3')*

RB[54:55] Segment Size	RB[63] L	RIC	RB[56:58] AP (Same as STE/SLBE[LII LP] Encoding)	Actual Page Size to be Invalidated
00	0	0,2	000	4 KB
00	0	0,2	101	64 KB
00	0	0,2	100	16 MB
01	0	0,2	000	4 KB
01	0	0,2	101	64 KB
01	0	0,2	100	16 MB

1. All other AP values must not be used when L = '0' (and effR = '0') and results in a machine check interrupt.  
 2. RB[54:55] = '00' corresponds to a 256 MB segment size and RB[54:55] = '01' corresponds to 1 TB segment size.  
 3. 16 GB page with a small segment (RB[54:55] = '00') is not a permitted combination and results in being treated as a NOP.  
 4. PRS = '1' and effR = '0' is an unsupported combination (invalid form).

*Table 5-18* shows the legal segment size and page size specifications for **tlbie** and **tlbiel** for the Power10 HPT translations (effR = '0') when L = '1' and RIC ≠ '3'.

*Table 5-18. Segment Size and Page Size Specifications for HPT **tlbie** and **tlbiel** (effR = '0', L = '1', and RIC ≠ '3')*

RB[54:55] Segment Size	RB[63] L	RIC	RB[44:51] LP	Base Page Size	Actual Page Size to be Invalidated
00	1	0,2	0000 0000	16 MB	16 MB
00	1	0,2	vvvv 0001	64 KB	64 KB
00	1	0,2	0000 1000	64 KB	16 MB
01	1	0,2	0000 0000	16 MB	16 MB
01	1	0,2	vvvv 0001	64 KB	64 KB
01	1	0,2	0000 0011	16 GB	16 GB
01	1	0,2	0000 1000	64 KB	16 MB

- All other LP values used when effR = '0', L = '1', and RIC ≠ '3' result in a machine check interrupt.
- "v" corresponds to AVA (AVPN) bits. The *Power ISA (Version 3.1B)* requires both the "v" bits in the LP field and the left-most 4 bits of the AVAL field in RB to also match these bits. However, the Power10 core only requires "v" bits in the LP field to match but does not require those AVAL bits to match for an invalidation to occur.
- RB[54:55] = '00' corresponds to 256 MB segment size and RB[54:55] = '01' corresponds to 1 TB segment size.
- A 16 GB page with a small segment (RB[54:55] = '00') is not a permitted combination and results in being treated as a NOP.

*Table 5-19. Segment Size and Page Size Specifications for HPT **tlbie** Cluster Bombs* (effR = '0', L = '0', and RIC = '3') (Note: **tlbiel** with RIC = 3 is an invalid instruction form and results in a machine check interrupt.)

RB[54:55] Segment Size	RB[63] L	RIC	RB[56:58] AP	Actual Page Size to be Invalidated
00	0	3	110	Eight consecutive 4 KB pages aligned on 32 KB boundary
00	0	3	111	Eight consecutive 64 KB pages aligned on 512 KB boundary
01	0	3	110	Eight consecutive 4 KB pages aligned on 32 KB boundary
01	0	3	111	Eight consecutive 64 KB pages aligned on 512 KB boundary

1. All other L and AP values and combinations used when effR = '0' and RIC = '3' result in a machine check interrupt.
2. RB[54:55] = '00' corresponds to 256 MB segment size and RB[54:55] = '01' corresponds to 1 TB segment size.
3. PRS = '1' and effR = '0' is an unsupported combination (invalid form).
4. The Power10 core has dropped support of range bombs.

**Note:** See *Appendix D tlbie(l) Encodings* on page 421 for details.

#### 5.10.10.4 Lockless TLBIE Support

The Power10 core supports lockless TLBIE operations. The architectural requirement that only one thread at a time can execute **tlbie/tlbsync** instructions during a page table modification need not be followed (see the Page Table Updates section of the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*). This was traditionally implemented with a single global lock for the entire page table modification sequences. The term lock-less **tlbie** refers to the Power10 core's ability to manage concurrent **tlbie/tlbsync** sequences from multiple threads without this global lock.

However, software must still ensure that concurrent, conflicting, racing PTE updates from more than one thread do not occur (the hardware performs the updates in some fashion, but the end result is undefined due to the racing of the updates to the same PTE entry) and therefore, software locks or some other synchronization discipline are still required to prevent these collisions as necessary.

The execution of **tlbie** instructions or the detection of snooped **tlbie** operations off the bus cause an index-based invalidate to occur in the TLB, if there is a match. In other words, an entry is invalidated only if there is a perfect match of the effective address supplied by the **tlbie** operation and the content of the TLB entry.

The Power10 core does not support the **tlbia** instruction. To purge the entire TLB, see *Section 5.10.22 TLB Invalidate All (tlbia) Instruction* on page 147.

Upon power-on, the Power10 core initializes each TLB entry to the invalid state.



### 5.10.11 Instruction, Data, and Translation Caches

Per the SMT4-core-resource, the Power10 core supports two effective address tagged L1 caches: one for instruction accesses and one for data accesses and a single 64-way, fully-associative effective-to-real-address translation (ERAT) cache that is shared between the instruction and data accesses. The ERAT is accessed upon instruction cache and data cache misses. An ERAT hit produces a host real address (hRA) that is then presented to either the L2 cache (for cacheable accesses) or the non-cacheable unit (NCU) for non-cacheable accesses. An ERAT miss results in a request to the memory management unit (MMU) to translate a given effective address (EA) into a host real address (hRA for radix or RA for HPT). The MMU either returns the translation mapping and associated attributes to the ERAT and caches or it presents the appropriate exception resulting an interrupt once the instruction that initiated the access becomes the next-to-complete instruction in program order.

#### 5.10.11.1 Instruction Cache

Each instruction cache entry is tagged using the following information:

- EA(0:56)
- Context Tag(0:3)
- MSR[HV, PR, IR, S, LE]

Instruction cache entries, that do not share the same translation context result in duplicate (synonym) entries, are tracked in two separate entries of the I-cache regardless of whether or not they map to the same hRA. The hardware ensures that all hRA synonyms are kept coherent.

Because the instruction cache is effective address indexed; whenever translation changes occur, instructions that are required by the Power ISA to invalidate cached translation information are also required to invalidate the pertinent lines in the instruction cache.

When an **sbia** instruction executes, all instruction cache lines for the thread performing the **sbia** instruction are invalidated. Similarly, when an **sbie** instruction executes, all instruction cache lines indexed by effective addresses that are being invalidated by the **sbie** instruction for the thread performing the **sbie** instruction are invalidated. For each **sbie** instruction, the following address ranges are searched in the instruction cache for the following forms:

- **sbie** for 1 TB segment – invalidate hardware thread translations when EA(0:23) match
- **sbie** for 256 MB segment – invalidate hardware thread translations when EA(0:35) match

The **sbieg** and **sbiag** instructions (that is, the global forms of the SLB invalidate instructions) will not invalidate any cache lines in the I-cache in the Power10 processor core.

The **tlbie** and **tlbiel** instructions filter invalidations based on context tags. Furthermore, the **tlbie** and **tlbiel** instructions invalidate all instruction cache lines that have a match according to the following criteria:

- EA(36:51) are matched for HPT **tlbie(I)** 4 KB page
- EA(12:51) are matched for Radix **tlbie(I)** 4 KB page
- EA(36:47) are matched for HPT **tlbie(I)** 64 KB page
- EA(12:47) are matched for Radix **tlbie(I)** 64 KB page
- EA(12:42) are matched for Radix **tlbie(I)** 2 MB page
- EA(36:39) are matched for HPT **tlbie(I)** 16 MB page



- EA(12:33) are matched for Radix **tlbie(I)** 1 GB page
- EA(24:29) are matched for HPT **tlbie(I)** 16 GB page

When a thread executes an **mtiamr** instruction, all instruction cache lines for that thread are invalidated. If that line is accessible by multiple threads (see *I-Cache Line Sharing* on page 139), the eviction applies to those threads as well.

Upon power-on, the I-cache is set to the invalid state.

#### *I-Cache Line Sharing*

In radix translation mode, instructions in the L1 cache are shared between threads when they have the same identified translation context, including when sharing the same LPID/PID. For HPT translation mode, instructions in the L1 cache are shared between threads when identified by an 8-entry shared instruction address alias table (SIAAT) that tracks instruction granules of 4 KB or 64 KB in size, regardless of the architected page size specified in the SLB or PTE. Invalidation of instruction alias table entries does not invalidate the cache-line state or change the ability of a cache line to be shared.

The SIAAT follows the same invalidation behavior as the I-cache. However, unlike the I-cache, invalidates to the SIAAT resulting from the execution of **mtpidr**, **mtlpidr** or **mtiamr** are specific to the thread executing those instructions and will not invalidate sharing between threads that are unaffected by the special purpose register (SPR) updates.

#### **5.10.11.2 Data Cache**

Each data cache entry is tagged using the following information:

- EA(0:56)
- Context Tag(0:3)
- MSR[HV, DR, S, LE]

Newly referenced D-cache lines that map to the same hRA as an existing entry but do not share the same identified translation context or were not identified as sharable, result in a cache eviction and replacement of the synonym entry.

Because the data cache is effective address indexed; whenever translation changes occur, instructions that are required by the Power ISA to invalidate cached translation information are also required to invalidate the pertinent lines in the data cache.

When an **sibia** or **slbie** instruction executes, all data cache lines for which the required context match are invalidated. For each **slbie** instruction, the following address ranges will be searched in the data cache for the following forms:

- **slbie** for 1 TB segment – invalidate hardware thread translations when EA(0:23) match
- **slbie** for 256 MB segment – invalidate hardware thread translations when EA(0:35) match

The **slbiag** and **slbieg** instructions (that is, the global forms of the SLB invalidate instructions) will not invalidate any cache lines in the D-cache in the Power10 core.

The **tlbie** and **tlbiel** instructions filter invalidations based on context tags. Furthermore, the **tlbie/tlbiel** instructions invalidate all D-cache lines that have a match according to the following criteria:



- EA(36:51) are matched for HPT **tlbie(l)** 4 KB page
- EA(12:51) are matched for radix **tlbie(l)** 4 KB page
- EA(36:47) are matched for HPT **tlbie(l)** 64 KB page
- EA(12:47) are matched for radix **tlbie(l)** 64 KB page
- EA(12:42) are matched for radix **tlbie(l)** 2 MB page
- EA(36:39) are matched for HPT **tlbie(l)** 16 MB page
- EA(12:33) are matched for radix **tlbie(l)** 1 GB page
- EA(24:29) are matched for HPT **tlbie(l)** 16 GB page

When a thread executes either an **mtdawr[0/1]** or **mtdawrx** instruction, all data cache lines for that thread are invalidated.

Upon power-on, the D-cache is set to the invalid state.

#### *D-Cache Line Sharing*

Entries in the D-cache can be shared between any of the threads in the SMT4-core-resource using various hardware mechanisms, see *Section 3.4.2 Data and Instruction Sharing* on page 53. For radix translation, threads with the same translation context, such as with the same LPID/PID, are granted the ability to share a cache line on installation. Other sharing mechanisms can promote cache lines to a shared state when an alias is detected at access time.

Cache lines for the following conditions are always shared between threads:

- Radix Host Quadrant 3: LPCR[HR] = '1' and EA(0:1) = '11' and MSR[HV, DR] = '11'
- Hypervisor Real Mode: MSR[S, HV, DR] = '010'
- Ultrervisor Real Mode: MSR[S, HV, DR] = '110'

Whenever a thread performs an **mtpidr**, **mtlpidr**, or **mtdawr** instruction, or whenever a change is made to a given thread's MSR value, the pertinent entries of the alias table are invalidated for that thread. This in turn revokes any access to thread sharing in the D-cache directory that was using the invalidated context alias tag entries. However, thread access to individual cache lines is still possible for threads not impacted by the context change.

When processing a **tlbie**, **tlbiel**, **slibie**, or **slibia** instruction, the alias table is searched similarly to the D-cache for invalidation criteria on the effective address. This in turn revokes any access to thread sharing in the D-cache directory that was using the invalidated context alias tag entries.

#### **5.10.11.3 Effective-to-Real Address Translation Cache**

The Power10 core includes a 64-entry, fully-associative effective-to-real address translation (ERAT) for fast translation of instruction or data effective addresses into real addresses. The ERAT is dynamically shared between all threads on a given core and the entries are tagged using the following information:

- EA(0:*p*) – see *Table 5-20*
- Context Tag(0:3)
- MSR[HV, DR, S]

The ERAT supports the following page sizes, where  $p$  is defined as shown in *Table 5-20* on page 141.

*Table 5-20. Supported Page Sizes Based on Value of “p”*

Page Size	Value of $P$	Radix Behavior	HPT Behavior
4 KB	51	<ul style="list-style-type: none"> <li>Flattened for nested radix if smaller page size of guest and host is 4 KB</li> <li>Normal translation for single-level radix</li> </ul>	<ul style="list-style-type: none"> <li>Normal for all HPT translations</li> </ul>
64 KB	47	<ul style="list-style-type: none"> <li>Flattened for nested radix if smaller page size of guest and host is 4 KB</li> <li>Normal translation for single-level radix is 64 KB</li> </ul>	<ul style="list-style-type: none"> <li>Normal for all HPT translations</li> </ul>
2 MB	42	<ul style="list-style-type: none"> <li>Flattened for nested radix if smaller page size of guest and host is 2 MB</li> <li>Normal translation for single-level radix is 2 MB</li> </ul>	<ul style="list-style-type: none"> <li>Unused for all HPT translations</li> </ul>
16 MB	39	<ul style="list-style-type: none"> <li>Flattened for nested radix if guest and host page sizes are both 1 GB. Entries are broken into multiple 16 MB entries.</li> <li>Single-level radix 1 GB entries broken into multiple 16 MB entries.</li> </ul>	<ul style="list-style-type: none"> <li>Normal for all HPT translations if 16 MB page size</li> <li>16 GB page sizes are broken into multiple 16 MB ERAT entries</li> </ul>

Hypervisor real mode accesses are required to hit in the ERAT. All accesses made by the processor in hypervisor real mode create a 2 MB page entry in the ERAT.

When the **sbia** instruction executes, ERAT entries are invalidated according to the *Power ISA Operating Environment Architecture - Book III (version 3.1B)* specifications for invalidating implementation-dependent lookaside buffers.

When the **sbie** instruction executes, all cache lines are invalidated according to the *Power ISA Operating Environment Architecture - Book III (version 3.1B)* specification for invalidating implementation-dependent look-aside buffers. For each **sbie** instruction, the following address ranges are searched in the ERAT for the following forms:

- sbie for 1 TB segment – invalidate hardware thread translations when EA(0:23) match
- sbie for 256 MB segment – invalidate hardware thread translations when EA(0:35) match

The **slbieg/slbiag** instruction never invalidate the ERAT entries in the Power10 processor core.

Furthermore, the **tlbie/tlbIEL** instructions invalidate all ERAT entries that have a match according to the following criteria. .

- EA(36:51) are matched for HPT **tlbie(I)** 4 KB page
- EA(12:51) are matched for Radix **tlbie(I)** 4 KB page
- EA(36:47) are matched for HPT **tlbie(I)** 64 KB page
- EA(12:47) are matched for Radix **tlbie(I)** 64 KB page
- EA(12:42) are matched for Radix **tlbie(I)** 2 MB page
- EA(36:39) are matched for HPT **tlbie(I)** 16 MB page
- EA(12:33) are matched for Radix **tlbie(I)** 1 GB page
- EA(24:29) are matched for HPT **tlbie(I)** 16 GB page

The **mtdawr[0/1]**, **mtdawrx**, and **mtiamr** instructions have no effect on the contents of the ERAT.



Upon power-on, the ERAT is set to the invalid state.

### 5.10.12 Segment Lookaside Buffer

For HPT translation, the Power10 core contains a unified (combined for both instruction and data), 32-entry, fully-associative SLB per thread. Although the *Power ISA (Version 3.1B)* supports both a hardware managed SLB, which caches the in-memory segment table (that is, UPRT = '1') and a strictly software-managed SLB (UPRT = '0'), the Power10 core only supports UPRT = '0' when performing HPT translation. Therefore, software must set the LPCR[UPRT] = '0' when using HPT translation or the results are undefined.

While the **slbieg** instruction does not invalidate SLB entries in the processor core when UPRT = '0', it is used to manage STEs cached by the NMMU. Outstanding **slbieg** instructions are ordered by the **slbsync** instruction per the *Power ISA (Version 3.1B)*. When the LPCR[UPRT] = '0', the segment table is not searched and all 32 entries for each thread can only be updated by software by using the **slbmte** instruction.

Information derived from the SLB can also be cached in the I-ERAT or the D-ERAT along with information from the TLB. As a result, many of the SLB management instructions have effects on the ERATs, as well as on the SLB itself.

The Power10 core supports both 256 MB and 1 TB segment sizes. Bit 0 of the SLB[B] field is ignored by the Power10 core and should always be set to '0' per the Power ISA for unimplemented segment-size encodings.

Because the SLB is managed by software (the operating system) either via the segment table or bolted entries, it is possible that multiple entries can be incorrectly set up to provide translations for the same effective address. If an effective address is translated by more than one SLB entry (that is, the ESID fields of the entries are identical or overlap), a machine check interrupt results with an indication that a parity error occurred when the SLB was accessed. When this happens the hardware logically ORs the data in the conflicting entries. The machine check handler can look at the SLB contents to try to determine if conflicting entries have been provided. When a parity error occurs not due to multiple entries, the entire SLB must be reloaded because the DAR does not contain an address indicating which entry caused the parity error. If the source of the error was due to multiple entries, the conflicting entries must be corrected for the translation to proceed, which might also be accomplished by reloading the entire SLB with good entries.

### 5.10.13 Discontinued Translation Support Items

#### 5.10.13.1 Address Space Register

Not to be confused with the Access Segment Descriptor Register (ASDR), the Address Space Register (ASR) has been removed from the *Power ISA (Version 3.1B)* and is not supported by the Power10 core. The ASDR is supported as described in the *Power ISA (Version 3.1B)*.

### 5.10.14 Block Address Translation

Although this facility existed in earlier versions of the architecture, it is no longer part of the *Power ISA (Version 3.1B)*. As a result, the Power10 core does not support block address translation.



#### 5.10.14.1 Support for 32-Bit Operating Systems

The Power10 processor does not support the optional bridge facility and instructions for 64-bit implementations described in the Bridge-to-SLB Architecture section of the *Power ISA Operating Environment Architecture - Book III (version 3.1B)*.

As a result, the following instructions are not supported in the Power10 processor:

- **mtrs** - Move to segment register
- **mtsrin** - Move to segment register indirect
- **mfsr** - Move from segment register
- **mfsrin** - Move from segment register indirect

#### 5.10.14.2 Real Mode

The Power10 core does not support real mode accesses that used the Real Mode Offset Register (RMOR) and Real Mode Limit Selector (RMLS) on previous generation processors. As such, per the *Power ISA (Version 3.1B)*, LPCR[0] is considered reserved and, when using HPT translation, the Power10 core behaves like previous generation processors (such as the POWER8 core) did when LPCR[0] = '1'. In other words, nonhypervisor real mode accesses for HPT translation are always treated as virtual real-mode accesses as per the *Power ISA (Version 3.1B)*. When radix translation in guest real mode (IR = '0' or DR = '0') is being used, the guest EA (gEA) equals the guest RA (gRA), which is then translated by the partition-scoped radix trees.

#### 5.10.15 Reference and Change Bits

When performing radix translation, the Power10 hardware triggers the appropriate interrupt (DSI, HDSI, ISI, or HISI) as defined in the *Power ISA (Version 3.1B)* for the mode and type of access whenever Reference (R) and Change (C) bits require setting in either the guest or host page-table entry (PTE). When performing HPT translation, the hardware performs the R and C bit updates non-atomically (same behavior as the POWER9 processor).

For HPT PTEs, the W and M bits in the PTE are assumed to be '01' respectively. If the change bit is updated, the W and M bit in the PTE are set to '01' respectively by the hardware.

The Power10 core can speculatively set R bits in the PTE. Whether or not this feature is enabled is under firmware control which determines whether speculative load instructions should reload the TLB in the event of a miss. If firmware is set to allow this behavior, the reference bit can be set on behalf of speculative loads (that is, loads that never actually complete from the program's perspective). In some rare circumstances, the Power10 core can speculatively set the page table entry C bit.

In the Power10 core, load instructions that miss the TLB miss and result in an exception (for example, storage protection type DSI/HDSI, DAWR, DSI/HDSI, cause an I = '1' DSI/HDSI, or cause an alignment interrupt) may still set the reference bit for the subject page. Similarly, store type instructions resulting in either a DSI or HDSI due to either a storage key protection, a DAWR match, or by accessing I = '1' page, or result in an alignment interrupt, set the change bit. However, the change bit is not set if a store instruction is denied access by page protection exception.



### 5.10.16 Storage Protection

For HPT translation, the *Power ISA (Version 3.1B)* states that whether an instruction fetch is permitted from a page marked “no access” is implementation dependent. In the Power10 core, these instruction fetches are permitted to continue without signaling an exception. The AMR, AMOR, UAMR (same physical SPR as AMR but accessible when PR = 1), and UAMOR are all implemented as 64-bit SPRs per the *Power ISA (Version 3.1B)*. The IAMR, despite being defined in the *Power ISA (Version 3.1B)* as a 64-bit SPR for which bit 0 of each key is reserved, physically implements only the odd number bits and is thus specified as being a 32-bit SPR in *Table 5-7* on page 106.

### 5.10.17 Ultravisor Real Mode Storage Control

The Power10 core supports the ability to control cacheability of data and instruction accesses while in hypervisor real mode based on the history block (also known as, page-based) mechanism described in the *Power ISA (Version 3.1B)* section on the Hypervisor Real Mode Storage Control (RMSC). Accesses performed in ultravisor real mode are cached in the ERAT’s, instruction caches and data caches as either 4 KB or 2 MB pages under control of a figtree dial. Note that the URMOR[43:51] are not implemented even though the page size is selectable down to 4 KB. See *Section 5.9.3 Ultravisor Real Mode Addressing Using URMOR* on page 122 for more information on the URMOR.

The Power10 core supports RMSC for data storage and instruction storage. The real memory in a system is often noncontiguous and the hypervisor data and instruction storage accesses can be scattered across the address space. The page-based RMSC architecture and implementation allows speculative access safely in system memory. The first time the HV = ‘1’ access is made in DR = ‘0’ and IR = ‘0’ mode, it is done nonspeculatively. After the first access to a given real page, a D-ERAT entry or an I-ERAT entry is established. If the first access to the page was cacheable, the page is installed in the ERAT with IG = ‘00’ and thus, all subsequent accesses to said page can be performed speculatively while still ensuring that the access is made to system memory. However, if the first access to the page was a noncacheable access, the page is installed in the ERAT with IG = ‘11’ and thus, all subsequent accesses to that page are considered noncacheable and are performed nonspeculatively as well.

If a page is already installed in the ERAT as IG = ‘00’ and a subsequent caching-inhibited load or store instruction (for example, **Ibzci**, **stbcix**) accesses that same page, a DS1 is taken with DSISR[62] set to ‘1’.



### 5.10.18 Hypervisor Real Mode Storage Control

The Power10 core supports the ability to control cacheability of data and instruction accesses while in hypervisor real mode based on the history block (also known as, page-based) mechanism described in the *Power ISA (Version 3.1B)* section on the Hypervisor Real Mode Storage Control (RMSC). Accesses performed in hypervisor real mode are cached in the ERAT for instruction and data accesses respectively as 2 MB pages. See *Section 5.9.4 Hypervisor Real Mode Addressing Using HRMOR* on page 123 for more information on the HRMOR.

The Power10 core supports RMSC for data storage and instruction storage. The real memory in a system is often noncontiguous and the hypervisor data and instruction storage accesses can be scattered across the address space. The page-based RMSC architecture and implementation allows speculative access safely in system memory. The first time the HV = '1' access is made in DR = '0' and IR = '0' mode, it is done nonspeculatively. After the first access to a given real page, a D-ERAT entry or an I-ERAT entry is established. If the first access to the page was cacheable, the page is installed in the ERAT with IG = '00' and thus, all subsequent accesses to said page can be performed speculatively while still ensuring that the access is made to system memory. However, if the first access to the page was a noncacheable access, the page is installed in the ERAT with IG = '11' and thus, all subsequent accesses to that page are considered noncacheable and are performed nonspeculatively as well.

If a page is already installed in the ERAT as IG = '00' and a subsequent caching inhibited load or store instruction (for example, **IbzciX** or **stbcix**) accesses that same page, a DSIR[62] is taken with DSISR[62] set to '1'.

### 5.10.19 Storage Access Modes (WIMG and ATT Bits)

Because the Power10 processor supports both HPT and radix translation, two methods of specifying storage attributes on a per page basis exist in the *Power ISA (Version 3.1B)*. For HPT, the WIMG bits determine this. For radix, the ATT bits determine this and the *Power ISA (Version 3.1B)* shows how ATT values correspond to their HPT WIMG equivalent values. The remainder of this section discusses storage attributes in HPT terms, but the radix ATT equivalent values also apply.

The Power10 core always assumes W = '0' and M = '1' independent of the value of these bits in the page table entry. For HPT PTEs, when the hardware is performing a change bit update, it writes the W and M bits as W = '0' and M = '1'. Per the Power ISA, accessing a page as both I = '0' and I = '1' is boundedly undefined. Software should avoid aliasing the I-bit on a page basis. Failing to do so can result in cache paradox situations, which can lead to memory corruption.



Table 5-21 summarizes the treatment of the WIMG bits in the Power10 core.

Table 5-21. WIMG Bits

ATT	WIMG	Description
00	x0x0	Treated as WIMG = '0010'
01	1110	Reserved. <b>Note:</b> This was formerly the SAO setting in the Power ISA. However, support for SAO was removed in <i>Power ISA (Version 3.1B)</i> . The Power10 core did not remove SAO support; however, the Power10 nest did. SAO should be considered not supported in the Power10 processor.
10	x1x1	Treated as WIMG = '0111'
11	01x0	Treated as WIMG = '0110'

For the noncacheable unit (NCU), the IG combination has the following meaning in the Power10 core to control the store ordering and store gathering (see Table 5-22).

Table 5-22. IG Bits

ATT	IG	Description
10	11	No gather, no reorder in NCU is allowed
11	10	Gather, reorder in NCU is allowed

In IG = '11' mode, caching-inhibited loads cannot be reordered relative to other caching-inhibited loads, and caching-inhibited stores cannot be reordered relative to other caching-inhibited stores. Caching-inhibited loads can be reordered relative to caching-inhibited stores and vice-versa. (If it is necessary to maintain ordering between loads and stores, barrier instructions must be used.) There is no defined ordering between caching-inhibited load or store operations from different threads.

In IG = '11' mode, gathering is not permitted for either load or store operations within or between threads.

In IG = '10' mode, caching-inhibited loads or stores from a given thread can be gathered and can be reordered. This mode allows for higher performance with a certain loss of control of the order in which the operations are completed or whether operations are gathered (barriers can be used where necessary to re-establish order). There is no defined ordering between caching-inhibited load or store operations from different threads.

### 5.10.20 Speculative Storage Accesses

The Power10 core can execute load instructions to nonguarded storage speculatively. This can occur when a load instruction is encountered on a predicted branch path or when a logically preceding instruction causes an interrupt. As a result, it is possible for a speculative load that misses in the on-chip cache hierarchy to initiate an external storage request even if that load instruction is not actually executed as part of the true instruction stream.



### 5.10.21 TLB Invalidate Entry (tlbie and tlbiel) Instruction

See *Section 5.10 Translation Architecture* on page 124 for details regarding whether **tlbie** and **tlbiel** are considered privileged or hypervisor privileged instructions.

See *Section 5.10.5 tlbie and tlbiel Instruction Format and Operands*, *Section 5.10.6 Radix Translation*, and *Section 5.10.9 Hashed Page Table (HPT) Translation* for details on **tlbie** and **tlbiel** instruction usage.

The *Power ISA (Version 3.1B)* describes a number of cases for **tlbie** and **tlbiel** as invalid forms. See *Section 5.10.28.7 Machine Check Interrupt* on page 158 for details on these cases.

The Power10 core truncates RS[0:31] and RS[32:63] to the supported size of PID and LPID respectively. No interrupt is generated for values that exceed the implement PID and LPID sizes. See *Section 5.10.3 Process Identification Register (PIDR)* and *Section 5.10.2 Logical Partition Identification Register (LPIDR)* for the details regarding the number of supported PID and LPID sizes respectively.

The Power10 core ignores RB[54] when effR = 0.

The Power10 core reports a machine check interrupt for unsupported AP and LP values specified in the RB register. See *Section 5.10.6.2 tlbie and tlbiel Encodings for Radix Translations* on page 132 and *Section 5.10.10.3 tlbie and tlbiel Usage for HPT Translations* on page 136 for details.

### 5.10.22 TLB Invalidate All (tlbia) Instruction

The **tlbia** instruction is not implemented in the Power10 core and if detected causes a hypervisor emulation assistance interrupt. If hypervisor software wishes to purge the entire TLB irrespective of the LPAR ID for a given core, it can do so by gathering all the threads on that core into hypervisor real mode and performing the following sequence:

```
tlbiel with IS = 3, SET = 0, RIC = 2, PRS = 0, R = 1 (purge TLB of partition scoped entries)
tlbiel with IS = 3, SET = 0, RIC = 2, PRS = 1, R = 1 (purge TLB of process scoped entries)
eieio
tlbsync
ptesync
```

### 5.10.23 TLB Synchronize (tlbsync) Instruction

On a given thread, the **tlbsync** instruction is used to synchronize the completion of the **tlbie** instruction. Only one **tlbsync** instruction is required to synchronize the completion of a group of **tlbie** instructions. See *Section 5.10 Translation Architecture* on page 124 for details regarding when **tlbsync** is considered a privileged or hypervisor privileged instruction. The instruction is otherwise implemented as described in the *Power ISA (Version 3.1B)*.

### 5.10.24 SLB Synchronize (slbsync) Instruction

On a given thread, the **slbsync** instruction is used to synchronize the completion of both the **slbieg** or the **slbiag** instruction. Only one **slbsync** instruction is required to synchronize the completion of a group of **slbieg** instructions. See *Section 5.10 Translation Architecture* on page 124 for details regarding when **slbsync** is considered a privileged or hypervisor privileged instruction. The instruction is otherwise implemented as described in the *Power ISA (Version 3.1B)*.



### 5.10.25 Support for Store Gathering

The Power10 core performs gathering of cacheable stores to reduce the store traffic into the L2 cache. Gathering of caching-inhibited stores is also supported and can be disabled with a mode bit in the noncacheable unit (NCU) configuration register.

### 5.10.26 Cache Coherency Paradoxes

Accesses to a given cache line as both cacheable and caching inhibited are not supported in either the *Power ISA (Version 3.1B)* or the Power10 chip. Because the value of the I-bit is cached by the ERATs inside the processor core, cacheable accesses can be performed speculatively and thus, software should avoid aliasing the I-bit (that is, the caching-inhibited bit) on a per page basis. Failure for software to adhere to this restriction can lead to cache corruption.

### 5.10.27 Handling Parity Error, Multi-Hit, and Uncorrectable Errors

#### 5.10.27.1 Parity Error

If there is a parity error in the D-cache; I-cache; ERAT; TLB or several other register files; SRAM dataflow or control structures (but not the SLB); the Power10 core sets the relevant FIR bit and initiates the instruction retry and recovery (IRR) process to “clean up” all the architected states and flush the caches, ERATs, and TLB, but keeps the SLB as is. Software restores the SLB. After the recovery process, a hypervisor maintenance interrupt (HMI) is generated. On a successful recovery, the HMER indicates a successful recovery.

If the same parity error occurs several times and reaches a threshold, the hypervisor can decide that the core is nonfunctional. The threshold counter is maintained by the hypervisor in software.

HID[5] must be set to ‘0’; otherwise, processor recovery does not work.

**Note:** The IRR process is engaged for detection of any recoverable parity error in the core or due to the firing of a control checker.

There is a separate FIR bit and FIR extension bits for a parity error in the I-cache, D-cache, SLB, ERAT, I-ERAT, TLB, and a few other structures. For all the other register files, there is one shared FIR bit to indicate parity error.

#### 5.10.27.2 Multi-Hit

If there is a multi-hit in the ERAT, TLB, or SLB, the core finishes the operation with a machine check interrupt and sets the proper DSISR bit to indicate where the multi-hit was detected.

A multi-hit in the ERAT and SLB can occur due to a hardware failure. Multi-hit means more than one entry matches the EA in the ERAT (ESID in the case of an SLB). Because of their CAM structure, the result is a “bitwise logical or” of the RA of the multiple entries (VSID in case of SLB). Because of this “bit-wise logical or”, a multi-hit is very likely to generate a parity error as well.

Because the SLB is managed by software with the Power ISA, a software bug can result in a multi-hit in SLB structures. There is no known case of a multi-hit in the ERAT that can produce a wrong result.

There are separate FIR bits for a multi-hit in the ERAT, TLB, and SLB.

### 5.10.27.3 Both Multi-Hit and Parity Error

If both multi-hit and parity errors happen in the ERAT or TLB, the processor core initiates an IRR process. No machine check is presented. However, after the recovery operation, the processor core provides an HMI interrupt.

For an SLB, any error causes the processor to take a machine check interrupt. The FIR bit setting indicates both multi-hit and parity error.

### 5.10.27.4 Uncorrectable Error Handling

If there is an uncorrectable error (UE) for a translate or a load operation, the instruction finishes with a machine check indication to the ISU. The instruction is flushed and re-executed without generating any machine check, and a counter is maintained to see how many UEs occurred. If the UE occurs more than a threshold, a machine-check interrupt is taken. For caching-inhibited load operation, a machine-check interrupt is taken on the first occurrence of the UE.

If there is a UE for a store operation, an asynchronous machine check interrupt results.

For the instruction side (I-side), if an instruction is executed and in the non-speculative path, only then is it treated as a UE. Otherwise, the I-side UE handling mechanism is similar to the data side (D-side).

The core provides the EA of the LSU operation that caused the UE in the DAR register. For a UE detected by the IFU for instruction fetches, SRR0 is set to the EA.

Table 5-23 summarizes how the Power10 processor handles parity, multi-hit, and uncorrectable errors.

Table 5-23. Power10 Behavior on Parity Error, Multi-Hit, and Uncorrectable Error Summary

Structure	Parity Error	Multi-Hit	Both Parity Error and Multi-Hit	Uncorrectable Error (UE)
SLB: I-side translation	MC, SRR1, SRR0	MC, SRR1, SRR0	MC, SRR1, SRR0	N/A
SLB: D-side translation, SLBFEE, MFSLB	MC, DSISR, DAR	MC, DSISR, DAR	MC, DSISR, DAR	N/A
TLB: I-side translation	IRR, HMI	MC, SRR1, SRR0	IRR, HMI	N/A
TLB: D-side translation, MFTLB	IRR, HMI	MC, DSISR, DAR	IRR, HMI	N/A
ERAT	IRR, HMI	MC, DSISR, DAR	IRR, HMI	N/A
Tablewalk: I-side initiated	IRR, HMI	N/A	N/A	MC, SRR1, SRR0
Tablewalk: D-side initiated	IRR, HMI	N/A	N/A	MC, DSISR, DAR
Load	IRR, HMI	N/A	N/A	MC, DSISR
CI Load	MC, DSISR	N/A	N/A	MC, DSISR
Store	IRR, HMI	N/A	N/A	MC, DSISR
Instruction fetch	IRR, HMI	N/A	N/A	MC, SRR1, SRR0
Any other structure (ERAT, other Register file, I-cache, D-cache and other SRAMs, data-flow hardware control checker)	IRR, HMI	N/A	N/A	N/A

**Notes:**

1. SRR0, SRR1, DSISR, DAR are various SPRs that are set on a machine-check interrupt.
2. In the TLB, a multi-hit cannot generate a parity error, but a parity error can generate a multi-hit. In the SLB and ERAT, a multi-hit probably generates a parity error.



### **5.10.27.5 TLB Parity Error and Multi-Hit Action**

Parity = 0 and Multi-hit = 0: No action.

Parity = 1 and Multi-hit = 0: Parity error detected, IRR followed by HMI (no machine check).

Parity = 0 and Multi-hit = 1: This case is probably caused by software setting up two TLB entries pointing to the same VSID.

Parity = 1 and Multi-hit = 1: Probably multiple bits flipped due to a soft-error that caused the parity error but also made two VSIDs look the same. The Power10 core does IRR and then HMI.

## 5.10.28 Interrupts

### 5.10.28.1 Interrupt Vectors

Exceptions implemented in the Power10 core are listed in *Table 5-24*.

*Table 5-24. Interrupt Vectors*

Exception Type	Exception Value
System Reset	x'00100'
Machine Check	x'00200'
Data Storage Interrupt (DSI)	x'00300'
Data Segment Interrupt	x'00380'
Instruction Storage Interrupt (ISI)	x'00400'
Instruction Segment Interrupt	x'00480'
External Interrupt	x'00500'
Alignment Interrupt	x'00600'
Program Interrupt	x'00700'
Floating-Point Unavailable	x'00800'
Decrementer Interrupt	x'00900'
Hypervisor Decrementer Interrupt	x'00980'
Directed Privileged Doorbell Interrupt	x'00A00'
Reserved	x'00B00'
System Call	x'00C00'
Trace Interrupt	x'00D00'
Hypervisor Data Storage Interrupt (HDSI)	x'00E00'
Hypervisor Instruction Storage Interrupt (HISI)	x'00E20'
Hypervisor Emulation Assistance Interrupt	x'00E40'
Hypervisor Maintenance Interrupt	x'00E60'
Directed Hypervisor Doorbell Interrupt	x'00E80'
Hypervisor Virtualization Interrupt	x'00EA0'
Performance Interrupt	x'00F00'
VMX Unavailable Interrupt	x'00F20'
VSX Unavailable Interrupt	x'00F40'
Facility Unavailable Interrupt	x'00F60'
Hypervisor Facility Unavailable Interrupt	x'00F80'
Directed Ultravisor Doorbell Interrupt	x'00FA0'
Soft Patch Interrupt	x'01500'
Debug Interrupt	x'01600'



#### **5.10.28.2 Alternate Interrupt Location**

Table 5-25 on page 153 summarizes the alternate interrupt location (AIL) and the hypervisor alternate interrupt location (HAIL) effects on interrupt processing when IR and DR have the same value before the interrupt occurs. If IR and DR have different values before the interrupt occurs, the interrupts are taken as if AIL = '00' or as if HAIL = '00', as appropriate for the circumstances, per the *Power ISA (Version 3.1B)*. AIL = '01' and AIL = '10' are not supported in the Power10 processor and should not be used.



Table 5-25. (H)AIL Effects on Interrupt Processing (IR = DR) (Sheet 1 of 2)

Interrupt Name	Init HV	Sets HV = 1?	HV After	IR and DR Before Interrupt	Use AIL, HAIL or Neither?	HAIL Value	AIL Value	Behavior	Comments	Rule #
Machine Check, SRI, HMI	X	Yes	1	X	AIL = HAIL = 0	X	X	IR/DR = 0, no offset added	Treated the same as if AIL = HAIL = 0	1
System Call Vectored	0	No	0	0	AIL = HAIL = 0	X	X	IR/DR = 0, normal effective address (00..0001_7xxx)		2, 9
	0	No	0	1	AIL	X	0	IR/DR = 0, normal effective address (00..0001_7xxx)		2, 7, 11
	0	No	0	1	AIL	X	3	IR/DR = 1, alternate effective address (0xC000_0000_0000_3    LEV    0b0_0000)		2, 7, 11
	1	No	1	0	AIL = HAIL = 0	X	X	IR/DR = 0, normal effective address (00..0001_7xxx)	Because HV = 1 after the interrupt, use HAIL	2, 9
	1	No	1	1	HAIL	0	X	IR/DR = 0, normal effective address (00..0001_7xxx)	Because HV = 1 after the interrupt, use HAIL	2, 6, 10
	1	No	1	1	HAIL	1	X	IR/DR = 1, alternate effective address (0xC000_0000_0000_3    LEV    0b0_0000)	Because HV = 1 after the interrupt, use HAIL	2, 6, 10
Rest of interrupts that (can) set HV = 1: sc(LEV = 1), ext(LPES = 0), HEAI, hyp doorbell, hyp decrementer, HDSI, HISI, hyp virtualization, hyp facility unavailable	0	Yes	1	X	HAIL	0	X	IR/DR = 0, no offset added	Because HAIL = 0, then IR/DR = 0 and no offset	6, 8
	0	Yes	1	X	HAIL	1	X	IR/DR = 1, offset	Because HAIL = 1, then IR/DR = 1 and offset added	6, 8
	1	Yes	1	0	AIL = HAIL = 0	X	X	IR/DR = 0, no offset added	Because IR/DR = 0 before and HV not changing, then IR/DR = 0 and no offset	9
	1	Yes	1	1	HAIL	0	X	IR/DR = 0, no offset added	Because HAIL = 0, then IR/DR = 0 and no offset	6
	1	Yes	1	1	HAIL	1	X	IR/DR = 1, offset	Because HAIL = 1, then IR/DR = 1 and offset added	6, 8

Table 5-25. (H)AIL Effects on Interrupt Processing (IR = DR) (Sheet 2 of 2)

Interrupt Name	Init HV	Sets HV = 1?	HV After	IR and DR Before Interrupt	Use AIL, HAIL or Neither?	HAIL Value	AIL Value	Behavior	Comments	Rule #
Rest of interrupts that preserve HV: DSI, ISI, data/instruction segment, alignment, FP / vector / VSX / facility unavailable, sc(LEV = 0), trace, performance monitor, ext(LPES = 1), program, decremener, priv doorbell	0	No	0	0	AIL = HAIL = 0	X	X	IR/DR = 0, no offset added	Because HV is preserved and IR/DR = 0 before the interrupt, then use AIL = 0 behavior.	9
	0	No	0	1	AIL	X	0	IR/DR = 0, no offset added		7, 11
	0	No	0	1	AIL	X	3	IR/DR = 1, offset		7, 11
	1	No	1	0	AIL = HAIL = 0	X	X	IR/DR = 0, no offset added	Because HV = 1 after the interrupt, use HAIL	9
	1	No	1	1	HAIL	0	X	IR/DR = 0, no offset added	Because HV = 1 after the interrupt, use HAIL	6, 10
	1	No	1	1	HAIL	1	X	IR/DR = 1, offset	Because HV = 1 after the interrupt, use HAIL	6, 10

Rules for Table 5-25 are as follows:

- | Rule # | Description   |
|--------|---|
| 1      | MC, SRESET, and HMI always result in HV = 1 with treatment like the legacy AIL = 0 behavior (should not be a change). |
| 2      | Aside from it being “vectored”, scv in is just like any other interrupt that does not change HV.                      |
| 3      | AIL = 1 and AIL = 2 are reserved.   |
| 4      | HAIL = 0 behavior is like AIL = 0 (when they matter).   |
| 5      | HAIL = 1 behavior is like AIL = 3 (when they matter).   |
| 6      | If HV = 1 after the interrupt, ignore AIL.  |
| 7      | If HV = 0 after the interrupt, ignore HAIL.   |
| 8      | If the interrupt causes HV to change, aside from 1 and 2 above, use HAIL.   |
| 9      | If the interrupt does not change HV and IR/DR = 0, use the old AIL = 0 behavior.                                      |
| 10     | If HV = 1 after the interrupt and IR/DR = 1 before the interrupt, use HAIL (except rule 1 above).                     |
| 11     | If HV = 0 after the interrupt and IR/DR = 1 before the interrupt, use AIL (except rule 1 above).                      |



### 5.10.28.3 Interrupt Definitions

Table 5-26 lists the implemented MSR and SRR1/HSRR1 bits.

Table 5-26. Implementation MSR and SRR1/HSRR1 Bits (Sheet 1 of 2)

Bits	MSR	SRR1/HSRR1
0	SF	SF
1	Reserved	Reserved
2	Not Implemented	Not Implemented
3	HV	HV
4	Not Implemented	Not Implemented
5	Reserved	Reserved
6:28	Not Implemented	Not Implemented
29:30	TS (Transactional State)	TS (Transactional State)
31	TM (Transactional Memory Available)	TM (Transactional Memory)
32	Not Implemented	Not Implemented
33	Not Implemented	Specific Interrupt Information
34	Not Implemented	Not Implemented
35:36	Not Implemented	Specific Interrupt Information
37	Not Implemented	Not Implemented
38	VMX	VMX
39	Not Implemented	Not Implemented
40	VSX	VSX
41	S	Secure
42:47	Not Implemented	Specific Interrupt Information
48	EE	EE
49	PR	PR
50	FP	FP
51	ME	ME
52	FE0	FE0
53:54	TE	TE
55	FE1	FE1
56	US	US
57	Not Implemented	Not Implemented
58	IR	IR
59	DR	DR
60	Not Implemented	Not Implemented



Table 5-26. Implementation MSR and SRR1/HSRR1 Bits (Sheet 2 of 2)

Bits	MSR	SRR1/HSRR1
61	PMM	PMM
62	RI	RI
63	LE	LE

#### 5.10.28.4 Synchronous Interrupts

In addition to the synchronous interrupts described in the *Power ISA (Version 3.1B)*, the Power10 core implements a soft-patch interrupt that can be used by hypervisor-level software to “trap” on particular instructions. When the conditions for trapping on the instruction are met (typically based on the instruction opcode), the soft-patch facility in the hardware generates a soft-patch interrupt and the hardware fetches the interrupt vector located at offset x'01500'. In addition, the hardware updates the Hypervisor Emulation Assist Interrupt Register (HEIR) with the Power ISA instruction. In some cases, the hardware modifies bits of the instruction image. A partial list of notable instructions that exhibit this behavior are listed in *Table 5-27*.

Table 5-27. HEIR Instruction Formatting for Branch-Like Instructions

Instruction	Primary Opcode	Secondary Opcode	Additional Fields
<b>sc or scv</b>	17		
sp_attn	00	256	
<b>rfscv</b>	19	82	
<b>rfid</b>	19	82	
<b>hrfid</b>	19	274	
<b>stop</b>	19	370	
<b>rfebb</b>	19	146	
<b>mtmsr</b>	31	146	L = 0
<b>mtmsrd</b>	31	178	L = 0
<b>ISTAT errors</b>	N/A	N/A	

**Note:** For the instructions listed in *Table 5-27*, HEIR[11:14] are set to all ‘1’s. The remaining bits are set as described in the *Power ISA (Version 3.1B)*.

#### 5.10.28.5 Asynchronous Interrupt Priorities

The Power10 core processes asynchronous interrupts and event-based branches (EBBs) in the following order:

1. System Reset Interrupt
2. Machine Check
3. Imprecise Floating Point Exceptions
4. Hypervisor Maintenance Interrupt
5. Hypervisor Virtualization External Interrupt
6. Mediated External Interrupt

7. Direct External Interrupt
8. Performance Monitor Interrupt
9. Hypervisor Decrementer Interrupt
10. Decrementer Interrupt
11. Hypervisor Doorbell Interrupt
12. Privileged Doorbell Interrupt
13. Event Based Branch

#### **5.10.28.6 System Reset Interrupt**

The system reset interrupt is a nonmaskable, asynchronous interrupt that is caused by an SCOM command for a soft reset.

**Note:** There is no explicit SRESET pin; SRESET must be invoked from the service processor.

*Table 5-28. System Reset Interrupt*

Register	Bits	Description
SRR0	0:63	Set to the effective address of the instruction that the processor would have executed next.
SRR1	0:31	Implemented bits loaded from the MSR.
	32	Set to '0'.
	33	LPAR mode switch occurred while the thread was in power savings mode.
	35:36	Set to '0'.
	42:45	Interrupt cause: 0000 Reserved by pervasive function. 0010 Interrupt caused by SCOM when not in power-saving mode or caused by back-to-back SRESET. 0011 Interrupt caused by hypervisor door bell. 0101 Interrupt caused by privileged door bell. 0100 Interrupt caused by SCOM when in power-saving mode. 0110 Interrupt caused by decrementer wake-up when in power-saving mode. 1000 Interrupt caused by external interrupt wake-up when in power-saving mode. 1001 Interrupt caused by hypervisor virtualization wake-up when in power-saving mode. 1010 Interrupt caused by HMI wake-up when in power saving mode. 1100 Interrupt caused by implementation-specific wake-up when in power-saving mode.
	46:47	Indicates if the interrupt occurs when the processor is in power-saving mode. 00 Interrupt did not occur while the processor was in power-saving mode. 01 Interrupt occurred while the processor was in power-saving mode. The state of all resources was maintained as if the processor was not in power-saving mode. 10 Interrupt occurred while the processor was in power-saving mode. The state of some resources was not maintained but the state of all hypervisor resources, including TB, PURR, and SPURR, was maintained as if the processor was not in power-saving mode and the state of all other resources is such that the hypervisor can resume execution. 11 Interrupt occurred while the processor was in power-saving mode. The state of some resources was not maintained, and the state of some hypervisor resources was not maintained or the state of some resources is such that the hypervisor cannot resume execution.
	62	Loaded from MSR[62] if recoverable. Otherwise, set to zero
	Others	Implemented bits loaded from MSR.



The Power10 core implements a 1-deep queue to remember the reason of a subsequent system reset interrupt while a system reset interrupt is pending. The reason of the most important subsequent system reset interrupt is remembered per the following priority:

1. Hypervisor doorbell-initiated system reset
2. Privileged doorbell-initiated system reset
3. SCOM-initiated system reset
4. HMI-initiated system reset
5. External-initiated system reset
6. Decrementer-initiated system reset
7. Implementation-specific initiated system reset

#### **5.10.28.7 Machine Check Interrupt**

There are several possible causes of machine check interrupts in the Power10 chip, some of which are generally recoverable and some of which are non-recoverable.

The following causes of machine check interrupts are precise and synchronous with the instruction that caused the operation which encountered the error (that is, SRR0 contains the address of the instruction that caused the operation).

1. The detection of either a parity error, or a multi-hit error, or both in the SLB during the execution of a load, store, **slbfee**, or **mfslb** instruction. If the interrupt is caused by a soft error, executing the appropriate sequence of instructions in the machine-check handler program clears the error condition without causing any loss of state, permitting the interrupted program to be resumed if MSR[RI] was a '1' when the instruction that encountered the error was executed.
2. If there is a multi-hit in the ERAT or TLB, the core finishes the operation with a machine-check interrupt and sets the proper DSISR bit to indicate where the multi-hit occurred.
3. If there is an uncorrectable ECC error when a load instruction is executed or when the page table is being searched in the process of translating an address, the instruction finishes with a machine-check indication to the instruction-sequencing unit. The instruction is flushed and re-executed without generating any machine check. A counter is maintained to see how many UEs occurred. If the UE occurs more than a pre-established threshold, a machine-check interrupt is taken.
4. For a caching-inhibited load operation, the machine-check interrupt is taken on the first occurrence of the UE.
5. For the I-side, if an instruction is executed and the instruction is in the nonspeculative path, only then will it be treated as a UE. Otherwise, the I-side UE handling mechanism is similar to the D-side.
6. For the I side, when an instruction fetch causes an out-of-range real address (L2 address error), a machine check interrupt is taken.
7. For the D side, when a load causes an out-of-range real address (L2 address error), a machine check interrupt is taken.
8. For the I and D side, if a tablewalk fetch causes an out-of-range address (L2 address error), a machine check interrupt is taken.
9. Anytime that a **tlbie** or **tlbiel** instruction has either an invalid form or an unsupported page-size encoding that is not supported by the hardware. The **tlbie(I)** instruction encoding cases are outlined in *Appendix D tlbie(I) Encodings* on page 421. The unsupported page-size cases correspond to any page-



size encodings not specified in *Table 5-15* on page 132, *Table 5-17* on page 136, *Table 5-18* on page 136, or *Table 5-19* on page 137.

The following are listed in the *Power ISA (Version 3.1B)* as invalid forms for **tlbie**:

- PRS = 1 and effR = 0 and RIC ≠ 2
- RIC = 1 and effR = 0
- RIC = 3 and effR = 1
- RIC = 1 and IS = 0
- RIC = 2 and IS = 0
- RIC = 3 and IS ≠ 0
- PRS = 0 and IS = 1
- effR = 0 and IS = 1 and RIC ≠ 2
- effR = 0, RIC = 2, PRS = 0, HV = 0, and IS = 2 or 3

The following are listed in the *Power ISA (Version 3.1B)* as invalid forms for **tlbiel**:

- PRS = 1 and effR = 0 and RIC ≠ 2
- RIC = 1 and effR = 0
- RIC = 3
- RIC = 1 and IS = 0
- RIC = 2 and IS = 0
- PRS = 0 and IS = 1
- effR = 0 and IS = 1 and RIC ≠ 2
- effR = 0, RIC = 2, PRS = 0, HV = 0, and IS = 2 or 3

Additionally, the Power10 core considers these two additional cases as invalid forms for both **tlbie** and **tlbiel**:

- L = 1 and IS ≠ 0
- L = 1 and effR = 1

10. During translation, if a host real address is in the control memory address range [RA(11) is a '1'].

11. If any instruction fetch, load or store access is in the control memory address range [RA(11) is a '1'].

In the cases described in items (2), (3), (4), and (5), no state is lost in the processor, but recovery of the correct data might not be possible.

For more traumatic errors or hard errors, these characteristics cannot be reliably provided on a machine check because it is likely that the failure will prevent reliable execution. In such cases, a checkstop will result. Additionally, a machine-check interrupt that occurs when MSR[ME] = '0' results in a checkstop.

In the Power10 core, there is only one cause of an asynchronous machine check interrupt: when a store instruction has an out-of-range real address associated with it. This case is in general a programming error. The core takes a machine check interrupt in this case and sets SRR1 to identify the cause to assist programmers in debugging bad code. All other causes of machine check interrupts are synchronous.

Information about the suspected source of the error condition is logged into either the SRR1 Register, the DSISR Register, or both as defined in *Table 5-29* on page 160 for synchronous and asynchronous machine checks.



*Table 5-29. Synchronous Machine Checks (Sheet 1 of 2)*

Register	Bits	Description																														
SRR0	0:63	Effective address of the next instruction that would have executed if the machine-check interrupt was not taken. For cases where this is a recoverable machine check due to a load that has surfaced an error, this will be the address of the load instruction itself. (The Power10 core allows the instruction to execute to surface the error, but inhibits the commitment of the results.) For cases where this is a recoverable machine check due to an instruction fetch surfacing an error, this will be the address of an instruction that initiated the memory/cache access. for asynchronous machine checks this address is meaningless																														
SRR1	42	When set to 0b1, the machine check interrupt is caused by a load/store detection of error. With the exception of the case where the interrupt results from an out-of-range address error (bad CResp) from a store, the cause of the error is recorded in the DSISR. The address error case is reported in SRR1 as described below for the '1101' case with SRR1[42] = '1'. All other causes of the interrupt, which result in SRR1[42] = '0', result from an instruction fetch.																														
	36, 43:45	<table> <tr><td>0000</td><td>Reserved.</td></tr> <tr><td>0001</td><td>Interrupt caused by a hardware uncorrectable error detected while doing an instruction fetch (but not translation related).</td></tr> <tr><td>0010</td><td>Interrupt caused by an SLB parity error while translating an instruction fetch address.</td></tr> <tr><td>0011</td><td>Interrupt caused by an SLB multiple hit, while translating an instruction fetch address. <b>Note:</b> This condition occurs if the ESID fields of two or more SLB entries contain the same value.</td></tr> <tr><td>0100</td><td>Interrupt caused by an ERAT multi-hit error resulting from an instruction fetch.</td></tr> <tr><td>0101</td><td>Interrupt caused by a TLB multi-hit error detected while translating an instruction fetch address. Note: This condition occurs if an address is mapped to both a small and large page in the SLB. This condition can also occur due to a software bug, when a software-managed TLB mechanism is used.</td></tr> <tr><td>0110</td><td>Interrupt caused by a hardware UE detected while doing a TLB reload for the I-side.</td></tr> <tr><td>0111</td><td>Instruction fetch to control memory address space.</td></tr> <tr><td>1000</td><td>Reserved.</td></tr> <tr><td>1001</td><td>Reserved.</td></tr> <tr><td>1011</td><td>Real address (CResp) error for an instruction fetch.</td></tr> <tr><td>1100</td><td>Real address (CResp) error for an instruction fetch tablewalk.</td></tr> <tr><td>1101</td><td>Asynchronous machine check due to a real address (CResp) error from a store.</td></tr> <tr><td>1110</td><td>Reserved.</td></tr> <tr><td>1111</td><td>I-side tablewalk used a host real address in the control memory address range.</td></tr> </table>	0000	Reserved.	0001	Interrupt caused by a hardware uncorrectable error detected while doing an instruction fetch (but not translation related).	0010	Interrupt caused by an SLB parity error while translating an instruction fetch address.	0011	Interrupt caused by an SLB multiple hit, while translating an instruction fetch address. <b>Note:</b> This condition occurs if the ESID fields of two or more SLB entries contain the same value.	0100	Interrupt caused by an ERAT multi-hit error resulting from an instruction fetch.	0101	Interrupt caused by a TLB multi-hit error detected while translating an instruction fetch address. Note: This condition occurs if an address is mapped to both a small and large page in the SLB. This condition can also occur due to a software bug, when a software-managed TLB mechanism is used.	0110	Interrupt caused by a hardware UE detected while doing a TLB reload for the I-side.	0111	Instruction fetch to control memory address space.	1000	Reserved.	1001	Reserved.	1011	Real address (CResp) error for an instruction fetch.	1100	Real address (CResp) error for an instruction fetch tablewalk.	1101	Asynchronous machine check due to a real address (CResp) error from a store.	1110	Reserved.	1111	I-side tablewalk used a host real address in the control memory address range.
0000	Reserved.																															
0001	Interrupt caused by a hardware uncorrectable error detected while doing an instruction fetch (but not translation related).																															
0010	Interrupt caused by an SLB parity error while translating an instruction fetch address.																															
0011	Interrupt caused by an SLB multiple hit, while translating an instruction fetch address. <b>Note:</b> This condition occurs if the ESID fields of two or more SLB entries contain the same value.																															
0100	Interrupt caused by an ERAT multi-hit error resulting from an instruction fetch.																															
0101	Interrupt caused by a TLB multi-hit error detected while translating an instruction fetch address. Note: This condition occurs if an address is mapped to both a small and large page in the SLB. This condition can also occur due to a software bug, when a software-managed TLB mechanism is used.																															
0110	Interrupt caused by a hardware UE detected while doing a TLB reload for the I-side.																															
0111	Instruction fetch to control memory address space.																															
1000	Reserved.																															
1001	Reserved.																															
1011	Real address (CResp) error for an instruction fetch.																															
1100	Real address (CResp) error for an instruction fetch tablewalk.																															
1101	Asynchronous machine check due to a real address (CResp) error from a store.																															
1110	Reserved.																															
1111	I-side tablewalk used a host real address in the control memory address range.																															
	62	Loaded from MSR[62] if recoverable. Otherwise, set to zero.																														
	others	Implemented bits loaded from MSR.																														

Table 5-29. Synchronous Machine Checks (Sheet 2 of 2)

Register	Bits	Description
DSISR	32:47	All zeros.
	48	Interrupt caused by a UE deferred error, but not for a tablewalk (D-side only). This bit will also come on for a cache inhibited load that has a bad address (cresp)
	49	Interrupt caused by a UE deferred error during a tablewalk (D-side).
	50	Reserved.
	51	Reserved.
	52	Interrupt caused by a ERAT multi-hit.
	53	Interrupt caused by a TLB multi-hit due to translation (D-side only) or MFTLB operation.
	54	Tlbie or tlbIEL programming error.
	55	Interrupt caused by an SLB parity error (translate lookup or <b>mfslbfee</b> ) due to a translation (D-side only), <b>slbfee</b> , or <b>mfslb</b> instruction.
	56	Interrupt caused by an SLB multi-hit (might not be recoverable) for translation (D-side only), <b>slbfee</b> , or <b>mfslb</b> instruction.
	57	Bad real address (CResp) for a load that is not cache inhibited.
	58	Bad address (CResp) for a load or store tablewalk address.
	59	D-side tablewalk used a host real address in the control memory address range.
	60	D-side operand access to control memory address space.
	61:63	Set to zeros.
DAR	0:63	<p>Effective address computed by a load or store instruction that caused the operation that encountered a parity error, or multi-hit, or both in the SLB, or which encountered a multi-hit in the TLB, or encountered a multi-hit in the ERAT, or encountered a UE while attempting to reload a TLB entry. For all other types of machine check interrupts, the DAR is undefined (including the case where the operand of the load instruction contains a UE).</p> <ol style="list-style-type: none"> <li>1. SLB parity error, multi-hit, or both: DAR is loaded with the EA of the target of the load or store instruction that caused the error.</li> <li>2. TLB multi-hit: DAR is loaded with the EA of the target of the load or store instruction that caused the error.</li> <li>3. ERAT multi-hit: DAR is loaded with the EA of the target of the load or store instruction that caused the error.</li> <li>4. UE on D-side tablewalk: DAR is loaded with the EA of the target of the load or store instruction.</li> <li>5. UE on instruction fetch: DAR is undefined.</li> <li>6. UE on I-side tablewalk: DAR is undefined.</li> <li>7. UE on load or store instruction: DAR is undefined (EA is not available in LMQ for loads; therefore, DAR cannot be loaded).</li> <li>8. CResp for load: DAR is set to the EA of the load that caused the error.</li> <li>9. CResp for a dside table-walk: DAR is undefined.</li> <li>10. Host real address to control memory address space: DAR is undefined.</li> <li>11. Tlbie or tlbIELL programming error: DAR is undefined.</li> <li>12. Asynchronous machine checks: DAR is not modified.</li> </ol>

**DSISR Implementation Note:** All the bits have been implemented in hardware.



#### *Machine-Check Interrupt Handler Notes:*

As mentioned previously, the machine check interrupt handler is expected to help hardware recover from certain types of ERAT, TLB, and SLB errors detected by the hardware. In general terms, the interrupt handler must check whether or not the machine check interrupt is recoverable (by looking at the state of the RI bit in SRR1). It must determine the type of error that caused the machine check (by looking at the state of the SRR1 and DSISR Registers). It must flush the contents of the array that reported the detected error (this process is slightly different for each of the possible arrays). Finally, it must return to the interrupted process.

#### **5.10.28.8 Hypervisor Maintenance Interrupt**

The Power10 hypervisor maintenance interrupt is implemented to replace the malfunction alert and thermal interrupt; and to provide support for recovery function. The HMER Register contains the sources of the interrupt, which can be masked by setting the HMEER enable bits to zero. For successful recovery, HMER setting indicates successful recovery.

#### **5.10.28.9 External Interrupt**

An external interrupt is classified as being either a direct external interrupt or a mediated external interrupt. Both cause an interrupt to x'500'.

##### *Direct External Interrupt*

The direct external interrupt is signaled by the assertion of the external interrupt input signal. The external interrupt signal must remain asserted until the processor has actually taken the interrupt. Failure to meet this requirement can lead the processor to not recognize the interrupt request.

When LPES = '0', the following registers are set.

*Table 5-30. Direct External Interrupt (LPES = '0')*

Register	Bits	Description
HSRR0	0:63	Set to the effective address of the instruction that the thread would have attempted to execute next if no interrupt conditions were present.
HSRR1	33:36	Set to '0'.
	42:47	Set to '0'.
	Others	Loaded from the MSR.
MSR	–	See the <i>Power ISA (Version 3.1B)</i> .

When LPES = '1', the following registers are set.

*Table 5-31. Direct External Interrupt (LPES = '1')*

Register	Bits	Description
SRR0	0:63	Set to the effective address of the instruction that the thread would have attempted to execute next if no interrupt conditions were present.
SRR1	33:36	Set to '0'.
	42:47	Set to '0'.
	Others	Loaded from the MSR.
MSR	–	See the <i>Power ISA (Version 3.1B)</i> .

#### *Mediated External Interrupt*

Mediated external interrupts are caused by the LPCR[MER] = '1', when the thread is in privileged (supervisor) or problem state mode.

When LPES = '0', the following registers are set.

*Table 5-32. Mediated External Interrupt (LPES = '0')*

Register	Bits	Description
HSRR0	0:63	Set to the effective address of the instruction that the thread would have attempted to execute next if no interrupt conditions were present.
HSRR1	33:36	Set to '0'.
	42	Set to '1'.
	43:47	Set to '0'.
	Others	Loaded from the MSR.
MSR	–	See the <i>Power ISA (Version 3.1B)</i> .

When LPES = '1', the following registers are set.

*Table 5-33. Mediated External Interrupt (LPES = '1')*

Register	Bits	Description
SRR0	0:63	Set to the effective address of the instruction that the thread would have attempted to execute next if no interrupt conditions were present.
SRR1	33:36	Set to '0'.
	42:47	Set to '0'.
	Others	Loaded from the MSR.
MSR	–	See the <i>Power ISA (Version 3.1B)</i> .



#### 5.10.28.10 Alignment Interrupt

See *Section 5.1.4.1 Alignment Interrupts* on page 77 for details on when the Power10 core takes alignment interrupts. The DAR is updated on an alignment interrupt as described in the *Power ISA (Version 3.1B)*. The DSISR register is not updated on an alignment interrupts per the *Power ISA (Version 3.1B)*.

#### 5.10.28.11 Trace Interrupt

In general, a trace interrupt is taken after every instruction when the MSR[TE] = '10' and after every branch instruction when MSR[TE] = '01'. In particular, for the case where MSR[TE] = '10' before a **mtmsr[d]** instruction is executed that alters the MSR[TE] bits, a trace interrupt also occurs. There are several instructions and conditions for which the preceding statements are not followed. See the *Power ISA (Version 3.1B)* for details. Additionally, a trace interrupt is taken when a CIABR match occurs. After a trace interrupt is taken, SRR0, SRR1, SIAR, and SDAR are set as shown in *Table 5-34*.

*Table 5-34. Trace Interrupt*

Register	Bits	Description
SRR0	0:63	Set as specified in the architecture.
SRR1	0:32	Implemented bits loaded from the MSR.
	33:34	'10'
	35	Set for a load instruction; otherwise, cleared. See note 1.
	36	Set for a store instruction; otherwise, cleared. See note 1.
	37:41	Loaded from the MSR.
	42	Loaded from the MSR, which is an unimplemented bit (therefore, always set to '0').
	43	Set to a '1', if a CIABR trace.
	44:47	Set to '0'.
	48:63	Implemented bits loaded from the MSR.
	1. Bit 35 and 36 are not set if an X-form load string or store string instruction specifies an operand length of 0.	
SIAR	0:63	Set to the effective address of the traced instruction; undefined if a CIABR trace.
SDAR	0:63	If the instruction that took the trace interrupt was a storage access instruction, the SDAR is set to the effective address of the storage access. SDAR is not set if an X-form load string or store string instruction specifies an operand length of 0; undefined if a CIABR trace.

The contents of SIAR and SDAR are undefined until a trace interrupt occurs.

#### 5.10.28.12 Performance Monitor Interrupt

The performance monitor interrupt is signaled when the MSR[EE] bit is set, the MMCR0[PMAE] bit is set, and any of the performance monitor counters overflow (this includes the eight performance counters defined in the SPR space, as well as the counters defined in MMIO space for the nest).

After such an event is detected, the Power10 core waits for previously dispatched instructions to complete, and then takes the interrupt.



### 5.10.28.13 Facility Unavailable Interrupt

The Power10 core implements the facility unavailable interrupt as defined in the *Power ISA (Version 3.1B)*.

### 5.10.28.14 Hypervisor Emulation Assistance Interrupt

The Power10 core implements the hypervisor emulation assistance interrupt as defined in the *Power ISA (Version 3.1B)*. However, the contents of the HEIR for the following instructions differ from the Power ISA instruction described as follows:

**mfblrbe** - Power ISA instruction bits 11:20 are recoded to indicate the internal SPR number assigned to each BHRB entry. SPR d'80' is assigned if the specified entry is outside of the available range. This instruction is sensitive to PCR and therefore can be made illegal.

**clrbhrb** - Power ISA instruction bits 11:20 are recoded to indicate the internal SPR address d'80'. This instruction is sensitive to PCR and therefore can be made illegal.

**bctar, bctarl** - This is what a recoded **bctar** looks like in the HEIR after an illegal instr interrupt, where '-' can be either 0 or 1.

0	1	2	3
01234567890123456789012345678901			
0-100-----11-0----01--10110000-			

This should only match the following ops, none of which go into HEIR. Therefore, assume **bctar** if the above compare matches:

ori\_nop  
ori.0  
ori.1  
oris.0  
oris.1  
subfic.0  
subfic.1

The following instruction fields can be found in the indicated HEIR bits. Note that bo(4) is lost.

1k = HEIR(1)  
bo(0:3) = HEIR(5:8)  
bo(4) = '0'  
bi(0) = HEIR(10)  
bi(1:4) = HEIR(15:18)  
bh(0:1) = HEIR(21:22)

**Stop fetch instructions** - The following instructions have Power ISA instruction bits 11:14 recoded to all '1's to indicate the "stop fetch" function:

scv  
rfscv  
sp\_attn  
rfebb



### 5.10.29 Strong Access Ordering Mode (SAO)

Unlike the POWER9 processor, the Power10 core does not support the SAO mode defined in versions of the Power ISA prior to version 3.1.

### 5.10.30 Graphics Data Stream Support

For caching-inhibited stores, the Power10 core provides store gathering with an intentional stall to maximize the amount of gathering that can occur.

### 5.10.31 Data Address Watchpoint Debug Support

Each Power10 core supports two instances of the Data Address Watchpoint Register (**DAWR0** and **DAWR1**) and the Data Address Watchpoint Register Extension (**DAWRX0** and **DAWRX1**), respectively. In general, the Power10 core implements support for these registers as specified in the *Power ISA (Version 3.1B)*.

The known deviations from the *Power ISA (Version 3.1B)* are as follows:

- In virtual real addressing mode while utilizing HPT translation, the ISA states that the high order 24 bits of the EA should be ignored by the hardware when determining whether a DAWR exception occurs. However, the Power10 core requires those bits to match in order for a DAWR exception to occur.
- In either hypervisor real addressing mode or ultravisor real addressing mode, EA(1:10) are not ignored as stipulated in the ISA in terms of determining whether or not a DAWR exception exists.
- As a consequence of the preceding bullet, if the address specified in **DAWR0** and **DAWR1** would overlap and all other conditions necessary for a match to occur are satisfied by both **DAWRX0** and **DAWRX1**, then the value reported in the **DAR** on the resulting DS1, may not reflect the lowest byte in the combined range as is stipulated in the ISA.

### 5.10.32 Performance Monitoring, Sampling, and Trace

Performance monitoring facilities have been incorporated into the Power10 core to enable the collection of performance related data and instruction traces. In general, the Power10 core supports the recommended architecture for performance monitoring as described in the *Power ISA (Version 3.1B)*.

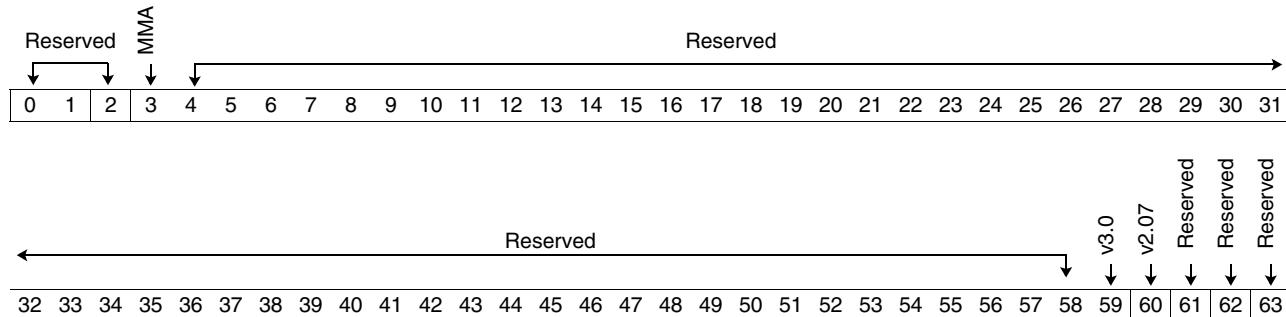
### 5.10.33 Processor Compatibility Mode

The Power10 core implements the Processor Compatibility Register (PCR) mostly as described in the Power ISA to facilitate partition migration for “N - 2” generations of hardware. Setting PCR[59] = ‘1’ disables all problem state instructions and facilities that were added in the *Power ISA (Version 3.1B)*. Thus, setting this bit effectively makes a Power10 core architecturally appear to problem state software as a *Power ISA version 3.0* core (that is, a POWER9 core). Setting PCR[60] = ‘1’ disables all problem state instructions and facilities that were added in *Power ISA (Version 2.07)*. Thus, setting this bit effectively makes a Power10 core architecturally appear to problem state software as a *Power ISA version 2.07* core (that is, a POWER8 core). Therefore, to migrate a partition from a POWER8 (Version 2.07) system to a Power10 (Version 3.1) system, PCR[59:60] should be set to ‘11’. The *Power ISA (Version 3.1B)* stipulates it is the responsibility of software to set both bits in this case. However, the Power10 processor will implicitly set PCR[59] to ‘1’ if software attempts to set only PCR[60] to ‘1’.

The Power10 processor also implements PCR[3] mostly as described in the *Power ISA (Version 3.1B)*. When set to ‘1’, this bit makes the Matrix Multiply Assist (MMA) instructions unavailable in any privilege state. However, note that the Power10 processor sets PCR[3] to a ‘1’ whenever software attempts to set either PCR[59] or PCR[60] to a ‘1’. This is an implementation detail not stipulated in the *Power ISA (Version 3.1B)*. It is possible to set PCR[3] to a ‘1’ while both PCR[59] and PCR[60] are set to a ‘0’.

All other PCR bits are treated as reserved and attempts to set them have no effect on the behavior of the Power10 core. Unless otherwise noted in the text, the PCR only affects instructions and SPR accesses that occur when the processor thread is operating in problem state (that is, MSR[PR] = ‘1’). See *Section 5.7.3.5 Move To/From Special Purpose Register Instructions* on page 104 for more details about how the PCR affects SPR accesses.

Unlike the POWER8 processor, there is no requirement to flush the I-cache after changing the state of the PCR with an `mtspr_PCR`.





## 6. L2 Cache

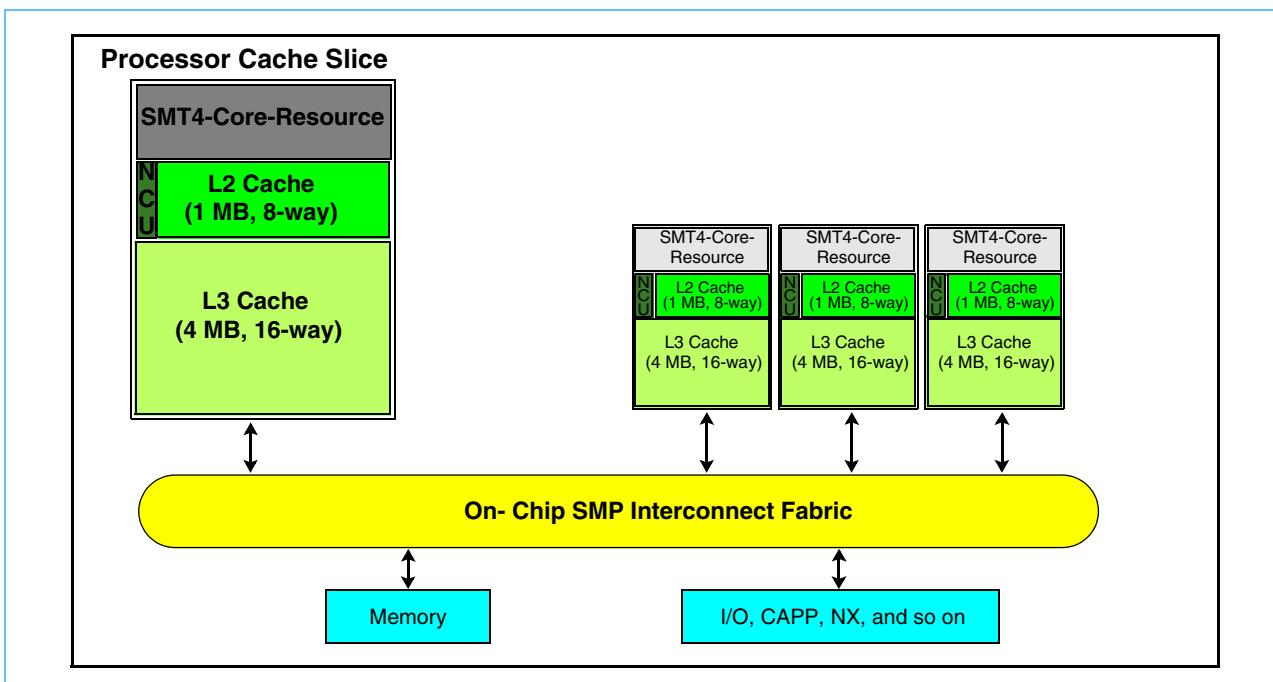
This section details the microarchitecture of the L2 cache for the SMT4 variant of the Power10 processor chip.

### 6.1 Overview

The L2 cache unit is contained within the processor cache slice, which consists of: one Power10 SMT4-core-resource, one 1 MB L2 cache, one 4 MB L3 cache, one NCU, and a portion of the on-chip SMP interconnect fabric logic that is referred to as PBex. The L2 cache is a unified cache that is accessed privately by a Power10 core. The L2 cache maintains full-hardware coherence within the system and can supply cache on-chip intervention data to other cores on this Power10 chip or off-chip intervention data to other Power10 chips. The L2 unit is a store-in cache that is fully inclusive for the Power10 core, which has an L1 D-cache and I-cache (note that the Power10 core is based on a store-through L1 D-cache design). The L2 unit also supports private bus access to a 4 MB L3 cache that is also private to this Power10 core for fast L3 hit data access and for storage of L2 victimization data.

*Figure 6-1 shows a high-level diagram of the Power10 chip with multiple Power10 SMT4 processor cache slices interconnected via the on-chip SMP interconnect fabric.*

*Figure 6-1. Power10 Block Diagram of Multiple SMT4 Processor Cache-Slices Interconnected via the On-Chip SMP Interconnect Fabric*





### 6.1.1 L2 Cache Features

The SMT4 L2 cache features are summarized as follows:

- 1 MB private cache per SMT4-core-resource:
  - 128-byte line, 8-way set associative.
  - Both instruction side (I-side) and data side (D-side) inclusive for a Power10 core.
  - Quad-banked cache design interleaved across a four consecutive cache-line boundary.
  - L2 cache can perform a read from one cache bank while writing to one of the other cache banks.
- 8-way directory, quad-banked multi-ported:
  - One processor read port, two snoop read ports, and one write port per physical bank.
  - The processor port operates at  $\frac{1}{2}$  the processor clock rate into a given bank (initiated on a 2:1 clock boundary).
  - The snoop port into a given bank operates at  $\frac{1}{2}$  the processor clock rate (initiated on a 2:1 clock boundary) allowing for up to four snoops per 2:1 clock across the four banks.
  - The quad-banked directory can initiate:
    - Up to five directory reads in a given 2:1 cycle (four snoop ports and one on processor port).
    - One write in a given processor clock cycle (where directory writes are scheduled on the second half of a 2:1 cycle, such that they never conflict with directory reads).
- $1024 \times 13$ -bit LRU arrays (logical configuration).
  - $2 \times 4$  LRU vector tracking tree with cache invalidate state biasing.
  - Supports LRU, direct map, single-member, and pseudo-random modes.
- Point of global coherency.
- Reservation stations: one per processor thread.
- Support for Power10 Synthetic\_TM core mode.
- Four snoop-bus ports selected by the cache-line “real-address” bits [55:56].
- Hardware directory line delete capabilities to support faulty L2 cache elements.

### 6.1.2 L2 RAS Features

The L2 cache RAS features are summarized as follows:

- Directory-array data protected by SEC/DED ECC
- Directory-array, single-bit, stuck-bit detection and correction
- Directory line-delete support
- Cache-array data protected by 8-byte SEC/DED ECC
- 8-byte ECC throughout the internal L2 data-flow and migration flow to the on-chip SMP interconnect fabric or L3 cache
- FIR/SCOM support
- L2-cache purge support

- Various L2 hardware end-to-end type control checkers (for example, end-to-end type protocol checking that check for unexpected on-chip SMP interconnect fabric `cresp` or data)

## 6.2 L2 Unit Internal Resources

The L2 unit contains a set of internal machine and enqueueing resources scheduled to handle the processing of load and store type requests from the Power10 core and requests from the on-chip SMP interconnect fabric. *Table 6-1* lists the type of resource and the size and function of these resources within the L2 unit.

*Table 6-1. L2 Resources (Share between Threads within a Power10 Core)*

Resource	Description	Size	Hash Bits
CIU_Load Request Queues	Handle the staging for load requests from the Power10 core that are pending access to the Read Claim (RC) machines dispatch pipe to be assigned to an RC machine or sent to an L3 prefetch machine.	12 demand 4 data prefetch 8 instruction fetch/prefetch 4 translate	none
L2 Store Queue (Gather Station)	Store requests from the Power10 core and gather stores that are from the same core thread and are to the same 128-byte cache line. The L2 store-queue gathering mechanism gathers stores into a single 64-byte block with up to two sectors of clustering to be processed by the L2's RC machine.	2 banks of 24 × 64 bytes	addr(56)
Read Claim Machines	Read claim machines manage all cacheable operations initiated by the local core. The RC handles: <ul style="list-style-type: none"> <li>Gaining ownership of the line (either via an L2 hit, L3 hit, or on-chip SMP interconnect fabric access).</li> <li>Updating the core with data for its request.</li> <li>Updating the L2 cache with the data.</li> <li>Updating the L2 directory with the current coherent state and inclusivity information for this line.</li> <li>Issuing any required I-side or D-side kills to the L1 caches in the core.</li> </ul>	2 banks of 8	addr(56)
Cast Out (CO) Machines	Castout machines manage moving victimized lines from the L2 cache to the L3 cache and sending kills to the Power10 core when the L2 cache is no longer tracking this line. Under certain conditions, the L2 castout machines can select to move the line to the memory controller.	2 banks of 8	addr(56)
Snoop (SNP) Machines	Snoop machines manage all cacheable operations initiated by the incoming on-chip SMP interconnect fabric operations. The snoop is responsible for: <ul style="list-style-type: none"> <li>Representing the current L2 directory state to the on-chip SMP interconnect fabric.</li> <li>Intervening data to the on-chip SMP interconnect fabric when necessary.</li> <li>Updating the L2 directory with the current coherent state for this line.</li> <li>Issuing any required I-side or D-side kills to the L1 caches in the core based on on-chip SMP interconnect fabric activity.</li> <li>Pushing modified data to memory when required to be by the snooped operation.</li> </ul>	4 banks of 4	addr(55:56)
Reservation	The reservation logic provides for execution of <code>lax/stcx</code> instructions. A reservation is provided for each thread within the core.	4 (one per thread)	thread



### 6.2.1 Description of L2 Control Flow

The L2 control flow handles the coordination of the dispatching of load and store requests from the Power10 core and dispatching snoop operations from the on-chip SMP interconnect fabric that were initiated by other cores or I/O type devices. This L2 control logic manages these accesses such that when contention or ordering is required among commands, the proper dispatch collision and ordering detection occurs such that the RC/CO machines and SNP machines perform their data references and storage updates in a coherent and consistent manner.

The L2 store queue is a 2 bank  $\times$  24-entry  $\times$  64-byte buffer that allows storage updates to the same cache line from a given thread to be gathered before being issued into the RC machine dispatch pipe. The L2 store queue issues stores into the RC dispatch pipe honoring the barriers that have been inserted by software. The L2 store queue takes full advantage of the weakly-ordered nature of the Power Architecture® by allowing as many RC machines to be running in parallel where ordering is not required.

The CIU load queues manage the enqueueing of load-type requests from the Power10 core for I-side, D-side, translate, and prefetch type requests that cannot be immediately serviced by the RC machines. When a backlog of load-type requests occur, due to resource contention, the CIU manages the detection of when the resource is free and the priority for which requests should be re-issued next into the RC dispatch pipe to achieve best performance and for fairness.

The set of sixteen RC machines are dispatched on behalf of load/stores from its private core. This set is responsible for acquiring the proper coherent authority to complete the command and is a unified cache that is accessed privately by a given Power10 core. Along with these RC machines, the CO machines also are conditionally dispatched to move any victimized lines out of the L2 cache to make room for the new lines the RC machines are bringing in.

When a given command is sent down the RC dispatch pipe, one RC machine is assigned the command. The RC machine then determines where it must go to acquire the proper coherency authority to perform the command and, if required, acquire a copy of the data. The RC machine can find the line that is already in the L2 cache (for example, an L2 hit) or in the L3 cache (for example, an L3 hit), or the RC machine might have to proceed to the on-chip SMP interconnect fabric to gain ownership of the line. In addition, when an RC machine is dispatched, a CO machine might also be assigned at RC dispatch time if the L2 cache must create a victim line to make room for the line the RC is now installing. The L2 CO machine has the resources to work independent of the RC machine and thus allow the CO machine to work in parallel to migrate the line down to its private L3 cache. The L2 CO machines are also responsible for sending any required invalidates to the Power10 cores when lines are invalidated from the L2 cache.

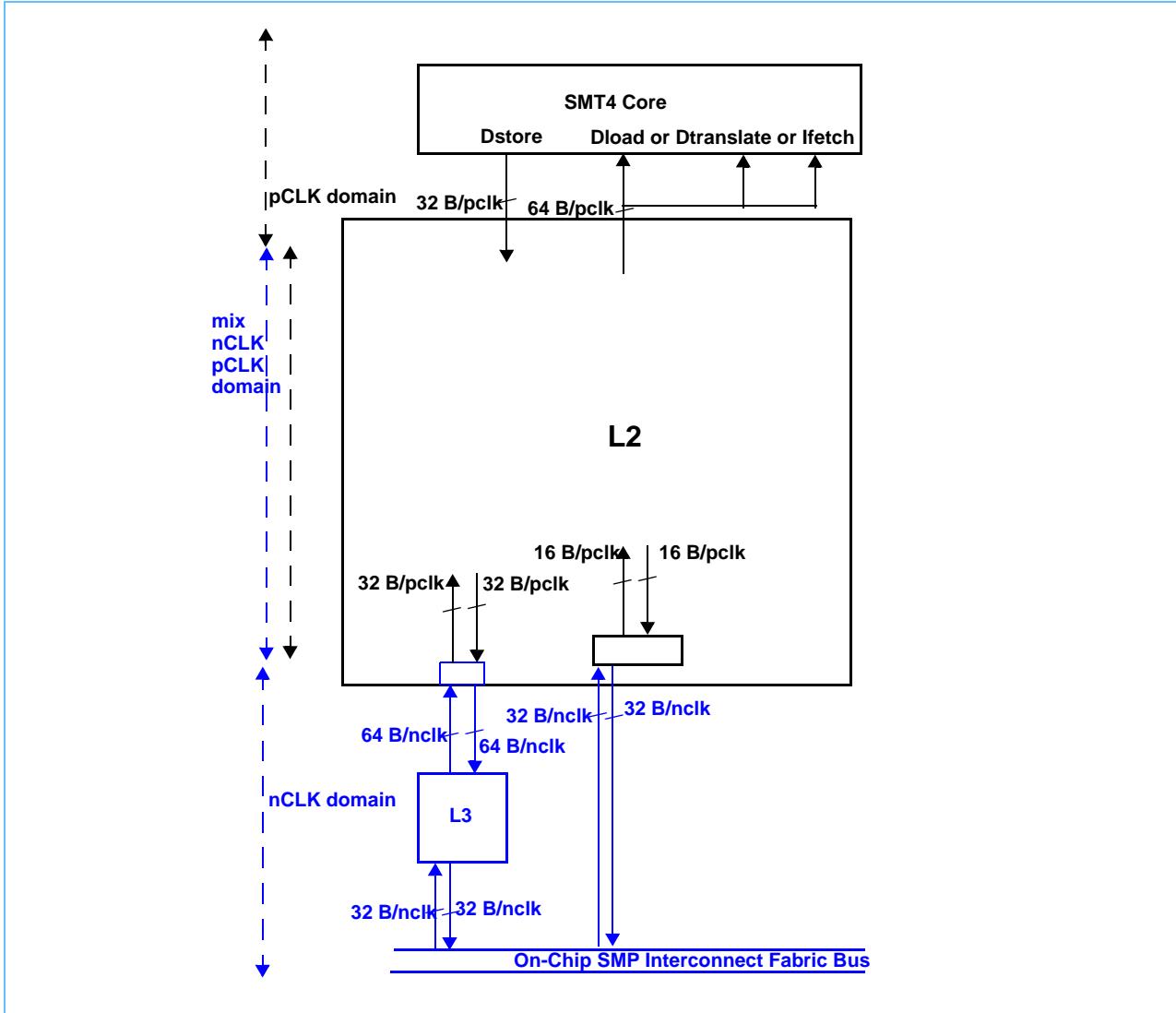
The L2 unit has four SNP dispatch pipes that control the assignment of the SNP machines to work on commands from the on-chip SMP interconnect fabric. These commands, initiated by on-chip SMP interconnect fabric masters, might include L2/L3 unit masters (on behalf of their cores commands) or by other masters such as I/O type devices. The L2 SNP machines are used to service these requests (L2 hits) by granting coherent authority from the L2 directory during SNP dispatch and (if required) providing a copy of the data via data intervention back to the requesting on-chip SMP interconnect fabric master. The L2 SNP machines are also responsible for sending any required invalidates to the Power10 core when lines are invalidated from the L2 cache.

Both the RC dispatch pipe and the SNP dispatch pipe have the ability to detect when a given line has already been assigned to either an RC, CO, or SNP machine. Contention by the subsequent command for the same line is detected and thus is delayed until after the previous machine has completed its work on the command it is working on.

## 6.3 Operational Flows and Bandwidths

Figure 6-2 shows the L2 units major input/output buses and the data bus bandwidth that represents the overall bandwidth capabilities in and out of the L2 cache.

Figure 6-2. L2 Bus Bandwidths





## 6.4 LRU

### 6.4.1 LRU Modes

- Normal LRU mode (13 bit,  $2 \times 4$ -way pointer mechanism)
- Direct Map: uses  $\text{addr}[44:46]$
- Single-Member Mode: configuration register specifies one of eight members
- Pseudo-Random Static Mode: LFSR based

### 6.4.2 Policies

- `invalid_line_lru_bias`: moves members in I state toward LRU
- `id_state_mru_bias`: moves members in invalid line-deleted (Id) state toward MRU

### 6.4.3 Line Disable

- Directory support for line delete (Id) on a per-directory location granularity
- Hardware or software-initiated line delete policies

## 6.5 Transactional Memory Support

### 6.5.1 Basic Policy

Transactional memory support is provided in the L2 cache to support the Power10 Synthetic\_TM core mode. In this mode, the L2 cache will not be asked to track any transactional load or store cache lines associated with a Synthetic\_TM transaction.

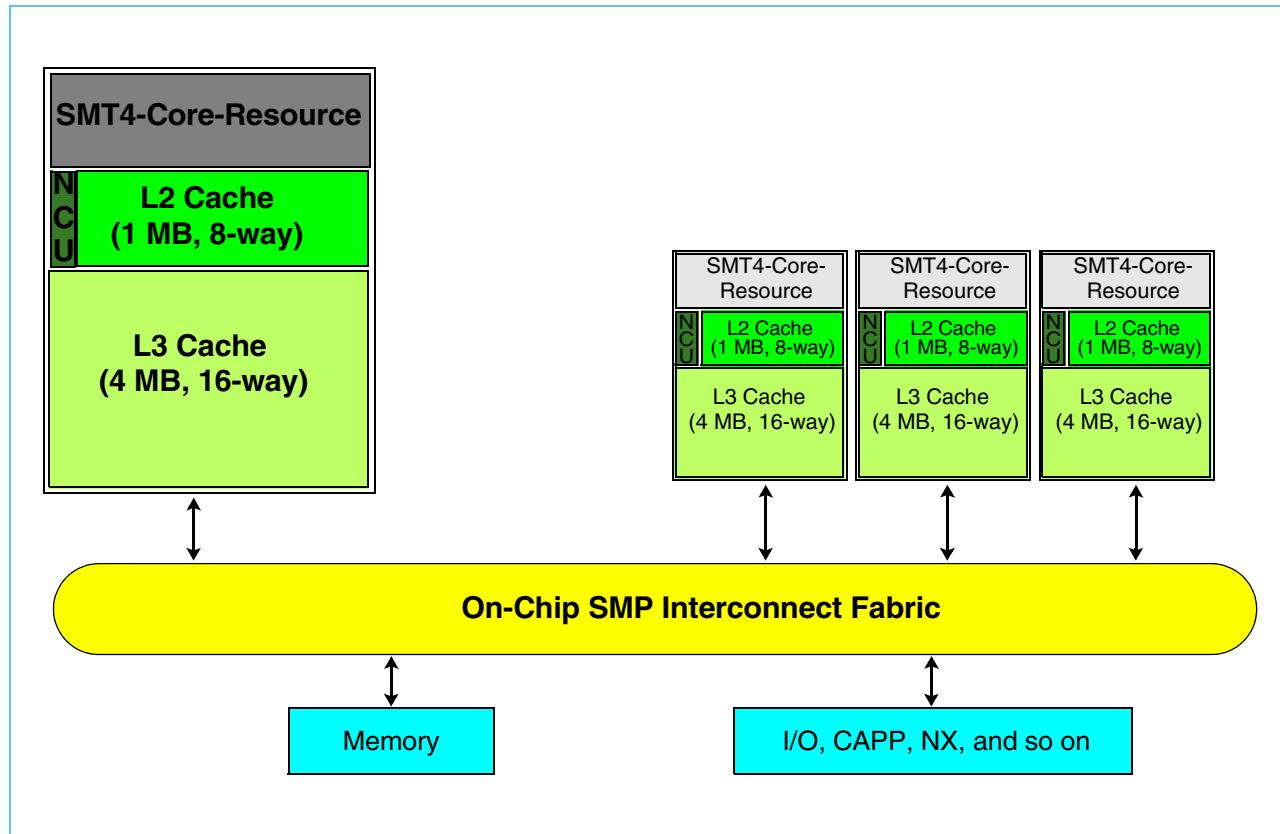
## 7. L3 Cache

This section details the microarchitecture of the L3 cache for the SMT4 variant of the Power10 processor chip.

The L3 cache unit is contained within a chiplet that consists of: one Power10 SMT4-core-resource, one 1 MB L2 cache, one 4 MB L3 cache, one NCU, and a portion of the on-chip SMP interconnect fabric logic. The L3 cache is a unified cache that is accessed privately by the L2 cache in the same chiplet and non-privately by other unit masters in the system via the on-chip SMP interconnect fabric. The L3 cache maintains full-hardware coherency within the system and can supply intervention data to other unit masters in the system. The L3 cache is a victim cache and thus typically holds cache lines that are different from those held by the L2 cache or the core's L1 cache.

*Figure 7-1 shows a high-level diagram of the Power10 chip, which includes multiple L3 regions interconnected via the on-chip SMP interconnect fabric.*

*Figure 7-1. Block Diagram of Multiple SMT4 Core/L2/Local-L3 Regions Interconnected via the On-Chip SMP Interconnect Fabric*





## 7.1 Interfaces

The interface to the L2 cache is used to service reads (associated with both load and store requests from the core) and castout requests from the L2 cache. The interface to the on-chip SMP interconnect fabric is used to snoop requests from other unit masters, to prefetch cache lines into the L3 cache based on requests from the L2 cache, to castout evicted cache lines to memory or lateral L3 caches, or to castout lines evicted from the L2 cache of the same chiplet to lateral L3 caches. Snooping can lead to providing intervention data to other unit masters, invalidating cache lines, or generating cache-line pushes to memory. The L3 cache also accepts lateral castouts from other on-chip L3 caches and injected writes from other unit masters.

The L3 cache supports a fast broadcast request interface to other on-chip caches and to the on-chip memory controllers for high-priority (demand) read requests that are L2 and L3 misses. Upon the occurrence of such a miss, the L3 cache broadcasts to these devices, ahead of the request to the slower coherent fabric interconnect. If any cache is in a state to deliver early data, it does so on the on-chip SMP interconnect fabric data bus. That cache then waits for the coherent request on the on-chip SMP interconnect fabric and responds in a manner that is consistent with its response to the fast request. When a fast broadcast arrives at the on-chip memory controllers, the memory controller begins to fetch the data even if a cache is also fetching data in parallel. However, the memory controller waits for the coherent request before sending data.

## 7.2 Features and Resources

- SMT4 L3 cache features:
  - Private 4 MB L3 cache; shared L3.1<sup>1</sup>.
  - 16-way set associativity.
  - 128-byte cache lines.
  - Data cache consists of 4 banks of high-efficiency SRAMs with interleaving for read/write overlapping.
  - 64-byte wide data bus to the L2 cache for reads.
  - 64-byte wide data bus from the L2 cache for L2 castouts.
  - Sixty-six, 600 kb SRAM macros; two of the SRAM macros are for redundancy.
  - All cache accesses have the same latency.
  - 16-way directory organized as four banks, with up to four reads or two reads and two writes every 2 pclk to differing banks. Physically implemented as eight  $512 \times 8$  way  $\times 50$ -bit SRAMs.
  - LRU algorithm for victim selection using 4-bit per member utilization tracking using data type and re-use awareness.
- L3 cache functionality:
  - Victim cache for the local L2 cache (L3.0).
  - Victim cache for the other on-chip L3 caches (L3.1).
  - Services read requests from the local L2 cache due to core load or stores that miss in the L2 cache.
  - Services prefetch requests that originate from a local core and are passed along by the local L2 cache.
  - Four snoop ports, address banked by address bits 55 and 56.

1. L3.1 means using selected on-chip L3 caches to which lateral-castouts occur.

- On-chip SMP interconnect fabric support.
- Cache injection support, including partial-line injection on any byte boundary and distribution of injected-lines on an L3 castout.
- Dual-class L3.0/L3.1 LRU support and L3.1 activity throttling.
- Fast broadcast of on-chip load request to other caches and memory controllers for L2 demand loads that miss in the L3 cache.
- Support for multi-level command scope on the on-chip SMP interconnect fabric.
- Entire L3 cache clocked at  $\frac{1}{2}$  core frequency.
- Peak bandwidths (2:1 clock = two core clocks):
  - 64-byte, 2:1 clock L2 reload capacity.
  - 64-byte, 2:1 clock L2 castout capacity.
  - 32-byte, 2:1 clock prefetch capacity.
  - 32-byte, 2:1 clock L3 castout capacity.
  - 32-byte, 2:1 clock full-line cache-inject capacity.
  - 32-byte, 2:1 clock intervention capacity.
- RAS features:
  - Data-cache contents are protected by 8 bits per 8 bytes SEC/DED ECC.
  - Directory contents are protected by SEC/DED ECC.
  - ECC propagation throughout recovery dataflow and dataflow buffers.
  - Directory single stuck bit support.
  - Physical data line delete support.

## 7.3 Queues

The L3 cache has multiple sets of state machines that act as queues for various types of requests.

### 7.3.1 Read Machines

The L3 cache has 16 instances of read (RD) state machines that handle L2 read requests. L2 read requests are a consequence of core loads and stores that miss in the L2 cache. The L3 cache first attempts to service the L2 read request from the L3 cache. If the request misses the L3 cache, it is either requested to the on-chip SMP interconnect fabric on behalf of the L2 cache to reduce latency or returned with a miss indicator to the L2 cache, which subsequently sends a request to the on-chip SMP interconnect fabric. Details about the RD state machines follow:

- 16 RD machines
- Prevent other state machines from acting on the same cache-line address
- Carry out an L3 directory invalidate on an L3 hit.
- No speculative cache reads; the directory is read first.



- Forwards request to the fast bus when an L2 read request misses the L3 cache and the L2 read request enables fast handling. Request to the fast bus is in parallel with an L3 miss indication to the L2 cache.
- In the case of a correctable data error, the RD re-requests a safe cache read (data run through an inline ECC correction) to resend all data beats.

### 7.3.2 Castin/Castout Machines

The L3 cache has 16 castin (CI) state machines that handle L2 castouts, line insertion associated with a prefetch request for which data has been received, LCOs that arrive from the on-chip SMP interconnect fabric, and full or partial cache line injections that arrive from the on-chip SMP interconnect fabric. The castin state machines read the L3 directory and then write the directory and the data cache. The L3 cache has 16 castout (CO) state machines that handle writing cache lines to the on-chip SMP interconnect fabric.

Details about the CO and CI state machines are as follows:

- CI machines:
  - 16 CI machines.
  - Prevents other state machines from acting on the same cache-line address.
  - Handles a CI that results from an L2 CO, prefetches, incoming LCOs, and incoming partial or full cache-line injects.
  - Private control/data interface for moving data from the L2 CO buffers to the L3 cache (no extra storage in the L3 cache).
  - Flow control mechanisms to provide fairness and to prevent L2 COs, PF, and WI from overflowing CI machines.
  - Handles line delete and purge/flush functions initiated via the SCOM register interface.
- CO machines:
  - 16 CO machines; each one paired with a CI machine.
  - Prevent other state machines from acting on the same cache-line address.
  - Handle L3 COs due to L3 CIs, prefetches, incoming LCOs, and incoming partial or full cache-line injects.
  - Handle L2 cast-through operations of cache lines not to be installed in the L3 cache. This includes cache line write to memory or to lateral L3 caches.
  - 1 × 128-byte buffer per CO machine (physically in the castout/push/intervention buffer).
  - Data-cache interlock: CI cache write held off until CO cache read is done.
  - Directory interlock: CO active for protection until a CI completes a directory write to destroy an old entry.

### 7.3.3 Prefetch Machines

The L3 cache has 48 prefetch (PF) state machines that make read requests to the on-chip SMP interconnect fabric. Each PF machine starts with a request that originates from the local core and is passed through the L2 cache. When a PF state machine receives a request to make a prefetch, it first reads the L3 directory. If the L3 cache already has the cache line, the PF machine does nothing further and goes idle. If the cache line is not in the L3 cache, the PF machine makes a read request to the on-chip SMP interconnect fabric. The on-chip SMP interconnect fabric request may indicate that the memory controller is allowed to discard the request if the memory controller is currently busy with higher priority requests. If prefetch data arrives from the fabric, the PF machine determines if a demand request has arrived for that cache line while the prefetch request to the fabric was pending. If no such request is pending, the PF machine forwards the request to a CI machine to have it put in the L3 cache. However, if a demand request has arrived, the data is forwarded directly to the L2 cache and not installed in the L3 cache.

Details about the PF state machines are as follows:

- 48 L3 PF machines.
- Prevent other state machines from acting on the same cache-line address.
- Handle prefetch requests from a local core.
- For entries that are L3.0 misses, the PF machine sends a request to the on-chip SMP interconnect fabric.
- When a RD machine is waiting for data, PF data bypasses to the L2 cache without installing in the L3 cache.
- When no RD machine is waiting for the data, the L3 PF machine uses the CI/CO machine to move data from the PFWI buffer to the L3 data cache.

### 7.3.4 Snoop Dispatch Pipes

The L3 cache has four dispatch pipes that handle incoming reflected-command (snoop) requests from the fabric. The snoop dispatch pipes perform an L3 directory read to determine how to handle the request. If the request requires the sending of intervention data, the executing of a snoop push, or the invalidating of the associated cache line, then the snoop dispatch pipe forwards the request to one of 16 snoop (SN) state machines. If the request is an incoming lateral cast-out (LCO) or a cache-injection that is accepted, the request is forwarded to both a snoop state machine (SN) and a write inject (WI) state machine.

### 7.3.5 Snoop Machines

The L3 cache has 16 snoop state machines which handle transactions to the on-chip SMP interconnect fabric that occur as a consequence of a snoop. The snoop machines can write the directory and read the cache and master push commands to the on-chip SMP interconnect fabric and send data to the on-chip SMP interconnect fabric. Details about the snoop machines are as follows:

- 16 SN machines.
- Prevents other state machines from acting on the same cache-line address.
- Handles intervention for snoops.
- Handles incoming LCO requests without data movement (state merge).
- Handles push command in reaction to snoops when required.



### 7.3.6 Write Machines

There are 16 write inject (WI) state machines that are started by the snoop dispatch pipes on incoming LCO requests or incoming cache-line injections. These machines wait for data to arrive from the fabric, then pass a request to a CI machine to have the cache line inserted into the L3 cache.

Details about the write machines are as follows:

- 16 WI machines; all can be used for partial-line cache injects.
- Prevent other state machines from acting on the same cache-line address.
- Handle incoming LCO requests with data movement and cache injects.
- L3 WI machines use CI/CO machines to move data from the PFWI buffer to the L3 data cache.

## 8. SMP Interconnect

The Power10 SMP interconnect is the underlying hardware used to create a scaleable cache-coherent multi-processor system. The Power10 SMP interconnect controller provides coherent and non-coherent memory access, I/O operations, interrupt communication, and system controller communication. The SMP interconnect provides all of the interfaces, buffering, and sequencing of command and data operations within the storage subsystem. The SMP interconnect is integrated on the Power10 chip with up to 16 SMT8 processor cores or 32 SMT4 processor cores and an on-chip memory subsystem. The Power10 chip has up to eight SMP external links that can be used to connect to other Power10 chips.

The external SMP interconnect link is a split-transaction, multiplexed command and data bus that can support up to sixteen Power10 chips in a system. Aggregation of data links between the same source and destination chips is supported to increase data bandwidth.

Cache coherence is maintained by using a snooping protocol. Address broadcasts are sent to the snoopers, snoop responses are sent back in order to the initiating chip, and a combined snoop-response broadcast is sent back to all of the snoopers. Multiple levels of snoop filtering are supported to take advantage of the locality of data and processing threads. This approach reduces the amount of interlink bandwidth required, reduces the bandwidth required for system-wide command broadcasts, and maintains hardware enforced coherency using a single-snooping protocol. When the transaction cannot be completed coherently using chip scope, the coherency protocol forces the command to be re-issued to an increased scope of the system.

### 8.1 SMP Interconnect Features

#### 8.1.1 General Features

- Master command/data request arbitration.
- Command requests are tagged and broadcast using a snooping protocol that enables high-speed cache-to-cache transfers.
- Multiple command scopes are used to reduce the bus-utilizations system wide. The SMP interconnect architecture uses cache states indicating the last known location of a line (sent off chip), information maintained in the system memory (memory domain indicator [MDI] bits), a coarse grained directory that indicates when a line has gone off the chip, and combined response equations that indicate if the scope of the command is sufficient to complete the command or if a larger scope is necessary.
- The command snoop responses specified by the SMP interconnect implementation are used to create a combined response that is broadcast to maintain system cache state coherency. Combined responses are not tagged. Instead, the order of commands from a chip source, using a specific command-broadcast scope, is the same order that combined responses are issued from that source. The order is also affected by the snoop bus usage as well.
- Data is tagged and routed along a dynamically selected path using staging/buffering along the way to overcome data routing collisions.
- Command throttling and retry command back-off mechanisms for livelock prevention.
- Multiple data links between chips are supported (link aggregation).



### 8.1.2 Power10-Specific Features

Some features for the Power10 SMP interconnect include:

- Command broadcast scopes (such as, snoop filtering)
  - Local Node Scope (LNS): Broadcast within a local chip with nodal scope. Node is defined as one chip.
  - Near Node Scope (NNS): Broadcast to a local chip and targeted chip in the local group.
  - Remote Node Scope (RNS): Broadcast to a local chip and targeted chip on a remote group.
  - Group Scope (GS): Broadcast to a local chip with access to the memory coherency directory (MCD).
  - Vectored Group Scope (VGS): Broadcast to a local chip and targeted remote chip.
- 1 - 8 socket, 1-hop system configuration support
- 1 - 4 group, 2-hop system configuration support
- 4× snoop bus support
- Memory controller fastpath support
- 256 Local master (LM) system-pump queue size (64 per snoop bus)
- 256 Group master (NM) group-pump queue size (64 per snoop bus)
- MCD with per-group bit vector
- Service processor accessible SCOM registers for configuration setup

### 8.1.3 On-Chip Features

- Eight EQ core chiplets. Each chiplet contains two SMT8 cores or four SMT4 cores with a shared PBI chiplet interface (4× data port)
- Four memory controller (MC) chiplets (2× data port)
- Two endcap chiplets (6× PAU, 2× PE, 2× nMMU, INT, MCD, VAS, NX, TP, Fabric master) (1× data port)
- Decentralized command-request arbitration
- Dynamic command rate throttling
- TLBI tokenizer
- Decentralized data-request arbitration
- 32-byte data arbitration size; unit specifies total transfer size



#### 8.1.4 Off-Chip External SMP Features

- 8× (A or X link), 2 × 9-bit 25G/32G links (asynchronous clocking)
  - 2.0 M length (module and board)
  - 22 UI MAX bit-lane skew
- Aggregate data-link support
- Indirect data-link support

#### 8.1.5 Power Management Features

- Variable nest clock frequency support (0.9 - 2.1 GHz)
- EQ chiplet power-management support
- Dynamic lane reduction support (external SMP links)

#### 8.1.6 RAS Features

- CRC link-level retry on external SMP links
- 100% ECC protection on internal data flow
- Hang recovery mechanism
- Trace array
- Performance monitor
- FIR error reporting
  - Protocol errors
  - Underflow/overflow checkers
  - Asynchronous drop/repeat checkers
  - Parity checkers on coherency register files
- Error injection
  - Single- and double-bit errors on external SMP links



## 8.2 External Power10 Fabric

The Power10 off-chip SMP interconnect is a highly scalable, multi-tiered, fully-connected topology. The off-chip links use 18-bit high-speed differential links running up to 32 Gbps and can be configured in either a 1-hop or 2-hop configuration.

In the 1-hop configuration, the Power10 processor chip can fully connect up to seven other processor chips to create an eight-chip SMP system. Each chip is a group using up to seven inter-group A-links for a maximum system of eight processor chips.

In the 2-hop configuration, the Power10 processor chip can fully connect up to three other processor chips to create a four-chip group. The intra-group links are designated as X-links. Each Power10 processor in a group connects to its corresponding processor chip in each other group. Three of the inter-group A-links are provided per chip supporting a total of four groups, each containing four processor chips. A full four-group system of four chips per group comprises a maximum system of 16 processor chips.

## 8.3 Terminology

*Table 8-1* defines some common terms used in this section.

*Table 8-1. Terminology*

Term	Description
Lane	Single Tx/Rx bit.
Link	Consists of multiple bit lanes organized by protocol layer in packets (DLL).
Single Link	Single link interconnect between two processors.
Paired Link	Pair of link interconnects between two processors.
Packet	Data payload generated by the data link layer.
Frame	Data payload generated by the transaction layer.

## 8.4 Power10 Fabric SMP Topology

Figure 8-1 and Figure 8-2 illustrates the external SMP topology.

Figure 8-1. 1-Hop SMP Topology

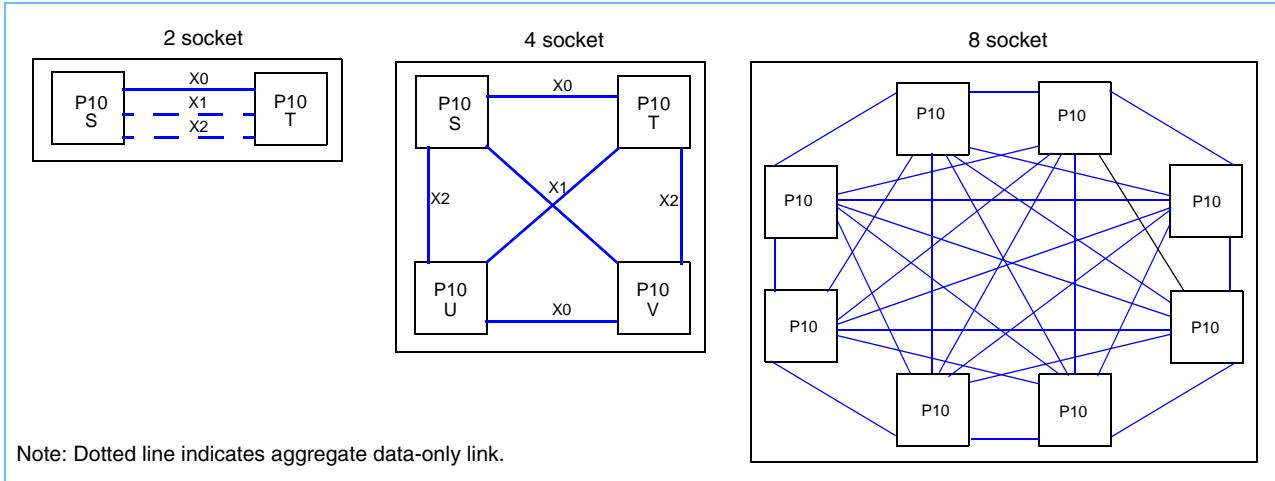
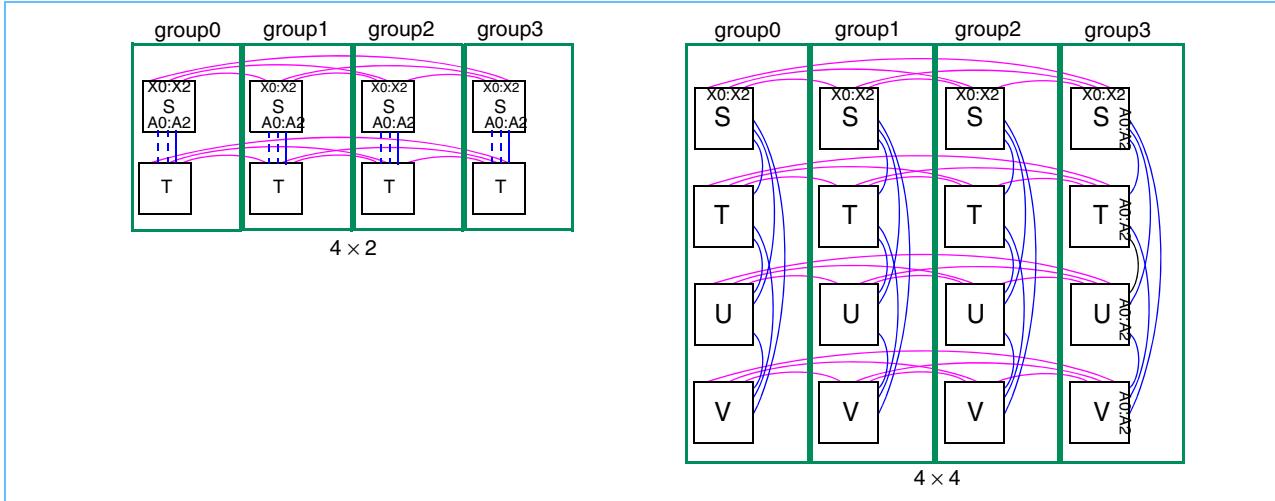


Figure 8-2. 2-Hop SMP Topology





#### 8.4.1 Protocol and Data Routing in Multi-Chip Configurations

The SMP ports configured for coherency are used for both data and control information transport. The buses are used as follows:

1. The chip containing the master that is the source of the command issues the reflected command and the combined response to all other chips in the SMP system. Partial responses are collected and returned to the chip containing the master.
2. Data is moved point-to-point. For read operations, the chip containing the source of the data directs the data to the chip containing the master. For write operations, the chip containing the master directs the data to the LPC that performs the write operation. The routing tag contains the chip and unit identifier information for this purpose.

### 8.5 Power10 Coherency Flow

#### 8.5.1 Broadcast Scope Definition

*Table 8-2* and *Table 8-3* describe the physical broadcast scope and the equivalent coherency scope.

*Table 8-2. 1-Hop Broadcast Scope Definition*

Command Scope		Physical Broadcast
LNS	Local node scope	Local chip
NNS	Near node scope	Local chip
GS	Group scope	Local chip
RNS	Remote node scope	Local chip and targeted remote chip <sup>1</sup>
VGS	Vectored group scope	Local group and remote group(s)

1. Requires A-bus BAR lookup.

*Table 8-3. 2-Hop Broadcast Scope Definition*

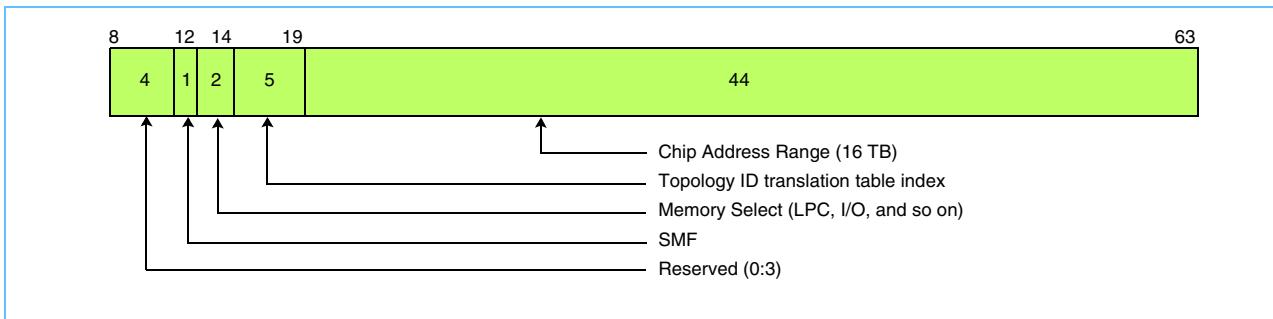
Command Scope		Physical Broadcast
LNS	Local node scope	Local chip
NNS	Near node scope	Local chip and targeted near chip <sup>1</sup>
GS	Group scope	Local chip and all near chips
RNS	Remote node scope	Local chip and targeted remote chip <sup>2</sup>
VGS	Vectored group scope	Local group and remote group(s)

1. Requires X-bus BAR lookup.  
2. Requires X-bus BAR lookup and A-bus BAR lookup.

### 8.5.2 Address Definition

Figure 8-3 illustrates the Power10 system real-address map.

Figure 8-3. Power10 System Real-Address Map

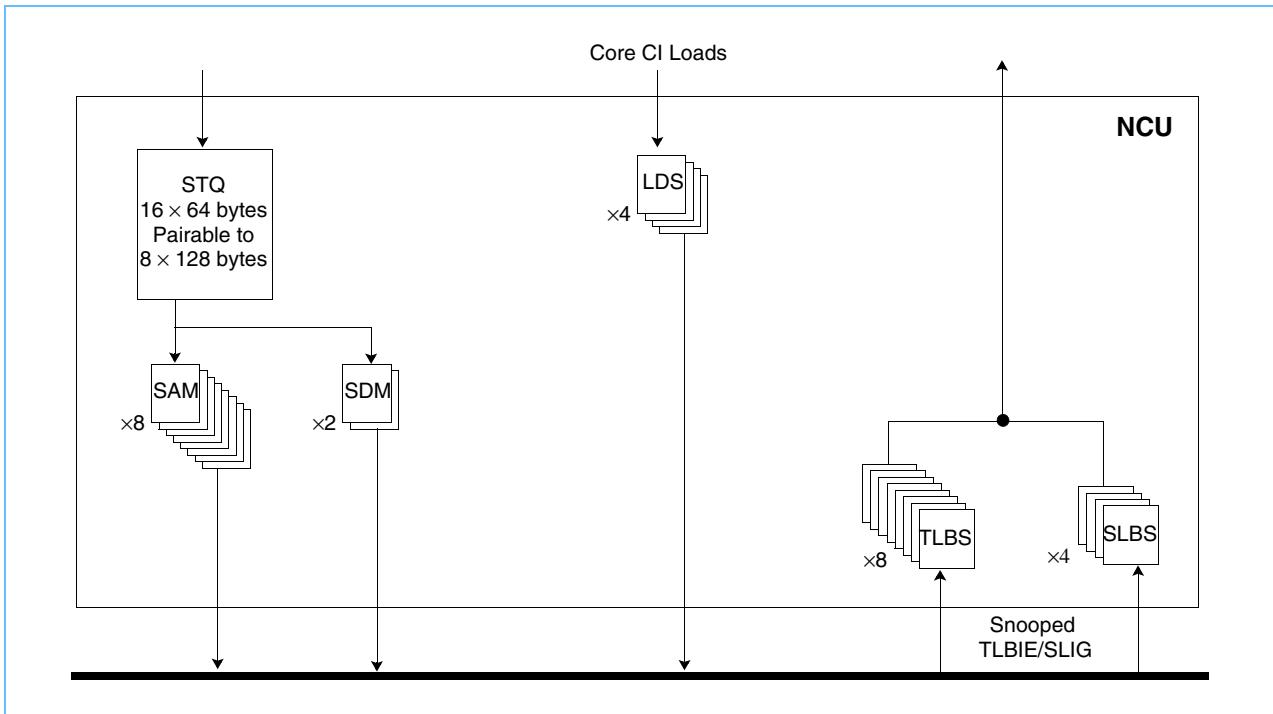




## 9. NCU

The Power10 Non-Cacheable Unit (NCU) is responsible for processing noncacheable load and store operations (load and store operations with a WIMG “I” bit setting of ‘1’ as described in the Power ISA), word and doubleword load and store atomic instructions (**Iwat**, **Idat**, **stwat**, **stdat**), and certain other uncacheable operations such as **tlbie** and portions of the various **sync** and **ptesync** instructions. All of these instructions support the behavioral definitions given in the Power ISA. One NCU unit is instantiated per SMT4-core-resource. This NCU handles appropriate operations for all four threads in the SMT4-core-resource.

Figure 9-1. NCU Block Diagram



The Power10 NCU provides one dedicated cache-inhibited load station (LDS) per thread to process cache-inhibited loads and load word or doubleword atomics (**Iwat**, **Idat**). Cache-inhibited loads (whether guarded [meaning the G bit of WIMG is set to ‘1’] or not) and load atomics are neither gathered nor are they reordered in the Power10 implementation. Although, with the exception of guarded cache-inhibited loads, they might be in future implementations. If ordering and/or non-gathering is required on unguarded caching-inhibited loads or on load atomics, appropriate barriers should be inserted to ensure future compatibility.

For cache-inhibited stores and store word and doubleword atomics (**stwat**, **stdat**), a store queue (STQ) consisting of sixteen 64-byte store gather stations is provided. The store gather stations are shared across the four core threads and hardware prevents any thread from blocking other threads in the store queue. A pair of 64-byte stations can “pair” together to gather up to 128 bytes.

The Power10 NCU supports gathering and reordering for cache-inhibited stores in the unguarded caching-inhibited (IG = ‘10’) space. In caching-inhibited, but guarded space (IG = ‘11’), cache-inhibited stores are neither reordered nor gathered as required by the architecture. Similarly, atomic word and doubleword stores (**stwat**, **stdat**) are never gathered, but might be re-ordered.



The Power10 NCU gathers aligned 1, 2, 4, 8, and 16-byte unguarded cache-inhibited stores. Gathering starts at the first such store in a 128-byte region and continues as long as the subsequent store (which might be of a different size than the previously gathered store) is contiguous with the previously gathered store and does not cross a 128-byte boundary.

The Power10 NCU provides eight store address machines (SAM) that manage the address tenure of the store allowing for up to eight outstanding cache-inhibited or store atomic word or doubleword instructions (**stwat**, **stdat**). A set of two store data machines (SDM) are used to manage the data tenures for the store address machines after the address tenures are complete.<sup>1</sup>

The Power10 NCU also masters hypervisor broadcast MSGSEND instruction through the store queue and store address and data machines. MSGSEND instructions are treated as a special type of store instruction.

Finally, the NCU provides eight snoop queues (TLBS) to process snooped TLBIE operations and four snoop queues (SLBS) to process SLBIE operations and forward these to the core. The instruction sequences provided in the Power ISA documentation for page table modification should be followed in using the TLBIE and SLBIEG instructions that cause these bus operations.

## 9.1 NCU Characteristics

### 9.1.1 Store Queue (STQ)

- 16 × 64 byte store gather stations (chainable to 8 × 128 byte gathered stores).
- The gather stations are shared across threads.

### 9.1.2 Store Modes (IG = '1X')

- IG = '11' mode, stores are done in-order and no gathering is allowed.
- IG = '10' mode, stores can be gathered and reordered.
- Atomic stores (**stwat**, **stdat**) are not gathered but can be reordered.

### 9.1.3 Loads

- One outstanding cache-inhibited (guarded or unguarded) load or atomic word or doubleword load (**lwat**, **ldat**) per thread.

---

1. Address tenures take longer than data tenures and data tenures can be fully satisfied by two store data machines.



## 10. Memory Controller

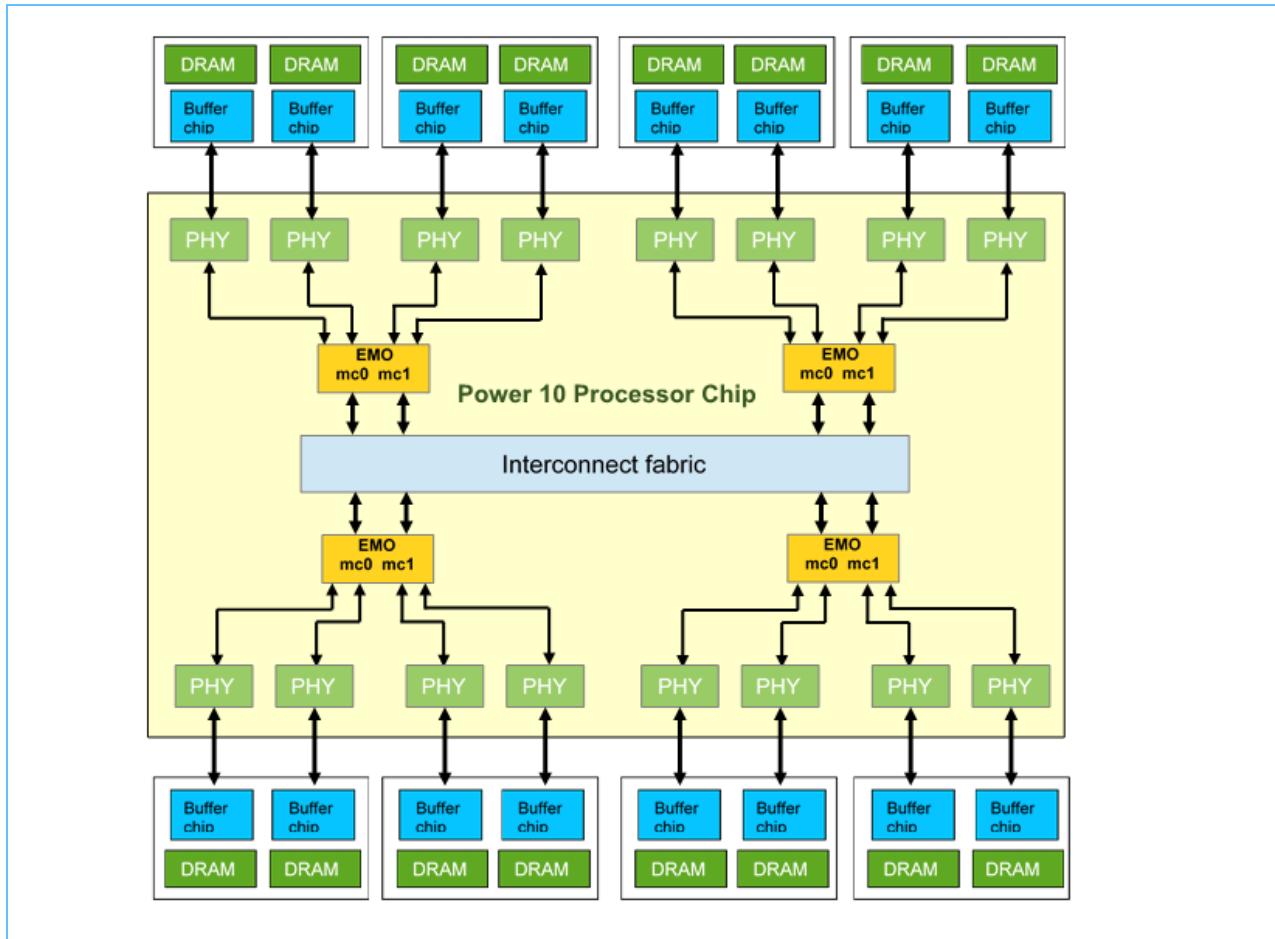
The Power10 memory controller unit (MCU) provides the system memory interface between the on-chip SMP interconnect fabric and the OpenCAPI memory interface (OMI) links. These OMI links can be attached to external memory buffer chips that conform to the OpenCAPI 3.1 specification (Memory Interface Class only). The buffer chips, in turn, directly connect to industry standard memory DIMM interfaces or other memory media (such as, storage-class memory). The MCU acts as a subordinate unit only. It does not source any commands to the SMP fabric. There are logically eight, essentially-independent MCU channels on the chip that interface to a total of 16 high-speed OMI links. Each memory channel supports two OMI links (referred to here as sub-channels). Physically, the MCUs are grouped into four instances of an extended memory OMI (EMO) unit. Each EMO unit contains two MCU channels. The MCUs process 128-byte read requests and 64-byte or 128-byte write requests from processor cores, caches, and I/O host bridges; 1 - 128-byte partial-line writes; and atomic memory operations (AMOs). The MCU also handles address-only operations for the purpose of address protection, acting as the lowest-point of coherency (LPC).

The eight MCUs on the chip can be configured into one or more address interleave groups. Within each group, the address space is divided into portions, such that each sequential portion is handled by a different MCU in a round-robin fashion. The maximum memory addressing per Power10 chip is 128 TB.

Within a single MCU channel, the two OMI sub-channels are always address interleaved on a 128-byte boundary (assuming both sub-channels are populated with memory).

The MCU resides in a single clock domain. That is, the memory controller asynchronous (MCA) domain runs asynchronously to the SMP interconnect fabric, but synchronously to the OMI interface.

Figure 10-1. Power10 System Memory High-Level Diagram



## 10.1 Basic Configuration/Grouping

Each MCU channel includes a set of Base Address Registers (BARs), which are programmed by boot firmware at IPL time. How they are configured depends on which OMI ports are populated with memory and the sizes of the memory behind each port. The following BAR registers are supported:

- MCFGP: Primary memory BAR
- MCFGPR-CFG: Configuration memory space BAR
- MCFGPR-MMIO: MMIO memory space BAR

The MCU architecture allows for 1, 2, 3, 4, 6, or 8 MCUs to be grouped together for address interleaving. As a group, the MCU channels then hash a contiguous address space among themselves to more efficiently distribute the memory workload.



For 2, 3, 4, 6, or 8 MCUs to be grouped, the total amount of memory defined by each MCU's primary memory configuration facilities must be the same. The DIMM configurations and sizes that make up the total amount of memory can be different, but the total memory size plugged behind each MCU/buffer chip in the group must be the same.

The total amount of physical memory behind an MCU can be less than the memory size specified by that MCU's primary memory configuration register. However, if the MCU decodes an address that falls within its programmed address range but does not decode to a valid physical DRAM address, an address error combined response is generated.

Each MCU receives a 56-bit address for each snoop operation from the SMP fabric, and forwards a 43-bit (8 TB) logical address to the buffer chip.

## 10.2 Atomic Memory Operations

Each MCU supports the processing of a set of atomic memory operations (AMOs). AMOs are read-modify-write type operations and have store data associated with them. The MCU implements an ALU; which operates on the store and memory fetch data. The intent is to provide high throughput of multiple AMOs targeting the same address. There are two major features that enable this high level of throughput:

- Multiple AMOs targeting the same address are allowed to be queued in the command list queue such that they are not serialized via retries on the SMP fabric interface. Each AMO command must be performed atomically, and the AMO commands are executed in the order in which there are snooped on the SMP fabric.
- The 64-entry  $\times$  128-byte RMW buffer is managed as a cache. When one AMO completes, its data is maintained in the RMW buffer so that a subsequent AMO to the same address receives its data directly from the buffer instead of having to re-fetch the data from memory. This caching capability is also used for partial writes, providing for the gathering of multiple partial writes before writing back to memory, and for MDI bit updates. Each command list entry representing data in the RMW buffer has three cache states: idle, valid, and clean. The command list logic controls the deallocation of RMW buffer entries in response to snoop traffic to free-up room in the RMW buffer.

AMOs are only supported in 4- and 8-byte sizes. Both big-endian and little-endian modes are supported. There are 23 different AMO types supported:

- Store Add
- Store XOR
- Store OR
- Store AND
- Fetch and ADD
- Fetch and XOR
- Fetch and OR
- Fetch and AND
- Compare and swap equal
- Compare and swap not equal
- Swap unconditional
- Fetch and increment bounded
- Fetch and increment equal
- Fetch and decrement bounded
- Store twin
- Store maximum unsigned



- Store maximum signed
- Store minimum unsigned
- Store minimum signed
- Fetch and maximum unsigned
- Fetch and maximum signed
- Fetch and minimum unsigned
- Fetch and minimum signed

All of the “Fetch and …” operations return the unmodified memory data to the SMP fabric, except for the following operations:

- Fetch and increment bounded
- Fetch and increment equal
- Fetch and decrement bounded

These three operations return either the unmodified memory data or the minimum unsigned-integer value based on the result of a compare of two adjacent granules of memory data.

From an MCU data flow and sequencing perspective, AMO operations are always 64 bytes in length, and partial writes and MDI updates are always 128 bytes in length.



## 10.3 Selective Memory Mirroring

To improve memory RAS for large systems, the memory controller can be configured for symmetric memory mirroring (SMM). In this configuration, memory sub-channels are grouped into mirrored pairs. Separate mirrored and non-mirrored BAR registers enable memory access to be targeted to either mirrored or non-mirrored space (thus the term “selective”). While mirroring in all MCFG configurations is supported, it is only supported if both sub-channels for all MCUs within an interleave group are populated with the same memory of equal capacity.

Memory mirroring takes effect when an incoming address lands in the mirrored address region as defined by the mirrored BAR and mirrored-size register MCFGPM.

Write operations are issued to both sub-channels of a mirrored pair. Read operations to a mirrored address space attempt to access memory first from the primary sub-channel (sub-channel A for real address bit 56 = 0 (RA[56] = 0), and sub-channel B for RA(56) = 1). If read data cannot be delivered from the primary sub-channel (because of a read data error or other sub-channel error conditions), the read is retried on the secondary sub-channel (sub-channel B for RA(56) = 0 and sub-channel A for RA(56) = 1), to get correct read data.

The MCFGP and MCFGPM BAR registers provide a non-mirrored and mirrored view of physical memory. Software must allocate mirrored and non-mirrored regions of memory such that these regions do not overlap in physical memory space.

## 10.4 Whole Memory Encryption

The memory controller supports Advanced Encryption Standard (AES) encryption/decryption of all traffic to system memory. Encryption is enabled via configuration bits accessible to firmware. Accesses to OMI configuration and MMIO spaces are never encrypted, because they are not part of the system memory media. Other than that, all traffic to system memory is encrypted (if enabled) or not. Selective encryption of specific system memory regions is not supported.

### 10.4.1 Modes of Operation

If utilized, persistent DIMM technology will keep the data stored inside the DIMMs even if the power is turned off. If a memory card is replaced, the data stored in the DIMM would leave the data center in the clear. Therefore, an AES block cipher with strong encryption in the form of an AES XTS mode is supported as part of the Power10 design to protect the data.

In addition to the XTS mode, the AES CTR mode is also supported. AES CTR is a low-latency AES block cipher and is intended for memory encryption of volatile memory only. The goal is to protect against physical attacks.

As memory latency in AES-XTS mode is higher than in AES-CTR mode, the encryption of XTS is much stronger. This stronger encryption of XTS is required for persistent memory. The reason being that an attacker has arbitrary access to the persistent DIMM data when the card leaves the Data Center for repair or replacement. With the volatile DIMMs, the data will be gone after the power goes down and therefore AES CTR is a choice to make it more difficult to physically gain data through the memory card interfaces with a reduced latency add.



The MCU crypto engine has been designed as fully pipelined. Fetch and store bandwidth are not compromised by either of the encryption modes. Meta bits will not be encrypted. All encryption relates to the data only.

#### 10.4.2 Encryption Keys

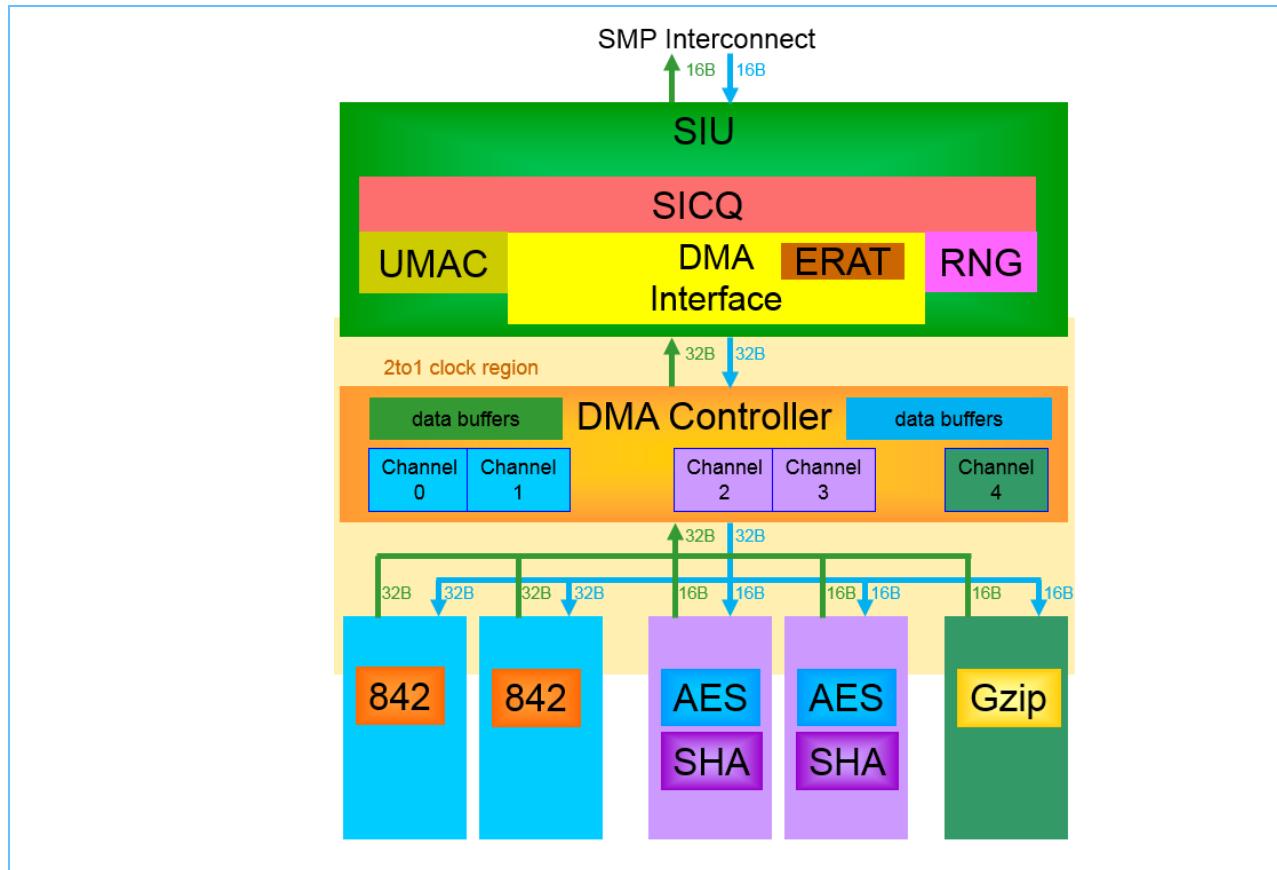
Both the XTS and CTR modes use 128-bit encryption keys. The host boot (HB) firmware sets up the crypto mode to be used based on the system configuration. The MCU implementation supports that each OMI sub-channel pair has its own keys. For CTR mode, such a key per OMI pair is recommended, but host boot can choose to either have a single system-wide key or a key per OMI sub-channel pair.

Note that not all systems in all markets can or will have this feature enabled. An eFuse on the Power10 chip can be destroyed (blown) to prevent the enabling of encryption. Once the eFuse has been blown, the entire memory encryption function can never again be enabled for that chip.

## 11. On-Chip Accelerators

The Nest Accelerator unit (NX) consists of cryptographic and memory compression/decompression engines (coprocessors) with support hardware. *Figure 11-1* shows a block diagram of the NX.

*Figure 11-1. NX Block Diagram*





## 11.1 NX Features

The NX coprocessors and their features are as follows:

### *Cryptographic Engines*

Advanced encryption standard (AES) engine:

- Modes: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Count (CTR), Counter with CBC-MAC (CCM), Galois Counter Mode (GCM), XCBC-MAC-96 (XMAC)
- Key lengths: 128 bits, 192 bits, 256 bits
- Two engines

Secure Hash Algorithm (SHA) engine:

- Modes: SHA-1, SHA-256, SHA-512, Message Digest 5 (MD5).
- Keyed-hash message authentication code (HMAC) supported for SHA.
- Two engines.
- Combined AES and SHA operations are supported on a single movement of data through an AES/SHA engine pair.

Random Number Generator (RNG):

- All digital design with dual noise sources
- NIST 800-90B draft standard compliant
- Hardware random number (RN) conditioning
- Continuously running health tests with noise source failover on test failure
- Produces 64-bit conditioned random numbers (CRN) and raw random numbers (RRN) readable by the darn instruction

Compression and Decompression:

- 842 compression/decompression
- IBM-proprietary algorithm with 8-byte, 4-byte, and 2-byte phrase parsings
- High throughput and low latency with good compression and silicon efficiency
- Two engines

Gzip Compression and Decompression:

- Industry standard DEFLATE RFC 1951 compliant
- Supports RFC 1950 (zlib) and RFC 1952 (gzip) file formats
- Fixed and dynamic Huffman table support
- Assist for dynamic Huffman table creation
- Ability to suspend an operation when a byte count limit is hit
  - Software can resume operation from a suspend point after adjusting the job parameters
- Bypass mode for data move
- High throughput with better compression efficiency (relative to 842)



Each one of the AES/SHA, 842, and Gzip units consist of a coprocessor type (CT). As such, NX has three coprocessor types.

To support coprocessor invocation by user code, use of effective addresses, high-bandwidth storage accesses, and interrupt notification of job completion, NX includes the following support hardware:

SMP interconnect unit (SIU):

- Interfaces to SMP interconnect and direct memory access (DMA) controller
  - Provides 16-bytes per cycle data bandwidth per direction to both
- Employs SMP interconnect common queue (SICQ) multiple parallel read and write machine architecture to maximize bandwidth
  - 16 DMA read machines, each with one cache-line data buffer storage
  - 16 DMA write machines, each with one cache-line data buffer storage
  - Three UMAC read machines, each with one cache-line data buffer storage
  - Three UMAC write machines, each with one doubleword data buffer storage
  - Eight RN read machines with buffer storage for eight doubleword CRN plus eight doubleword RRN
- User-mode access control (UMAC) coprocessor invocation block
  - After the Virtual Accelerator Switchboard (VAS) accepts a CRB that was initiated by a copy/paste instruction, the UMAC snoops the VAS's notification for an available coprocessor request block (CRB or job).
  - Supports one high- and one low-priority queue per coprocessor type
  - Retrieves CRBs from queues and dispatches CRBs to the DMA controller
- Effective-to-real address translation (ERAT) table stores 32 recently used translations
  - Interfaces to nest memory management unit (NMMU) for address translation services
  - Translates all effective addresses (EA) from DMA controller to real addresses (RA)
  - Returns translation faults to DMA controller
- Snoops **darn** instruction command for RN delivery

DMA Controller:

- Decodes CRB to initiate coprocessor and move data on behalf of coprocessors
- Uses effective addresses for all CRB storage accesses
  - Issues paste command to VAS to dispense CRB with translation fault to per-partition fault queue
- 5-channel data mover, one per each instance of AES/SHA, 842, Gzip engine, with buffers for data to and from engines
- Two CRB queue positions per channel: one for current CRB (currently executing on a coprocessor) and one for pending CRB (awaiting execution)
  - Can prefetch coprocessor parameter block (CPB) and source data for pending CRB
- Provides prefetch hints to memory controller to reduce read latency
- Supports byte-aligned source and target data
- Supports scatter/gather through data descriptor list (DDL)
- Supports interrupt notification on CRB completion
- 16 bytes per cycle data bandwidth per direction to/from SIU
- 16 bytes per cycle data bandwidth toward 842 engines, 8 bytes per cycle toward AES/SHA, Gzip
- 16 bytes per cycle data bandwidth from 842 engines, 8 bytes per cycle from AES/SHA, Gzip



Most of the NX unit operates on the nest clock domain, but the DMA controller operates on a clock period twice that of the nest clock (see “2:1 clock domain” in *Figure 11-1* on page 197). To match the SIU data bandwidth, the DMA controller buses are twice as wide as the SIU buses, as shown in the figure.

## 11.2 Using NX Coprocessors

NX coprocessors can be invoked through library or operating system kernel calls that use the Power ISA copy/paste facility. See [GITHUB](#) for links to both Linux and AIX operating systems, as well as low-level gzip engine invocation details for library or kernel coders.



## 12. Virtual Accelerator Switch

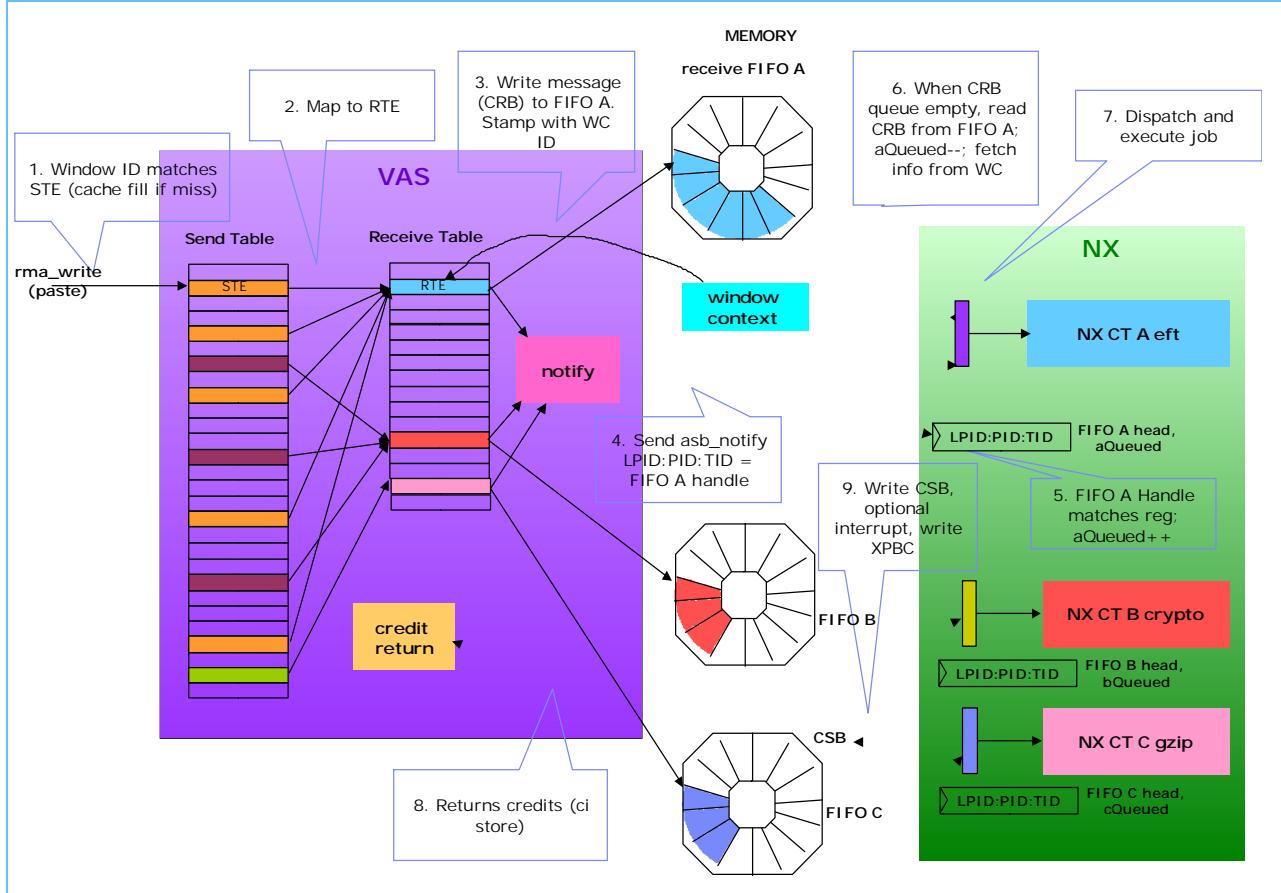
### 12.1 Overview

The main function of the Virtual Accelerator Switch (VAS) unit is to enable user-level software code running on a processor core direct access to the Nest Accelerator (NX) unit accelerator engines without the requirement for an expensive hypervisor call to initiate accelerator usage. This allows user-level code direct access to the cryptographic and compression accelerator engines on the Power10 chip.

### 12.2 Flow for NX Invocation Through the VAS

When an operating system wants to initially enable a user process access to an NX accelerator, it must communicate with the hypervisor. The hypervisor sets up a Send Window Table Entry (STE). A send window can be unique per process, or can be shared by an operating system among its many user processes. When the hypervisor establishes the send window, it assigns it a quantity of credits indicating how many operations are allowed at one time and returns an address handle to the operating system. Then, the operating system returns an effective address to the user. This effective address allows the user process to directly post entries to the FIFOs associated with the NX accelerators. A separate send window is necessary for each accelerator a process uses.

Figure 12-1. Flow for NX Invocation through the VAS



1. After a send window has been established, the user process can begin using the NX accelerator. First, it must create a coprocessor request block (CRB). The NX specification defines the format of the CRB. This CRB is sent to the VAS unit by using the Power10 copy/paste instructions. The copy instruction places the CRB into the copy buffer. The user process then issues a paste instruction using the effective address given by the operating system during send window creation to store the copy data to the VAS. The copy data contains the 128-byte CRB. The effective address is translated to a real address by translation hardware in the core. The store to the real address is issued to the SMP interconnect as a remote memory access write (RMA\_write) command and has the send window identifier embedded within the real address. The 128-byte RMA\_write payload (the CRB) is stored into one of 64 VAS data buffers.

The VAS has the ability to hold 128 unique window contexts. Upon snooping the RMA\_write, the VAS uses the send window identifier to fetch the Send Window Table Entry from memory if not already resident with the VAS window cache logic.

2. The VAS reads the Receive Window Identifier field in the send window context to determine which receive window the send window from the RMA\_write points to. Each NX coprocessor type (CT) has a unique receive window corresponding to a unique FIFO for each of the accelerators.

If the receive window is not cached, it will be fetched from memory.



As shown on the left side of *Figure 12-1* on page 202, many different send windows, each associated with a user process, can point to the same receive window. In fact, all send windows that are using the same NX CT can point to the same receive window, because there is one receive FIFO per CT.

3. Using the FIFO address from the receive window context, VAS stores the RMA\_write payload to memory, thereby placing the CRB onto the NX accelerator FIFO. VAS stamps or overlays a portion of the CRB with the send and receive window identifiers. NX uses this information when processing the CRB. In particular, the send window identifier in the CRB is used by NX to fetch the send window and obtain translation information for the addresses contained within the CRB.

The receive FIFOs are implemented as circular queues. After reaching the end of the FIFO, VAS wraps back to the beginning of the FIFO and writes the next entry.

4. After writing the CRB to the FIFO, VAS sends an ASB\_notify command on the SMP interconnect. The ASB\_notify contains a logical partition identifier (LPID), process identifier (PID), and thread identifier (TID).
5. Each NX FIFO has a particular LPID:PID:TID combination associated with it. When NX snoops an ASB\_notify that matches its programmed LPID:PID:TID, it increments the corresponding counter for the associated FIFO, indicating a new work item has been placed on the accelerator FIFO.
6. When an NX CT queue is empty and its counter is nonzero, NX reads the next CRB from the receive FIFO. As soon as the CRB is read from the FIFO, NX does a memory mapped (MMIO) store to the VAS unit to return a credit. VAS ensures that the receive FIFO does not overflow by managing credits. The hypervisor initializes the receive window with credits equal to the number of CRBs that can be stored to the receive FIFO based on the size of the FIFO. VAS decrements the receive credit count when it stores a CRB to the receive FIFO and increments the count when NX returns a credit via MMIO store after NX pulls the CRB off of the FIFO.

NX uses the stamped information from the CRB to read the send window context from memory and decrements its internal counter.

7. NX dispatches the job to the associated CT, which can have multiple acceleration engines, and executes the CRB.
8. Upon completion of the job, NX returns a send window credit to VAS via an MMIO store. Each send window, when created by the hypervisor, is assigned a number of send credits. This allows the hypervisor to implement quality of service by managing numerous users sharing the same accelerator resource, and preventing one process from using more than its share. When an RMA\_write command is received by VAS, VAS decrements the send credit for the associated send window. VAS increments the count when NX completes the CRB and returns a send credit with an MMIO store.
9. NX writes a coprocessor status block (CSB) and can optionally send an interrupt, which notifies the user that the job has completed. NX also updates the accelerator processed byte count (XPBC) in the send window indicating the number of bytes that were processed on behalf of the user.

As shown in *Figure 12-1* on page 202, many different send windows can point to the same receive window. This occurs when many different processes are using and sharing the same NX CT. Each process writes a FIFO entry onto the NX queue, independently, with no ordering implied nor maintained between different processes and different send windows.



## 12.3 Features

The following features are implemented in support of NX:

- 64 KiB window support per VAS. For a large SMP with 16 VAS in the system, this gives 1 MiB window accessibility.
- Data stamping of send and receive window information into the CRB.
- Send window credits.
- Receive window credits.
- N → 1 support. Multiple sending windows are allowed to point to one receive window.
- NX utilization reporting (XPBC support).
- ASB\_Notify notification.
- Ability to pin send or receive window contexts in the cache to avoid casting out frequently-used windows.
- Caching of 128 unique windows contexts.

## 13. OpenCAPI Processing in the POWERAccel Unit

This section describes the Open Coherent Accelerator Processor Interface (OpenCAPI) interfaces and the parts of the POWERAccel unit (PAU) that provide the transaction layer functionality for those interfaces.

The OpenCAPI interface enables an attached functional unit (AFU) to connect to the Power10 on-chip SMP interconnect bus in a high-speed, cache-coherent manner. For example, this structure can be used to connect a Power10 chip to an FPGA containing an acceleration function or a data storage function. The accelerator or data storage function represent the AFUs on the FPGA. The Power10 chip and the AFU have the ability to coherently read and write memory attached to either chip. The AFU can use caching operations to act as a peer on the Power10 chip's interconnect bus. Supporting OpenCAPI on the Power10 chip requires OpenCAPI-capable PHYs, datalink layer logic, and transaction layer logic. The PHYs are the physical connection to the OpenCAPI interconnect. The datalink layer provides link training, CRC generation and checking, and the replay of failed packets. The transaction layer executes the cache-coherent and data-movement commands on the Power10 chip.

The PAU provides the transaction layer functionality for the OpenCAPI links on the Power10 chip. This functionality includes accepting commands from the OpenCAPI datalink logic, converting them into sequences of on-chip SMP interconnect commands, and then generating responses based on the results of the on-chip SMP interconnect commands. The responses are sent back to the OpenCAPI link through the datalink logic. The supported commands include reads, writes, interrupts, and cache-management operations. The PAU sends reads, writes, and cache management operations to the AFU over the OpenCAPI link as a result of operations seen on the on-chip SMP interconnect buses. In addition, transaction-layer command response and data credits are passed in both directions over the link.

An OpenCAPI connection to the Power10 chip is composed of one or more individual links. Each link provides a separate stream of commands. All ordering requirements are enforced independently for each link. The Power10 PAU supports two OpenCAPI links. Six PAUs are instantiated on the Power10 chip for a total of twelve possible OpenCAPI links.

### 13.1 Functions Supported by the Power10 PAU Over OpenCAPI Links

In addition to AFU connectivity, the Power10 PAU supports the following two functions using OpenCAPI links and are described in the following subsections.

- Memory inception
- Integrated networking

#### 13.1.1 Memory Inception

Memory inception (MI) enables one Power10 chip to access memory on another Power10 chip when the two Power10 chips are not connected via an SMP interconnect bus. The MI connection between the two chips is made using one or more OpenCAPI links that are configured in a peer-to-peer mode.

With this connection in one example use case, each Power10 chip can lease memory space from the other Power10 chip. Such as a cluster of systems where a Power10 chip in one of the systems has a large amount of storage class memory attached to it. Software in another system could be given the ability to use some of that space. The system with the Power10 chip that has the memory attached to it is referred to as the server. The server leases space in the memory to the other system which is referred to as the client. When this is



done, the client incepts the space into its own SMP memory map. Software running on the client sees the incepted memory space as fully coherent memory. In this particular example, user-level software on the server system is not allowed to access memory that is leased to a client.

### **13.1.1.1 Memory Inception Mirroring**

The Power10 PAU supports a mirroring mode with memory inception. In this mode, one of the OpenCAPI links supported by the PAU is designated as primary and the other as secondary. Writes are sent to both the primary and secondary links. Reads are only sent to the primary link. If the primary link fails, reads can be rerouted to the secondary link.

### **13.1.2 Integrated Networking**

A peer-to-peer OpenCAPI connection between Power10 chips can be used, with support in the Power10 PAU, to create a network of systems. The PAU provides virtual channel support that allows several network configurations to be constructed. These include the following:

- Ring topology,
- Two-tier Dragonfly topology
- Four-dimensional Torus topology

In these networks, command routing is accomplished using a combination of SMP-interconnect routing and OpenCAPI address translation. This combination allows a write command originating on one Power10 chip to be delivered, through a series of intermediary Power10 chips, to a memory location attached to a destination Power10 chip, following a predetermined route. The route can be direct, meaning a route with the least number of intermediate nodes; or indirect where an alternate, albeit longer, route is used to help balance bandwidth by offloading the links in the direct route. In integrated-networking mode, write commands are used to send information to the destination node. Read commands are not supported.

## **13.2 Features**

A summary of the features are as follows:

- OpenCAPI link bandwidths:
  - Peak read bandwidth per link: 32.0 GBps (25.6 GBps for an FPGA AFU)
  - Peak write bandwidth per link: 32.0 GBps (25.6 GBps for an FPGA AFU)
- The PAU supports two OpenCAPI links
- Effective bandwidths including command and response overhead:
  - Read bandwidth per link: 28.4 GBps (22.7 GBps for an FPGA AFU)
  - Write bandwidth per link: 28.4 GBps (22.7 GBps for an FPGA AFU)
  - Total read bandwidth for the PAU: 56.9 GBps (45.5 GBps for an FPGA AFU)
  - Total write bandwidth for the PAU: 56.9 GBps (45.5 GBps for an FPGA AFU)
- L2 directory size: 16K entry (1K × 4)

- Address translation sizes and rates are as follows:
  - Address translation cache (ATC): 4K entry ( $1K \times 4$ )
  - Context cache: 512 entry ( $128 \times 4$ )

### 13.3 Power10 AFU Transaction Examples

The following sections contain examples of transactions between Power10 and OpenCAPI AFU chips using the OpenCAPI link and the on-chip SMP interconnect.

#### 13.3.1 Read from AFU to Power10 Memory

*Table 13-1 shows an example of a 128-byte read command sent over an OpenCAPI link.*

*Table 13-1. Read from AFU to Power10 Memory*

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	SMP Interconnect	From SMPI	Comment
1		→ RD_WNITC			→ NON_CACHING_READ		The AFU requests a read-with-no-intent-to-cache (RD_WNITC). The PAU sends a non-caching read command to the SMP interconnect.
2					Memory_Ack	←	The Power10 memory controller acknowledges the command.
3					Data	←	The Power10 memory controller sends data for the read command to the PAU over the SMPI data bus.
4		READ_RESPONSE Data	←				The PAU sends a response containing the data over the OpenCAPI link to the AFU.

**Note:** OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response



### 13.3.2 Read for Shared Access from AFU to Power10 Memory

*Table 13-6 shows an example of a 128-byte read for shared access command sent over an OpenCAPI link.*

*Table 13-2. Noncaching Read from AFU to Power10 Memory*

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	SMP Interconnect	From SMPI	Comment
1		→ READ_S			→ CACHE_LINE_READ		The AFU requests a read for shared access (READ_S). The PAU sends a CACHE_LINE_READ to the SMP interconnect.
2					Memory_Ack, Shared_OK	←	The Power10 memory controller acknowledges the command. The Power10 cache state allows the AFU to get a shared copy.
3					Data	←	The Power10 memory controller sends data for the read command to the PAU over the SMPI data bus.
4		CL_RD_RESP Data	←				The PAU sends a response containing the data over the OpenCAPI link to the AFU.

**Note:** OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response

### 13.3.3 AFU Writes to Power10 Memory

*Table 13-3 shows an example of a series of 128-byte write commands sent over the OpenCAPI link to the Power10 chip. In this example, it is assumed that the final write must occur after the previous writes have completed. A case where this is required is when the final write is an indication to software that the previous writes were completed.*

*Table 13-3. AFU Writes to Power10 Memory (Sheet 1 of 2)*

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	On-Chip SMP Interconnect	from SMPI	Comment
1		→ DMA_W_0			→ DMA_INJECT_0		The AFU sends a stream of three DMA write commands on the same link. The PAU sends the write commands onto the SMP interconnect as DMA_INJECT commands.
2		→ DMA_W_1			→ DMA_INJECT_1		
3		→ DMA_W_2			→ DMA_INJECT_2		
4					Cache_Ack_1	←	The responses for the inject commands can arrive out-of-order. DMA_INJECT_1 hit a cache and was handled by the cache controller. The other two injects were handled by the Power10 memory controller. All of the responses indicate that the writes are globally visible.
5					Memory_Ack_0	←	
6					Memory_Ack_2	←	

**Note:** OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response



Table 13-3. AFU Writes to Power10 Memory (Sheet 2 of 2)

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	On-Chip SMP Interconnect	from SMPI	Comment
7		Write-Response_1	←		→ data_1		The PAU sends the data for the DMA_INJECT commands and sends responses for each of the DMA writes to the AFU.
8		Write-Response_0	←		→ data_0		
9		Write-Response_2	←		→ data_2		
10		→ DMA_W_3			→ DMA_INJECT_3		Because the final write must occur after the previous writes are done, the AFU waits for all of the responses for the previous writes before sending the final one. The PAU sends this write command onto the SMP interconnect as a DMA_INJECT command.
11					Memory_Ack_3	←	The response indicates that the inject command is complete on the SMP interconnect.
12		Write-Response_3			→ data_3		The PAU sends the data for the DMA_INJECT command and sends a response for the final write.

Note: OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response



### 13.3.4 AFU Posted Writes to Power10 Memory

Table 13-4 shows an example of a series of 128-byte posted write commands sent over the OpenCAPI link to the Power10 chip. In this example, it is assumed that the final write must occur after the previous writes have completed. A case where this is required is when the final write is an indication to software that the previous writes were completed.

Table 13-4. AFU Posted Writes to Power10 Memory

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	On-Chip SMP Interconnect	from SMPI	Comment
1	→ DMA_W.T.P_0			→ DMA_INJECT_0			The AFU sends a stream of three posted DMA write commands on the same link. These writes use previously translated addresses. The PAU sends the writes onto the SMP interconnect as DMA_INJECT commands.
2	→ DMA_W.T.P_1			→ DMA_INJECT_1			
3	→ DMA_W.T.P_2			→ DMA_INJECT_2			
4	→ DMA_W.T.P.S_3						Because the final posted write must occur after the previous writes are done, the AFU sends it as a dot-s (presync) form.
5				Cache_Ack_1	←		The responses for the inject commands arrive out-of-order. DMA_INJECT_1 hit a cache and was handled by the cache controller. The other two injects were handled by the Power10 memory controller. All of the responses indicate that the writes are globally visible.
6				Memory_Ack_0	←		
7				Memory_Ack_2	←		
8				→ data_1			The PAU sends the data for the DMA_INJECT commands.
9				→ data_0			
10				→ data_2			
11				→ DMA_INJECT_3			The PAU insures that all previous writes in the stream are complete and then sends the dot-s form write onto the SMP interconnect as a DMA_INJECT.
12				Memory_Ack_3	←		The response indicates that the inject command is complete on the SMP interconnect.
13				→ data_3			The PAU sends the data for the DMA_INJECT command.

**Note:** OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response

### 13.3.5 Read from Power10 Chip to AFU M1 Memory

Table 13-5 shows an example of a Power10 read of the AFU memory.

*Table 13-5. Power10 Read of AFU M1 Memory*

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	On-Chip SMP Interconnect	From SMPI	Comment
1					READ	←	A read command targeting AFU memory appears on the SMP interconnect.
2		RD_MEM	←	→	PAU_Ack		The PAU accepts the read command, acknowledges it, and sends a RD_MEM command to the AFU.
3		→ MEM_RD_RESPONSE		→	Data		When the AFU responds with the data for the read, the PAU places it on the SMP interconnect as the data transfer for the read command that was accepted in step 1.

1. OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response.

2. AFU M1 memory is only accessible through the Power10 chip. There is no direct access from the AFU itself.

### 13.3.6 Write from Power10 Chip to AFU M1 Memory

Table 13-6 shows an example of a Power10 write to the AFU M1 memory.

*Table 13-6. Power10 Write to the AFU M1 Memory*

Step	AFU Drives OCL	OpenCAPI Link	PAU Drives OCL	PAU Drives SMPI	On-Chip SMP Interconnect	from SMPI	Comment
1					WRITE	←	A write command targeting the AFU memory appears on the SMP interconnect.
2				→	PAU_Ack		The PAU accepts the write command and responds with an acknowledge.
		WRITE_MEM	←		Data	←	The requester of the write command sends the write data to the PAU. The PAU then sends a WRITE_MEM command to the AFU.
3		→ MEM_WRITE_RESPONSE					When the AFU responds to the write, the command is complete.

1. OCL = OpenCAPI Link; Ack = Acknowledge; Resp = Response.

2. AFU M1 memory is only accessible through the Power10 chip. There is no direct access from the AFU itself.





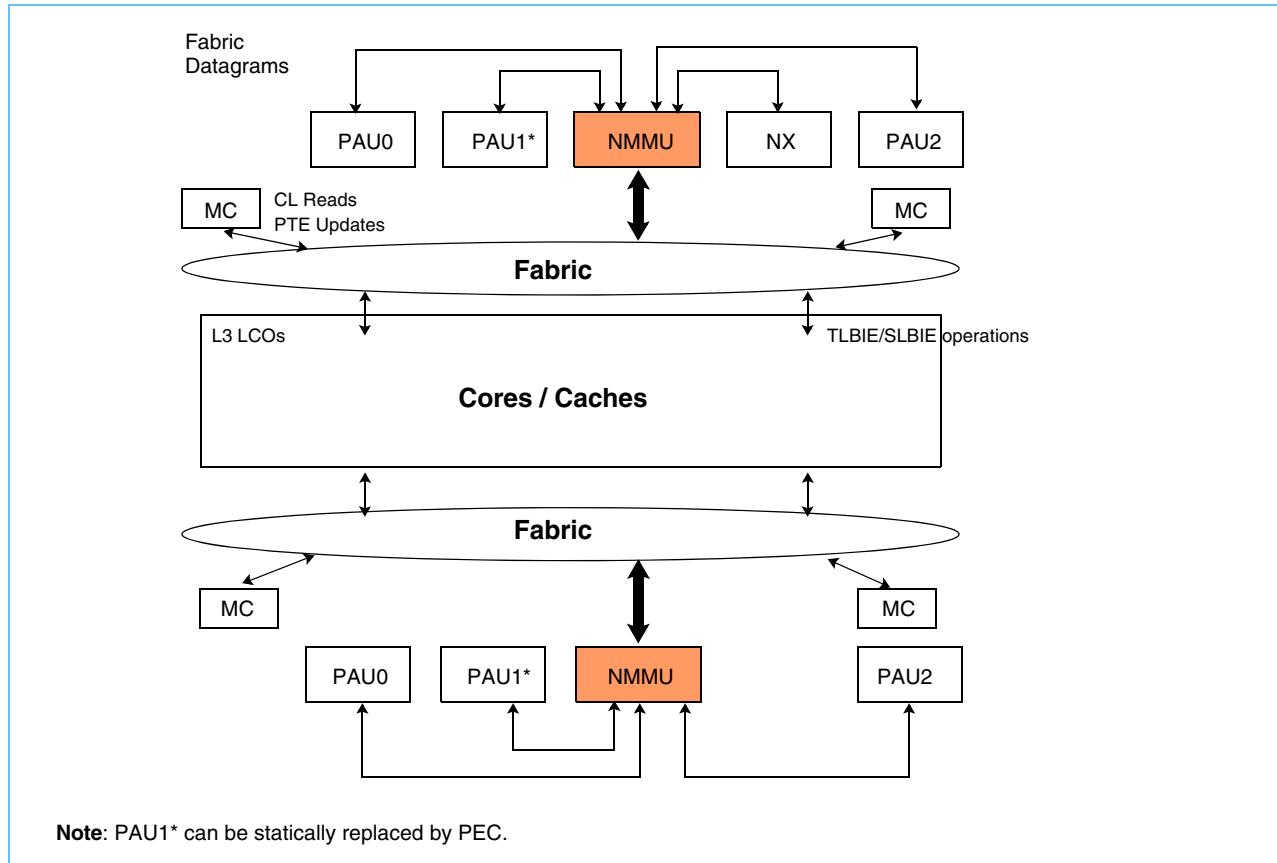
## 14. Nest MMU

This chapter describes the overall microarchitecture for the nest memory management unit (NMMU) implemented in the Power10 processor. The primary goal of this unit is to provide effective address (EA) to real address (RA) translation for the various accelerator agents within the processor's storage subsystem. In addition, the NMMU protects the pages that are being translated by ensuring that only tasks with the proper authorization are allowed to access them.

On the Power10 chip, two NMMU units reside within the chip as shown in *Figure 14-1* on page 214. Each NMMU services the translation clients within its own endcap of the chip. In the north endcap, the NMMU services three PAU units (formerly known as NPU on the POWER9 chip) and the NX. The NMMU in the south endcap services three PAU units. Note that the PEC may statically replace PAU1 as a translation client for the Power10 chip as a PCIe unit requiring address translation to be serviced by the NMMU in each endcap. The NMMU's primary function is to translate effective (logical) addresses into real (physical) addresses for memory accesses on behalf of these accelerator agents. The unit's focus is on data accesses to memory generated by loads and stores. In its primary PowerPC mode, the NMMU's translation mechanism is defined by segment descriptors and page tables, as set up by the hypervisor. In addition to translation, the NMMU provides various levels of access protection on a per segment and page basis.

As shown in *Figure 14-1* on page 214, the Power10 nest MMU primarily communicates with external units through the system bus (Fabric). All translation protocols with the accelerator units are run over the fabric via data-only operations. The NMMU also interacts with memory to perform tablewalks and to update the page tables, as required. In addition, cache management instructions (SLB and TLB invalidates) are sourced by the core/NCU in the system and are snooped and managed by the NMMU on behalf of the attached accelerator units.

Figure 14-1. Power10 Nest MMU





## 14.1 NMMU Features

A summary of the NMMU features follows:

- Translation mechanisms supported:
  - Single-level translation
    - PowerPC hashed page table (HPT) approach (via POWERVM)
  - Dual-level translation
    - Radix-on-radix page table approach (via Linux over [KVM](#))
  - Single-level radix translations (process-scoped, partition-scoped)
- Functions supported:
  - EA-to-RA translations
  - Memory protection at segment and page levels
  - Page sizes supported:
    - Radix: 4 KB, 64 KB, 2 MB, 1 GB
    - HPT: 4 KB, 64 KB, 16 MB, 16 GB
  - Segment sizes supported: 256 MB, 1 TB for HPT translations
  - 64-bit effective address (EA), 68-bit virtual address (VA), 56-bit real address (RA)
  - Supports 12 simultaneous tablewalks
    - Responsible for acquiring segment table entries (STEs) and page table entries (PTEs) from segment and page tables residing in main memory
  - Optional TLB/SLB invalidation management on behalf of the inclusive accelerators (NMMU supports **slbie** and **tlbie**)
- Translation protocol:
  - Checkout phase
    - Agent requests that the tandem NMMU resolve a given translation
  - Check-in phase (applies to inclusive agents only)
    - Upon completion of relevant processing (active eviction) or local cache castout (passive eviction), the agent signals that the translation is no longer in use by checking it back into the NMMU's TLB.
  - Invalidation phase (applies to inclusive agents only)
    - Due to SLB/TLB invalidations snooped by the NMMU (**slbie**, **tlbie**)
    - Due to LRU castout of NMMU cache (TLB/SLB)
- Primary customers (nest accelerators that require translation):
  - NX: eight concurrent checkout/check-in operations total, one invalidate/barrier operation
  - PAU: eight checkout operations per instance (three instances per Power10 endcap, one instance can be replaced by a PEC)
- Accelerator agent/NMMU interface communication mechanism
  - Via Fabric data bus (for common platform, floorplan flexibility, and future extendability)
- Cache resources available for translations
  - Local cache (ERAT) resides within the accelerator agent
    - NX: 32-entry local cache (ERAT) arranged in a [CAM/RAM](#) structure
    - PAU: Assuming a 64-deep intermediate buffer ahead of the ERAT held in Nvidia's GPU chip

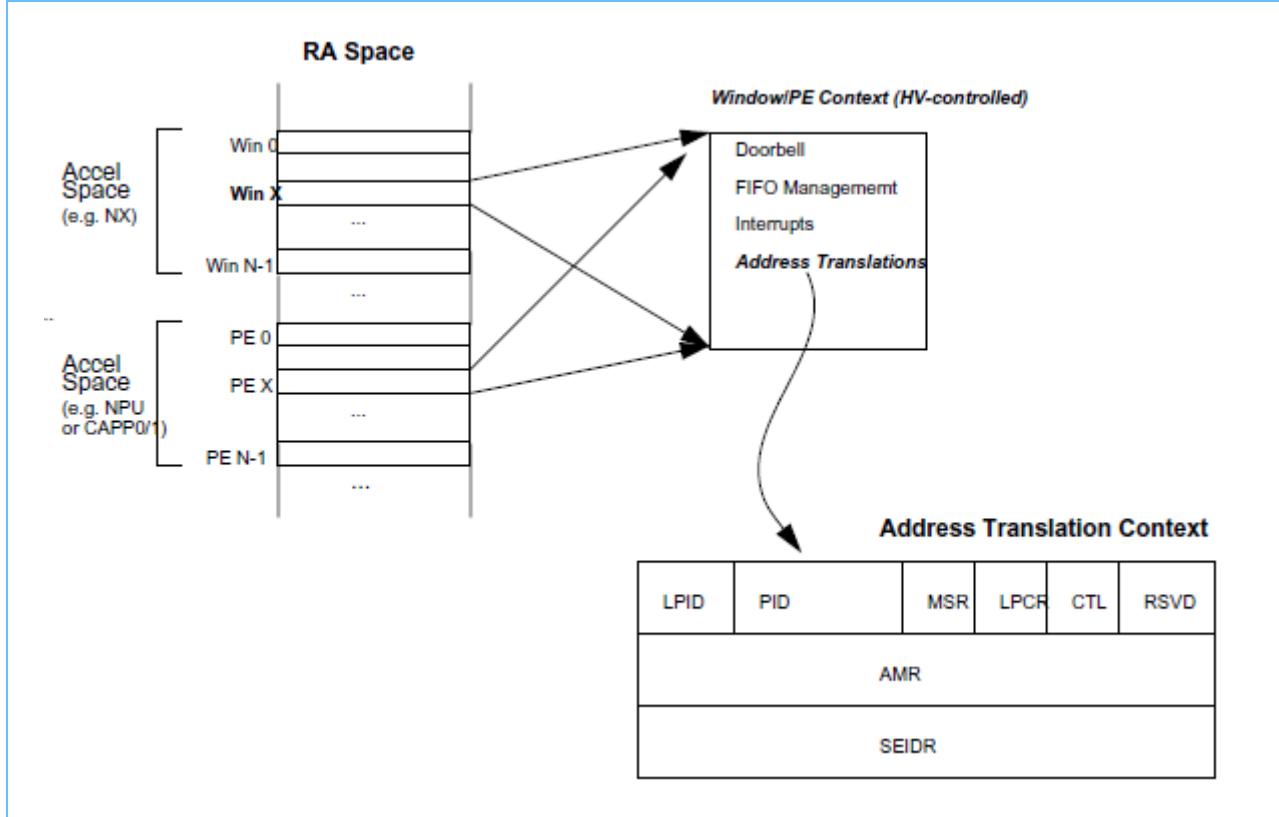


- SLB
  - 256-entry, 16-way set associative (number of entries targeted at matching number of cached windows)
  - Used to cache the most recent copies of STEs
- TLB
  - 8192-entry, 16-way set associative
  - Used to cache the most recent copies of PTEs, inclusive with agent's ERAT
- Radix Page Walk Cache (PWC):
  - L1: 256-entry, 16-way set associative (128 LPID/PID pairs per guest and host)
  - L2: 512-entry, 16-way set associative (128 pairs × 2 branches per guest and host)
  - L3: 1024-entry, 16-way set associative (128 pairs × 2 branches × 2 sequential entries/branch per guest/host)
  - L4: 2048-entry, 16-way set associative (128 pairs × 2 branches × 4 sequential entries/branch per guest/host)
- Fabric snoop interface:
  - Four sets of snoop and partial response interfaces (ports 2-3 added for LCO support)
  - One 16-byte inbound data ramp
  - One 16-byte outbound data ramp
  - Primarily for agent communication, SLBI/TLBI-related commands (assuming no MMIOs required)
- Fabric master interface:
  - One master interface (single command bus, four combined response ports)
  - Used for in-memory table reads (process table), STEG/PTEG lookups, and PTE updates
- Array parity error recovery mechanism
- Clocking/frequencies:
  - Nest clock frequency (2 GHz), 1:1 with Fabric

## 14.2 Window/Process Element Context

For the accelerators in the nest, a window or process element (PE) is the communication mechanism where trusted software provides the context required by the hardware to process a given packet or message. The context for a window/process element is located in main memory, as illustrated in *Figure 14-2*, and is maintained by the hypervisor. A window/process element is the portal that links system software and hardware together and provides the common structures that dictate how tasks through this window are processed. Ultimately, a window ID (or process element ID) is the index into the context stored in memory.

Figure 14-2. Window/Process Element Context



#### 14.2.1 Rules for Managing Address Translation Contexts

Any agent that requires use of the NMMU to translate an effective address (or guest virtual address) must provide the address translation context (see *Figure 14-2*) with the corresponding checkout request. For the format of the address translation context within a process element, see the *Coherent Accelerator Interface Architecture*.

The following general rules describe the behavior of contexts with respect to the NMMU.

- Note that an address translation context is tied to a particular LPID/PID combination and is assumed to be static. Therefore, any changes to the address translation context during the NMMU's runtime must be accompanied by invalidates (**s1bie/tlbie** operations) for any pending translations for the given LPID/PID pair before the updated context takes effect.
- From a context perspective, the LPID/PID refers to the effective LPID and effective PID.
- From an NMMU perspective, multiple window/process contexts can exist for the same LPID/PID pair, but the address translation context must be identical across the window/process contexts (with the potential exception for AMRs used to “color” accesses differently based upon particular context IDs).
- For a given LPID, even subtle differences in the context must map to separate and distinct contexts. This is the preferred way to set up contexts with similar address xlat contexts, as opposed to incurring the snoop invalidate overhead penalty for changing the contexts for a given LPID/PID pairing with each update.



- Hypervisor real mode translations must be run through a dedicated LPID/PID pair (context) and cannot change until the next IPL. This is due to there not being an invalidate path for HVreal mode operations.
- For changes to the AMR (for HPT virtual protection checks or for key0 of Radix EAA protection) of the address translation context, the following subset of rules apply.
  - For NX usage, software is encouraged to use multiple contexts (or process elements) for a given LPID/PID pair where only the AMR varies. This enables software to “color” regions of access based upon the context/process element ID, which is compared in the NX ERAT, and permits it to avoid snoop invalidate overhead in its code.
  - Any updates to the AMR within a given context/PE for the NX requires **tlbie** operations to invalidate any old ERAT entries prior to the change.
  - For PAU usage, software is required to issue **tlbie** operations for any change to the AMR anywhere in the context. It is expected that this would be a rare case for the exclusive use of radix translations in the PAU.



## 15. Interrupt Controller

The Power10 interrupt controller (INT) consists of three major units: the virtualization controller (P3VC), the presentation controller (P3PC), and the Power10 Fabric bus interface common queue (P3CQ). These units work together to take triggers from interrupt sources and deliver exceptions to the appropriate processor thread. This section provides an overview of the interrupt architecture, describes the INT units and their interfaces, and also describes how they operate with the interrupt sources and software in the Power10 infrastructure.

**Note:** The “P3” prefix in the unit acronym name refers to version 3 of the interrupt architecture. The previous version was referred to as version 2.

### 15.1 External Interrupt Virtualization Engine

The Power10 interrupt architecture can be configured to present a binary compatible interface with the first generation External Interrupt Virtualization Engine architecture. It significantly reduces the interrupt code overhead/path length and improves performance compared to the previous architecture. Other advantages of this architecture over previous versions are:

- Enables direct user-level I/O device drivers:
  - Direct I/O adapter interrupts to user-level event-based branches
  - Significantly simplifies CAPI models and path length
- Enables direct user-level virtual I/O signaling:
  - Significantly simplifies scalable inter-processor/partition signaling
  - Compliments scalable virtual super-sockets
- Combines all notification mechanisms into one architecture:
  - External interrupts, including inter-processor interrupts (IPI), targeting:
    - Operating system (OS)
    - Hypervisor
    - Event-based branch (EBB)
  - Thread resume signaling
  - Enables authorized signaling by:
    - I/O device
    - Platform service
    - Program at any privilege level
  - Adds routing to the dispatched logical server in addition to the physical thread:
    - Required combination of LPID, VP/VT, PID, TID
  - Removes hypervisor from the path, except:
    - When required to dispatch a logical server
    - To handle extreme scalability
    - To handle corner cases in page migration



## 15.2 High-Level Block Diagram

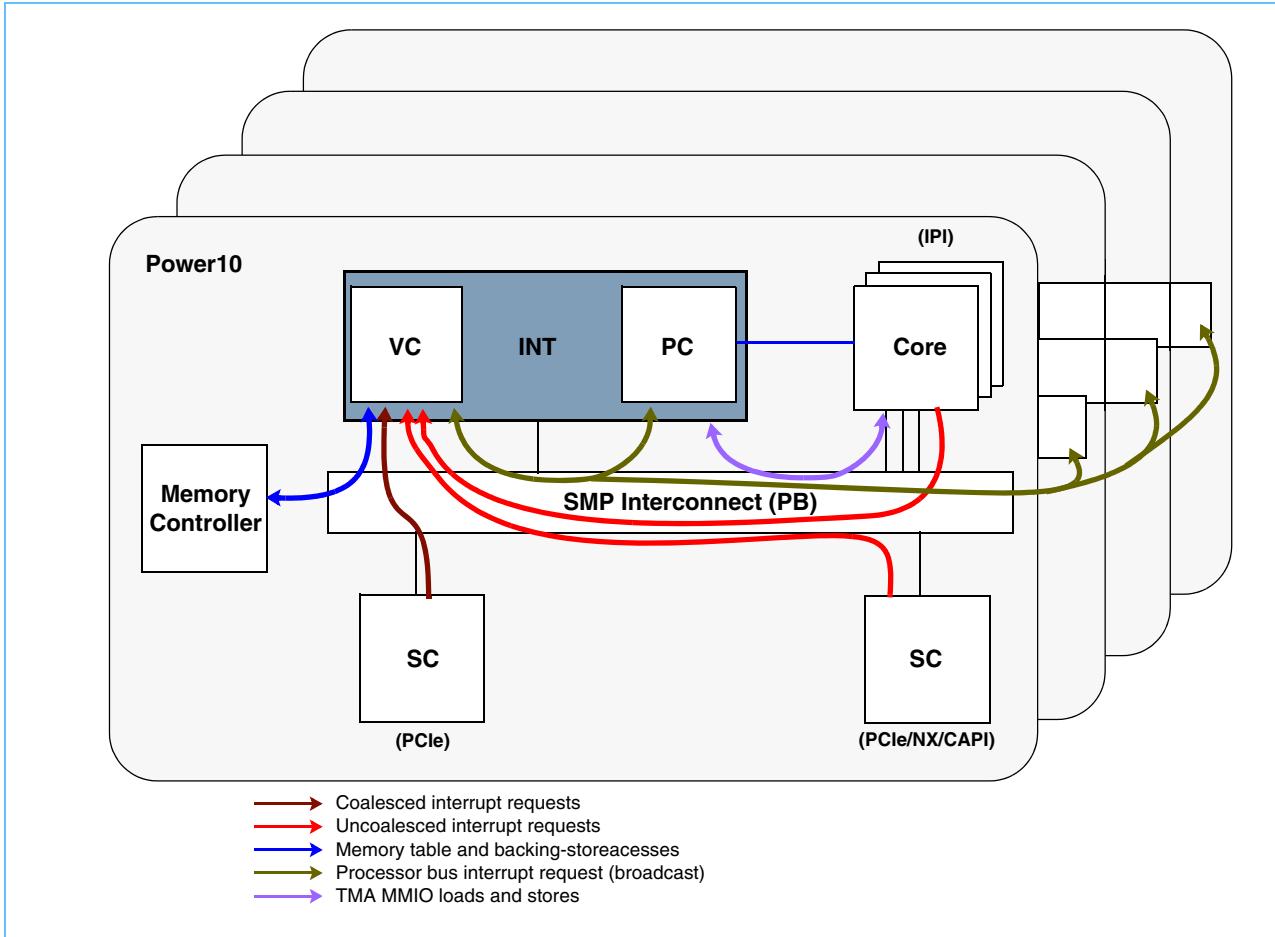
The high-level diagram, *Figure 15-1* on page 221, depicts the conceptual interaction among sources and the controller blocks in interrupt signaling and notification. The individual elements are interconnected and communicate via the Power10 Fabric bus.

The P3VC receives notification triggers from interrupt source controllers (P3SCs) via a Power10 Fabric bus store operation (for example, a cache-inhibited write: ci\_wr). It processes the notification using information contained in the event assignment entry (EAE) that is located in main memory and associated with the specific trigger. This processing can include updating an event queue entry and then forwarding the notification to the P3PC, which signals an exception to one of the processor threads. The P3VC also handles notification redistribution if a state change to the assigned processor thread preclude it from handling the interrupt or notification escalation, if there is no processor thread that is currently capable of handling the interrupt.

The P3PC has an exception bus towards the cores to notify the individual processor threads. Three signals are created for each thread, one to generate hypervisor interrupts, another to generate operating-system interrupts, and a third to generate an event-based branch. Associated with each of the exception-notification signals in the P3PC is prioritization and exception-queuing logic that prevents less favored events from pre-empting more favored ones or from loss due to dropping an event. Associated with each of the exception notification wires is one or more logical server numbers stored in CAM-like lines. This structure is also referred to as the thread interrupt management area (TIMA). These logical server numbers identify which software entities are currently dispatched on the specific physical processor thread. When the P3VC issues fabric bus operations to route an event notification, these CAM-like lines are searched to identify candidate processor threads. In addition to the CAM-like lines, priority and exception-queuing logic mentioned previously, each interrupt-generating exception has logic to track how much interrupt work has been handled by the associated processor thread. This information is used to evenly distribute interrupt processing load among the candidates.

The P3CQ serves as the Power10 Fabric bus interface controller between the interrupt logic and the rest of the Power10 chip. This unit is responsible for sequencing the appropriate fabric bus protocol when the interrupt controller drives or receives commands. It performs compares to determine if the interrupt controller is the destination of a command (for example, a store operation used for an interrupt trigger). It is also responsible for driving the fabric bus histogram, poll, and assign commands to find the correct presentation controller for an interrupt trigger. Another key P3CQ function is sending and receiving the AIB interface to the virtualization and presentation controllers.

Figure 15-1. Interrupt Presentation Interaction



## 15.3 Fabric Bus Interrupt Command

A P3PC receives interrupt fabric bus commands (histogram, poll, and assign) and generates responses based on the contents of the CAM lines and other state information in the unit.

### 15.3.1 Message Send (Msgsend)

The interrupt controller logic supports the internal fabric bus *msgsend* command. The P3CQ snoops the SMP fabric for *msgsend* commands. If it determines an address match, it asserts *lpc\_ack* and passes the command on to the thread context (TCTXT) portion of the P3PC. This logic then decodes the appropriate threads and activates a wire to the appropriate threads of the Power10 chip.



## 16. PCI Express Controller

The PCIe Express controller (PEC) provides a PCIe root-complex port to connect to an adapter, slot, or as a link to a PCIe switch. It acts as a PCIe host bridge (PHB) from the internal, coherent SMP interconnect (also known as the processor bus) to the PCIe I/O. A PEC can support 16 lanes of Gen4 PCIe, which can be configured as 1 - 3 PCIe stacks, or it can support eight lanes of Gen5 PCIe plus eight lanes of Gen4 PCIe, which can be bifurcated.

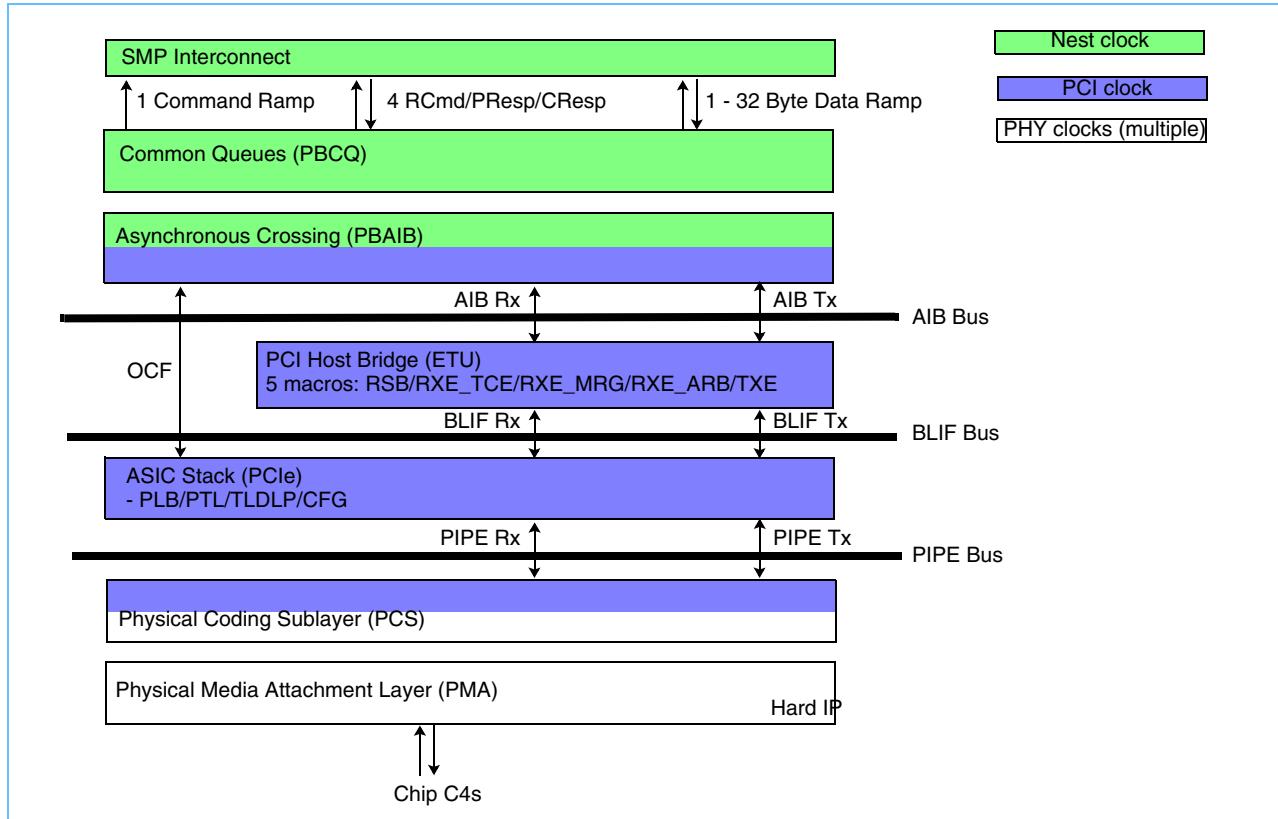
### 16.1 Overview

The PEC is composed of six major building blocks:

- Processor bus common queue (PBCQ)
- Processor bus to AIB interface (PBAIB), which is an asynchronous boundary crossing
- Express transaction unit (ETU)
- PCIe ASIC building block (PCIASIC)
- Physical coding sublayer (PCS)
- Physical media access (PMA)

*Figure 16-1* shows an overview of the major blocks and defined interfaces.

*Figure 16-1. PCIe Major Blocks*





### 16.1.1 Processor Bus Common Queues

The processor bus common queue (PBCQ) logic is responsible for managing the transactions on the coherent processor/cache fabric, the SMP interconnect.

Key features of the PBCQ are as follows:

- Inbound DMA capability
  - Supports 96 DMA read transactions on the SMP interconnect. DMA read transactions are sourced from non-posted read transactions from the PCIe.
  - Supports 48 DMA write transactions on the SMP interconnect. DMA write transactions are sourced from posted write transactions from the PCIe.
  - PCIe 4- and 8-byte atomic operations.
- Outbound MMIO capability
  - Two Base Address Registers (BARs) for external MMIO address ranges
  - 32 MMIO stores
  - 128 MMIO loads
- The ability to share resources with more than one PHB stack

### 16.1.2 Processor Bus AIB Interface

The PBAIB logic provides an asynchronous boundary crossing between the PBCQ and the AIB 2.0 interface.

### 16.1.3 Express Transaction Unit

The ETU is responsible for address translation, interrupt management, and error isolation.

Key features of the ETU are as follows:

- 512 KB (256 KB) partitionable endpoints
- 1 KB (512 KB) 4-way set-associative translation cache
- 4K (2K) MSI interrupts supported
- Eight LSI interrupts supported

### 16.1.4 PCIe ASIC Intellectual Property

The PCIe ASIC building block is composed of the packet buffer layer (PBL), the packet transaction layer (PTL), the transaction and data link layer (TLDLP), and the PCIe Configuration Register core (CFG). These blocks implement the PCIe transaction and data link layers.

### 16.1.5 Physical Coding Sublayer

The PCS manages the low-level networking protocol and signaling between the physical media and the higher-level link protocol layer across the PIPE interface. The 16 lanes of the PCS can be bifurcated into two  $\times 8$  lanes or trifurcated into one  $\times 8$  and two  $\times 4$  lanes.

### 16.1.6 Physical Media Access

The PMA provides the SERDES and analog protocols necessary to connect to the chip C4s. It also provides the PLLs used to drive the PCI clock grid.

## 16.2 Power10 PCIe Configurations

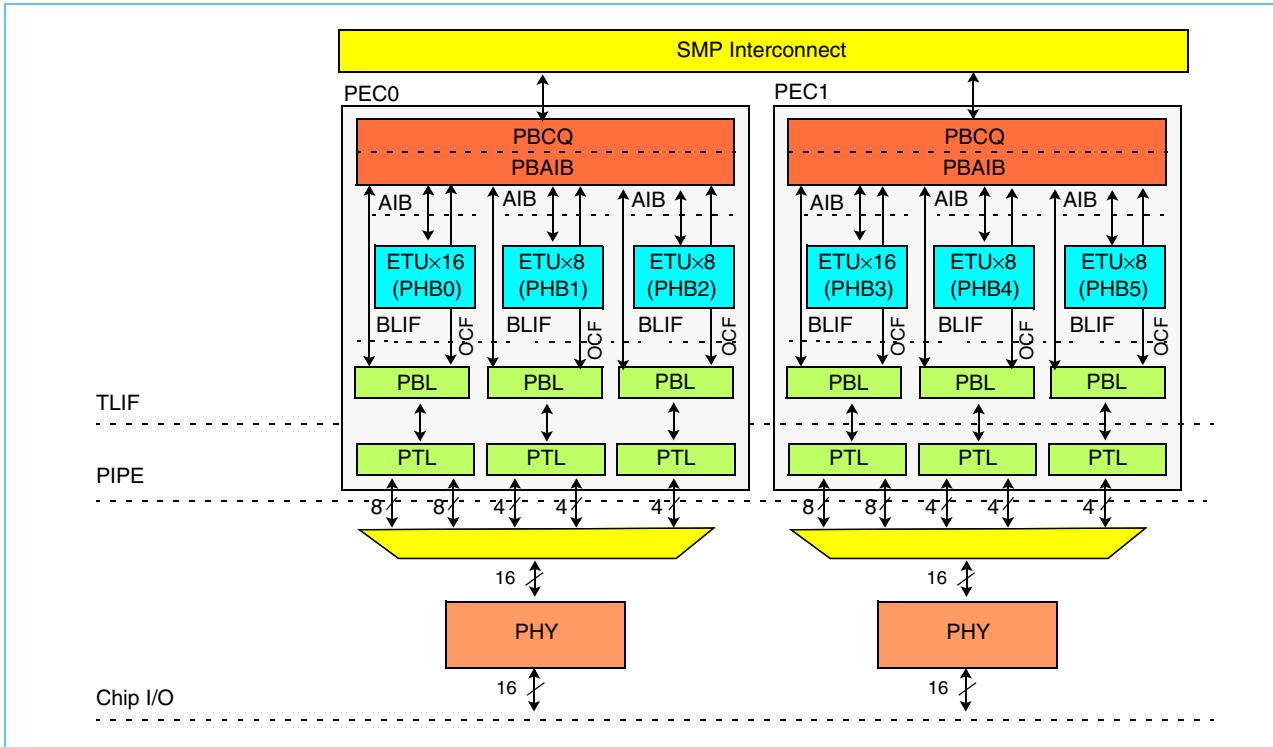
The Power10 chip has two PCIe controllers of 16 lanes each for a total of 32 lanes of PCIe Gen4 I/O. The two PECs can be configured as follows:

- Gen4: One  $\times 16$  lanes, two  $\times 8$  lanes (bifurcation), or one  $\times 8$  lane and two  $\times 4$  lanes (trifurcation)
- Gen5: One  $\times 8$  lanes plus eight lanes of Gen4 PCIe, which operate as a one  $\times 8$  stack or two  $\times 4$  stacks

Each grouping of lanes is called a stack and each stack has dedicated ETU and PCIe blocks. Each set of 16 lanes have only one PBAIB and PBCQ pair to interface to the SMP interconnect. The resources of the PBCQ are shared between the stacks that it services. If not all stacks in a PEC are used, the stacks should be used left to right as shown in *Figure 16-2*. For example, if only one stack will be used in PEC0, PHB0 should be used first.

Within a stack grouping, the lanes can be swapped to facilitate board wiring.

*Figure 16-2. Power10 PCIe High-Level Diagram*



## 16.3 Reliability, Availability, and Serviceability (RAS)

### 16.3.1 Bit-Level RAS

- End-to-end data protection from the processor bus ECC to the PCI packet LCRC/ECRC
- Arrays have SEC/DED ECC
- Register files have parity (some have SEC/DED)
- Support all SMP interconnect parity/ECC
- Major control registers have parity protection

### 16.3.2 Enhanced Error Handling (EEH)

If an error can be isolated to an endpoint, this endpoint is blocked from introducing new transactions until the error can be resolved.

### 16.3.3 Freeze Mode

An error that requires a reset of a stack enters freeze mode. Freeze mode blocks all new transactions to and from a stack. Outstanding operations on the SMP interconnect run to completion, marking data as bad if required. Reset and initialization can be performed on the stack without a checkstop of the chip. A freeze on one stack does not affect the actions of another stack even if they share a PEC.

## 17. Power Management

Like the last several POWER processor generations, the Power10 processor uses a number of traditional dynamic power-savings techniques built into the hardware circuitry. Techniques such as clock gating<sup>1</sup> latches and arrays when they are not required reduce peak power and, therefore, the thermal design point, which is also referred to as the total design power (TDP). The Power10 processor can also dynamically clock gate or power gate<sup>2</sup> individual processor cores and their associated caches when they are not being used.

Additionally, the Power10 processor continues to support “adaptive power management” techniques to manage the average power and to proactively take advantage of variations in workload, environmental conditions, and overall system usage. The EnergyScale™ firmware, coupled with the policy direction from both the customer and feedback from the hypervisor and operating system that is running on the machine, determine the modes of operation and the best power and performance trade-off to implement during runtime to meet customer goals and achieve the best possible performance. The Power10 processor contains an on-chip controller (OCC) to run the EnergyScale firmware, which supports software-requested performance states (Pstates), Idle (Stop) states, chip and system thermal management and protection, and power-supply current over-limit management.

Managing the power and performance trade-off is a complex problem. There are many ways to control the behavior of the hardware, but these also have a number of side effects that vary based on the workload being processed. Because there is no single policy that can be implemented, the Power10 processor, like its predecessors, supports an adaptive approach to the problem in the form of a joint hardware, firmware, and software solution.

### 17.1 Policies and Modes of Operation

EnergyScale removes the “ugly” details of a low-level hardware implementation to provide policies (operational modes) that allow the customer to achieve the required level of power and performance efficiency within specified bounds. Implementations differ depending on which hypervisor is running on the system: Power KVM or PowerVM.

The Power10 chip supports multiple power management choices for system operation, which can be selected by the customer depending on the situation at any given time or for a particular data center’s constraints.

The Power10 power management supports Workload Optimized Frequency (WOF), a mechanism where the chip can be deterministically pushed to the thermal and electrical current limits of the socket to maximize performance. The WOF mechanism, when enabled, takes advantage of clocked-off and powered-off cores and caches to increase the frequency of those remaining. Additionally, the WOF firmware will increase the frequency of a chip whose cores are drawing less power due to other factors such as lower Pstates, lighter workloads, per-core instruction throttling, and dynamic bus width reduction.

---

1. Clock gating involves deactivating clocks for portions of a circuit that are not in use.  
2. Power gating involves turning off the current to portions of a circuit that are not in use.



### 17.1.1 Power Management in Linux-Based Systems (Power KVM)

OpenPOWER systems run a Linux-based hypervisor. The primary interface to EnergyScale in Linux-based systems is from the Linux Governor<sup>1</sup>. The following links describe the available capabilities:

- [\*CPU frequency scaling\*](#)
- [\*The Ondemand Governor\*](#)

### 17.1.2 Power Management in PowerVM-Based Systems

On IBM-branded PowerVM-based systems, the modes and policies provided to the customer are similar to previous Power systems and are described in greater detail in the following white-paper:

- [\*IBM EnergyScale for POWER8 Processor-Based Systems\*](#)

## 17.2 Architected Control Registers

### 17.2.1 Power Management Control Register (PMCR)

The PMCR is the mechanism used by the hypervisor firmware (for example, PowerKVM) to request PState changes. The 8-byte register is implemented as a generic special purpose register per core. The format and content of the register is defined by firmware and is intended to contain Pstate requests along with other performance and power optimization hints, such as workload priority and quality of service expectation, dependent on the version of firmware using it.

**Note:** The layout of this register is unchanged from the POWER9 definition.

Table 17-1 describes the PMCR.

Table 17-1. Power Management Control Register (Version x'1')

Bits	Field	Description
0:7	UpperPS	Global Pstate request (for future enhanced Pstate support). This field is ignored in version x'1' of this register format.
8:15	LowerPS	Pstate request. Where x'00' represents the fastest frequency supported and subsequent increasingly positive values (x'00', x'01', x'02', ...) are decreasing frequency in 16.667 MHz steps from the fastest frequency as the base. In version x'1' of this register format, this field provides both the local and global Pstate request.
16:59	Reserved	Reserved for future enhancements. <b>Note:</b> In version x'1', these bits are ignored and must be all 0's.
60:63	Version	Indicates the format of the fields in this register expected by firmware. x'0' POWER8 format, not supported but treated the same as version x'1' by the Power10 core. x'1' POWER9 format, using only bits 8:15. x'2' Future extensions for enhanced Pstate support.

1. The Governor is the component of Linux software responsible for managing the work allocation, power, and performance of the microprocessor cores.

### 17.2.2 Power Management Idle Control Register (PMICR)

The PMICR (previously defined in the POWER8 chip) is not implemented on the POWER9 or Power10 chip. The function is subsumed by the per-thread PSSCR Register, which is defined in the Stop instruction architecture.

### 17.2.3 Power Management Status Register (PMSR)

The PMSR enables energy management firmware to communicate to the hypervisor running on the Power10 core. The 8-byte register is implemented as a generic special purpose register with only one instance per SMT4-core-resource. The format and content of the register is defined by firmware, intended to communicate metrics for energy efficiency and quality of service, such as the actual global and local Pstate achieved.

**Note:** The format of this register is unchanged from the POWER9 definition, except for formally removing the internal voltage regulation fields that were also unused on POWER9 chip.

*Table 17-2 on page 229 describes the PMSR.*

*Table 17-2. Power Management Status Register (Sheet 1 of 2)*

Bits	Field	Description
0:7	Global Actual Pstate	Global Actual Pstate. Represents the Pstate value that pertains across all cores on the chip and represents the maximum value currently possible. The value x'00' represents the fastest frequency supported and subsequent positive values are decreasing frequency in 16.667 MHz steps from the fastest frequency as a base.
8:15	Local Actual Pstate	Local Actual Pstate. Represents the presently operating Pstate value for this core. This value is less than or equal to the Global Actual Pstate. The value x'00' represents the fastest frequency supported and subsequent positive values are decreasing frequency in 16.667 MHz steps from the fastest frequency as a base. <b>Engineering Note:</b> For the Power10 core, the Local Actual Pstate present applies to all cores on the same chip.
16:23	Pmin	Pstate Minimum. Reads from this field return the presently established minimum Pstate for this core as set by the platform. This value can change autonomously based on the current policy in place and the physical constraints of the platform. The value x'00' represents the fastest frequency supported and subsequent positive values are decreasing frequency in 16.667 MHz steps from the fastest frequency as a base.
24:31	Pmax	Pstate Maximum. Reads from this field return the presently established maximum Pstate for this core as set by the platform. This value can change autonomously based on the current policy in place and the physical constraints of the platform. The value x'00' represents the fastest frequency supported and subsequent positive values are decreasing frequency in 16.667 MHz steps from the fastest frequency as a base.
32	PMCR Disabled	SPR-Based Energy Management Disabled. Reads from this field indicate whether the platform has disabled the PMCR SPR to control the core PState. 0 PMCR enabled. 1 PMCR disabled.



Table 17-2. Power Management Status Register (Sheet 2 of 2)

Bits	Field	Description
33	Safe Mode	Safe Mode. Reads from this field indicate whether the chiplet has been put into a fixed safe mode (frequency and voltage setting), where Pstate requests have been suspended due to errors or for externally forced reasons (such as, firmware updates). 0 Not in safe mode. 1 Safe mode engaged.
34:35	Reserved	Previously used for Internal voltage management status.
36:59	Reserved	Reserved for future enhancements. Set to all 0's.
60:63	Version	Version of PMCR (and PMSR) format supported by the current version of EnergyScale Firmware running on this system.

#### 17.2.4 Power Management Memory Activity Register (PMMAR)

The PMMAR defined in POWER8 is not implemented on the POWER9 chip or the Power10 chip.

## 17.3 Architected Idle Modes (Stop States)

The Stop instruction works in conjunction with the per-thread Power Save Status and Control Register (PSSCR), as described in the *Power ISA (Version 3.1B)*, except for an additional requirement and behavior as indicated in *Section 17.3.8.4 Ultrervisor and Hypervisor StopAPI Requirement* on page 234 and *Section 17.3.8.2 SPR State Loss* on page 233.

With this instruction, the operating system or hypervisor can stop a thread that is not required from occupying the core pipeline. In addition, the hypervisor can disable the thread to free up core resources to the remaining threads, which can cause the core to switch SMT modes.

### 17.3.1 Summary of Stop Level Categories

The ISA makes Stop level definitions implementation dependent. The Power10 chip maintains the same notion of Stop categories as the POWER9 chip, but does not necessarily implement all the same Stop levels. When all threads on a core (in a given core clock region) enter a Stop state, the entire core enters into the Stop state, saving additional power. This enables varying levels of power savings, each with an increasing amount of power saved but higher latency to resume operation. Sixteen Stop levels can be requested, defined as four ranges of state loss each with four levels of “deepness” encoded in the requested Stop level:

- Level 0 - 3: Lowest latency core states, with no or only selectable SMT thread, state loss.  
Can be executed by the operating system or hypervisor with no state loss when PSSCR[ESL] = ‘0’. Only the hypervisor can select SMT mode change with thread state loss via PSSCR[ESL] = ‘1’. These levels wake up to the next instruction when PSSCR[EC] = ‘0’, or to SRESET when executed by the hypervisor based on PSSCR[EC] = ‘1’. Timing facilities are maintained. Operating system requests less than PSSCR[PSLL] do not involve the hypervisor. The hypervisor can choose to convert defined operating system Stop requests into an interrupt back to the hypervisor should that additional latency imposed by the operating system become a problem.
- Level 4 - 7: Higher latency core states, involving core power loss.  
Some hypervisor state is lost, so its execution is only supported by the hypervisor with PSSCR[ESL] = ‘1’. Timing facilities are maintained. Only the state critical to executing code at the SRESET vector is saved and restored. Because PSSCR[EC] must be set to ‘1’, these levels, like legacy stop states, always wake up to an SRESET vector and not the address of the exception that caused the wakeup.
- Level 8 - 11: High latency states also affecting circuits outside the core (for example, in the Quad).  
Also only supported by the hypervisor with PSSCR[ESL] = PSSCR[EC] = ‘1’.  
On the Power10 chip, the timing facilities state is not lost in these levels.
- Level 12 - 15: Highest latency states reserved for future chip or system idle states.  
Stop15, the deepest Stop level, is also used to indicate that a core has not yet been started during system IPL.

For levels 4 and above, an API is available for the hypervisor (or ultrervisor) to request particular register content be restored automatically during the wakeup process by the Stop exit sequencing code, and for a subset of that register content to be automatically saved upon Stop entry for later restoration. For a more detailed description, see *Section 17.3.8 State Loss and Restoration* on page 233.



### 17.3.2 Management of Stop Entry and Exits

Software requests a Stop state per thread. If any thread is still running, the core cannot enter a Stop state. Each SMT4-core-resource only requests a Stop state based on the lowest Stop state of all its threads (for example, if three threads request Stop state 15 and one thread requests Stop state 2, the core enters Stop state 2). When all threads on a SMT4-core-resource enter a Stop state, higher-level actions can be taken to save additional power by clocking or powering off the entire core and possibly their cache regions.

When a thread executes a Stop instruction with PSSCR[ESL] = '1', an SMT thread reconfiguration occurs in the core hardware to remap that thread's resources to the remaining threads. As part of this process, a subset of caches (L1 instruction and data cache, ERATs, context cache, but not the TLB) are invalidated.

Unlike the POWER9 processor, states greater than Stop 5 are not limited by artificially imposed SMT8 core quad-level cache or clocking structures. On the Power10 chip, Stop levels apply to a SMT4-core-resource and its associated caches and are orthogonal to the Stop states of other core resources in the same quad or on the same chip. Stop states are managed on physical clock and power region boundaries. For SMT8 cores, if all 4 odd-numbered threads enter Stop while at least one even-numbered thread is still running (or vice versa), the corresponding odd or even numbered EL region (SMT4-core-resource) comprising the SMT8 core enters a Stop state to save half of that core's power.

### 17.3.3 Unsupported Stop Levels

Not all possible Stop levels are supported by EnergyScale firmware on the Power10 chip. If a nonimplemented Stop level greater than '1' is requested, the Power10 chip enters the next lower supported state as defined in *Section 17.3.1 Summary of Stop Level Categories* on page 231.

### 17.3.4 Auto-promote of Stop Levels

The Power10 chip does not honor the optional auto-promote feature provided in the PSSCR. The requested level field (PSSCR[RL]) in conjunction with PSSCR[PSLL] is always used. The values in the maximum transition level field (PSSCR[MTL]) as well as PSSCR[TR] are ignored in this design.

### 17.3.5 Recommended Stop Code Sequence to Support Lab Debug Tools

When PSSCR[EC] = '0' and MSR[EE] = '0', the thread wakes up and resumes execution at the address of the instruction after the Stop instruction, as per the architecture. In all other cases, wakeup from Stop resumes execution at the appropriate interrupt vector.

However, there are certain lab debug tools that give the ability to stop and start instructions running on selected processor core threads. If this debug activity occurs in a Stop state in the absence of a wakeup condition, the thread incorrectly resumes execution at the address after the stop instruction, because core instruction dispatch was restarted without an interrupt to service the wakeup.

Therefore, for Stop states where PSSCR[EC] = '1', the hypervisor should put a branch minus x'4' immediately after the Stop instruction to support lab debug tooling. Similarly, the privileged user should also put a branch minus x'4' after the Stop instruction when PSSCR[EC] = '0' and MSE[EE] = '1'. Therefore, when running lab debug tools that start and stop a processor thread in a Stop state, it instead branches back to the Stop instruction and re-enters the Stop state. Otherwise, the thread accidentally starts executing code after the Stop, which is not what happens in a nondebug mode.

### 17.3.6 Power-Savings Level Status

As per the ISA, PSSCR[PLS] indicates the deepest core Stop level that thread has been in since executing the Stop instruction. The core can potentially enter and exit Stop many times, in a variety of Stop levels, while one or more threads remain in a Stop state. Therefore, on wakeup, all threads might not contain the same value in PSSCR[PLS].

Stop15 is indicated in PSSCR[PLS] for a core that has not yet entered a Stop state since System IPL. The QME is responsible for setting the PLS for both regular and special wakeup events for Stop greater than '1'.

### 17.3.7 Regular Wake Up Events

When EC = '0', any interrupt will cause an exit from a Stop state. When EC = '1', only interrupts enabled by PECE fields in the LPCR (and for ultravisor enabled interrupts, by SMFCTRL fields) cause a Stop exit.

When PSSCR[ESL] = '1', any hypervisor (or ultravisor) state lost required for the core to resume execution is first restored by the QME, by performing SCOMs and running a self-restore routine on the core itself, before an SRESET is seen by the hypervisor.

### 17.3.8 State Loss and Restoration

Only the hypervisor can initiate state loss with PSSCR[ESL] = '1'. The hypervisor is responsible to save any non-hypervisor thread context (for example, GPRs, VSRs, FPRs) it wishes to not be lost upon execution of the Stop instruction and then restore those values after wakeup. Additional state loss behavior is described in the following sections.

#### 17.3.8.1 Timing Facilities

Unlike the POWER9 chip, all Stop states on the Power10 chip maintain the state and synchronization of the timing facilities (TB, VTB, DEC, HDEC, PURR, SPURR, and TFMR). This is accomplished by shadowing these facilities outside of the core and cache region into the always running region of the Quad superchiplet during Stop entry, and restoring them back into the core upon Stop wakeup, all while maintaining synchronization with the chip Time of Day (TOD) mechanism.

#### 17.3.8.2 SPR State Loss

The only SPR state that might be lost for Stop levels less than 4 when PSSCR[ESL] = '1' are the following:

- CR, LR, CTR, FPSCR, VSCR, XER, TAR, SPRG2, and HSPRG1

Note that although some minimal hypervisor state (HSPRG1) might be lost, SRR1(46:47) is reported as '10'. This does not technically follow the description in the *Power ISA (Version 3.1B)*, which states that any hypervisor state loss should be reported as '11'. This is a carryover from the POWER9 chip, where this indication was required to distinguish Stop levels for which the timebase was lost.

State loss incurred for Stop levels greater than or equal to 4 must be managed as described in *Section 17.3.8.4 Ultravisor and Hypervisor StopAPI Requirement* on page 234, because power is turned off to the core.



#### 17.3.8.3 SPR Loss Differences From the POWER9 Chip

Although the list in the previous section is complete, a description of the differences in the state lost on the Power10 chip compared to the previous generation might prove useful.

As mentioned previously, the timing facility SPRs that were lost for Stop levels greater than 5 on the POWER9 chip are now preserved for all Stop levels.

The following registers, that were preserved for Stop levels less than 11 on the POWER9 chip, are lost for Stop levels greater than 3:

- DPDES, SPRC, SPRD, HMER, HMEER, PSSCR, PMCR, L2QOSR, RWMR, CTRL

The following registers, that were preserved on the POWER9 chip for Stop levels less than 4, are now lost:

- HPRG1, TAR, SPRG2

The following registers, that were lost on POWER9, are now preserved for Stop levels less than 4:

- DSCR, AMR, IAMR, UAMOR, AMOR, DAWR, DAWRX

#### 17.3.8.4 Ultravisor and Hypervisor StopAPI Requirement

The ultrvisor, if enabled for a secure system, must call the StopAPI during system IPL to save the state of all ultrvisor-protected registers that must be restored upon subsequent Stop wakeup from powered-off states. Before executing the Stop instruction for levels greater than 3, the hypervisor is responsible for calling a Stop-API supported by the power management firmware for saving a subset of hypervisor SPR values necessary to properly execute the SRESET interrupt vector immediately upon wakeup. Although not specified in the POWER ISA v3.0, this requirement simplifies the hardware implementation and reduces entry and exit latency. These values are restored during wakeup by the power management infrastructure before the SRESET interrupt is taken by the hypervisor, appearing as if they were not lost. These registers typically include SPRs important to the initial processor context such as the HRMOR, LPCR, and HSPRG0. The PSSCR and LPCR are required to be saved by the hypervisor via the StopAPI for Stop level greater than 3; otherwise, after a special wakeup, there are scenarios that:

- An unintended Stop11 might result while in the Stop state because PSSCR scan flushes to Stop level 15 by default and
- The wakeup conditions might no longer be honored.

Although not necessarily required for the hypervisor to resume operation upon an SRESET, other registers are also supported by this mechanism to expedite the Stop wakeup process. These registers typically include, but are not limited to, registers such as the HMEER, LDBAR, HID, MSR, and DAWR. SPRs handled by the StopAPI that are lost due to SMT thread reconfiguration are only restored for Stop greater than 3 (that is, calling the StopAPI to save any SPR does not restore it for core Stop levels less than or equal to 3).

System and host firmware is required to call a SCOM Restore API for any SCOM registers changed during runtime that must be restored on core or cache power-on during Stop wakeup.

As per the Power ISA, upon wakeup from a Stop state, SRR(46:47) on every thread indicates the amount of architected state lost and is summarized in *Table 17-3* on page 235.

### 17.3.9 Summary of Power10 Supported Stop Levels

*Table 17-3. Supported STOP Instruction Behavior*

STOP Level >	STOP0	STOP2	STOP3	STOP5 <sup>(1)</sup>	STOP11 <sup>(2)</sup>
Core	Dispatch halted	Clock Grid Off	$V_{DD} = V_{MIN}$	$V_{DD}$ and $V_{CS}$ Power Off	
L2 Cache	–	–	–	–	–
L3 Cache	–	–	–	–	Power Off
SRR1[46:47]	'01' if PSSCR[ESL] == '0' else '10'			'10'	

1. Stop5 is not supported by hypervisor code and firmware as per plan of record.  
 2. Stop11 is limited such that the hypervisor can request at most one entry or exit per Quad at a given time.

Any Stop levels not listed in *Table 17-3* are rounded down to the next smallest numbered level (for example, a Stop4 request is rounded down to Stop3). When a thread executes any Stop state, it immediately stops fetching or dispatching any new instructions. A core only enters a Stop state when all of its threads enter Stop.

The list of supported core Stop states with a high-level description of each follows:

- Stop0 on all threads effectively quiesces the core pipelines and enables all dynamic clock gating in the core to engage. The L2 cache remains coherent. Less than 1% of the latches in the core receive clocks in this state, and those latches experience very little data switching. Note that if PSSCR[ESL] = '1', dropping to lower SMT modes might invalidate a subset of the L1 caches (L1 instruction and data caches, ERATs, context cache, but not TLB).
- Stop1 is no longer supported as a unique state, because the Power10 core is already heavily clock gated when quiesced. Requests for this level also invalidate all L1 caches (instruction and data caches, ERATs, context cache, and TLB) and incur additional entry and exit latency, but enjoy no additional reduction in power. This level can also be indicated in PSSCR[PLS] if a wakeup event causes an abort when the core is in the process of entering a deeper Stop level.
- Stop2 stops clocks and turns off the clock grid to the entire core and its L2 cache. L1 caches (instruction and data, ERATs, context cache, and TLB) are invalidated and the L2 cache state is flushed to preserve memory coherency. Increased latency is predominately due to the L2 cache purge operation.
- Stop3 additionally lowers the  $V_{DD}$  voltage of the core and its L2 cache to  $V_{MIN}$  to save leakage power, while leaving  $V_{CS}$  at full rail. (Note that the small amount of power saved by dropping the  $V_{CS}$  voltage does not justify the complexity or cost of implementation.)
- Stop4 is not supported.
- Stop5 is no longer supported as plan of record, but powers off both  $V_{DD}$  and  $V_{CS}$  to the core and its L2 cache, requiring electrical fencing plus scan, SPR, and SCOM re-initialization on wakeup.
- Stop6 - 10 is not supported. The relatively small amount of power saved (by stopping clocks to the L3 cache or dropping its  $V_{DD}$  or  $V_{CS}$  to  $V_{MIN}$ ) in any of these states is far outweighed by the latency incurred by flushing the L3 cache and fencing it from the SMP fabric memory coherence protocol.
- Stop11 additionally powers off the  $V_{DD}$  and  $V_{CS}$  to the L3 cache, requiring electrical fencing plus additional scan and SCOM re-initialization on wakeup. Unlike previous POWER generations, Stop11 usage is limited to only special firmware operations such as unlicensed cores, concurrent array repair, or concurrent firmware update, with the requirement that the hypervisor might request at most one Stop11 entry or exit per Quad at a given time.
- Stop12 - 14 are not supported.



- Stop15 is used to denote deconfigured or “bad” cores, as well as “good” cores that have not yet been started during IPL.

### 17.3.10 Latency and Power Savings in each Stop Level

*Table 17-4* estimates the latency of each Stop state and the relative amount of per-core chip power saved by that state compared to the execution of a low-priority idle loop. Actual power savings varies based on the chip sort-point per system type and  $V_{DD}$  voltage set-point based on the workload on the remaining cores. The latency and power numbers are estimated pre-silicon projections for the Power10 chip and are based on hardware measurements for a POWER9 chip running at nominal in the lab.

*Table 17-4. Projected Stop State Latency and Power Savings (with POWER9 Comparison)*

Power Savings Technique	Idle State Latency (% idle chip power at $V_{MIN} \rightarrow V_{MAX}$ )	
	Power10 Chip	POWER9 Chip
Stop Dispatch	Stop0 Projected: 0.1 $\mu$ s (83 → 63%)	Stop0 0.1 $\mu$ s (79%)
Core Execution Units Clock Off	–	Stop1 2 $\mu$ s (74%)
Core Clock Grid Off	Stop2 Projected: 20 $\mu$ s (74 → 54%)	Stop2 20 $\mu$ s (38%)
Core $V_{MIN}$ (Power10: plus MMA power off)	Stop3 Projected: 30 $\mu$ s (74 → 39%)	–
Core Power Off (Power10: L3 cache active, no rVRM)	Stop5 – not supported Projected: 300 $\mu$ s (55 → 31%)	Stop5 250 $\mu$ s (26%)
Core Plus L3 Power Off	Stop11 – limited use Projected: 3000 $\mu$ s (51 → 25%)	Stop11 6000 $\mu$ s (22%)

## 18. Security

Secure boot and security capabilities in the Power10 processor include those security enhancements made in legacy designs, as well as several additional features. The Power10 design includes support for a secure and trusted boot of the hypervisor through a sequence of verification operations that extend the initial trust in the hardware and secure boot code in SEEPROM through the firmware components and finally to the hypervisor itself.

The security design for the Power10 chip also includes the following security features:

- A secure firmware stack that includes the PowerVM hypervisor.
- A secure memory facility (SMF), which provides a trusted execution environment at the VM level for untrusted hypervisor environments such as KVM.
- Hardware features to protect the processor state and customer data from unauthorized access after the system is up and running and during debug data collection.
- Secure and trusted boot with supply-chain threat detection enhancement
- Secure access protection, which is a register filtering mechanism to control access to sensitive processor states
- Secure channel in a multi-node system
- Dynamic root of trust for measurement (DRTM) support
- Secure debug which provides the capability to securely dump the processor state in the event of a debug requirement
- Side channel attack prevention
- Pervasive memory encryption

The Power10 security architecture only provides partial protection in the following aspects of security:

- Physical attacks at data centers and customer locations: Pervasive memory encryption provides a line of defense against a class of physical attacks that attempt to obtain any secret data by reading main memory (DRAM) content or snooping data traffic such as cold boot attacks or memory bus snooping attacks. However, physical access may still permit various other class of attacks; for example, attacks that target data integrity in main memory, such as replay attacks.
- Denial of service: The virtualization architecture in the Power10 chip provides a limited degree of support for trusted hypervisor and firmware stack; to provide partial protection against a rogue logical partition (LPAR) causing denial-of-service to all co-resident LPARs in the same system.



## 18.1 Secure Memory Facility

In the virtualized cloud computing model, client provided virtual machines (VMs) or logical partitions are often hosted by third-party open source hypervisors; such as XEN or KVM. Because a hypervisor can typically access any memory location in the host real address space, as well as observe any residual register state during a hypervisor call or hypervisor interrupt, all client data is effectively available to the hypervisor. Hypervisors are complex with many million lines of code and many security exploits have been found in hypervisors over the years. For example, security exploits such as user code running under a VM can break out into a hypervisor via privilege escalation attacks. Thus, an attacker can compromise all client data from all VMs running in a server once they compromise the hypervisor from a VM. Moreover, if the cloud provider is complicit, the hypervisor itself can be actively malicious and intentionally compromise client data. Similarly, client provided containers run atop an untrusted computing stack of operating system and possibly a hypervisor and face similar security threats.

The overall goal of the Secure Memory Facility (SMF) is to provide register and memory isolation of a client compute stack (such as, a secure VM) from the rest of the untrusted system software stack (such as hypervisor or other untrusted VMs executing on the same machine). The processor architecture provides this facility via implementing a privilege state bit defined as the Secure [S] bit in the Machine State Register. This new state bit is used in conjunction with existing privilege level state bits to implement a higher privilege state known as an “ultravisor” state defined as S = ‘1’, HV = ‘1’ and PR = ‘0’. This new state supersedes the hypervisor state (S = ‘0’, HV = ‘1’, PR = ‘0’) in both system privilege and trust. Any software executing in this privilege state is known as the ultravisor and its correctness is critical to fulfill the goals of the SMF function, such as register isolation. In addition, the host real address space is divided up into secure and nonsecure regions demarcated by a designated host real address bit, thus providing memory isolation.

Any developer attempting to leverage the Secure Memory Facility (SMF) by designing software executing in an ultravisor state must only do so on IBM-approved firmware that supports this feature.

### 18.1.1 Protected Execution Facility

Protected execution facility (PEF), which leverages the Secure Memory Facility (SMF) present in hardware, provides protection against the types of attack from the hypervisor on to the virtual machines by creating the abstraction of a protected partition, known as a secure virtual machine (abbreviated as Secure VM or SVM) against which the previously mentioned attacks will fail to extract any information from. To achieve this, a new privilege layer (the previously described ultravisor) intercepts any call or interrupt into the hypervisor enabling it to clean up the VM’s register state and provide register isolation before handing off control to the hypervisor. The ultravisor also establishes and manages memory regions (known as secure memory regions) that are exclusively accessible to trusted parts of the client computing stack (the applications and operating system belonging to a secure VM) and the ultravisor. This achieves memory isolation of trusted software components. The ultravisor system software component is considered trusted, because this solution provides isolation at the granularity of an entire VM, which is also referred to as coarse-grain SMF in some literature.

The security claims of PEF are upheld by combined contribution of the following:

- Coarse-grain SMF facility in hardware.
- Ultravisor software guarantees consisting primarily of translation in the memory management unit (MMU).
- Interrupt privilege state transitions in hardware.
- Certain code procedural sequences to conceal state information of secure virtual machines.

- Checks implemented in the ultrvisor software as secondary mechanisms. Certain modifications to hypervisor software are required for useful operation, but the correctness of those modifications is not a factor for upholding security properties.

### **18.1.2 Deviations from the SMF Architecture Specification in the Power10 Implementation**

Certain architectural features mentioned in the POWER ISA are not supported or have restricted use in the Power10 implementation of the secure memory facility. In some cases, they deviate from the specification found in the *Power ISA (Version 3.1B)*. This section lists such unsupported and restricted use features, as well as any notable deviations. Note that these deviations have no impact on data integrity guarantees of secure virtual machines provided by the PEF. The impact of these deviations only results in restricted software implementation choices or longer code sequences in the software stack.

#### **18.1.2.1 Implementation Restriction: Only URMOR[12:42] Bits are Implemented**

The Power10 processor core implements URMOR bits [12:42] as described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B), Section 3.2 Ultrvisor Real Mode Offset Register (URMOR)*. All other bits are reserved and return zero when read. For additional details, see *Section 5.9.3 Ultrvisor Real Mode Addressing Using URMOR* on page 122.

#### **18.1.2.2 Implementation Restriction: the UILE Bit is Not Implemented and it is a Constant Zero, Ultrvisor Must Execute in Big-Endian Mode**

The implementation-specific UILE bit defined in the *Power ISA Operating Environment Architecture - Book III (version 3.1B) Section 3.3 "Ultrvisor Interrupt Little Endian (UILE) Bit"* is not implemented in any special purpose register in the Power10 processor family. The Power10 processor assumes a fixed value of zero. Hence, MSR[LE] is set to '0' whenever an interrupt that results in an ultrvisor state occurs. In the Power10 processor family, the ultrvisor software must also ensure that the MSR[LE] is set to '0' when operating in the ultrvisor state.

#### **18.1.2.3 Implementation Restriction: SMFCTRL[E] Bit is Not Implemented per Thread Only per Core**

The SMF Enable [E] bit described in the *Power ISA Operating Environment Architecture - Book III (version 3.1B) Section 3.4 "Secure Memory Control Facility Register (SMFCTRL)"* is not implemented per physical thread of a core, but rather per core. Hence, any modification of the bit from one thread will have its effect on all threads of the core.

#### **18.1.2.4 Implementation Restriction: SMFCTRL[62:63] Bits are Not Implemented**

The implementation-specific bits [62:63] of the ultrvisor privileged special purpose register SMFCTRL described in *Power ISA Operating Environment Architecture - Book III (version 3.1B) Section 3.4 "Secure Memory Facility Control Register (SMFCTRL)"* are not implemented, and hence, do not have any function.

### **18.1.3 Secure Memory Bit in System Memory Map**

In the Power10 processor family, host real address bit 12 is designated to differentiate secure versus non-secure real address access. Bit 12 in the system (host) real address is set to '1' to indicate an attempt to access secure memory when the SMF function is enabled.



*Table 18-1. System Memory Map for 56-Bit System Address (8:63)*

Bit 8:11	Bit 12	Bit 13:14	Bit 15:19	Bit 20:63
Reserved	Secure bit	Memory select	Topology ID translation table index	Chip address range

#### 18.1.4 Mandatory Software Procedures Followed by Ultravisor for Maintaining an SVM for Coarse-Grain SMF

Because the security guarantees of the first generation PEF are upheld by a combination of ultravisor software and SMF hardware extensions, it is imperative for the ultravisor to follow certain guidelines and implement certain checks. The ultravisor software layer can implement additional functions and checks but the listed items are a bare minimum necessity.

Although the architecture and implementation does not limit SMF to radix translation only, it is expected that the SMF will only be used in radix translation mode in IBM products. Hence, the following discussions only provide examples relevant to radix translation mode.

##### 18.1.4.1 Essential Elements of Code Sequence to Convert a Non-secure Virtual Machine into a Secure Virtual Machine

The actual code sequence to launch a secure VM by converting a non-secure VM might have many additional steps and actions in an actual implementation. However, the following steps must be present in some form.

- Supervisor software from a non-secure virtual machine invokes the ultravisor via an ultravisor system call (**sc 2**) with an indication of converting the non-secure VM into a secure VM.
- Ultravisor code prepares a VM to transform into a secure VM (which involves moving or copying all or some of its pages into secure memory and creating duplicates of all the necessary page tables in secure memory).
- Then, the ultravisor sets up USRR0/1 to the entry point of the secure VM: USRR0 = “EA of VM entry point”, USRR1(S, HV, PR) = ‘100’. and performs a **urfid** instruction.

*Table 18-2 on page 240 summarizes these essential code sequence elements to launch an SVM.*

*Table 18-2. Essential Elements of Code Sequence to Launch a Secure Virtual Machine*

VM requesting conversion to secure VM makes an ultracall ( <b>sc 2</b> ) into the ultravisor
After performing necessary validation on this ultracall, the ultravisor marks the partition table entry of this partition as secure and begins editing the entire translation structure of this partition
The ultravisor moves the data pages and page tables of the non-secure VM into secure memory, this involves moving all the host real pages into secure memory and also moving all guest and host page tables associated with those translation into secure memory, as well as editing all necessary table entries to point to the new secure memory addresses
Set USRR0 = <target instruction address in new secure VM>
Set USRR1(S, HV, PR) = ‘100’
<b>urfid</b>

#### 18.1.4.2 Ensure Isolation of Register State of a Secure VM from Hypervisor

This section describes the procedure used to ensure the isolation of the register state of a secure VM from the hypervisor (via intercepting guest SVM hypervisor privileged interrupts by the ultrvisor).

All hypervisor interrupts (hypervisor decrementer interrupt [HDEC], hypervisor instruction exception [HISI], hypervisor data exception [HDSI], hypervisor emulation interrupt, machine check, hypervisor mediated external) or software-initiated system call to the hypervisor (**sc 1**) that cause hypervisor interrupts are intercepted by the ultrvisor when SMF is enabled and are routed to an effective address of the associated interrupt + URMOR offset in ultrvisor real mode. Note that on such an interception, MSR[LE] is always set to zero, which implies all ultrvisor code must be big endian.

Upon invocation of the ultrvisor execution context via the previous mechanism, the following steps must be performed by the ultrvisor to ensure that no residual register state from the secure VM is left over for the hypervisor to access.

1. Save all context-specific registers that are considered sensitive data and should not be read by an untrusted hypervisor in secure memory. This includes general purpose registers (GPR); vector-scalar registers (VSR); floating-point registers (FPR); and any sensitive SPRs such as decrementer (DEC), and depending on the interrupt type, SRR0/SRR1 or HSRR0/HSRR1. The register contents must be saved to secure memory (RA[12] = '1' space).
2. Populate the registers saved in step 1 with random garbage values except for (H)SRR0/(H)SRR1, because these two registers are used to convey information regarding the interrupt to the hypervisor.
3. Encrypt the pages possibly required by the hypervisor and store the encrypted data in the normal memory area (RA[12] = '0' space).
4. Update (H)SRR1[S] = '1', before the interrupt is reflected into the hypervisor as if it came from the VM directly. The S bit in HSRR1 is used to indicate to the hypervisor that this interrupt originated from a secure VM as opposed to it originating from a non-secure VM.
5. Set up USRR0/1 to reflect the interrupt to the hypervisor: USRR0 = "EA of interrupt", USRR1(S, HV, PR) = "010".
6. The ultrvisor performs a **urfid** instruction to start execution in the HV state at the USRR0 address + HRMOR value.
7. The hypervisor handles the interrupt.
8. At the exit of the hypervisor handler tests (H)SRR1[S] bit:
  - If (H)SRR1[S] = '1', perform an **sc 2** (ultrvisor call) to return to ultrvisor, because the original interrupt came from a secure VM.
  - If (H)SRR1[S] = '0', perform a **(h)rfid** instruction to directly return to the non-secure VM
  - Note that if a malicious hypervisor does not implement this step properly, it creates a disruption of execution but does not violate the memory and register state isolation guarantees.
9. Ultrvisor restores the saved registers in step 1.
10. Check that (H)SRR0 or (H)SRR1 are still the same as when the interrupt came to the UV (to prevent any misdirection attack from the hypervisor via returning to a different address in the secure VM).
11. Perform a **(h)rfid** in the ultrvisor to return to the guest secure VM.



#### **18.1.4.3 Ensure Secure VM Translations for Secure Pages are Immutable by Hypervisor**

To ensure that translations to secure memory pages can only be altered by the ultravisor, the following structures must be maintained within secure memory so that the hypervisor is required to make ultravisor system calls (**ucall**) to perform any changes. Hence, ultravisor software can perform the necessary checks. Hardware ensures that the root register for locating the partition table, namely the Partition Table Control Register (PTCR) is only ultravisor writable. The rest of the conditions must be enforced by the ultravisor software.

1. The partition table is maintained by the ultravisor and must be placed in secure memory.
2. Partition scoped page tables (used during guest real address translation for a guest OS) for a secure partition/SVM must be kept in secure memory.
3. Process-scoped page tables (used during translation from a guest application as well as when a guest OS runs with translation enabled) must be kept in secure memory.

Alongside these restrictions, the ultravisor also must ensure that there is no function in the ultravisor that can be called by the hypervisor to change the PTCR, and all ultravisor calls have appropriate validation.

#### **18.1.4.4 Ensure Secure Memory Region Separation between Different Secure VMs**

The ultravisor must ensure that during creation of a secure VM, all the host real pages that are mapped to the secure VM belong within the bounds of the allocated area for that particular logical partition within the secure memory region. This is analogous to how hypervisors maintain partition separation.

In addition to the original allocation during creation of a secure VM, the ultravisor must act on behalf of the hypervisor whenever a hypervisor fault (HISI, HDSI) occurs on the secure memory pages belonging to an SVM, because the host page tables for secure VMs are located in secure memory. The hypervisor can perform page table allocation and all policy decisions but must use an ultravisor system call (**sc 2**) to ultimately update the page table entries in secure memory.

The ultravisor must check the following when such a page installation is taking place.

- The LPID for which the translation is being installed is indeed a secure partition as per the partition table.

The secure host real address (RA[12] = '1') being installed in a leaf page-table entry must be checked against the data structures in the ultravisor that contain the host real-size bounds of each secure partition to ensure that one secure VM is not overlapping with another secure VM.

### 18.1.5 Code Sequence to Change Value of URMOR Register

Table 18-3 shows a code sequence that the ultrvisor privileged software can use to update the URMOR value in a core with multiple hardware threads and also potentially across the multiple cores of a system. The thread changing the value of URMOR is considered to be the primary thread while all others are considered to be secondary threads.

Table 18-3. Code Sequence to Set Value of URMOR Register

Primary	Secondary
Thread sync up point 1	Thread sync up point 1
Jump to instruction located at address with EA[0] = '1'	Jump to instruction located at address with EA[0] = '1'
Thread sync up point 2	Thread sync up point 2
Change URMOR (via <b>mtspr</b> on same core, scom write to other cores)	
Thread sync up point 3	Thread sync up point 3
<b>isync</b>	<b>isync</b>
<b>sibia IH = x'7'</b>	<b>sibia IH = x'7'</b>
<b>isync</b>	<b>isync</b>
Thread sync up point 4	Thread sync up point 4
Jump to instruction EA[0] = '0', new URMOR value will take effect now	Jump to instruction EA[0] = '0', new URMOR value will take effect now

### 18.1.6 Additional Restrictions

Certain instructions, such as **mtmsrd**, **rfid**, or **hrfid**, can enable entering the state MSR[S,HV,PR] = '111'. However, the processor behavior is not defined in this state and hence, ultrvisor software should avoid entering this state.



## 18.2 Dynamic Execution Control

There are several aspects of dynamic execution control (micro-architectural, as well as architectural) in the processor that can either favor security or favor performance. This feature enables user software to have a degree of freedom in choosing these aspect control values to allow for performance or security instead of “always on” enforcement. There are special purpose instructions and registers to control aspects of execution in both architecture and micro-architecture in a dynamic fashion to enable these performance versus security trade-offs.

### 18.2.1 Dynamic Execution Control Instructions

Certain special instructions exist in the Power10 implementation for special micro-architectural behavior control related to security. These instructions are listed here:

- **bcctr 2,0,0** (machine code 0x4C400420 in big-endian format) is an invalid form of a **bcctr** instruction as per the architecture (because BO[2] = ‘1’, decrement and test CTR is invalid), which in addition to the functional behavior described in *Section 5.1.2.5 Branch Processor Instructions with Undefined Results* on page 75, performs a flush of count cache and hardware link stack structure as a microarchitectural side effect. System software must place this instruction between context switches (such as an application to the operating system (OS) kernel and back to a different application, OS kernel to hypervisor kernel and back to a different OS kernel) in system software to flush these microarchitectural structures to prevent Spectre variant 2 and Spectre return stack buffer (Spectre-RSB) attacks across two different contexts in the same privilege state. Application software should also execute a system call where the privileged state software must execute this instruction before returning to the application code, when switching from a trusted part of application code to an untrusted part of the same in application sandbox environments.
- This method provides software-enforced mitigation of Spectre variant 2 and Spectre-return stack buffer attacks across the same address space within the same hardware thread between a trusted and untrusted region of code.
- **ori r31,r31,0** (machine code 0x63FF0000 in big-endian format) is a NOP as per architectural behavior with the following micro-architectural side effect of execution serialization. This instruction will serialize dispatch in the out-of-order processor pipeline. The primary function of this instruction is to prevent speculative execution past this instruction, until instructions prior to this have committed.

### 18.2.2 Dynamic Execution Control Registers

Section 9.3 of the *Power ISA Operating Environment Architecture - Book III (version 3.1B)* provides a detailed description of three special purpose registers: the Dynamic Execution Control Register (DEXCR), the Hypervisor Dynamic Execution Control Register (HDEXCR), and the implementation-dependent Ultravisor Dynamic Execution Control Register (UDEXCR). These special purpose registers provide dynamic means of software control on several execution aspects such as branch prediction behavior and execution behavior of certain instructions, as well as enablement for Return Oriented Programming mitigation. Note that the Power10 processor does not implement the UDEXCR register.

## 19. Performance Profile

This chapter supplements the Power10 design descriptions and architectural features supported with additional specifications relevant for software optimization.

### 19.1 Core

#### 19.1.1 Microarchitecture and Pipeline Overview

See *Section 3 Power10 Processor Core* on page 39 for an overview of the Power10 core features and micro-architecture.

Additionally, the Power10 core is supported by the following facilities:

- A connected cache subsystem providing a dedicated L2 and L3 cache region per core, as well as shared and local-castout (LCO) utilization of on-chip caches; providing up to 120 MB of on-chip L3 cache.
- Support for accessing the on-chip accelerator subsystem (NX) via the cut and paste architecture and for accessing an on-chip random number generator.
- Simultaneous multi-threading (SMT) that operates in one of three modes: ST (single-thread), SMT2 (up to two threads), and SMT4 (up to four threads) for each SMT4-core-resource. See *Section 4 Thread and LPAR Management* on page 55 for more details on the core SMT modes and core variants.

The following added stages for a value-add over the POWER9 core offset with enhanced IPC and reduced execution latency are:

- Add a stage to decode, enables prefix/fusion.
- Add three stages to the issue/register file pipeline:
  - Focus on efficiency; move to efficient register file; sliced target file.
  - Enable wide pipes (128 bits  $\times$  4 SIMD).
- Reduced execution latency:
  - 1- and 0-cycle fixed latency via fusion (minus 1, 2).
  - 5-cycle float-to-float (minus 1 average).

*Figure 19-1* on page 246 shows a high-level pipeline diagram of the POWER9 core versus the Power10 core.

Figure 19-1. High-Level Pipeline Diagram (POWER9 Core versus Power10 Core)

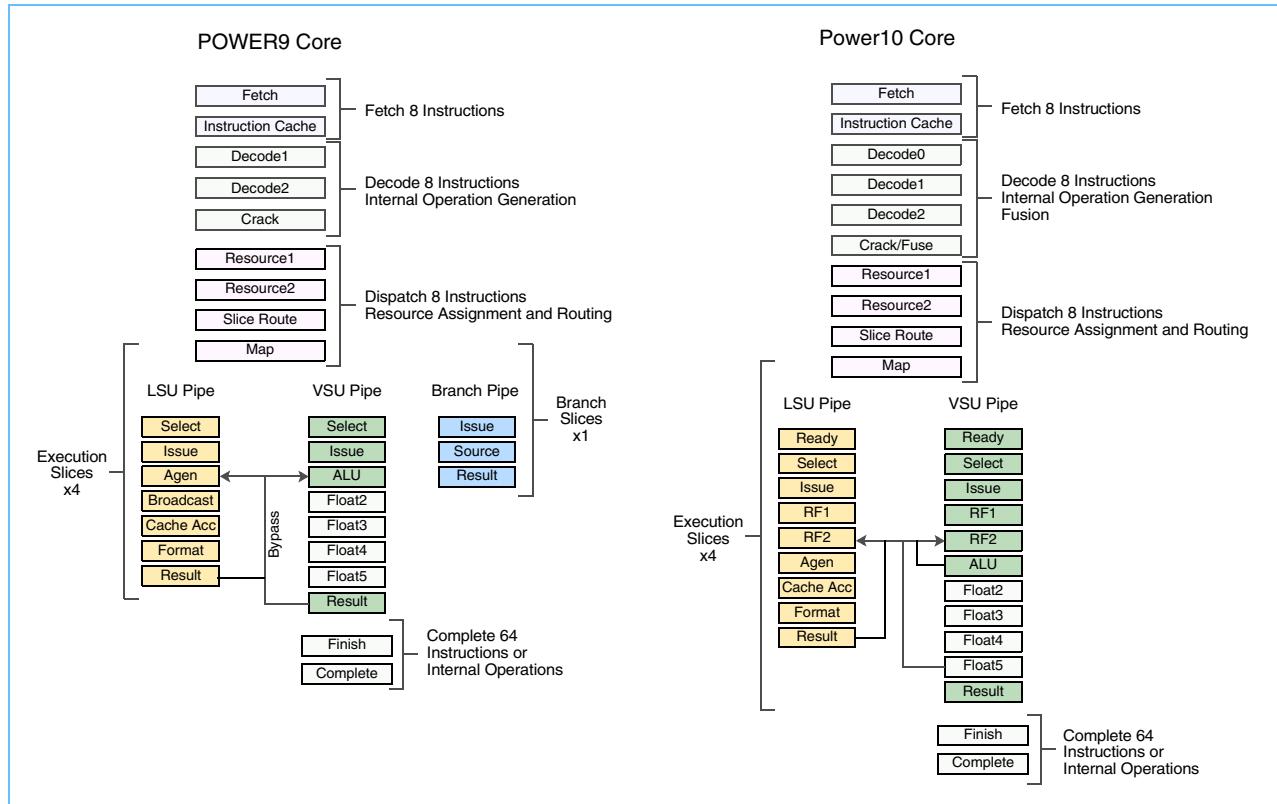
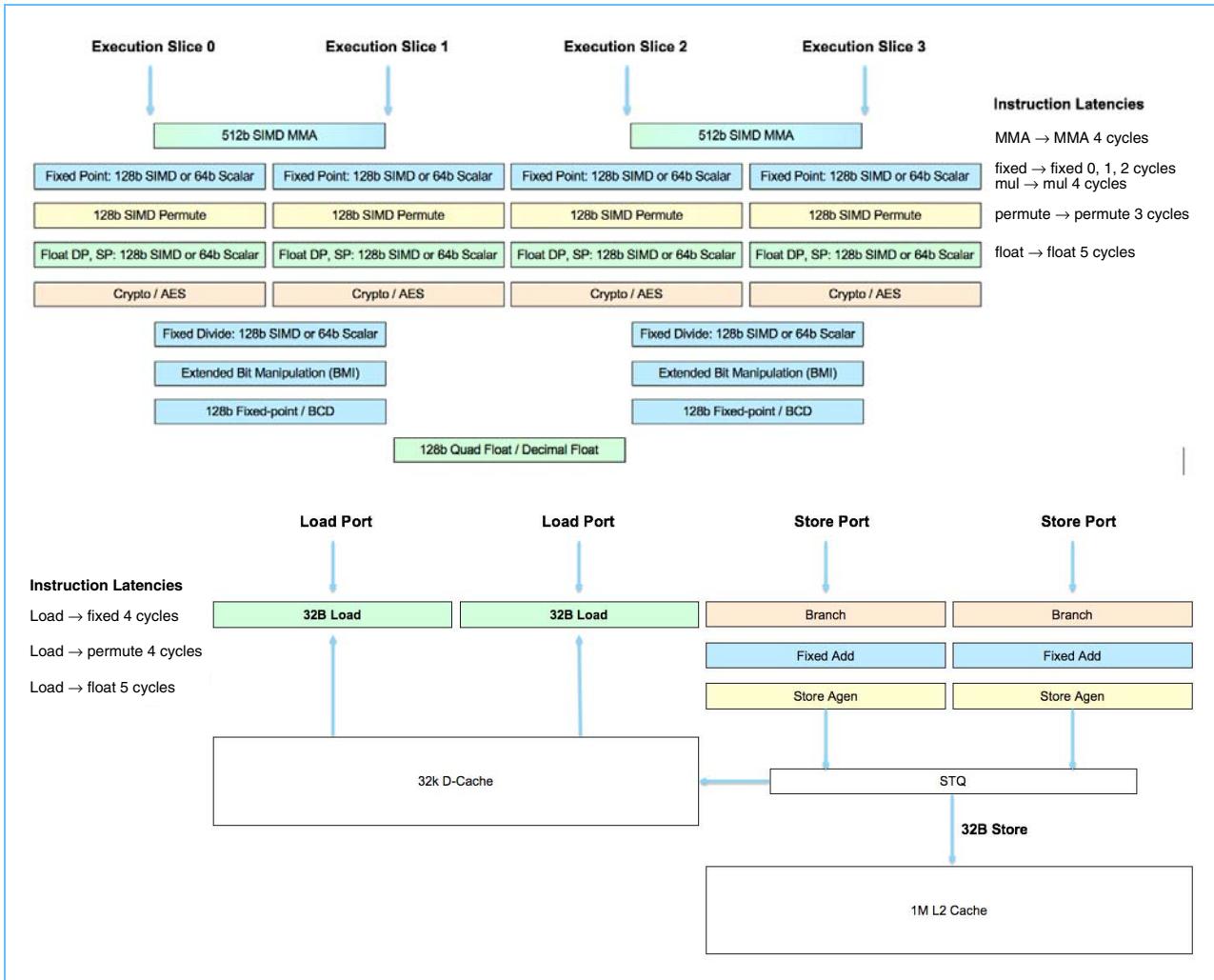


Figure 19-2 illustrates the execution capability block diagram.

*Figure 19-2. Execution Capability Block Diagram*





### 19.1.2 SMT Modes and Thread Count Sensitivity

Generally, all resources are shared within the pipeline between threads unless otherwise stated. Portions of the Power10 pipeline and other resources are partitioned between threads depending on the active SMT mode. See *Section 4.1 SMT Overview* on page 55 for additional details.

The most significant partitioning related to threads occurs when more than two threads are active per SMT4-core-resource, placing the core in SMT4 mode. In SMT4 mode, the decode/dispatch pipeline is split into two pipelines, each pipeline is four IOPs wide and each pipeline serves two threads. The split decode/dispatch pipes each feed one of the two superslices, providing two execution slices for each pair of threads.

Additionally, the instruction completion table (ICT), which tracks IOPs in flight, is statically partitioned based on the number of active threads.

Other pipeline and resource implications of SMT mode are discussed in subsequent sections.

### 19.1.3 Instruction Fetch

In each cycle, a thread can fetch up to eight instruction words (4 bytes each) from the L1 instruction cache (I-cache), branch prediction is performed on all valid instructions, and an instruction prefetcher fetches lines speculatively to reduce I-cache miss occurrences. The compiler should align critical-code segments (targets, inner loops) on a quadword boundary.

The fetched instructions are packed eight at a time into the IBUF or bypassed, up to eight at a time, into the decode stage.

#### 19.1.3.1 Thread Priority

In SMT2 and SMT4 modes, fetches are rotated between threads, round-robin style, until the IBUF is full for the thread. Requests for thread priority at fetch and decode are honored per specification of thread priority (see *Section 5 Architecture Compliance* on page 73). The Power10 core also optimizes thread-priority selection to improve inter-thread efficiency during long-latency stall events. By lowering thread priority during a low-productivity state, additional processing resources are provided to other threads.

Fetch priority is determined by the thread priority set by PPR and RPR. Priority levels are as follows:

1. Very low: decode every 128 cycles (non-relative).
2. Low: decode every 16 cycles (non-relative).

**Note:** This is a new feature for the Power10 core to assist in managing power from low-priority threads.

3. (3+) depends on relative priority compared with other threads.

To enable power savings or allocate processing resources to another thread, use the pause form of the wait instruction (for example, **pause\_short**). This is preferred over setting the PPR to “very low”.



#### 19.1.3.2 L1 Instruction Cache

Each core contains a 48 KB, 6-way set-associative L1 instruction cache (I-cache). The I-cache is allocated in 128-byte lines with 32-byte sector valid tracking. The replacement algorithm is pseudo LRU. The I-cache is banked and allows concurrent reads and writes to different banks. The I-cache and associated effective address directory (EADIR) are accessed using EA bits 51:56 and tagged using EA bits 41:51. Entries in the I-cache directory are also tagged with MSR bits PR, LE, and HV. The EADIR is used to predict the way selection for each I-cache access. An EADIR hit that is later determined to be an I-cache miss adds approximately eight cycles to the base miss latency.

The I-cache can be accessed on any 16-byte quadword boundary and returns up to two 32 consecutive bytes per cycle, which results in fetching eight instruction words for in-line code. A branch target fetches 5 - 8 instructions in one cycle based on target alignment, maximized for depending on quadword alignment, except when crossing a 128-byte cache-line boundary. Instruction fetches do not span a cache-line boundary in a single cycle.

For each thread, aliasing can occur when a given EA(41:56) cache line is required for more than one real address or a combination of MSR bits. Only one of the two combinations can be valid in the I-cache for a given thread at any one time, and an EADIR invalidate is required before fetching the other alias. Multi-thread EADIR aliasing results when two threads map the same EA(41:56) to two different real addresses or MSR combinations. When address aliasing occurs between threads, lines are brought in as private to the thread after first invalidating the existing entries.

On instruction fetches, effective address bits are used to index into the I-cache and the directory. An effective-to-real-address-translation (ERAT) table is shared with the L1 data cache and resides physically in the load store unit (LSU).

#### 19.1.3.3 Instruction Prefetch

An instruction prefetch mechanism is used to fetch additional lines after an I-cache miss is detected. The instruction prefetcher uses the I-cache miss history to make decisions about the depth of prefetching on a per miss basis, ranging from 0 - 7 lines ahead. The Power10 processor enables this prefetcher in all SMT modes, but with an adjusted prefetch depth depending on the SMT mode. The bandwidth of the instruction prefetcher is extended for SMT8 cores when there is only one thread active.

The banked cache design of the Power10 instruction cache allows a concurrent read and write (as long as they reference different cache banks), so that writing prefetched lines into the instruction cache does not steal cycles from the fetching of instructions from the cache.

#### 19.1.3.4 Software-Initiated Instruction Prefetch

An **icbt** instruction initiates the prefetch of a line into the L3 cache for use by the instruction cache.

When an L1 instruction fetch miss is critical to avoid for performance, the programmer can attempt to pre-load the line in the L1 cache using the static hint bits to force an instruction prefetch by intentionally predicting down the wrong path. See section *Branch Direction Prediction Using Static Prediction* on page 251 for details on how to enable the hint bits for this code sequence such as inserting a special NOP. Because this method causes a pipeline flush, it should only be used when experimentation to demonstrate performance advantage can be performed on the target system. With this method, some instructions are speculatively executed or processed to some extent by the instruction fetch logic before they are discarded. The instruction in the



(wrongly) predicted path can be used as a hint instruction to the memory subsystem. For example, software prefetching of instructions from location “Line\_to\_touch” can be initiated by forcing a branch misprediction as follows (the “a” bit in the **bc** instruction indicates “must agree with static prediction”).

Short distance touches:

```
bc Line_to_touch          // Static prediction taken, but CR bit is set to  
                           "not-taken"
```

Long distance touches:

```
bc Next                  // Static prediction not-taken, but CR bit is set to  
                           "taken"  
b Line_to_touch          // Initiate prefetch for cache line "Line_to_touch"  
Next:...                 // Instructions in the actual instruction stream
```

This type of software prefetching is useful if the line to prefetch is in the L3 cache or beyond. Because of the high penalty for branch misprediction, it might not be beneficial if the referenced line is already in the L2 cache and even harmful if it is already in the I-cache. It is beneficial if the compiler makes special attempts to schedule code around such a branch that reduces the misprediction penalty. Attempts to reduce the forced branch misprediction penalty can be made by:

- Setting the CR bit used by the “bc” as early as possible.
- Scheduling such a branch in a code segment, where there are relatively few branches so that the branch does not wait too long in the branch issue queue behind other branches.
- Trying to overlap a likely D-cache miss with the forced branch misprediction.
- Scheduling such a branch after an existing long chain of flow dependency.

**Note:** The mechanism for block prefetch into the L3 cache is not suitable for the instruction-side prefetch.

#### 19.1.3.5 Branch Prediction

As instructions are fetched, they are scanned for branches. Up to eight branches are simultaneously processed by the branch prediction logic that predicts both the direction and/or target of the branches, depending on the branch type.

Branch direction prediction for conditional branches is performed using four branch history tables, a tagged orientation predictor (TOP), and a TAGE predictor. For nonrelative branches, target prediction is performed using a link stack for link returns and a count cache and tagged indirect predictor (TIP) for other indirect branches. For relative branches, the target address is computed precisely. All conditional branches are predicted using the branch predictors. An incorrectly predicted branch results in a pipeline flush after the branch is executed.

The pipeline latency for a predicted taken branch from the BHT, link stack, and count cache predictors is three cycles, and the latency for a TAGE, TIP, and TOP prediction is five cycles. The Power10 processor also includes a BTAC predictor that reduces the predicted branch taken latency to one cycle, including for indirect branches predicted by the link stack or local count cache.

Static branch direction prediction is performed using hints as defined by the *Power ISA User Instruction Set Architecture - Book I (version 3.1B)*. Branches that are statically predicted are always predicted in the direction of the hint.



These predictors are described in more detail in the following subsections.

#### *Branch Direction Prediction Using the Branch History Tables*

The Power10 core uses a set of four branch history tables (BHTs) to predict the direction of branch instructions, supplemented by a TAGE and TOP predictor with the following entries:

- Local predictor: 16K entries. Predict branch taken based on history. Indexed by the branch address.
- Global predictor: 8K entries. Predict branch taken based on the path of execution to reach the branch. Indexed with a global-history-vector (GHV), formed from a taken branch history hashed with the address of the branch.
- Selector: 8K entries. Track which of the local or global predictor results should be used. Indexed by the global predictor index.
- Local selector: 16K entries. Tracks branches predicted well by the local predictor and prevents them from installing in (polluting) the global predictor. Indexed by the global predictor index.

Unconditional branches (including branches with the BO field set to ‘1z1zz’) and statically predicted conditional branches (such as branches with the “a” bit set to ‘1’) do not have an entry in the BHTs.

#### *Branch Direction Prediction Using the TAGE*

While the main BHTs are not tagged, the tagged geometric history length predictor (TAGE) has a 10-bit tag per entry to uniquely identify the optimal GHV length for a particular branch. The TAGE consists of four different global history predictors, each uses a different length of GHV. All four predictors are accessed concurrently and the longest predictor with a matching tag is used. The TAGE maintains a usefulness indicator with each table entry, when the indicator is showing poor prediction, the next longest table is updated. The TAGE prediction overrides a BHT prediction only if the usefulness value is greater than a threshold.

For TAGE to correctly predict the exit of a loop with a consistent code path, the loop should have less than 100 total of taken branches and sector crossings. A two-cycle penalty is incurred when the TAGE overrides the BHT prediction.

#### *Branch Direction Prediction Using the Tagged Orientation Predictor*

The tagged orientation predictor (TOP) is a direction predictor that is locally tagged and overrides the BHT but not the TAGE prediction. The TOP provides a supplemental prediction for branches that are not predicted well by the other predictors.

#### *Branch Direction Prediction Using Static Prediction*

The Power10 core normally ignores any software that attempts to override the dynamic branch prediction by setting the “a” bit in the BO field. This is done because historically programmers and compilers have made poor choices for setting the “a” bit, which limited the performance of codes where the hardware can do a superior job of predicting the branches. However, the “a” bit can still be an important tool for certain performance-sensitive cases, such as those identified after careful analysis of branch misprediction on a Power10 system. The Speculative Branch Hint Enable (SBHE) bit of the Dynamic Execution Control Register (DEXCR) controls whether or not the static prediction is obeyed by the hardware. Alternatively, static prediction can be temporarily honored via a special NOP instruction; which is useful when the value of the SBHE is unknown or when hints are important for performance only in a specific code section (specifically, an unconditional bc+4 (b-form) with BO(3) = 1). The honoring takes effect two cycles after the NOP is fetched.



The temporary honoring of static prediction due to the special NOP lasts for a count of 127 total of:

- taken branches, plus
- sequential cache-line crossings for instruction fetches.

In terms of number of instructions, this is:

- minimum of approximately 120 instructions (if every instruction is a taken branch)
- maximum of approximately 5,600 instructions (if no taken branches)
- typically approximately 1,400 instructions (if taken branch every eight instructions)

Note that a **lарx** instruction will reset the branch + crossings counter to 8, which could shorten the amount of time that static prediction is honored following the special NOP.

Incorrect setting of the “at” bits results in a pipeline flush. Therefore, setting of the “a” bit should be avoided except in extraordinary cases where a thorough analysis has been performed.

The following cases are the only suggested uses of the “a” bit. When the “a” bit is set, the “t” bit of the BO field specifies ‘1’ for taken and ‘0’ for not-taken. These uses are honored even when the hardware is configured to ignore the “a” bit.

- For the branches that close out a lock acquisition sequence, it is desirable to force the branch prediction based on the hotness of the lock. If the lock is not contended, optimal performance will occur when the branch is predicted as not taken (predict lock acquisition). If the lock is highly contended (a “hot lock”), then optimal performance will occur when the branch is predicted as taken (predict no lock acquisition). The following example illustrates setting “branch to predict not taken” for the case where speculation into the critical section is desired.
  - Without static prediction, if the lock is not acquired in the first iteration, the branch history mechanism works to update the prediction to predict *taken*; that is, predict lock acquisition failure and cause more “**lwarx**” traffic for the next iteration. When the hardware detects a **I\*arx** instruction near a static prediction, the static prediction is honored.

```
top: lwarx
      add
      stwcx
      bc- top    <-- Power10 core predicts this branch to be not taken, through
                  software directives that properly set the “a” and “t” bits.
```

#### *Branch Target Address Prediction Using the Link Stack*

The Power10 core uses a link stack to predict the target address for a branch-to-link instruction that it believes corresponds to a subroutine return. By setting the hint bits in a branch-to-link instruction, software communicates to the processor whether the instruction represents a subroutine return, or a target address that is likely to repeat, or neither (see *Table 19-1*).

When instruction fetch logic fetches a branch and link instruction either unconditional or conditional but predicted taken, it pushes the address of the next instruction into the stack. When it fetches a branch-to-link instruction with “taken” prediction and with hint bits indicating a subroutine return, the stack is popped and instruction fetching starts from the popped address.

In the Power10 core, the link stack is 64-entries deep and is split per thread mode: 32 entries per thread in SMT2 and 16 entries per thread in SMT4 mode.

*Table 19-1* summarizes the handling of the Power10 **bclr** and **bclrl** instructions.

*Table 19-1. Handling of bclr and bclrl Instructions*

Instruction	BH Field	Power10 Design	Power ISA
<b>bclrl</b>	xx	If the branch is predicted taken, the link stack address is used as the predicted target address; however, the link stack is not popped.	Same as for <b>bclr</b> .
<b>bclr</b>	00	If the branch is predicted taken, the link stack is popped and the popped address is used as the predicted target address.	The branch is a subroutine return.
<b>bclr</b>	01	If the branch is predicted taken, the target is predicted using the count cache. The count cache data and confidence fields might be updated when the branch is executed and resolved. No action is taken by the link stack.	Target address is likely to repeat.
<b>bclr</b>	10	Same as BH = '00'.	Reserved.
<b>bclr</b>	11	Same as BH = '01'.	Target is not predictable.

#### *Branch Target Address Prediction Using the Count Cache*

The count cache is used to predict the target address for branch-to-count (**bcctr[I]**) instructions and branch-to-link (**bclrl[I]**) instructions, which do not set the BH field to indicate a subroutine return and therefore, are not predictable by the link stack.

The count cache predicts the branch target based on previous target addresses from previous executions of the same instruction. By setting the hint bits appropriately, software communicates to the hardware whether the target address for such branches are predictable using a cache. See *Table 19-1* on page 253 and *Table 19-2*.

**Note:** The count caches can only be accessed for one branch per cycle. The **bcctr[I]** and **bclrl[I]** instructions use the count cache. Therefore, no more than one such branch should be placed per aligned 32-byte or octoword block. Other branches that do not access the count cache are still predicted using the other predictors. (The tagged indirect predictor helps to mitigate this limitation.)

The Power10 core maintains two count cache arrays:

- A global 512-entry array maintaining only the lower 32 bits of the target address (the upper bits are assumed to be unchanged). It is indexed similar to the global BHT using a GHV hashed with the fetch address.
- A local 256-entry array with a full 64-bit target address in each entry. It is indexed using the fetch address.

**Note:** Due to indexing using the fetch rather than instruction, address, branches that use the count cache can populate different count cache entries if they are accessed both via an aligned and unaligned fetch.

A 2-bit selector value is stored in the local array to select between use of the local and global caches.

*Table 19-2. Handling of bcctr and bccctrl Instructions (Sheet 1 of 2)*

Instruction	BH field	Power10 Design	Power ISA
<b>bcctr, bccctrl</b>	00	If the branch is predicted taken, the target address is predicted using the count cache. Update the count cache when the branch is executed, if the branch is resolved as taken. For the <b>bccctrl</b> instruction, if the branch is predicted taken, push in the link stack the address of the next sequential instruction when the <b>bcctr</b> instruction is fetched.	Target address is likely to repeat.



Table 19-2. Handling of **bcctr** and **bcctrl** Instructions (Sheet 2 of 2)

Instruction	BH field	Power10 Design	Power ISA
<b>bcctr, bcctrl</b>	01	Same as BH = '00'.	Reserved.
<b>bcctr, bcctrl</b>	10	Same as BH = '00'.	Reserved.
<b>bcctr, bcctrl</b>	11	Same as BH = '00'.	Target is not predictable.

#### Branch Target Address Prediction Using the Tagged Indirect Predictor (TIP)

The TIP is a supplemental predictor used for predicting branches that are not well-predicted by the count cache. The TIP uses a separate history vector to index according to indirect branch target patterns.

#### Branch Target Address Prediction Using the BTAC

The branch target address cache (BTAC) is used to provide target fetch addresses for the current fetch group without wasting any fetch cycles due to a delay in a taken branch prediction. That is, the BTAC has a 1-cycle latency versus the 3 - 5 cycles of latency for other predictors.

The BTAC is only active in ST mode (per small core), and predicts both relative and indirect targets.

The BTAC is best used for a frequent branch that is consistently taken. To optimally utilize the BTAC without performance penalties, a branch should be the first taken branch in the fetch group. Therefore it is safest for software to schedule a BTAC-dependent branch at least an octword after the previous branch.

For a given branch, each unique instruction path, based on the prior fetch, requires a unique entry in the BTAC. Therefore for optimal BTAC utilization, the approach to the branch should be consistent for each iteration of the branch.

Branch-taken redirects start a new fetch group at the target:

- Next cycle if predicted by the BTAC
- Three cycles later (2 bubbles) if not predicted by the BTAC or TAGE
- Five cycles later (4 bubbles) if predicted by the TAGE

A branch-and-link instruction, whose address is predicted by the BTAC, can corrupt the link stack if a subsequent branch-to-link occurs within two clock cycles. Therefore, it is advised (if possible) to separate such link stack “push” and “pop” instructions in code sequences.

#### Obtaining the Next Instruction Address/BC+4 Handling

On the Power10 core, the preferred method of obtaining the next instruction address is to use the prefixed PC-relative instructions; for example, **paddi**, **pld**, **pstd**.

Codes that instead use unconditional B-form branches with the link bit set and a displacement of '4' to set the address of the next instruction into the link register are handled specially. Architecturally, these branches are taken and go to the next instruction. The hardware handles BCL + 4, with a BO = x'20' (unconditional taken) as a special case. For this special case, the branch is always treated as *not taken* for fetch, resulting in no fetch penalty. When the special branch executes, it updates the link register, and does not cause a flush (even though fetch processed it as *not taken* and architecturally it was *taken*). The **PMU** sees these special branches as requiring direction prediction, direction predicted correctly, and branch not taken counts.

### Branch Prediction Power Down

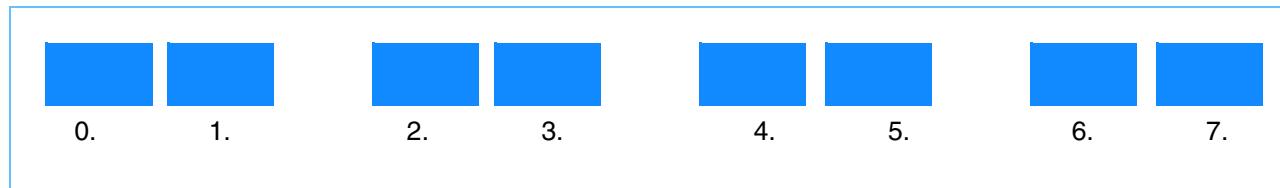
Branch history tables can be temporarily powered down based on sequences of consecutive branches predicted correctly by the local BHT. The count cache can be temporarily powered down in the absence of indirect branches.

#### 19.1.4 Instruction Decode and Dispatch Pipeline

Instructions can be 4 bytes or 8 bytes (prefixed). After instructions are fetched from the I-cache, they are decoded into IOPs and then dispatched to slices for execution scheduling.

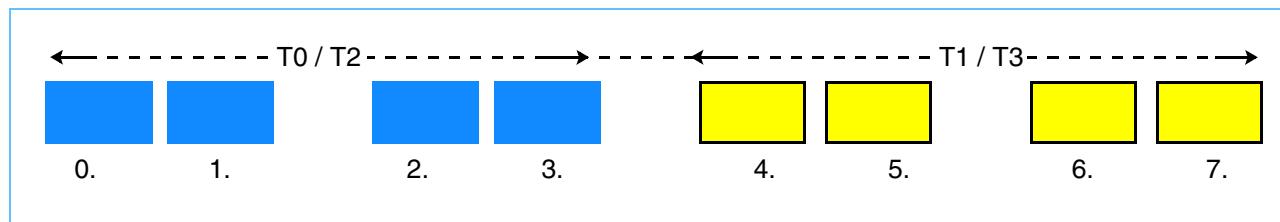
The decode/dispatch pipeline can process up to eight IOPs (for up to eight instructions) per cycle per SMT4-core-resource (see *Figure 19-3*).

*Figure 19-3. Decode/Dispatch Pipeline*



In SMT4 mode, it is split into two (mostly) independent pipelines, each handling up to two threads and each processing up to four IOPs per cycle (see *Figure 19-4*).

*Figure 19-4. Decode/Dispatch Pipeline (SMT4 Mode)*



##### 19.1.4.1 Instruction Buffer

When the decode stage cannot accept all the fetched instructions due to the high fetch rate, or due to stalls in the pipeline, instructions are stored in the IBUF. This allows the instruction fetch and branch prediction to proceed during the pipeline stall condition.

Instructions that are not bypassed are written into the IBUF. Bypass from an instruction fetch occurs when the IBUF is empty. The IBUF holds 128 instruction entries and is partitioned per thread statically in SMT2 mode and SMT4 mode. Fetch for a given thread does not occur unless there are at least eight entries available in the IBUF.

New instructions, wait-pause (**pause\_short**) and wait-reserve (**waitrsv**) allow holding instructions at the IBUF. These provide a means to limit power consumption and SMT resource consumption by the program.



#### 19.1.4.2 Effective Address Tracking

The effective address table (EAT) stores branches and retains a correlation of branches to instruction addresses. The EAT holds 48 entries, 24 entries per thread in SMT2 mode, and 12 entries per thread in SMT4 mode. A new EAT entry is consumed for each new 128-byte cache line that is accessed as part of the in-flight instruction stream and for each predicted-taken branch. Fetch is held for a thread when there are no remaining EAT entries for the thread, thus limiting the number of predicted-taken branches in flight to 48 for the core.

#### 19.1.4.3 Instruction Cracking, Expansion, Fusion, and Prefixing

Instruction decode maps instructions into IOPs.

Most instructions in the Power ISA are not cracked and are decoded into a single IOP that takes up one decode lane. However, some instructions are cracked or expanded into more than one IOP, and/or into two or more decode lanes. Those instructions always start on an EVEN itag, resulting in a 50% chance of inserting an empty slot for alignment.

There are three categories of instruction cracking and expansion:

- 2-way cracked or “single”
  - Consumes two slots at decode, dispatch, ICT (two itags).
  - Example: load-float-single cracks into:
    - Load-single (load-pipe).
    - Convert-to-single (CTS) (fixed-pipe).
  - Example: **addc.** is a “single”, it takes two decode slots/itags but only uses one IOP in the pipeline after dispatch.
- 4-way cracked
  - Consumes four slots at decode, dispatch, ICT (four itags).
  - Requires a full-dispatch group, such that in ST and SMT2 modes, the other four decode slots are unused.
  - Examples: **amo**, **mtxer (full)**, **addco.**, **subco.**.
- Expanded
  - Split into more than four IOPs or has a variable number of IOPs.
  - There is one expansion engine per SMT4-core-resource, meaning only one expansion can be decoded at a time. This engine is shared between the two split decode pipelines in SMT4 mode.
  - Expansion produces up to four IOPs per cycle.
  - There is a 3-cycle decode startup penalty for each expanded instruction; that is, there are three cycles in the pipeline for which no instructions are decoded. In SMT4 mode, the decode penalty applies only to the split decode pipeline on which the expansion instruction is detected.
  - Examples of expanded instructions are: load-multiple or AMO instructions.
- Fused
  - Always a pair of 4-byte instructions, itags
  - Might take a pair of issue-queue entries

- Prefixed
  - Always takes two slots at decode, dispatch, ICT (two itags)
  - Might take a pair of issue-queue entries

Cracking is done independently per split pipeline in SMT4 mode. The cracked IOPs must decode together. For example, this can cause IOPs to shift decode to the following cycle if the first IOP takes the last decode slot.

A listing of cracked and expanded instructions can be found in *Table A Instruction Properties* on page 295, by examining the “Instruction Type” column.

*Figure 19-5* on page 257 shows an example sequence with a 2-way crack and forced empty slot.

*Figure 19-5. Example of a Sequence with 2-Way Crack and Forced Empty Slot*

*Example sequence with 2-way crack and forced empty slot*

add add add 1fs add add

**Note:** 50% of the time, if **Ifs** does not align to an even slot, there will be an empty decode slot inserted to align the cracked operation.



**Note:** Care should be taken for sequences that reduce decode efficiency (for example, high IPC codes): crack, non-crack, crack, non-crack.... introduce empty decode slots at a rate of  $\frac{1}{4}$ . Place 2-way cracked instructions adjacently in code, because a maximum of only one empty decode slot would be inserted for a string of consecutive cracked instructions.

#### 19.1.4.4 Instruction/IOP Completion Table

The instruction/IOP completion table (ICT) tracks IOPs from dispatch through completion and is allocated at decode time. Each IOP from a crack or microcode expansion consumes one entry in the ICT.

The ICT contains 512 entries and is split for SMT: 256 entries per thread in SMT2 mode and 128 entries per thread in SMT4 mode. ICT entries are grouped into an even/odd pair.

Certain instructions start on an even itag. Each pair of itags that share an ICT entry decodes, dispatches, and completes together. This does not mean that the operations execute together:

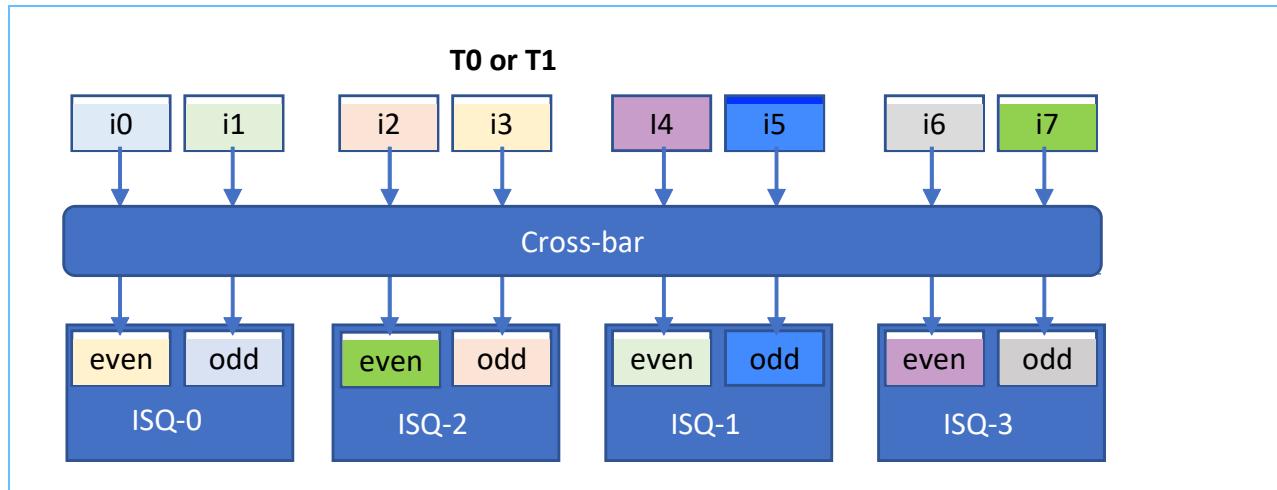
- Independent operations execute separately.
- Some fused-cracked and prefixed operations share a pair of issue-queue entries (“paired”) in the same slice (hold more information or coordinate execution).
- For example, back-to-back, fixed-point instructions can issue back-to-back operations and PC-relative load/store instructions issue as one operation.

#### 19.1.4.5 Dispatch

Dispatch is in-order, per thread and takes place on a decode block of instructions. If all eight slots cannot be dispatched, the group takes additional cycles until all slots are dispatched. The instructions are presented from decode (one thread at a time), and dispatch does not blend threads.

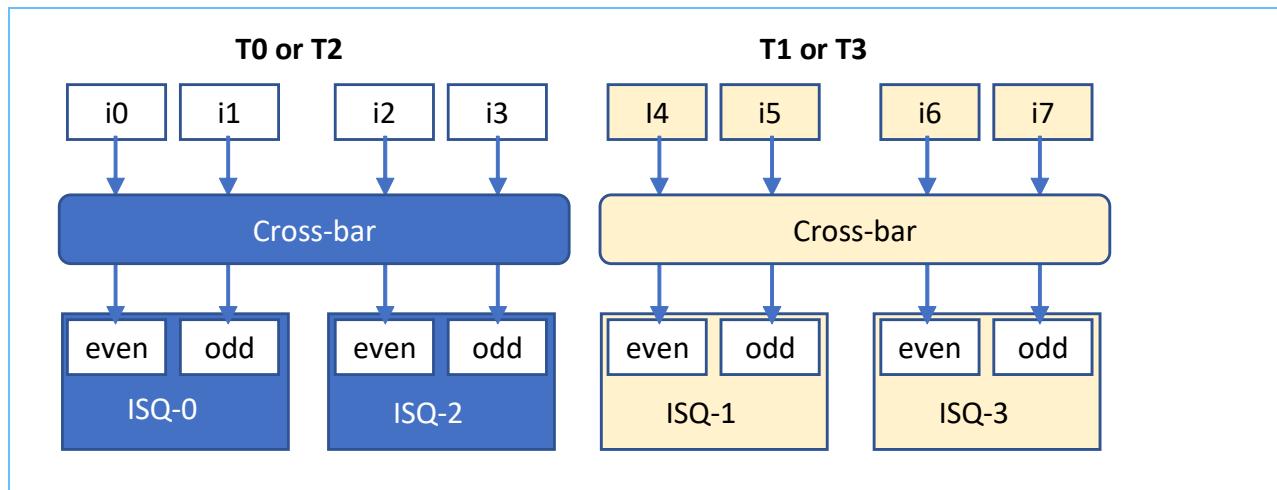
Figure 19-6 shows dispatch-issue queue assignments for the ST and SMT2 modes.

Figure 19-6. Dispatch - Issue Queue Assignments for ST and SMT2 Mode



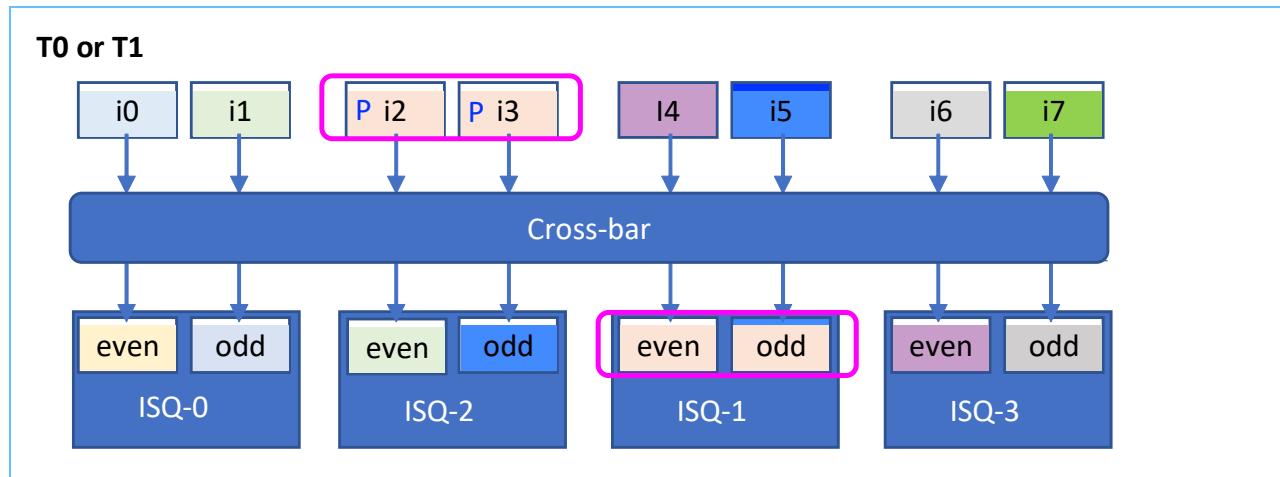
SMT4 is similar but with a split design to match the split decode pipe. Note that only one thread is presented per cycle per side. Figure 19-7 shows the dispatch-issue queue assignments for the SMT4 mode.

Figure 19-7. Dispatch - Issue Queue Assignments for SMT4 Mode



Issue-queue paired instructions require a matching even and odd issue queue entry to be available for dispatch. They transfer together from the even/odd decode lanes into an even/odd entry in the ISQ. Figure 19-8 shows dispatch for paired instructions.

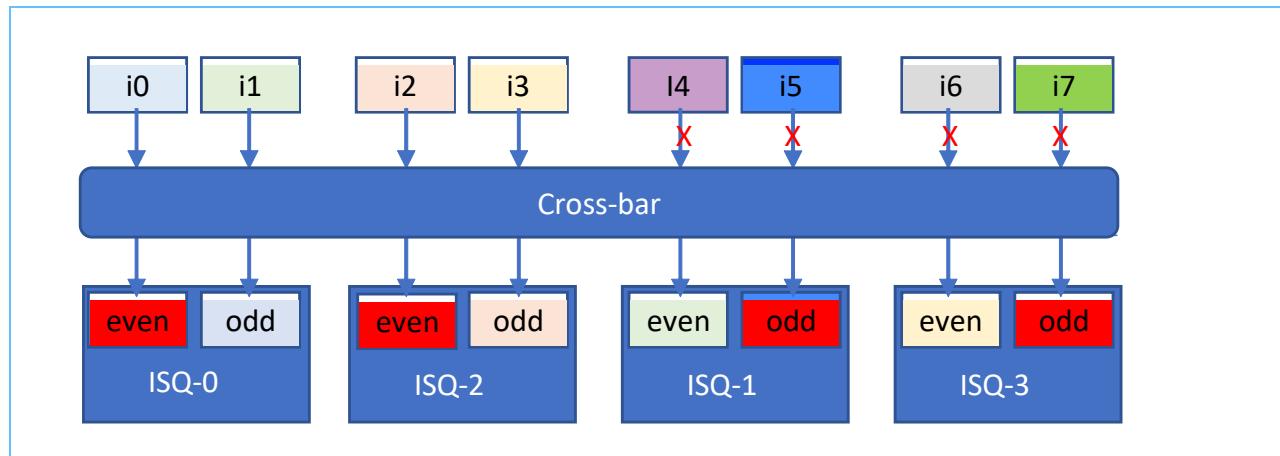
Figure 19-8. Dispatch (Paired Instructions)



As individual issue queues become full or run out of resources including register renames, the dispatcher will continue to dispatch to available slices.

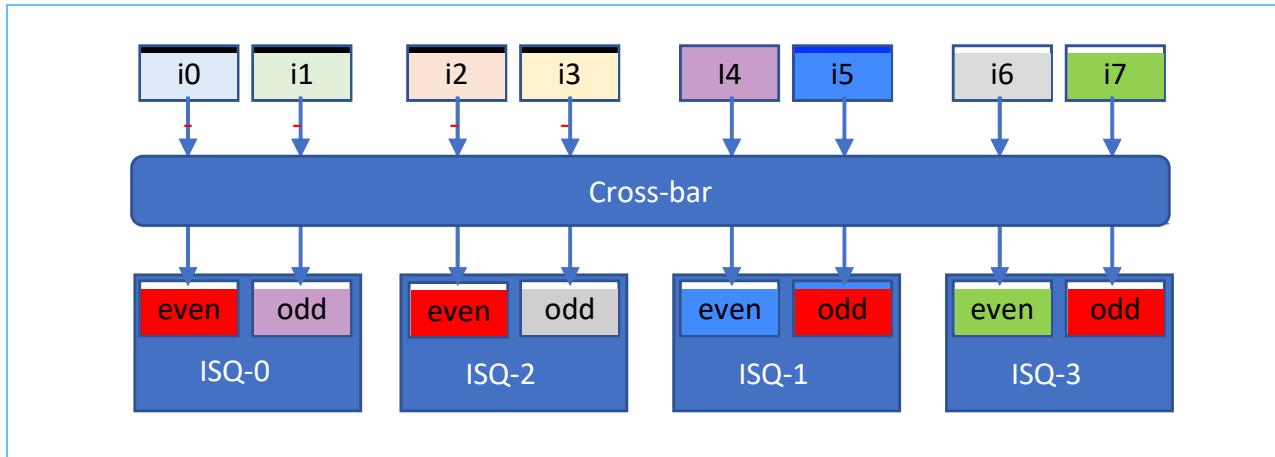
A fullness indicator exists per half-slice, enabling dispatch to a half-slice if the other half-slice is full, provided that both instructions in an even/odd dispatch lane pair have an available half-slice. Therefore, if four of the issue queue receiving slots are full, four instructions can dispatch. See the examples shown in Figure 19-9 and Figure 19-10.

Figure 19-9. Cycle 1 with Four Slices Full



If only these slots remain full, the other four instructions will be dispatched in the next cycle to the same ports.

Figure 19-10. Cycle 2 with Four Slices Full



In the previous example, if any of the input operations are re-paired, it will not be able to dispatch until a matching even/odd entry are free.

#### 19.1.4.6 Register Renaming

##### Zero Cycle Moves

Zero cycle register moves avoid the latency of the producer to consumer when moving a register within the shared STF register file.

The types of register state kept in the STF include:

- GPR
- FPR
- VSR
- VR
- CTR
- LR
- TAR
- SPRG2
- HSPRG1

Zero cycle moves are achieved in hardware through instruction fusion. The following list shows supported zero cycle moves:

1. GPR → LR (**mtlr**) followed adjacently by **bclr(l)**
2. GPR → CTR (**mtctr**) followed adjacently by **bcctr(l)**



*Table 19-3* lists the zero-cycle register move combinations supported on the Power10 processor and preferred move types that might be supported on a future processor. Code should use these preferred methods of moving registers even when alternate methods are available.

*Table 19-3. Zero-Cycle Register Move Combinations*

Move Destination	Move Source	IOPs	Zero Cycle Move Supported
GPR	GPR	<b>mr</b> (or Ry, Rx, Rx)	N
FPR	FPR	<b>fmr</b> FRT, FRB (Rc=0)	N
VSR	VSR	<b>vmr</b> (vor Vx, Vy, Vy)	N
GPR	FPR	<b>mffprd</b>	N
GPR	VSR	<b>mfvrd</b>	N (no 128 bit → 64 bit)
GPR	CNT	<b>mfsp9 9 (mfctr)</b>	N
GPR	LNK	<b>mfsp8 8 (mflr)</b>	N
GPR	TAR	<b>mfsp815</b>	N
GPR	SPRG2	<b>mfsp274</b>	N
GPR	HSPRG1	<b>mfsp305</b>	N
FPR	GPR	<b>mtfprd</b>	N
VSR	GPR	<b>mtvrd</b>	N (no 64 bit → 128 bit)
CNT	GPR	<b>mtspr9 9 (mtctr)</b>	Y
LNK	GPR	<b>mtspr8 8 (mflr)</b>	Y
TAR	GPR	<b>mtspr815</b>	N
SPRG2	GPR	<b>mtspr274</b>	N
HSPRG1	GPR	<b>mtspr305</b>	N

Constant moves are NOP at dispatch (no travel through the pipe) and bypass any dependency. *Table 19-4* describes supported zero cycle constant assignments.

*Table 19-4. Zero Cycle Constant Assignments* (Sheet 1 of 2)

IOP	Name	Description
<i>Constant Zero Move</i>		
<b>xor</b> RT,RS,RB	Exclusive OR	RS = RB RT ← RS ^ RB (sets RT to 0)
<b>andi</b> RA,RS,UI	AND Immediate	UI = 0 RA ← RS & 0 (sets RA to 0)
<b>andis</b> RA,RS,UI	AND Immediate Shifted	UI = 0 RA ← RS & 0 (sets RA to 0)



*Table 19-4. Zero Cycle Constant Assignments (Sheet 2 of 2)*

IOP	Name	Description
<i>Constant Immediate Move</i>		
<b>addi</b> RT,RA,SI	Add Immediate	RA = 0, SI = 0, 1, -1 RT $\leftarrow$ 0 + EXTS(SI) (sets RT to SI)
<b>addis</b> RT,RA,SI	Add Immediate Shifted	RA = 0, SI = 0 RT $\leftarrow$ 0 + SI (sets RT to 0)
<b>li</b> RX,value	Load Immediate	value = 0,1,-1 RX $\leftarrow$ value (set RX to value)
<b>xxlxor</b> XT, XA, XB	VSX Vector Exclusive OR	XA = XB XT $\leftarrow$ XA $\wedge$ XB (sets XT to 0)
<b>vxor</b> VRT,VRA,VRB	Vector Exclusive OR	VRA = VRB VRT $\leftarrow$ VRA $\wedge$ VRB (sets VRT to 0)
<b>xxleqv</b> XT,XA,XB	VSX Vector Equivalence	XA = XB XT $\leftarrow$ XA = XB (sets XT to -1)
<b>veqv</b> VRT,VRA,VRB	Vector Equivalence	VRA = VRB VRT $\leftarrow$ VRA = VRB (sets VRT to -1)
<b>xxspltib</b> XT,IMM8	VSX Vector Splat Immediate	IMM8 = 0, 255 XT $\leftarrow$ 0 if IMM8 = 0 (sets XT to 0) XT $\leftarrow$ -1 if IMM8 = 255 (sets XT to -1)

#### *Scoreboarding and Dispatch Serialization*

There are three scoreboards on the Power10 core at dispatch.

Scoreboards have setters (SS) and checkers (CS). If an SS is in the pipeline (from dispatch through completion) for the thread, then a CS will stall at dispatch.

**Note:** In the following list, SPR means not one of XER, CR, FPSCR, VSC.

The three scoreboards are:

- Scoreboard0: most SPR access is not in scoreboard1
- Scoreboard1: non-renamed SPRG and HSPRG
- FPSCR: **mtfpscr** and FPSCR reader

Some operations serialize at dispatch; they can serialize before, after, or both. For these operations, a dispatch flush is possible to enhance performance in SMT mode. A dispatch flush includes flushing back to fetch and holding the instructions at the IBUF; TLBIE and scoreboard wait are examples.

## Register Allocation

There are 360, 128-bit entries in the main Slice Target File (STF) registers supporting the following register types in a unified structure:

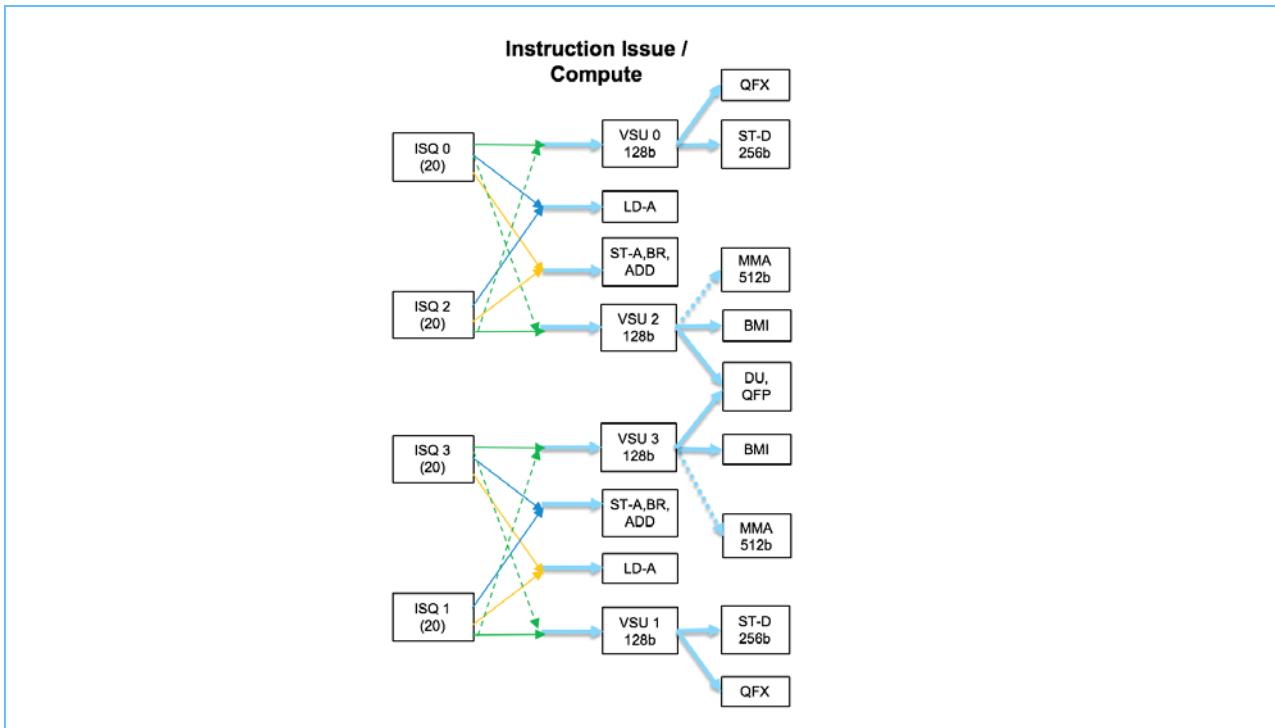
- GPR
- FPR
- VSR
- VR
- CTR
- LR
- TAR
- SPRG2
- HSPRG1

The 128-bit registers in the main STF pool are allocated as “full entries”, essentially even/odd pairs of “half entries.” The 64-bit registers in the STF pool are allocated as “half entries” using a half based statically on the thread, such that there are effectively double the number of entries available. Therefore, in SMT2/SMT4 mode, the register rename space is increased if 128-bit entries are cleared out when not in use such as by zeroing out the VSRs. For example, this could be done on a context switch.

### 19.1.5 Instruction Issue

Figure 19-11 shows the flow for the issue of instructions.

Figure 19-11. Issue of Instructions





#### **19.1.5.1 Load/Store AGen Issue**

Two load AGen ports (one per superslice) handle iops routed to the LD pipelines (see Pipe Class in *Table A-1. Instruction Properties* on page 295) for load iop address generation.

Likewise, two store AGen ports (one per superslice) handle iops routed to the ST pipelines for store iop address generation. The store AGen ports are also used for Branch (BR) and Simple FX (SX) issue.

Store data issues on the VSU issue port, after the AGen issues.

Load and store tags are allocated at dispatch, with virtual tags available to avoid dispatch stalls in case no physical load or store queue entry is available at dispatch time. AGen requests are prevented from issuing until a physical load/store queue entry is available.

#### **19.1.5.2 Execution Pipeline Issue-to-Issue Latencies**

*Table 19-5* on page 265 shows the issue-to-issue latency between IOP pipelines for main destination, GPR, FPR, VSR, and VR. The latency depends not on the main destination register type (for example, GPR or VR) but rather on the execution pipelines of the producer and consumer.



Table 19-5. Issue-to-Issue Latency Between IOP Pipelines

Source Route		Ld/St AGen	Simple FX	BR (CR rdy)	FX		FX2		PM	Store Data	DX	GPR MUL	VMX MUL	BF	CY	DF	DV	MMA
Load Data	LD	4	4	6	4		4		4	4	5	5	5	5	5	5	5	
Load Data Algebraic	LD	5	5		5		5		5	5	5	5	5	5	5	5	5	
Store (Update)	ST	2	2		2		2		2	2	3	3	3	3	3	3	3	
Simple Fixed Point	SX	2	2		2		2		2	2	3	3	3	3	3	3	3	
Fixed Point 1-Cycle	FX	2	2	2	2	1	2	1	2	2	3	3	3	3	3	3	3	
Fixed Point 2-Cycle	F2	3	3	3	3		3		3	3	4	4	4	4	4	4	4	
Permute	PM		3		3		3		3	3	4	4	4	4	4	4	4	
Decimal Fixed Point	DX		5	5	5		5		5	5	4	4	4	4	4	4	4	
GPR Multiply	MU	5	5		5		5		5	5		4						
Vector Multiply	vMU		7		7		7		7	7	6		6	6	6	6	6	
Binary Floating Point	BF		7		7		7		7	7	6		6	5	6	6	6	
Crypto	CY		7		7		7		7	7	6		6	6	4	6	6	
Quad Decimal Floating Point	DFq		15	15	15		15		15	15	14	14	14	14	14	14	14	
Decimal Floating Point	DF		13	13	13		13		13	13	12	12	12	12	12	12	12	
Divide-Fast	DV		13		13		13		13	13	12	12	12	12	12	12	12	
Matrix Multiply Accumulate	MM		8.5		9.5		9.5		9.5	9.5	8.5		8.5	8.5	8.5	8.5	4	



#### Additional notes for *Table 19-5*:

- Back-to-back (1-cycle latency) for fixed-point instructions is only supported if two operations are fused for back-to-back and not fused for zero-cycles. Furthermore, for instructions whose destination register is indicated by bits 11:15 (RA) of the instruction binary, back-to-back is only supported for a consumer source indicated by bits 11:15 (RA, FRA, VRA, XA) of the binary. Example: rldicl-add is supported for back-to-back; rldicl-rldicr is not.
- ADDI is routed to the SFX versus the VSU based on the history of the instruction density.
- For a given VSU issue port, if a write-back collision occurs, the shorter latency operation result is delayed to avoid the collision (excluding collisions with greater than 6-cycle operations).

XER, FPSCR, VSCR, CR (XFVC) latencies:

- VSU XFVC producer → S3 of FX/FX2/PM: no additional delay (includes back-to-back)
- VSU XFVC producer → S3 of other VSU: +2 cycles
- LSU XFVC producer → S3 of any consumer: +1 cycle; for example, load-compare-imm fusion
- Any XFVC producer → S0/S1/S2 of any consumer: +2 cycles
- CR-logical optimized for BA as low-latency source (routed to s3)

The Power10 core implements “fast” SPR operations (FX latency) for the SPRG registers that are not renamed. This enables quick access for scratch use by the supervisor/hypervisor. Other **mfsp** operations take 11 cycles.

Stores issue from a single ISQ entry, first an agen operation, followed by a data beat. The 32-byte stores issue from two issue queue entries (paired), first from agen, then two data-beats.

For execution units which only exists once per SS, the issue queue manages collisions through an LRU ping-pong between the two slices.

- Load, store, branch: these toggle between slices to choose the “winner” if there is a conflict. For most cases, the operation re-issues the next cycle.
- VSU: a conflict occurs when an ISQ issues “across” to the neighbor slice VSU port at the same time as an operation issues from the neighbor (native) ISQ to the same port.
  - FX-DIV, **mfker**, mma, store-data, bmi
  - The winner is chosen by an LRU toggle (same as load/store/branch).
  - The loser issues in a subsequent cycle.

For operations issued to the DFU, if there are operations in both super-slices in the ISQ, then either ISQ can only issue every other cycle (ping-pong) with no consideration of an instruction “ready” for issue (that is, even if dependencies are not met).

Long latency operations (greater than six cycles, including the DFU and SPR) will block issue for dependency wakeup. They are not self-blocking, but will block shorter latency operations for a cycle:

- The **mfsp** blocks the SFX pipe.
- The DFU quad blocks two cycles.

Instructions with an XFVC source are “2-cycle issue” block instructions with an XFVC source on the second cycle.

### 19.1.5.3 Branches

Branches issue from the main issue queue even if the CR source is not ready. In that case, they will be resolved from the BCQ, when the CR result is snooped. This frees up space in the main issue queue and gets targets such as LR and CTR updated quickly.

If two branches issue on the same super-slice on the same cycle, they will be staged serially (back-to-back) for execution and no branches may issue the following cycle.

Branch with LR/CTR target handling:

- Branch and link: the link writer issues first (never a dependency), followed by the branch
- Branch and link on CTR occupies a “paired” ISQ entry. One entry will issue the link writer independently and the other entry will issue the branch once the CTR dependency is met

### 19.1.5.4 Fusion

Fusion identification takes place at predecode. Actuation takes place just prior to decode (pre-cracking).

- “Zero cycle” FX fusion: results in a single issue with normal producer latency (for example, 2 cycles to consumer)
  - ADD G1 ←; ADD G1← G1;  
single operation “3way add”
- “Back-to-back” FX fusion: results in back-to-back issue from a *paired ISQ* entry only for the identified pair. This only occurs if other sources are ready; that is, the operations need not issue back-to-back.
- Non-destructive “zero cycle”: the second operation issues as a zero-cycle fused operation. Independently the first operation issues.
  - ADD G1 ←; ADD G2 ← G1;  
single operation “3-way add” G2; Add G1 (independent)
- Load-load: issues as if it is a wider load
- Store-store: issues as if it is a wider store
- Permute-legacy, VSX destructive: issues as a single operation
- Extended-agen: issues as a single load or store from a *paired ISQ* entry
- Load-compare-immediate: issues as a normal load but with a CR result. Note that a CR result is available as soon as four cycles.

Table 19-6. Supported Fusion Sequences (Sheet 1 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
addadd	add rx,ra,rb	add rt,rx,rc		
addadd	vaddudm vx,va,vb	vaddudm vt,vx,vc		
addadd_rb	add rx,ra,rb	add rt,rc,rx		
addadd_rb	vaddudm vx,va,vb	vaddudm vt,vc,vx		
addlogical	subf rx,ra,rc	or rt,rx,rb		
addlogical	add rx,ra,rc	and rt,rx,rb		
addlogical	subf rx,ra,rc	and rt,rx,rb		



*Table 19-6. Supported Fusion Sequences (Sheet 2 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
addlogical	add rx,ra,rc	nor rt,rx,rb		
addlogical	subf rx,ra,rc	nor rt,rx,rb		
addlogical	add rx,ra,rc	nand rt,rx,rb		
addlogical	subf rx,ra,rc	nand rt,rx,rb		
addlogical	add rx,ra,rc	or rt,rx,rb		
addlogical_rb	add rx,ra,rc	or rt,rb,rx		
addlogical_rb	subf rx,ra,rc	or rt,rb,rx		
addlogical_rb	add rx,ra,rc	and rt,rb,rx		
addlogical_rb	subf rx,ra,rc	and rt,rb,rx		
addlogical_rb	add rx,ra,rc	nor rt,rb,rx		
addlogical_rb	subf rx,ra,rc	nor rt,rb,rx		
addlogical_rb	add rx,ra,rc	nand rt,rb,rx		
addlogical_rb	subf rx,ra,rc	nand rt,rb,rx		
loadcmpi	lbz rx,d(ra)	cmpli 0,1,rx,0		
loadcmpi	lbz rx,d(ra)	cmpli 0,1,rx,1		
loadcmpi	lbz rx,d(ra)	cmpli 0,1,rx,-1		
loadcmpi	lbzx rx,ra,rb	cmpli 0,1,rx,0		
loadcmpi	lbzx rx,ra,rb	cmpli 0,1,rx,1		
loadcmpi	lbzx rx,ra,rb	cmpli 0,1,rx,-1		
loadcmpi	lhz rx,d(ra)	cmpli 0,1,rx,0		
loadcmpi	lha rx,d(ra)	cmpli 0,L,rx,0		
loadcmpi	lhz rx,d(ra)	cmpli 0,1,rx,1		
loadcmpi	lha rx,d(ra)	cmpli 0,L,rx,1		
loadcmpi	lhz rx,d(ra)	cmpli 0,1,rx,-1		
loadcmpi	lha rx,d(ra)	cmpli 0,L,rx,-1		
loadcmpi	lhzx rx,ra,rb	cmpli 0,1,rx,0		
loadcmpi	lhax rx,ra,rb	cmpli 0,L,rx,0		
loadcmpi	lhzx rx,ra,rb	cmpli 0,1,rx,1		
loadcmpi	lhax rx,ra,rb	cmpli 0,L,rx,1		
loadcmpi	lhzx rx,ra,rb	cmpli 0,1,rx,-1		
loadcmpi	lhax rx,ra,rb	cmpli 0,L,rx,-1		
loadcmpi	lwz rx,d(ra)	cmpli 0,1,rx,0		
loadcmpi	lwa rx,d(ra)	cmpli 0,L,rx,0		
loadcmpi	lwz rx,d(ra)	cmpli 0,1,rx,1		
loadcmpi	lwa rx,d(ra)	cmpli 0,L,rx,1		
loadcmpi	lwz rx,d(ra)	cmpli 0,1,rx,-1		
loadcmpi	lwa rx,d(ra)	cmpli 0,L,rx,-1		



Table 19-6. Supported Fusion Sequences (Sheet 3 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
loadcmpi	lwzx rx,ra,rb	cmpi 0,1,rx,0		
loadcmpi	lwax rx,ra,rb	cmpi 0,L,rx,0		
loadcmpi	lwzx rx,ra,rb	cmpi 0,1,rx,1		
loadcmpi	lwax rx,ra,rb	cmpi 0,L,rx,1		
loadcmpi	lwzx rx,ra,rb	cmpi 0,1,rx,-1		
loadcmpi	lwax rx,ra,rb	cmpi 0,L,rx,-1		
loadcmpi	ld rx,d(ra)	cmpi 0,1,rx,0		
loadcmpi	ld rx,d(ra)	cmpi 0,1,rx,1		
loadcmpi	ld rx,d(ra)	cmpi 0,1,rx,-1		
loadcmpi	ldx rx,ra,rb	cmpi 0,1,rx,0		
loadcmpi	ldx rx,ra,rb	cmpi 0,1,rx,1		
loadcmpi	ldx rx,ra,rb	cmpi 0,1,rx,-1		
loadcmpi	lbz rx,d(ra)	cmpli 0,L,rx,0		
loadcmpi	lbz rx,d(ra)	cmpli 0,L,rx,1		
loadcmpi	lbzx rx,ra,rb	cmpli 0,L,rx,0		
loadcmpi	lbzx rx,ra,rb	cmpli 0,L,rx,1		
loadcmpi	lhz rx,d(ra)	mpli 0,L,rx,0		
loadcmpi	lhz rx,d(ra)	mpli 0,L,rx,1		
loadcmpi	lhzx rx,ra,rb	mpli 0,L,rx,0		
loadcmpi	lhzx rx,ra,rb	mpli 0,L,rx,1		
loadcmpi	lwz rx,d(ra)	cmpli 0,L,rx,0		
loadcmpi	lwz rx,d(ra)	cmpli 0,L,rx,1		
loadcmpi	lwzx rx,ra,rb	cmpli 0,L,rx,0		
loadcmpi	lwzx rx,ra,rb	cmpli 0,L,rx,1		
loadcmpi	ld rx,d(ra)	cmpli 0,1,rx,0		
loadcmpi	ld rx,d(ra)	cmpli 0,1,rx,1		
loadcmpi	ldx rx,ra,rb	cmpli 0,1,rx,0		
loadcmpi	ldx rx,ra,rb	cmpli 0,1,rx,1		
loadload	lwz rt,ds(ra)	lwz ru,ds(ra)+4		Disabled.
loadload	ld rt,ds(ra)	ld ru,ds(ra)+8		Disabled.
loadload	lxv xt,ds(ra)	lxv xu,ds(ra)+16		Disabled.
loadload	lfd ft,ds(ra)	lfd fu,ds(ra)+8		Disabled.
loadload	lxsd xt,ds(ra)	lxsd xu,ds(ra)+8		Disabled.
loadload	lwz rt,ds(ra)	lwz ru,ds(ra)-4		Disabled.
loadload	ld rt,ds(ra)	ld ru,ds(ra)-8		Disabled.
logical	and rx,ra,rb	and rt,rx,rc		
logical	andc rx,ra,rb	and rt,rx,rc		



*Table 19-6. Supported Fusion Sequences (Sheet 4 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical	eqv rx,ra,rb	and rt,rx,rc		
logical	nand rx,ra,rb	and rt,rx,rc		
logical	nor rx,ra,rb	and rt,rx,rc		
logical	or rx,ra,rb	and rt,rx,rc		
logical	orc rx,rb,ra	and rt,rx,rc		
logical	xor rx,ra,rb	and rt,rx,rc		
logical	and rx,ra,rb	andc rt,rx,rc		
logical	andc rx,ra,rb	andc rt,rx,rc		
logical	eqv rx,ra,rb	andc rt,rx,rc		
logical	nand rx,ra,rb	andc rt,rx,rc		
logical	nor rx,ra,rb	andc rt,rx,rc		
logical	or rx,ra,rb	andc rt,rx,rc		
logical	orc rx,rb,ra	andc rt,rx,rc		
logical	xor rx,ra,rb	andc rt,rx,rc		
logical	and rx,ra,rb	eqv rt,rx,rc		
logical	andc rx,ra,rb	eqv rt,rx,rc		
logical	eqv rx,ra,rb	eqv rt,rx,rc		
logical	nand rx,ra,rb	eqv rt,rx,rc		
logical	nor rx,ra,rb	eqv rt,rx,rc		
logical	or rx,ra,rb	eqv rt,rx,rc		
logical	orc rx,rb,ra	eqv rt,rx,rc		
logical	xor rx,ra,rb	eqv rt,rx,rc		
logical	and rx,ra,rb	nand rt,rx,rc		
logical	andc rx,ra,rb	nand rt,rx,rc		
logical	eqv rx,ra,rb	nand rt,rx,rc		
logical	nand rx,ra,rb	nand rt,rx,rc		
logical	nor rx,ra,rb	nand rt,rx,rc		
logical	or rx,ra,rb	nand rt,rx,rc		
logical	orc rx,rb,ra	nand rt,rx,rc		
logical	xor rx,ra,rb	nand rt,rx,rc		
logical	and rx,ra,rb	nor rt,rx,rc		
logical	andc rx,ra,rb	nor rt,rx,rc		
logical	eqv rx,ra,rb	nor rt,rx,rc		
logical	nand rx,ra,rb	nor rt,rx,rc		
logical	nor rx,ra,rb	nor rt,rx,rc		
logical	or rx,ra,rb	nor rt,rx,rc		
logical	orc rx,rb,ra	nor rt,rx,rc		



Table 19-6. Supported Fusion Sequences (Sheet 5 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical	xor rx,ra,rb	nor rt,rx,rc		
logical	and rx,ra,rb	or rt,rx,rc		
logical	andc rx,ra,rb	or rt,rx,rc		
logical	eqv rx,ra,rb	or rt,rx,rc		
logical	nand rx,ra,rb	or rt,rx,rc		
logical	nor rx,ra,rb	or rt,rx,rc		
logical	or rx,ra,rb	or rt,rx,rc		
logical	orc rx,rb,ra	or rt,rx,rc		
logical	xor rx,ra,rb	or rt,rx,rc		
logical	and rx,ra,rb	orc rt,rx,rc		
logical	andc rx,ra,rb	orc rt,rx,rc		
logical	eqv rx,ra,rb	orc rt,rx,rc		
logical	nand rx,ra,rb	orc rt,rx,rc		
logical	nor rx,ra,rb	orc rt,rx,rc		
logical	or rx,ra,rb	orc rt,rx,rc		
logical	orc rx,rb,ra	orc rt,rx,rc		
logical	xor rx,ra,rb	orc rt,rx,rc		
logical	and rx,ra,rb	xor rt,rx,rc		
logical	andc rx,ra,rb	xor rt,rx,rc		
logical	eqv rx,ra,rb	xor rt,rx,rc		
logical	nand rx,ra,rb	xor rt,rx,rc		
logical	nor rx,ra,rb	xor rt,rx,rc		
logical	or rx,ra,rb	xor rt,rx,rc		
logical	orc rx,rb,ra	xor rt,rx,rc		
logical	xor rx,ra,rb	xor rt,rx,rc		
logical	vand vx,va,vb	vand vt,vx,vc		
logical	vand vx,va,vb	vandc vt,vx,vc		
logical	vand vx,va,vb	veqv vt,vx,vc		
logical	vand vx,va,vb	vnan dvt,vc,vx		
logical	vand vx,va,vb	vnor vt,vx,vc		
logical	vand vx,va,vb	vor vt,vc,vx		
logical	vand vx,va,vb	vorc vt,vx,vc		
logical	vand vx,va,vb	vxor vt,vc,vx		
logical	vandc vx,va,vb	vand vt,vx,vc		
logical	vandc vx,va,vb	vandc vt,vx,vc		
logical	vandc vx,va,vb	veqv vt,vx,vc		
logical	vandc vx,va,vb	vnan dvt,vc,vx		



*Table 19-6. Supported Fusion Sequences (Sheet 6 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical	vandc vx,va,vb	vnor vt,vx,vc		
logical	vandc vx,va,vb	vor vt,vc,vx		
logical	vandc vx,va,vb	vorc vt,vx,vc		
logical	vandc vx,va,vb	vxor vt,vc,vx		
logical	veqv vx,va,vb	vand vt,vx,vc		
logical	veqv vx,va,vb	vandc vt,vx,vc		
logical	veqv vx,va,vb	veqv vt,vx,vc		
logical	veqv vx,va,vb	vnand vt,vc,vx		
logical	veqv vx,va,vb	vnor vt,vx,vc		
logical	veqv vx,va,vb	vor vt,vc,vx		
logical	veqv vx,va,vb	vorc vt,vx,vc		
logical	veqv vx,va,vb	vxor vt,vc,vx		
logical	vnand vx,va,vb	vand vt,vx,vc		
logical	vnand vx,va,vb	vandc vt,vx,vc		
logical	vnand vx,va,vb	veqv vt,vx,vc		
logical	vnand vx,va,vb	vnand vt,vc,vx		
logical	vnand vx,va,vb	vnor vt,vx,vc		
logical	vnand vx,va,vb	vor vt,vc,vx		
logical	vnand vx,va,vb	vorc vt,vx,vc		
logical	vnand vx,va,vb	vxor vt,vc,vx		
logical	vnor vx,va,vb	vand vt,vx,vc		
logical	vnor vx,va,vb	vandc vt,vx,vc		
logical	vnor vx,va,vb	veqv vt,vx,vc		
logical	vnor vx,va,vb	vnand vt,vc,vx		
logical	vnor vx,va,vb	vnor vt,vx,vc		
logical	vnor vx,va,vb	vor vt,vc,vx		
logical	vnor vx,va,vb	vorc vt,vx,vc		
logical	vnor vx,va,vb	vxor vt,vc,vx		
logical	vor vx,va,vb	vand vt,vx,vc		
logical	vor vx,va,vb	veqv vt,vx,vc		
logical	vor vx,va,vb	vnand vt,vc,vx		
logical	vor vx,va,vb	vnor vt,vx,vc		
logical	vor vx,va,vb	vor vt,vc,vx		
logical	vor vx,va,vb	vorc vt,vx,vc		
logical	vor vx,va,vb	vxor vt,vc,vx		
logical	vorc vx,va,vb	vand vt,vx,vc		

Table 19-6. Supported Fusion Sequences (Sheet 7 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical	vorc vx,va,vb	vandc vt,vx,vc		
logical	vorc vx,va,vb	veqv vt,vx,vc		
logical	vorc vx,va,vb	vnand vt,vc,vx		
logical	vorc vx,va,vb	vnor vt,vx,vc		
logical	vorc vx,va,vb	vor vt,vc,vx		
logical	vorc vx,va,vb	vorc vt,vx,vc		
logical	vorc vx,va,vb	vxor vt,vc,vx		
logical	vxor vx,va,vb	vand vt,vx,vc		
logical	vxor vx,va,vb	vandc vt,vx,vc		
logical	vxor vx,va,vb	veqv vt,vx,vc		
logical	vxor vx,va,vb	vnand vt,vc,vx		
logical	vxor vx,va,vb	vnor vt,vx,vc		
logical	vxor vx,va,vb	vor vt,vc,vx		
logical	vxor vx,va,vb	vorc vt,vx,vc		
logical	vxor vx,va,vb	vxor vt,vc,vx		
logical_rb	and rx,ra,rb	and rt,rc,rx		
logical_rb	andc rx,ra,rb	and rt,rc,rx		
logical_rb	eqv rx,ra,rb	and rt,rc,rx		
logical_rb	nand rx,ra,rb	and rt,rc,rx		
logical_rb	nor rx,ra,rb	and rt,rc,rx		
logical_rb	or rx,ra,rb	and rt,rc,rx		
logical_rb	orc rx,rb,ra	and rt,rc,rx		
logical_rb	xor rx,ra,rb	and rt,rc,rx		
logical_rb	and rx,ra,rb	andc rt,rc,rx		
logical_rb	andc rx,ra,rb	andc rt,rc,rx		
logical_rb	eqv rx,ra,rb	andc rt,rc,rx		
logical_rb	nand rx,ra,rb	andc rt,rc,rx		
logical_rb	nor rx,ra,rb	andc rt,rc,rx		
logical_rb	or rx,ra,rb	andc rt,rc,rx		
logical_rb	orc rx,rb,ra	andc rt,rc,rx		
logical_rb	xor rx,ra,rb	andc rt,rc,rx		
logical_rb	and rx,ra,rb	eqv rt,rc,rx		
logical_rb	andc rx,ra,rb	eqv rt,rc,rx		
logical_rb	eqv rx,ra,rb	eqv rt,rc,rx		
logical_rb	nand rx,ra,rb	eqv rt,rc,rx		
logical_rb	nor rx,ra,rb	eqv rt,rc,rx		
logical_rb	or rx,ra,rb	eqv rt,rc,rx		



*Table 19-6. Supported Fusion Sequences (Sheet 8 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical_rb	orc rx,rb,ra	eqv rt,rc,rx		
logical_rb	xor rx,ra,rb	eqv rt,rc,rx		
logical_rb	and rx,ra,rb	nand rt,rc,rx		
logical_rb	andc rx,ra,rb	nand rt,rc,rx		
logical_rb	eqv rx,ra,rb	nand rt,rc,rx		
logical_rb	nand rx,ra,rb	nand rt,rc,rx		
logical_rb	nor rx,ra,rb	nand rt,rc,rx		
logical_rb	or rx,ra,rb	nand rt,rc,rx		
logical_rb	orc rx,rb,ra	nand rt,rc,rx		
logical_rb	xor rx,ra,rb	nand rt,rc,rx		
logical_rb	and rx,ra,rb	nor rt,rc,rx		
logical_rb	andc rx,ra,rb	nor rt,rc,rx		
logical_rb	eqv rx,ra,rb	nor rt,rc,rx		
logical_rb	nand rx,ra,rb	nor rt,rc,rx		
logical_rb	nor rx,ra,rb	nor rt,rc,rx		
logical_rb	or rx,ra,rb	nor rt,rc,rx		
logical_rb	orc rx,rb,ra	nor rt,rc,rx		
logical_rb	xor rx,ra,rb	nor rt,rc,rx		
logical_rb	and rx,ra,rb	or rt,rc,rx		
logical_rb	andc rx,ra,rb	or rt,rc,rx		
logical_rb	eqv rx,ra,rb	or rt,rc,rx		
logical_rb	nand rx,ra,rb	or rt,rc,rx		
logical_rb	nor rx,ra,rb	or rt,rc,rx		
logical_rb	or rx,ra,rb	or rt,rc,rx		
logical_rb	orc rx,rb,ra	or rt,rc,rx		
logical_rb	xor rx,ra,rb	or rt,rc,rx		
logical_rb	and rx,ra,rb	orc rt,rc,rx		
logical_rb	andc rx,ra,rb	orc rt,rc,rx		
logical_rb	eqv rx,ra,rb	orc rt,rc,rx		
logical_rb	nand rx,ra,rb	orc rt,rc,rx		
logical_rb	nor rx,ra,rb	orc rt,rc,rx		
logical_rb	or rx,ra,rb	orc rt,rc,rx		
logical_rb	orc rx,rb,ra	orc rt,rc,rx		
logical_rb	xor rx,ra,rb	orc rt,rc,rx		
logical_rb	and rx,ra,rb	xor rt,rc,rx		
logical_rb	andc rx,ra,rb	xor rt,rc,rx		
logical_rb	eqv rx,ra,rb	xor rt,rc,rx		



Table 19-6. Supported Fusion Sequences (Sheet 9 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical_rb	nand rx,ra,rb	xor rt,rc,rx		
logical_rb	nor rx,ra,rb	xor rt,rc,rx		
logical_rb	or rx,ra,rb	xor rt,rc,rx		
logical_rb	orc rx,rb,ra	xor rt,rc,rx		
logical_rb	xor rx,ra,rb	xor rt,rc,rx		
logical_rb	vand vx,va,vb	vand vt,vc,vx		
logical_rb	vand vx,va,vb	vandc vt,vc,vx		
logical_rb	vand vx,va,vb	veqv vt,vc,vx		
logical_rb	vand vx,va,vb	vnand vt,vc,vx		
logical_rb	vand vx,va,vb	vnor vt,vc,vx		
logical_rb	vand vx,va,vb	vor vt,vc,vx		
logical_rb	vand vx,va,vb	vorc vt,vc,vx		
logical_rb	vand vx,va,vb	vxor vt,vc,vx		
logical_rb	vandc vx,va,vb	vand vt,vc,vx		
logical_rb	vandc vx,va,vb	vandc vt,vc,vx		
logical_rb	vandc vx,va,vb	veqv vt,vc,vx		
logical_rb	vandc vx,va,vb	vnand vt,vc,vx		
logical_rb	vandc vx,va,vb	vnor vt,vc,vx		
logical_rb	vandc vx,va,vb	vor vt,vc,vx		
logical_rb	vandc vx,va,vb	vorc vt,vc,vx		
logical_rb	vandc vx,va,vb	vxor vt,vc,vx		
logical_rb	veqv vx,va,vb	vand vt,vc,vx		
logical_rb	veqv vx,va,vb	vandc vt,vc,vx		
logical_rb	veqv vx,va,vb	veqv vt,vc,vx		
logical_rb	veqv vx,va,vb	vnand vt,vc,vx		
logical_rb	veqv vx,va,vb	vnor vt,vc,vx		
logical_rb	veqv vx,va,vb	vor vt,vc,vx		
logical_rb	veqv vx,va,vb	vorc vt,vc,vx		
logical_rb	veqv vx,va,vb	vxor vt,vc,vx		
logical_rb	vnand vx,va,vb	vand vt,vc,vx		
logical_rb	vnand vx,va,vb	vandc vt,vc,vx		
logical_rb	vnand vx,va,vb	veqv vt,vc,vx		
logical_rb	vnand vx,va,vb	vnand vt,vc,vx		
logical_rb	vnand vx,va,vb	vnor vt,vc,vx		
logical_rb	vnand vx,va,vb	vor vt,vc,vx		
logical_rb	vnand vx,va,vb	vorc vt,vc,vx		
logical_rb	vnand vx,va,vb	vxor vt,vc,vx		



*Table 19-6. Supported Fusion Sequences (Sheet 10 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logical_rb	vnor vx,va,vb	vand vt,vc,vx		
logical_rb	vnor vx,va,vb	vandc vt,vc,vx		
logical_rb	vnor vx,va,vb	veqv vt,vc,vx		
logical_rb	vnor vx,va,vb	vnand vt,vc,vx		
logical_rb	vnor vx,va,vb	vnor vt,vc,vx		
logical_rb	vnor vx,va,vb	vor vt,vc,vx		
logical_rb	vnor vx,va,vb	vorc vt,vc,vx		
logical_rb	vnor vx,va,vb	vxor vt,vc,vx		
logical_rb	vor vx,va,vb	vand vt,vc,vx		
logical_rb	vor vx,va,vb	vandc vt,vc,vx		
logical_rb	vor vx,va,vb	veqv vt,vc,vx		
logical_rb	vor vx,va,vb	vnand vt,vc,vx		
logical_rb	vor vx,va,vb	vnor vt,vc,vx		
logical_rb	vor vx,va,vb	vor vt,vc,vx		
logical_rb	vor vx,va,vb	vorc vt,vc,vx		
logical_rb	vor vx,va,vb	vxor vt,vc,vx		
logical_rb	vorc vx,va,vb	vand vt,vc,vx		
logical_rb	vorc vx,va,vb	vandc vt,vc,vx		
logical_rb	vorc vx,va,vb	veqv vt,vc,vx		
logical_rb	vorc vx,va,vb	vnand vt,vc,vx		
logical_rb	vorc vx,va,vb	vnor vt,vc,vx		
logical_rb	vorc vx,va,vb	vor vt,vc,vx		
logical_rb	vorc vx,va,vb	vorc vt,vc,vx		
logical_rb	vorc vx,va,vb	vxor vt,vc,vx		
logical_rb	vxor vx,va,vb	vand vt,vc,vx		
logical_rb	vxor vx,va,vb	vandc vt,vc,vx		
logical_rb	vxor vx,va,vb	veqv vt,vc,vx		
logical_rb	vxor vx,va,vb	vnand vt,vc,vx		
logical_rb	vxor vx,va,vb	vnor vt,vc,vx		
logical_rb	vxor vx,va,vb	vor vt,vc,vx		
logical_rb	vxor vx,va,vb	vorc vt,vc,vx		
logical_rb	vxor vx,va,vb	vxor vt,vc,vx		
logicaladd	and rx,ra,rb	add rt,rx,rc		
logicaladd	andc rx,ra,rb	add rt,rx,rc		
logicaladd	eqv rx,ra,rb	add rt,rx,rc		
logicaladd	nand rx,ra,rb	add rt,rx,rc		
logicaladd	nor rx,ra,rb	add rt,rx,rc		



Table 19-6. Supported Fusion Sequences (Sheet 11 of 12)

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
logicaladd	or rx,ra,rb	add rt,rx,rc		
logicaladd	orc rx,rb,ra	add rt,rx,rc		
logicaladd	xor rx,ra,rb	add rt,rx,rc		
logicaladd	and rx,ra,rb	subf rt,rx,rc		
logicaladd	andc rx,ra,rb	subf rt,rx,rc		
logicaladd	eqv rx,ra,rb	subf rt,rx,rc		
logicaladd	nand rx,ra,rb	subf rt,rx,rc		
logicaladd	nor rx,ra,rb	subf rt,rx,rc		
logicaladd	or rx,ra,rb	subf rt,rx,rc		
logicaladd	orc rx,rb,ra	subf rt,rx,rc		
logicaladd	xor rx,ra,rb	subf rt,rx,rc		
logicaladd_rb	and rx,ra,rb	add rt,rc,rx		
logicaladd_rb	andc rx,ra,rb	add rt,rc,rx		
logicaladd_rb	eqv rx,ra,rb	add rt,rc,rx		
logicaladd_rb	nand rx,ra,rb	add rt,rc,rx		
logicaladd_rb	nor rx,ra,rb	add rt,rc,rx		
logicaladd_rb	or rx,ra,rb	add rt,rc,rx		
logicaladd_rb	orc rx,rb,ra	add rt,rc,rx		
logicaladd_rb	xor rx,ra,rb	add rt,rc,rx		
logicaladd_rb	and rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	andc rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	eqv rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	nand rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	nor rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	or rx,ra,rb	subf rt,rc,rx		
logicaladd_rb	orc rx,rb,ra	subf rt,rc,rx		
logicaladd_rb	xor rx,ra,rb	subf rt,rc,rx		
multiplyadd	mulld rx,ra,rb	add rx,rx,rc	Y	
multiplyadd_rb	mulld rx,ra,rb	add rx,rc,rx	Y	
sha3assist	rldicl rx,rs,1,0	xor rt,rx,rb		
sha3assist	rldicr rx,rs,1,63	xor rt,rx,rb		
shiftadd	sldi rx,rc,3	add rt,rx,ra		
shiftadd	sldi rx,rc,3	subf rt,rx,ra		
shiftadd	sldi rx,rc,6	add rt,rx,ra		
shiftadd	sldi rx,rc,6	subf rt,rx,ra		
shiftadd_rb	sldi rx,rc,3	subf rt,ra,rx		
shiftadd_rb	sldi rx,rc,3	add rt,ra,rx		



*Table 19-6. Supported Fusion Sequences (Sheet 12 of 12)*

Category	Mnemonic1	Mnemonic2	Destructive-Only	Note / Restriction
shiftadd_rb	sldi rx,rc,6	subf rt,ra,rx		
shiftadd_rb	sldi rx,rc,6	add rt,ra,rx		
storestore	stw rt,ds(ra)	stw ru,ds(ra)+4		
storestore	std rt,ds(ra)	std ru,ds(ra)+8		
storestore	stfd ft,ds(ra)	stfd fu,ds(ra)+8		
storestore	stxsd xt,ds(ra)	stxsd xu,ds(ra)+8		
storestore	stw rt,ds(ra)	stw ru,ds(ra)-4		
storestore	std rt,ds(ra)	std ru,ds(ra)-8		
wideimmediate	li rx,si	addis rx,rx,SI	Y	rx > 0; si ≥ 2
wideimmediate	addi rx,ra,si	addis rx,rx,SI	Y	ra > 0; rx > 0; si ≥ 0
wideimmediate	addis rx,ra,si	addi rx,rx,SI	Y	ra > 0; rx > 0; si ≥ 0
wideimmediate	lis rx,si	ori rx,rx,UI	Y	
wideimmediate	ori rx,ra,ui	oris rx,rx,UI	Y	
wideimmediate	oris rx,ra,ui	ori rx,rx,UI	Y	
wideimmediate	xori rx,ra,ui	xoris rx,rx,UI	Y	
wideimmediate	xoris rx,ra,ui	xori rx,rx,UI	Y	
wideimmediate	addis rx,ra,si	lbz rx,d(rx)	Y	ra(3) = ra(4) = si(0), ra > 0
wideimmediate	addis rx,ra,si	lhz rx,d(rx)	Y	ra(3) = ra(4) = si(0), ra > 0
wideimmediate	addis rx,ra,si	lwz rx,d(rx)	Y	ra(3) = ra(4) = si(0), ra > 0
wideimmediate	addis rx,ra,si	ld rx,d(rx)	Y	ra(3) = ra(4) = si(0), ra > 0
zeromove	mtctr	bctr		
zeromove	mtctr	bcctrl		
zeromove	mtlr	bclr		
zeromove	mtlr	bclrl		

## 19.1.6 IOP Execution

### 19.1.6.1 Execution Pipeline Hazards

The following sticky bits are handled with a special reordering logic to facilitate out-of-order execution of exception setters and calculate the architected value at completion:

- From the XER: SO
- From the FPSCR: FX, OX, UX, ZX, XX, VXSAN, VXISI, VXIDI, VXZDZ, VXIMZ, VXVC, VXSOFT, VXSQRT, VXCVI
- From the VSCR: SAT

## XER-SO

The processor core speculates that the XER[SO] bit will not change during execution of operations; for example, that the SO bit is rarely cleared. When a change to SO is detected, the instructions younger than the SO-changing instruction are flushed from the pipeline once the SO changer becomes NTC and are re-fetched for execution.

## FPSCR

An FPSCR scoreboard is implemented for instructions that write to an FPSCR exception enable bit or can clear an FP exception, such that only one such instruction is supported between dispatch and completion per thread. The dispatch stalls if an FPSCR-scoreboarded instruction appears before the previous one completes.

The following instructions are subject to the FPSCR scoreboard:

```
"111111-----0----0010000110-" - mtfsfi[.] W = 0
"1111111-----1011000111-" - mtfsf[.] L = 1
"1111110-----0----1011000111-" - mtfsf[.] L = 0 W = 0
"111111-----0001000110-" - mtfsb0[.] all forms
"111111100000-----0000100110-" - mtfsb1[.] BT = 0 - FX
"11111110101-----0000100110-" - mtfsb1[.] BT = 21- VXSOFT
"11111111-----0000100110-" - mtfsb1[.] BT = 24..31
"111111-----00001----1001000111-" - mffsce
"111111-----0----0001000000-" - mcrfs BFA = 0,1,2,3
"111111-----101----0001000000-" - mcrfs BFA = 5
"111111-----1011----1001000111-" - mffscrn[i]
```

**Note:** In general, lightweight move-to FPSCR instructions that only modify the control field of the FPSCR including **mffscdrn**, **mffscdrni**, **mffsl**, as well as **mtfsb1** that only updates the control field, does not cause a flush and can always execute out-of-order. Exceptions to this optimization include **mffscrn**, **mffscrni**, and **mtfsb0** (these are intended to be corrected in a future processor). For **mffscrn[i]**, the scoreboardding penalty can be avoided by reading the current rounding mode via **mffsl** to see if it already matches the desired mode and suppressing the write if the two match.

## VSCR-SAT

A VSCR scoreboard is implemented to allow only one move-to-VSCR instruction between dispatch and completion per thread. The dispatch stalls if a move-to-VSCR instruction appears before the previous one completes.



### 19.1.6.2 FPR Result Forwarding Restrictions

The internal dataflow was optimized for common cases of data type exchange. Certain internal data type representations are not eligible for forwarding. If forwarding occurs because of an instruction dependency, a flush is generated. Avoid the following dependent scenarios:

- Convert float-to-integer feeding a floating-point input
- Floating-point output feeding convert integer-to-float
- 64-bit floating-point result feeding a 32-bit floating-point input operand; for example, **xvadddp** feeding **xvcvspuxds**
- 32-bit floating-point result feeding a 64-bit floating-point input operand; for example, **xscvdpsp** feeding **fadd**

### 19.1.7 Load/Store Processing

There is no address slicing in the LSU. Instead, ports are associated with each super-slice:

- Two load ports (up to 32 bytes each); one per SS.
- Two store issue ports (up to 32 bytes each agen); one per SS.
  - Data is received as two beats of up to 16 bytes for 32-byte stores per SS (from the VSU).
    - The implication is that for SMT4 mode, the maximum sustained store rate per thread is 16 bytes per cycle for an individual thread.
- One store drain up to 32 bytes, from up to two consecutive (age-ordered) store queue entries.

The load and store pipeline flow is managed in the LSU as it is in the POWER9 core. That is, once the operations are issued to the LSU, the operations remain in the LSU pipeline and are managed locally to handle hazards.

#### 19.1.7.1 Tracking Load and Store Ordering

Load IOPs are tracked by a load reorder queue (LRQ) and store IOPs are tracked by a store reorder queue (SRQ) to maintain correct architectural storage ordering and cache coherency.

New operations issued to load and store pipes from the ISQ are written to the LLQ (or the LRQ extension for re-launch) for loads and SRQ for stores. The LLQ and SRQ manage the pipeline including giving priority for re-cycled operations, while new operations are pipelined through and re-issued. This includes a “+1 cycle” pipelined delay for a load collision (dependent operations must recycle, but the original operation issued from the ISQ can proceed after only one delay clock).

Load latency:

- 4 cycles to FX/PM/AGEN/ST-DATA pipes
- 5 cycles to other (for example, DP and others)

Unaligned loads: cross 128-byte cache line or for 32-byte load, if it is not 4-byte aligned.

- +6 cycle load recycle (second pipe pass)
  - Ordered left-to-right resolution
    - For example, if left = miss and right = hit, then latency adder applies after miss

- Hazards on the left extend the right, as do hazards on the right
- Prefetch spawned for right side immediately to cover left = miss, right = miss latency

Unaligned stores: cross 128-byte cache line

- +8 cycle recycle

Data cache: 32 KB, 8-way

- No read/read bank conflicts
- Each load pipe accesses 32 bytes; word granularity
- Banking: 8 banks

LRQ: 128 entries:

- ST: 64 entries
- SMT2:  $64 \times 2$  entries
- SMT4:  $32 \times 4$  entries
- Entries track up to 32 bytes

SRQ: 80 entries:

- ST: 40 entries
- SMT2:  $40 \times 2$  entries
- SMT4:  $20 \times 4$  entries
- Entries are 16 bytes each; therefore, 32-byte stores take two entries.

#### **19.1.7.2 L1 Data Cache**

The L1 D-cache is the first level of cache available to load and store operations. It is 32 KB and organized as 8-way set-associative with 128-byte cache lines. The L1 D-cache is reloaded by the L2 cache at a rate of up to 64 bytes per cycle. The L2 reload bus is dynamically shared with the other core of a core-pair (when active).

The L1 D-cache has eight banks. Cache writes and reads to different banks occur simultaneously from each execution pipe.

When load and store operations execute, they use a set prediction directory (SETP) to reduce cache access latency. The SETP provides a cache-hit and set selection indication. In parallel to accessing the SETP, the L1 D-cache directory and ERAT are also accessed and are used to confirm the SETP cache-hit and set-selection.

When a load encounters an SETP hit and ERAT miss, a translation is performed to confirm the SETP prediction. If the translation results in a TLB miss or an SLB miss for a hashed-page table (HPT) translation mode, the load is flushed from the pipeline and is re-fetched while the translation process proceeds.

When a load encounters an L1 D-cache miss, the following occurs:

- A request is made to the L2 to retrieve the cache line.
- A load miss queue (LMQ) entry is allocated for the cache line or the request is merged into an existing LMQ entry for a matching cache line.



- An LRQ entry associated with the load waits for the cache line to return from the cache hierarchy before waking the load back up for re-issue into the execution pipeline.

The LMQ holds 12 cache-line miss requests per core.

If there is already an LMQ entry active for the same cache line, the load becomes dependent on the same LMQ entry for re-execution, with no limit on the number of loads that are dependent on a particular LMQ entry. Otherwise, if there are no remaining LMQ entries, the load re-arbitrates for an LMQ entry once an entry becomes available.

The L1 D-cache is store-through. Stores that miss the cache write into the L2 cache after they are complete, but do not allocate an entry in the L1 D-cache. If a store hits the SETP/L1 D-cache, it writes the cache once it is complete as it is being sent to the L2 cache.

A write into the L1 D-cache by a store makes the line private to the thread of the store performing the write. If any other thread requests to access the line that is marked as private, the line is evicted from the L1 D-cache. If the request from the other thread was a load, the line is brought back into the cache in a non-private state available to all threads.

#### **19.1.7.3 Effective-to-Real-Address-Translation (ERAT)**

The instruction/data shared ERAT has 64 entries with a lower latency (7.5 cycles) to TLB; and a pipelined TLB offset; and a smaller effective ERAT versus the POWER9 core which improves the overall performance.

The ERAT is a 64-entry, fully set-associate CAM for low-latency, effective-to-real-address-translation. Conceptually, it is the “Level 0” translation cache; in that it is the smallest, fastest translation structure maintained by the MMU. While other MMU structures provide caching at different virtualization layers, the ERAT contains the flattened translation for loads, stores, and instruction fetches. An ERAT hit requires a match on various EA bits, as well as specific MSR bits (HV, PR, S, US, DR/IR) that are used to determine the translation context. The ERAT supports four page sizes: 16 MB, 2 MB, 64 KB, and 4 KB; each of which are built into the CAM compare logic within the array itself.

Access to the ERAT is granted by an arbiter controlled by the TIQ; such that only one load, store, or fetch can access the ERAT in a given cycle. On a hit, the resulting real address (RA) is broadcast to the L1 RA directory, as well as the L2 cache. If an exception is found, the ERAT forces a miss to cancel an L2 data request and generate a trouble code to send to the MMU.

When a translation misses the ERAT, it is sent to the TLB pipeline and the TLB reloads the ERAT with the valid translation. The victim ERAT entry is selected using a pseudo-LRU replacement algorithm. When reloading the ERAT, one cycle will be stolen in the miss pipeline, which blocks other translations from accessing the ERAT until the reload is complete. (The miss pipeline refers to the EA cache miss pipeline that is used to translate, check the RA directory, and send out L2 data requests.)

The ERAT also has an invalidation interface built into the array itself. The flash invalidate can do a precise EA compare for various page sizes or a class-based invalidation. The **sibia** instruction does the class-based or MSR(IR/DR) based invalidation, and the **tlbie(I)** instruction does the precise compare at the ERAT. Although most of the ERAT functionality is invisible to software, class-based invalidation is used by system software to optimize performance by preserving kernel pages in the ERAT. Additionally, the ERAT also supports a thread-based invalidate and an invalidation based on the MSR(IR/DR) bit.

#### 19.1.7.4 Translation Look-Aside Buffer

- TLB: 4096 entries of child TLB or L3 page-walk cache (PWC) for radix.
- 512 entries of parent TLB, each covering a 1 GB range.
- All child entries must maintain pointer to an active parent or they are evicted.

The TLB has 4096 entries of child TLB or L3 page-walk cache (PWC) for radix. It supports translation for ERAT misses. The TLB uses a pseudo-LRU replacement algorithm.

Whereas in previous designs the TLB was a simple four-way set-associative array where any page size could occupy an entry, the Power10 TLB has a hierarchical design based on page size to increase not only TLB storage efficiency but also reduce ERAT miss latency. Additionally it allows for a fully pipelined TLB to increase translation throughput. With the addition of radix on the POWER9 processor, this throughput has become increasingly important. In this type of design, the structure exploits access locality of programs to reduce redundant storage of the high-order bits of the effective address by sharing these bits across many small pages. Sharing is achieved by storing small-page entries in a separate array than large-page entries; where the small-page entries contain reference pointers to the large TLB entry that covers a wider range of memory. This means that each entry in the large-page size TLB can be the parent of many small-page entries.

The parent TLB is a  $64 \times 8$ -way set-associative array, where each entry covers 1 GB of virtualized address space. The parent TLB can be either a 1 GB TLB entry for large page sizes or a parent entry to many small page child entries. In HPT, 16 GB PTEs are broken into multiple 1 GB TLB entries. In radix, the parent TLB also acts as an L2 PWC, because the parent entry compare is the same as an L2 page directory entry qualifier.

The child TLB is composed of two  $256 \times 8$ -way set-associative arrays. Radix stores the 4 KB, 64 KB, and 2 MB TLB entries such that the 2 MB compare is jointly used as an L3 PWC entry if the resulting page size of the translation is less than 2 MB. In HPT, the child TLB contains 4 KB, 64 KB, and 16 MB pages. Because the child TLB accepts only two page sizes per cycle, a two-pass lookup design is required. If the two page sizes on the first lookup do not hit in the TLB, then the TLB must be looked up again for the third page size.

If a translation request does not hit in the TLB, a tablewalk is initiated that loads the TLB and ERAT. Up to four outstanding tablewalks can be active at one time. The implementation allows tablewalks for speculative instructions but does not update the Ts or change (C) bit in a PTE entry unless the instruction is NTC when the PTE entry is found. The TLB is reloaded with the corresponding PTE entry even if the instruction that requested the translation is speculative.

If a store misses the TLB after missing translation, and the C bit is not on, then a second tablewalk is done after the iop is NTC, so that the C bit can be updated.

There are four 32-entry SLBs, one per thread, that are accessed in parallel with the TLB during HPT mode. All entries of the SLB are architected and can be loaded by the software.



#### 19.1.7.5 Store Forwarding

Store forwarding for load/store overlap:

- “zero cycle” latency adder
  - Initiate based on EA 40:63 match; if not full match, reject until drained
- No background data. The store must be “fully contained” in a single 16-byte store entry.
  - **Note:** if not fully contained and back-to-back in an instruction stream, then it is at risk of “flush-to-break pair” and avoid (for example, stb, ld to the same address).

As loads are issued to each load execution pipe (LD0, LD1) into each LS-slice pipeline, they check each SRQ entry for older stores on which they are dependent (that is, for stores with overlapping address ranges). A load-hit-store (LHS) is when an overlapping store is found. When a store with an overlapping EA(40:63) is found, the load is identified as a candidate for store forwarding with zero-cycle execution delay.

For store forwarding to occur, two additional requirements must be met after a load hit store (LHS) address overlap condition with a store is detected against the SRQ:

1. The youngest SRQ store entry for which an LHS address overlap was detected must contain all of the bytes required by the load.
2. The load must hit in the L1 effective address directory (EA Dir) against the same EA Dir entry that the SRQ entry hit against when the store was issued. This second condition ensures that the load and store match against each other with a full EA and full RA, making the store forwarding fully valid.

For the case where a load and store have two different EA(40:63) but have the same full RA, store forwarding will not be done and the load will reissue once the store with the same RA drains.

For the case where a load and a store have the same EA(40:63) but have a different full EA (and different full RA), load hit store will be falsely detected, and forwarding will not be allowed. For this case, the load will be reissued immediately and upon the reissue of the load, the false LHS detection will be suppressed for that store and allow the load to execute properly by missing that SRQ entry.

Store forwarding cannot take place if the load IOP or store IOP are on pages that are caching inhibited or guarded.

#### 19.1.7.6 Out-of-Order Load/Store Execution

##### Store-Hit-Load (SHL)

When a load overlaps in storage with an older store, but the load executes first; it must be flushed from the pipeline and re-executed. Out-of-order SHL avoidance is managed by hardware by two mechanisms. Both are supported by a load-hit-store (LHS) table, which tracks recent stores in instruction order. Loads query the LHS table to form a dependency on older stores.

The mechanisms are as follows:

- Proactive mechanism: A load matches the B,X or B,D fields (register and displacement) from the store.
- Reactive mechanism: An SHL creates an entry in the SHL table which stores the IOP distance from the store to the load. Loads query the SHL table at fetch time and create dependencies on stores in the LHS table with matching IOP distance. More than one distance cannot be handled.

Mixing prefixed and non-prefixed storage instructions can create store-hit-load performance problems. Consider this scenario, which was found to cause an SHL flush every 1,000 cycles:

```
pla r31, global
while
    std rX, r31(0)
    variable target call (aggravates SHL learning)
        func1
            save r31
            ...
            restore r31
        func2
            pld rY, global
            ...
```

Replacing the **std** by pc-relative **pstd** will substantially reduce the number of SHL flushes. Replacing the global variable by a static local variable will also accomplish that and eliminate the "**pla r31, global**" instruction. Randomizing register allocation more will also help improve performance.

#### *Load Penalty for Auto-Aligned VMX Loads that are not Naturally Aligned*

If an auto-aligned VMX load (**lvehx**, **lviewx**, **lvx**, **lvxl**) is not naturally aligned to its element size and has a cache-line address match with an entry in the store queue, the load takes an additional pass through the load execution pipe.

#### **19.1.7.7 Load-to-Use Latency**

Iops dependent on register results from older loads are able to issue with a nominal issue-to-issue latency of four core cycles.

**Note:** The effective issue latency for single-precision floating-point load IOPs increased because they are cracked instructions. Each instruction is cracked into a load IOP followed by a conversion IOP that takes an additional three cycles to execute before a dependent instruction can issue.

#### **19.1.7.8 Load/Store Throughput**

Each of the two load units is capable of executing a load of up to 32 bytes per cycle. The two store units have shared access to one 32-byte store drain write port of the L1 data cache, providing a combined store execution capability of 32 bytes per cycle.

After they complete, up to two consecutive store queue entries of a given thread can drain simultaneously. If the stores are adjacent in memory, they will be merged into one store request to be sent to the L1 and the L2 cache. This leads to a maximum drain rate of 32 bytes of store data from the STQ over an indefinitely sustained interval for the case where all stores are 16-byte (vector) sequential stores and up to four store instructions drained per cycle for the case where all stores are 8-byte sequential stores and the instructions were also fused.



#### 19.1.7.9 Load/Store Pipeline Hazards

The following conditions detected in the load/store pipeline result in a pipeline flush at or after the problem instruction:

- SETP hit but tag miss, see *Section 19.1.7.2 L1 Data Cache* on page 281.
- TLB miss after an SETP hit, ERAT miss, see *Section 19.1.7.2* on page 281.
- TLB miss followed by SETP/L1 D-cache miss, ERAT miss; invalidate or eviction in translation window.
- Load-hit-store flush: RA does not match, but forwarding was selected, see *Section 19.1.7.5 Store Forwarding* on page 284.
- Store-hit-load flush: Older store executes after younger load to same address, see *Section 19.1.7.6 Out-of-Order Load/Store Execution* on page 284.
- Cache inhibited load ( $I = '1'$ ) detected in PTE entry (not explicit cache-inhibited instruction).
- Out-of-order **lax** detected for same thread
- Snoop or store from the other thread, invalidates the younger load data while the older load in the pipeline is flushed.
- Snoop or store from other thread invalidates part of load-quad
- Snoop or store from other thread invalidates load while **sync** is pending
- Snoop or store from other thread invalidates load while **tend** is pending
- TLBIE snoop response expedited and flush impacted stores and load misses
- ECC: UE on data from memory

#### 19.1.7.10 D-Cache Misses

Load, store, and data-cache misses all go through single point of arbitration: “TIQ”. TIQ is the access point to address translation. All loads and stores through the TIQ take an extra 5 - 6 cycles through the pipe and take a second pipe pass.

However, four cycles after a given load or store data-cache miss is sent through the TIQ to allocate a new cache line, subsequent loads and stores to the same cache line will no longer data-cache miss and be routed to the TIQ as the new line is seen as allocated in the L1 directory. For the case that the subsequent operation is a load, if the data has not yet been returned from the L2 cache for the newly allocated line, the load is put to sleep in the LRQ until the required cache line data is returned to the L1 cache.

Load misses are tracked in a 12-entry LMQ:

- Infinite merging per entry (like the POWER9 core)
- Load CDF wakeup hits the L1 cache in 12 cycles; other (merged loads) hit in 14 cycles.

Store misses allocate the data cache but do not write data because of the EA cache design. All store misses therefore must go through the TIQ to allocate. After draining, the store entry is returned to the LRQ for re-allocation. This action puts temporary pressure on the L1 cache, creating overhead for store-miss streams.



#### **19.1.7.11 Store Drain and Merge**

The store pipe to the L2 cache supports up to 32 bytes per cycle. The SRQ merges two consecutive SRQ entries of up to 16 bytes each to form a 32-byte store on the L2 interface. The merging function supports two stores within the same aligned 32 bytes.

Two stores that are consecutive in program order can potentially merge and drain to the L1 and the L2 cache in one cycle. Two stores are eligible for merging even if there are other non-store instructions in between them.

The additional conditions for two stores to merge are:

- The two stores are adjacent in memory (either ascending or descending).
- The two stores do not cross a 64-byte boundary in memory.
- The two stores, when taken together, do not access more than four consecutive doublewords of memory.

The consecutive stores can be any mix of size or store instruction op code as long as they are adjacent in memory.

#### **19.1.7.12 Store Pre-Allocate**

The prefetcher has a store pre-allocate mode to avoid the 5 - 6 cycle TIQ overhead and second pipe pass impact for store miss streams.

#### **19.1.7.13 Data Prefetch**

There are 16 prefetch streams supported per SMT4-core-resource. Each stream is tracked in a prefetch queue (PRQ) entry. Additionally, there are 48 entries in the L3 cache to support outstanding line fetches for each SMT4-core-resource.

The data prefetch engine can recognize sequentially increasing or decreasing accesses to adjacent cache lines and then request anticipated lines from more distant levels of the cache/memory hierarchy. The usefulness of these prefetches is reinforced, as repeated demand references are made along such a path or stream. The depth of prefetch is then increased until enough lines are being brought into L1, L2, and L3 that much or all of the load latency can be hidden. The most urgently needed lines are prefetched into the nearest cache levels.

##### *Data Prefetch Features per SMT4-Core-Resource*

- Sixteen active streams tracked
- EA based
- Software eDCBT/ST control support
- L1 and L3 prefetching
- Stride-N detection
- Stream direction detection
- Duplicate stream removal
- Finite stream length support
- Full LRU
- Full set of user controls (HID, LPCR, DSCR, SCAN)
- Fully pipelined, 4 prefetches in-flight in core
- 48 prefetch engines in the L3 cache
- Bandwidth sensitivity controls: adaptive prefetch



### Prefetch Controls and Interaction with Adaptive Mechanisms

The Power10 processor dynamically adapts prefetch behavior via mechanisms in the core, caches, and memory controller. These mechanisms balance prefetch resource consumption and aggressiveness across a wide variety of program types and data access patterns.

The DSCR settings can have a significant impact on program performance and provide a means to guide or even over-ride aspects of the dynamic prefetch control. The default prefetch behaviors can vary by system type and aggregate workload and therefore the programmer should adhere to the following guidelines:

1. DSCR[SNSE] (Stride-N) prefetching is designed to work in all SMT levels on the Power10 processor. It is recommended to enable stride-N by default by setting DSCR[SNSE] = '1' (hypervisor, supervisor, program).
2. The DSCR(URG) field is the primary indicator of how much a program benefits from prefetching on a data cache miss. The processor identifies a per thread effective urgency value using adaptive methods and the URG default value. Before setting the URG field, a program should be tested to ensure that it does or does not benefit significantly from speculative prefetching by altering the URG values under a range of workload conditions.
  - Programs that rely very heavily on speculative prefetching for performance can indicate a high urgency of 0b110. This value will enable the program to override some adaptive behaviors such as those that monitor aggregate benefit of prefetching versus volume of traffic.
  - In cases where the system has a well-defined use for a program and interference with other programs is not of high concern, the highest urgency of 0b111 can be indicated. This value can enable the program to override additional adaptive measures to aggressively generate prefetches.
  - Programs that do not benefit from general prefetching on a cache miss but still want to enable prefetching for high-confidence streams should set a low urgency value of 0b001. This is often a good alternative to disabling a prefetch completely.
3. Setting DSCR[SSE] (store stream enable) can be beneficial for optimizing store throughput, depending on the other demands of the workload and system configuration. Experimentation is suggested on target systems if store streams are to be enabled.
4. The DSCR[DPFD] is a primary indicator for how deep (how far ahead) streams should prefetch. The default behavior includes adaptive depth mechanisms; however, a program can discover by experimentation that it does not benefit or does benefit from deeper streams and can modify the DPFD accordingly.

#### 19.1.7.14 Software-Initiated Data Prefetch

Streams allocated by software evict streams allocated by hardware. The maximum depth of streams for the Power10 core is increased and adaptive.

The **dcbt** and **dcbtst** instructions, described in *Section 5.6.2.6 Data Cache Block Touch to a Single Block Characteristics* on page 90, cause a single-line prefetch (**dcbt** into the L1 cache and **dcbtst** into the L3 cache).

The **dcbt/dcbtst** TH = x'8', x'B', x'A', x'E' values identify start and stop streams that bring data into the cache hierarchy as described in *Section 5.6.2.7 Data Cache Block Touch Streaming Characteristics* on page 91.

- A stream takes a PRQ entry
- Streams can specify a finite length (UNITCNT) or unlimited. If the UNITCNT is never reached, the stream stays active occupying a PRQ entry until it is converted to a software stream on a context switch.



- At creation, software streams initiate prefetches until either the specified depth (as indicated in DSCR[DPFD] or LPCR[DPFD]) is reached or the UNITCNT is reached, which ever comes first.
- Software streams are converted to hardware streams as follows:
  - On a context switch
  - When prefetch stop is indicated
  - When the UNITCNT is reached
- Transient indication impacts L3 cache replacement, biasing toward being more easily replaced when set.

The **dcbt/dcbtst** TH = x'E' should be used in situations where the program wishes to fetch a contiguous block of data of a known size (up to 128 KB) from main memory to the L3 cache. The address should be specified with a TH = 8 instruction with UG = '0', which should be followed by a TH = x'E' instruction which specifies the number of 128-byte units of data to be fetched in the UNITCNT field. For maximum efficiency, this TH = 8/TH = E sequence should be performed as far in advance as practicable before the data being fetched is to be actually used by the program. The TH = 8/TH = E sequence will fetch the data to the L3 cache as soon as possible and will not halt to wait for the data to start being used by the program. The entire block specified will always be fetched. As such, if performed in advance, this sequence significantly lowers the latency of the instructions that actually use the fetched data. However, it should only be used when the program is certain the entire block will be required. In practice, scheduling the prefetch (**dcbt/dcbtst**) far enough ahead for misses is challenging due to aggressive pipelining. To hide the latency of a memory miss, a long stream or a long lead time is required. For example, if there is more than one block of data to be accessed, initiating a prefetch for a future block of data while processing the current block is advised when possible. When used effectively, software prefetch streams are a very powerful tool for boosting performance.

Software streams can also be used to guarantee that a particular stream of importance is allowed to gain maturity and remain active and not be evicted if it is deemed critical. This is because the software streams are able to stay resident and are eligible for eviction from the PRQ until they are either stopped or encounter a context switch.

#### 19.1.7.15 Storage Priming/Zeroing Using **dcbz**

The **dcbz** instruction enables invalidation and zeroing of lines before storing to them, which reduces traffic to the memory controller and reduces time to ownership of a line. The **dcbz** instruction is a powerful tool when software knows that a line is fresh (writable) without consequence.

The best practices are to issue the **dcbz** ahead of where stores will occur to a region.

### 19.1.8 Special Instruction Sequences

#### 19.1.8.1 **lарx/stcx** Instruction

Load and reserve and store conditional (**lарx/stcx**) instruction sequences give the programmer/compiler the ability to share storage in an effective manner by exchanging shared locks and updating shared atomic variables between program threads.

The Power10 core has improved lock performance for both uncontested and contested locks. However, care must be taken to follow the Exclusive Access Hint (EH) bit coding guidelines for **lарx** instructions (see the various **lарx** instruction descriptions in Book II of the *Power ISA (Version 3.1B)* specification and in the introduction to the section that describes those instructions).



### *EH Bit*

The lарx EH bit is honored by the Power10 core and should be used consistently to distinguish between:

- Critical sections (EH = '1'), where a lock data element in memory is set to a "locked" state by a **lарx/stcx** pair and remains "locked" while work is performed in a critical section. When the work in the critical section is completed, the lock is released by a store or **lарx/stcx** pair that either partially or completely overwrites the lock value in memory to an "unlocked" value. (If a **lарx/stcx** pair is used to release a lock, the EH value for that **lарx/stcx** pair should actually be '0' as the releasing **lарx/stcx** pair is itself an atomic update, as described in the following bullet.)
- Atomic updates (EH = '0'), where a **lарx/stcx** pair atomically updates a data element in memory and there are no further updates in the program flow directly coupled to this update. For example, a fetch-and-add operation implemented using a **lарx/stcx** would set EH = '0'; whereas, a **lарx/stcx** that obtains a lock, and necessarily implies a later storage update to release the lock, would use EH = '1'.

If the usage of a given **lарx/stcx** pair is unknown, the compiler/library routine should set the EH bit to the atomic update case (EH = '0'). Incorrect usage of the EH bit will in most cases cause a performance degradation and that degradation is more pronounced when an EH = '1' setting is used incorrectly, especially on a heavily contested atomic update.

### *Contested Locks and Atomically Updated Variables*

The data element for a heavily contested lock variable or a heavily contested variable being atomically updated should often be kept in its own cache line with no other data in the cache line. Padding of memory might be required by the programmer or compiler to achieve this separation. This isolation both removes unnecessary contention for the cache line and enables the various mechanisms implemented in the Power10 core for locks and atomic updates to perform in an optimal fashion.

Looping on a contested lock and re-reading the lock value when the lock variable is in a "locked" state is a common software strategy. While looping in such cases (and in the initial read of the lock), reading the lock with a normal load and then attempting a **lарx** instruction only after the normal load observes the lock in an "unlocked" state performs better in many cases rather than re-reading the location only using **lарx** instructions.

It is also common to go into a lower thread-priority state while polling. However, NOPs to alter thread priority state should be avoided within the polling loop itself, and instead should be inserted before starting and after exiting the polling loop. See *Section 19.1.3.1 Thread Priority* on page 248 for additional information.

#### **19.1.8.2 Lock Critical Section Management**

For contested locks, critical section access should be carefully considered. The section *Branch Direction Prediction Using Static Prediction* on page 251 describes a method to use hint bits to bias toward or away from a critical section depending on if a lock is known to be uncontested or contested.

#### **19.1.8.3 icbi Instruction**

See *Section 5.6.2.2 Instruction Cache Block Invalidate (icbi)* on page 89 for details.



#### **19.1.8.4 *isync* Instruction**

An **isync** pauses briefly at NTC to check if any special conditions have occurred. If no special conditions have occurred, it completes with no effect. However, if a special condition has occurred since the last flush of the pipeline, all subsequent instructions are flushed from the pipeline and re-fetched.

See *Section 5.6.2.3 Instruction Cache Synchronize (isync)* on page 89 for details.

#### **19.1.8.5 *ptesync* Instruction**

A **ptesync** is held at dispatch until all loads have been executed; for example, including no outstanding load misses. It synchronizes at NTC with the L2 cache to ensure that no snoop or store from another thread matches any of the loads in the pipeline. If any does match, it causes a flush of the pipeline and a refetch of subsequent instructions.

#### **19.1.8.6 *sync* Instruction**

The **sync (hwsync)** instruction synchronizes at NTC with the L2 cache to ensure no snoop or store from another thread matches any loads in the pipeline. If a match is detected, a pipeline flush occurs along with a refetch of the subsequent instructions.

#### **19.1.8.7 *eieio* Instruction**

An **eieio** instruction is held at dispatch until all loads have been executed; for example, including no outstanding load misses. Cache-inhibited loads are rejected when an older **eieio** is active in the pipeline.



## 19.2 Instruction Properties

Characteristics for instructions are listed in *Table A-1. Instruction Properties* on page 295 including various latency, throughput, and interlock specifications:

- **Instruction Mnemonic:** For cracked and expanded operations, this field is only valid for the first IOP and subsequent rows indicate the behaviors for additional IOPs. When applicable, the mask or sub fields are shown as a hexadecimal suffix appended to the mnemonic.
- **Single/Cracked/Expanded:**
  - S - single
  - C2 - cracked into 2 IOPs
  - C4 - cracked into 3 IOPs
  - X - expanded
- **Op Num:** the number of the IOP for the instruction
  - “-” - single IOP instruction
  - 1-N - IOP number
- **Pipe Class:** designates the pipeline
  - AA
  - BF
  - BR
  - CY
  - DF
  - DV
  - DX
  - F2
  - FX
  - LD
  - MM
  - MU
  - PM
  - ST
  - SX
  - vMU
- **Main DST:** Indicates the main register target type.
  - Accumulators (ACC)
  - Count Register (CTR)
  - Floating-Point Register (FPR)
  - General Purpose Register (GPR)
  - Link Register (LR)
  - Target Address Register (TAR)
  - Vector-based register (VR)
  - Vector Scalar Register (VSR)
- **CR DST:** Indicates a CR target, if any.
- **XER/FPSCR/VSCR DST:** Indicates XER, FPSCR, or VSCR as a target.
- **Ideal Maximum Operations Per Cycle:** The maximum rate of executing this IOP within the processing pipeline: for multicycle instructions, this is shown as a ratio less than one. This is not always an indication



of the peak sustainable execution rate for the specific instruction as various throughput limitations might apply.

- **Latency (Minimum):** The minimum latency for executing a dependent IOP relative to execution of this IOP for the main register destination, if any. Additional latency for the non-main register destinations is added as specified in the consuming IOP’s field “Additional Latency for CR/XER//FPSCR/VSCR Source”. Note that issue-to-issue latencies should be consulted in *Section 19.1.5.2 Execution Pipeline Issue-to-Issue Latencies* on page 264 to identify additional latency components incurred when exchanging data between pipelines and between various data types. For load instructions, the latency reflects optimal address alignment. Additional latency can also be incurred as shown in *Section 19.1.7.7 Load-to-Use Latency* on page 285.
- **Latency (Maximum):** The maximum latency for executing a dependent IOP relative to execution of this IOP for the main register destination, if any.
- **Pipe Busy Cycles (Minimum):** The number of cycles for which the execution pipeline is blocked by executing this IOP. Multicycle instructions have a cycle count greater than one.**Dispatch Rule:** See section *Section 19.1.4.5 Dispatch* on page 258 for additional details.
  - P - Dispatches together with previous IOP from the same cracked instruction to the same superslice
- **Dispatch Rule:** See section *Section 19.1.4.5 Dispatch* on page 258 for additional details.
  - P - pair, 2 operations to full ISQ entry
- **Dispatch Interlock:** See section *Section 19.1.4.5 Dispatch* on page 258 for additional details.
  - CS0/1 - This IOP checks a dispatch scoreboard. If the scoreboard has been set, the IOP is held at dispatch until the scoreboard setter completes.
  - SS0/1 - This IOP sets a dispatch scoreboard. The scoreboard remains set while this IOP is in the ICT.
- **CR / XER / FPSCR Source:** Indicates Cr, XER, or FPSCR as a source.
- **Issue Next-to-Complete:** Indicates that this IOP is held in the issue queue until completion of all older IOPs.





## Appendix A. Instruction Properties

*Table A-1.* lists the Power10 instruction characteristics including latency, throughput, and interlock specifications. See *Section 19.2 Instruction Properties* on page 292 for descriptions of the table headings and values.

*Table A-1. Instruction Properties* (Sheet 1 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
add	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
add.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
addc	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
addc.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
addco	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
addco.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
adde	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
adde.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
addeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
addex	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addg6s	C2	1	FX		-	XER	4	1	3	1	-	-	-	-
		2	F2	GPR	-	-	4	3	4	1	-	-	XER	-
addi	-	-	SX	GPR	-	-	2	2	3	1	-	-	-	-
addi	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
addic	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
addic.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
addis	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
addme	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addme.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
addmeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addmeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-



Table A-1. Instruction Properties (Sheet 2 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
addo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
addo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
addpcis	C2	1	SX	GPR	-	-	2	2	3	1	-	-	-	-
		2	FX	GPR	-	-	4	1	3	1	-	-	-	-
addze	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addze.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
addzeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
addzeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
and	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
and.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
andc	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
andc.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
andi.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
andis.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
b	-	-	BR		-	-	1	2	2	1	-	-	-	-
ba	-	-	BR		-	-	1	2	2	1	-	-	-	-
bc_always	-	-	BR		-	-	1	2	2	1	-	-	-	-
bc_both	-	-	BR	CTR	-	-	1	2	2	1	-	-	CR	-
bc_cr	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bc_ctr	-	-	BR	CTR	-	-	1	2	2	1	-	-	-	-
bca_always	-	-	BR		-	-	1	2	2	1	-	-	-	-
bca_both	-	-	BR	CTR	-	-	1	2	2	1	-	-	CR	-
bca_cr	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bca_ctr	-	-	BR	CTR	-	-	1	2	2	1	-	-	-	-
bcctr_always	-	-	BR		-	-	1	2	2	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 3 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
bcctr_both	-	-	BR	CTR	-	-	1	2	2	1	-	-	CR	-
bcctr_cr	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bcctr_ctr	-	-	BR	CTR	-	-	1	2	2	1	-	-	-	-
bcctrl_always	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-
bcctrl_both	C2	1	BR	CTR	-	-	1	2	2	1	P	-	CR	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bcctrl_cr	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bcctrl_ctr	C2	1	BR	CTR	-	-	1	2	2	1	P	-	-	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bcdadd.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdcfn.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdcfsq.	-	-	DF	VR	CR	-	1/31	38	38	30	-	-	-	-
bcdcfz.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdcpsgn.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdctn.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdctsq.	-	-	DF	VR	CR	-	1/17	24	24	16	-	-	-	-
bcdctz.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcds.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdsetsgn.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdsr.	-	-	DF	VR	CR	-	2	12	13	1	-	-	-	-
bcdsub.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdtrunc.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdus.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcdutrunc.	-	-	DX	VR	CR	-	2	4	5	1	-	-	-	-
bcl_always	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-



**Table A-1. Instruction Properties (Sheet 4 of 86)**

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
bcl_both	C2	1	BR	CTR	-	-	1	2	2	1	P	-	CR	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bcl_cr	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bcl_ctr	C2	1	BR	CTR	-	-	1	2	2	1	P	-	-	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bcla_always	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-
bcla_both	C2	1	BR	CTR	-	-	1	2	2	1	P	-	CR	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bcla_cr	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bcla_ctr	C2	1	BR	CTR	-	-	1	2	2	1	P	-	-	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bclr_always	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bclr_both	-	-	BR	CTR	-	-	1	2	2	1	-	-	CR	-
bclr_cr	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bclr_ctr	-	-	BR	CTR	-	-	1	2	2	1	-	-	-	-
bclr_l_always	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bclr_l_both	C2	1	BR	CTR	-	-	1	2	2	1	P	-	CR	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bclr_l_cr	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bclr_l_ctr	C2	1	BR	CTR	-	-	1	2	2	1	P	-	-	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bctar_always	-	-	BR		-	-	1	2	2	1	-	-	-	-
bctar_both	-	-	BR	CTR	-	-	1	2	2	1	-	-	CR	-
bctar_cr	-	-	BR		-	-	1	2	2	1	-	-	CR	-
bctar_ctr	-	-	BR	CTR	-	-	1	2	2	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 5 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
bctarl_always	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-
bctarl_both	C2	1	BR	CTR	-	-	1	2	2	1	P	-	CR	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bctarl_cr	-	-	BR	LR	-	-	1	2	2	1	-	-	CR	-
bctarl_ctr	C2	1	BR	CTR	-	-	1	2	2	1	P	-	-	-
		2	BR	LR	-	-	1	2	2	1	-	-	-	-
bl	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-
bla	-	-	BR	LR	-	-	1	2	2	1	-	-	-	-
bpermd	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
brd	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
brh	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
brw	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
cbcdtd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cdtbcd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cfuged	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
clrhrb	S	-	SX		-	-	2	2	3	1	-	-	-	N
cmp	-	-	FX		CR	XER	4	1	3	1	-	-	-	-
cmpb	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
cmpeqb	-	-	F2		CR	XER	4	3	4	1	-	-	-	-
cmpi	-	-	FX		CR	XER	4	1	3	1	-	-	-	-
cmpl	-	-	FX		CR	XER	4	1	3	1	-	-	-	-
cmpli	-	-	FX		CR	XER	4	1	3	1	-	-	-	-
cmprb	-	-	F2		CR	XER	4	3	4	1	-	-	-	-
cntlzd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cntlzd.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 6 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
cntlzdm	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
cntlzw	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cntlzw.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
cnttzd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cnttzd.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
cnttzdm	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
cnttzw	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
cnttzw.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
copy	-	-	ST		-	-	2	2	3	1	-	-	-	-
cp_abort	S	-	ST		-	-	2	2	3	1	-	-	-	-
crand	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
crandc	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
creqv	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
crnand	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
crnor	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
cror	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
crorc	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
crxor	-	-	FX		CR	-	2	1	3	1	-	-	CR	-
dadd	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
dadd.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
daddq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
daddq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
darn	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 7 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
dcbf	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbfl	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbflp	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbfps	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbst	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbstps	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcbt	-	-	LD		-	-	2	4	6	1	-	-	-	-
dcbtst	-	-	LD		-	-	2	4	6	1	-	-	-	-
dcbz	-	-	ST		-	-	2	2	3	1	-	-	-	-
dcffix	-	-	DF	FPR	-	FPSCR	1/26	33	33	25	-	-	FPSCR	-
dcffix.	C2	1	DF	FPR	-	FPSCR	1/26	33	33	25	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dcffixq	C2	1	DF	FPR	-	FPSCR	1/26	34	34	25	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	1/26	34	34	25	-	-	-	-
dcffixq.	C2	1	DF	FPR	-	FPSCR	1/26	34	34	25	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
dcffixqq	C2	1	DF	FPR	-	FPSCR	1/35	43	43	34	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	1/35	43	43	34	-	-	-	-
dcmopo	-	-	DF		CR	FPSCR	2	12	13	1	-	-	FPSCR	-
dcmopoq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dcmpu	-	-	DF		CR	FPSCR	2	12	13	1	-	-	FPSCR	-
dcmpuq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dctdp	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 8 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
dctdp.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dctfix	-	-	DF	FPR	-	FPSCR	1/19	26	26	18	-	-	FPSCR	-
dctfix.	C2	1	DF	FPR	-	FPSCR	1/19	26	26	18	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dctfixq	C2	1	DF	FPR	-	FPSCR	1/19	26	26	18	P	-	FPSCR	-
		2	DF		-	FPSCR	1/19	26	26	18	-	-	-	-
dctfixq.	C2	1	DF	FPR	-	FPSCR	1/19	26	26	18	P	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dctfixqq	C2	1	DF	VR	-	FPSCR	1/23	30	32	22	P	-	FPSCR	-
		2	DF		-	FPSCR	1/23	30	32	22	-	-	-	-
dctqpq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
dctqpq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
ddedpd	-	-	DF	FPR	-	-	2	12	13	1	-	-	-	-
ddedpd.	C2	1	DF	FPR	-	-	2	12	13	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
ddedpdq	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	DF	FPR	-	-	2	12	13	1	-	-	-	-
ddedpdq.	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
ddiv	-	-	DF	FPR	-	FPSCR	1/33	40	100	32	-	-	FPSCR	-
ddiv.	C2	1	DF	FPR	-	FPSCR	1/33	40	100	32	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 9 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
ddivq	C2	1	DF	FPR	-	FPSCR	1/34	42	174	33	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	1/34	42	174	33	-	-	-	-
ddivq.	C2	1	DF	FPR	-	FPSCR	1/34	42	174	33	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
denbcd	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
denbcd.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
denbcdq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
denbcdq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
diex	-	-	DF	FPR	-	-	2	12	13	1	-	-	-	-
diex.	C2	1	DF	FPR	-	-	2	12	13	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
diexq	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	DF	FPR	-	-	2	12	13	1	-	-	-	-
diexq.	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
divd	-	-	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
divd.	C2	1	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divde	-	-	DV	GPR	-	-	2/11	12	41	10	-	-	-	-
divde.	C2	1	DV	GPR	-	-	2/11	12	41	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divdeo	-	-	DV	GPR	-	XER	2/11	12	41	10	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 10 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
divdeo.	C2	1	DV	GPR	-	XER	2/11	12	41	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divdeu	-	-	DV	GPR	-	-	2/11	12	41	10	-	-	-	-
divdeu.	C2	1	DV	GPR	-	-	2/11	12	41	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divdeuo	-	-	DV	GPR	-	XER	2/11	12	41	10	-	-	-	-
divdeuo.	C2	1	DV	GPR	-	XER	2/11	12	41	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divdo	-	-	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
divdo.	C2	1	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divdu	-	-	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
divdu.	C2	1	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divduo	-	-	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
divduo.	C2	1	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divw	-	-	DV	GPR	-	-	2/11	12	20	10	-	-	-	-
divw.	C2	1	DV	GPR	-	-	2/11	12	20	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divwe	-	-	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
divwe.	C2	1	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divweo	-	-	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
divweo.	C2	1	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-



Table A-1. Instruction Properties (Sheet 11 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
divweu	-	-	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
divweu.	C2	1	DV	GPR	-	-	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divweuo	-	-	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
divweuo.	C2	1	DV	GPR	-	XER	2/11	12	25	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divwo	-	-	DV	GPR	-	XER	2/11	12	20	10	-	-	-	-
divwo.	C2	1	DV	GPR	-	XER	2/11	12	20	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
divwu	-	-	DV	GPR	-	-	2/11	12	20	10	-	-	-	-
divwu.	C2	1	DV	GPR	-	-	2/11	12	20	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
divwuo	-	-	DV	GPR	-	XER	2/11	12	20	10	-	-	-	-
divwuo.	C2	1	DV	GPR	-	XER	2/11	12	20	10	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
dmul	-	-	DF	FPR	-	FPSCR	1/18	25	40	17	-	-	FPSCR	-
dmul.	C2	1	DF	FPR	-	FPSCR	1/18	25	40	17	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dmulq.	C2	1	DF	FPR	-	FPSCR	1/13	21	87	12	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	1/13	21	87	12	-	-	-	-
dmulq.	C2	1	DF	FPR	-	FPSCR	1/13	21	87	12	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
dqua	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
dqua.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dquai	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 12 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
dquai.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dquaiq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
dquaiq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
dquaq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
dquaq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
drdpq	C2	1	DF	FPR	-	FPSCR	1/18	25	25	17	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	1/18	25	25	17	-	-	-	-
drdpq.	C2	1	DF	FPR	-	FPSCR	1/18	25	25	17	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
drin	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
drin	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
drin	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX	FPR	CR	-	4	1	3	1	-	-	FPSCR	N
drin	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	N
drin	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 13 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
drintxq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
drrnd	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
drrnd.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
drrndq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
drrndq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
drsp	-	-	DF	FPR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
drsp.	C2	1	DF	FPR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dscli	-	-	DF	FPR	-	-	2	12	13	1	-	-	-	-
dscli.	C2	1	DF	FPR	-	-	2	12	13	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dscliq	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	DF	FPR	-	-	2	12	13	1	-	-	-	-
dscliq.	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
dSCRI	-	-	DF	FPR	-	-	2	12	13	1	-	-	-	-
dSCRI.	C2	1	DF	FPR	-	-	2	12	13	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dSCRIQ	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	DF	FPR	-	-	2	12	13	1	-	-	-	-
dSCRIQ.	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N



**Table A-1. Instruction Properties** (Sheet 14 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
dss	-	-	FX		-	-	4	1	3	1	-	-	-	-
dst	-	-	FX		-	-	4	1	3	1	-	-	-	-
dstst	-	-	FX		-	-	4	1	3	1	-	-	-	-
dsub	-	-	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
dsub.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dsubq	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	DF	FPR	-	FPSCR	2	12	13	1	-	-	-	-
dsubq.	C2	1	DF	FPR	-	FPSCR	2	12	13	1	P	-	FPSCR	-
		2	FX	FPR	CR	-	4	3	3	1	-	-	FPSCR	N
dtstdc	-	-	DF		CR	FPSCR	2	12	13	1	-	-	-	-
dtstdcq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	-	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dtstdg	-	-	DF		CR	FPSCR	2	12	13	1	-	-	-	-
dtstdgq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	-	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dtstex	-	-	DF		CR	FPSCR	2	12	13	1	-	-	-	-
dtstexq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	-	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dtstsfi	-	-	DF		CR	FPSCR	2	12	13	1	-	-	-	-
dtstsfiq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	-	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dtstsfq	C2	1	DF		CR	FPSCR	2	12	13	1	P	-	-	-
		2	DF		-	FPSCR	2	12	13	1	-	-	-	-
dxex	-	-	DF	FPR	-	-	2	12	13	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 15 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
dxex.	C2	1	DF	FPR	-	-	2	12	13	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
dxexq	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	DF		-	-	2	12	13	1	-	-	-	-
dxexq.	C2	1	DF	FPR	-	-	2	12	13	1	P	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
eieio	S	-	ST		-	-	2	2	3	1	-	-	-	-
eqv	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
eqv.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
extsb	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
extsb.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
extsh	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
extsh.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
extsw	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
extsw.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
extswsli	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
extswsli.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
fabs	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fabs.	C2	1	FX	FPR	-	-	4	3	3	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fadd	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fadd.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fadds	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fadds.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 16 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
f fid	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f fid.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f fid s	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f fid s.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f fid u	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f fid u.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f fid us	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f fid us.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f cmpo	-	-	F2		CR	FPSCR	4	3	4	1	-	-	FPSCR	-
f cmpu	-	-	F2		CR	FPSCR	4	3	4	1	-	-	FPSCR	-
f cpsgn	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
f cpsgn.	C2	1	FX	FPR	-	-	4	3	3	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f ctid	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f ctid.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f ctid u	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f ctid u.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
f ctid uz	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
f ctid uz.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 17 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
fctidz	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fctidz.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fctiw	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fctiw.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fctiwu	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fctiwu.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fctiwuz	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fctiwuz.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fctiwz	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fctiwz.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fdiv	-	-	BF	FPR	-	FPSCR	4/22	27	27	7	-	-	FPSCR	-
fdiv.	C2	1	BF	FPR	-	FPSCR	4/22	27	27	7	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fdivs	-	-	BF	FPR	-	FPSCR	4/17	22	22	5	-	-	FPSCR	-
fdivs.	C2	1	BF	FPR	-	FPSCR	4/17	22	22	5	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmadd	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fmadd.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmadds	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 18 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
fmadds.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmr	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fmr.	C2	1	FX	FPR	-	-	4	3	3	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmrgew	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fmrgow	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fmsub	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fmsub.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmsubs	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fmsubs.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmul	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fmul.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fmuls	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fmuls.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fnabs	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fnabs.	C2	1	FX	FPR	-	-	4	3	3	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fneg	-	-	FX	FPR	-	-	4	3	3	1	-	-	-	-
fneg.	C2	1	FX	FPR	-	-	4	3	3	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fnmadd	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 19 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
fnmadd.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fnmadds	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fnmadds.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fnmsub	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fnmsub.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fnmsubs	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fnmsubs.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fre	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fre.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fres	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fres.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frim	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frim.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frin	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frin.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frip	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frip.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 20 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
friz	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
friz.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frsp	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frsp.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frsq rte	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frsq rte.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
frsq rtes	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
frsq rtes.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fsel	-	-	BF	FPR	-	-	4	5	7	1	-	-	-	-
fsel.	C2	1	BF	FPR	-	-	4	5	7	1	-	-	-	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fsqr t	-	-	BF	FPR	-	FPSCR	4/31	36	36	10	-	-	FPSCR	-
fsqr t.	C2	1	BF	FPR	-	FPSCR	4/31	36	36	10	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fsqr ts	-	-	BF	FPR	-	FPSCR	4/21	26	26	5	-	-	FPSCR	-
fsqr ts.	C2	1	BF	FPR	-	FPSCR	4/21	26	26	5	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fsub	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fsub.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fsubs	-	-	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
fsubs.	C2	1	BF	FPR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 21 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
		2	FX		CR	-	4	1	3	1	-	-	FPSCR	N
fdiv	-	-	F2		CR	-	4	3	4	1	-	-	-	-
ftsqrt	-	-	F2		CR	-	4	3	4	1	-	-	-	-
hashchk	C2	1	LD		-	-	2	4	6	1	-	-	-	-
		2	DF		-	-	2	12	13	1	-	-	-	-
hashchkp	C2	1	LD		-	-	2	4	6	1	-	-	-	-
		2	DF		-	-	2	12	13	1	-	-	-	-
hashst	C2	1	DF		-	-	2	12	13	1	-	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
hashstp	C2	1	DF		-	-	2	12	13	1	-	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
hrfid	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
icbi	-	-	ST		-	-	2	2	3	1	-	-	-	N
icbt	-	-	LD		-	-	2	4	6	1	-	-	-	-
isel	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
isync	C2	1	ST		-	-	2	2	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
lbarx	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lbz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lbz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lbzcix	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lbzu	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lbzux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 22 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lbzx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
ld	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
ldarx	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
ldat	X	1	ST		-	-	2	2	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
		3	ST		-	-	2	2	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
		5	ST		-	-	2	2	3	1	-	-	-	N
		6	FX		-	-	4	1	3	1	-	-	-	N
		7	LD	GPR	-	-	2	4	6	1	-	-	-	N
		8	ST		-	-	2	2	3	1	-	-	-	N
ldbrx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
ldcix	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
ldu	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
ldux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
idx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lfd	-	-	LD	FPR	-	-	2	4	6	1	-	-	-	-
lfd	-	-	LD	FPR	-	-	2	4	6	1	-	-	-	-
lfdp	C2	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	SX	FPR	-	-	2	3	3	1	-	-	-	-
lfdpx	C2	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	SX	FPR	-	-	2	3	3	1	-	-	-	-
lfdu	C2	1	LD	FPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 23 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lfdx	C2	1	LD	FPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lfdx	-	-	LD	FPR	-	-	2	4	6	1	-	-	-	-
lfiwax	-	-	LD	FPR	-	-	2	4	6	1	-	-	-	-
lfiwzx	-	-	LD	FPR	-	-	2	4	6	1	-	-	-	-
lfs	C2	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
lfsu	C4	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
		3	SX	GPR	-	-	2	2	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
lfsux	C4	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
		3	SX	GPR	-	-	2	2	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
lfsx	C2	1	LD	FPR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
lha	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhax	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhau	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lhaux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lhax	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhbrx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 24 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lhz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhzcix	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lhzu	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lhzux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lhzx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lmd	X	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	LD	GPR	-	-	2	4	6	1	-	-	-	-
		3	LD	GPR	-	-	2	4	6	1	-	-	-	-
		4	LD	GPR	-	-	2	4	6	1	-	-	-	-
lmw	X	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	LD	GPR	-	-	2	4	6	1	-	-	-	-
		3	LD	GPR	-	-	2	4	6	1	-	-	-	-
		4	LD	GPR	-	-	2	4	6	1	-	-	-	-
lq	C2	1	LD	GPR	-	-	2	4	6	1	P	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lqarx	C2	1	LD	GPR	-	-	2	4	6	1	P	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lswi	X	1	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		2	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		3	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		4	LD	GPR	-	-	2	4	6	1	-	-	XER	-



Table A-1. Instruction Properties (Sheet 25 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lswx	X	1	SX		-	-	2	2	3	1	-	-	XER	N
		2	FX		-	-	4	1	3	1	-	-	-	-
		3	FX		-	-	4	1	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
		5	LD	GPR	-	-	2	4	6	1	-	-	-	-
		6	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		7	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		8	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		9	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		10	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		11	LD	GPR	-	-	2	4	6	1	-	-	XER	-
		12	LD	GPR	-	-	2	4	6	1	-	-	XER	-
lvebx	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lvehx	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lvewx	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lvsl	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
lvsr	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
lvx	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lvxl	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lwa	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwarx	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 26 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lwat	X	1	ST		-	-	2	2	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
		3	ST		-	-	2	2	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
		5	ST		-	-	2	2	3	1	-	-	-	N
		6	FX		-	-	4	1	3	1	-	-	-	N
		7	LD	GPR	-	-	2	4	6	1	-	-	-	N
		8	ST		-	-	2	2	3	1	-	-	-	N
lwaux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lwax	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwbrx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwz	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwzcix	S	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lwzu	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lwzux	C2	1	LD	GPR	-	-	2	4	6	1	-	-	-	-
		2	SX	GPR	-	-	2	2	3	1	-	-	-	-
lwzx	-	-	LD	GPR	-	-	2	4	6	1	-	-	-	-
lxsd	-	-	LD	VR	-	-	2	4	6	1	-	-	-	-
lxwdx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxsbzx_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxsbzx_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxsihxz_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxsihxz_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 27 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lxsiwax	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxsiwzx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxssp	C2	1	LD	VR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
lxsspx	C2	1	LD	VSR	-	-	2	4	6	1	P	-	-	-
		2	F2		-	-	4	3	4	1	-	-	-	-
lxv_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxv_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvb16x_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvb16x_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvd2x	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvdsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvh8x_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvh8x_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvkq	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
lxvl_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvl_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvll_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvll_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvp	C2	1	LD	VSR	-	-	2	4	6	1	P	-	-	-
		2	SX	VSR	-	-	2	2	3	1	-	-	-	-
lxvpx	C2	1	LD	VSR	-	-	2	4	6	1	P	-	-	-
		2	SX	VSR	-	-	2	2	3	1	-	-	-	-
lxvrbx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvrdx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvrhx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 28 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
lxvrx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvw4x	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvwsx_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvwsx_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvx_vec	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
lxvx_vsx	-	-	LD	VSR	-	-	2	4	6	1	-	-	-	-
maddhd	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
maddhdu	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
maddld	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mcrf	-	-	FX		CR	-	4	1	3	1	-	-	CR	-
mcrfs	S	-	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N
mcrfs	-	-	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	-
mcrfs	S	-	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N
mcrxrx	-	-	FX		CR	-	2	1	3	1	-	-	XER	-
mfblrbe	S	-	SX	GPR	-	-	2	2	3	1	-	-	-	N
mfcrr	C2	1	F2	GPR	-	-	2	3	4	1	P	-	CR	-
		2	F2		-	-	2	3	4	1	-	-	CR	-
mfctr	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mfefs_p8	S	-	FX	FPR	-	-	2	3	3	1	-	-	FPSCR	N
mfefs_p8	S	-	FX	FPR	-	-	2	3	3	1	-	-	FPSCR	N
mfefs_p8	C2	1	FX	FPR	-	-	2	3	3	1	-	-	FPSCR	N
		2	FX		-	-	4	1	3	1	-	-	-	N
mfefs	S	-	FX	FPR	-	-	2	3	3	1	-	-	FPSCR	N
mfefs_p8	S	-	FX	FPR	CR	-	2	3	3	1	-	-	FPSCR	N
mfefs_p8	S	-	FX	FPR	CR	-	2	3	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 29 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mffs._p8	C2	1	FX	FPR	CR	-	2	3	3	1	-	-	FPSCR	N
		2	FX		-	-	4	1	3	1	-	-	-	N
mffs.	S	-	FX	FPR	CR	-	2	3	3	1	-	-	FPSCR	N
mffscdrn	S	-	FX	FPR	-	FPSCR	4	3	3	1	-	-	FPSCR	-
mffscdrni	S	-	FX	FPR	-	FPSCR	4	3	3	1	-	-	FPSCR	-
mffsce	C2	1	FX	FPR	-	FPSCR	4	3	3	1	-	-	FPSCR	-
		2	FX	FPR	-	-	2	3	3	1	-	-	FPSCR	N
mffscrn	S	-	FX	FPR	-	FPSCR	4	3	3	1	-	-	FPSCR	-
mffscrni	S	-	FX	FPR	-	FPSCR	4	3	3	1	-	-	FPSCR	-
mffsl	S	-	FX	FPR	-	FPSCR	2	3	3	1	-	-	FPSCR	-
mfgsr	S	-	SX		-	-	2	2	3	1	-	-	-	-
mfhsprg1	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mfir	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mfmsr	S	-	SX	GPR	-	-	2	2	3	1	-	CS0	-	-
mfocrf	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
mfspr_amor	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_amr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_asdr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dar	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dawr0	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dawr1	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dawrx0	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	N
mfspr_dawrx1	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-



Table A-1. Instruction Properties (Sheet 30 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfspr_dscr_ru	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dscr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_dsisr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_hdar	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_hdsisr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_hmrnor	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_iamr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_l2qosr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	N
mfspr_lpqr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_lpidr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_mmcr1_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_mmcr1	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_mmcr2	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_mmcr4_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_mmcr4	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_pidr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_ptcr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_sdar_ru	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	N
mfspr_sdar	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	N
mfspr_smctrl	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-



**Table A-1. Instruction Properties** (Sheet 31 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfspr_tidr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_uamor	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_uamr	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfspr_umor	S	-	MFL	GPR	-	-	1	12	13	1	-	CS0	-	-
mfsprbescr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprbescrr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_bescrru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprbescrs	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_bescrsu	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprcfar	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprciabr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprctrl_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprctrl	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprdec	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprdexcr_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprdexcr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprdpdes	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprebbhr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprebbr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprfscr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprg2	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mfspr_hashkey	S	-	DF	GPR	-	-	2	12	13	1	-	CS0	-	-



Table A-1. Instruction Properties (Sheet 32 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfsprhash pkey	S	-	DF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprhdec	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr hdexc_r_u	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprhdxcr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprheir	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprhfscr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprhid	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprhmeer	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprhmer	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprhsprg0	S	-	MU	GPR	-	-	2	4	5	1	-	CS1	-	-
mfsprhsrr0	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprhsrr1	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspric	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprimc	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprldbar	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr mmcr0_r_u	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprmmcr0	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprm- mcr2_r_u	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprm- mcr3_r_u	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprmmcr3	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprmmcrc	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprpcr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpir	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N



Table A-1. Instruction Properties (Sheet 33 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfspr pmc1_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc1	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr pmc2_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc2	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr pmc3_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc3	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr pmc4_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc4	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr pmc5_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc5	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr pmc6_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmc6	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmcr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpmrsr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprppr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprppr32	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpspb	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfspr psscr_su	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprpsscr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprpurr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprpvrv	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprrpr	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-

*Table A-1. Instruction Properties* (Sheet 34 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfsprrwmr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprsiar_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsiar	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsier_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsier	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_siera_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsiera	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfspr_sierb_ru	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsierb	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprspc	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprsprd	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprsprg0	S	-	MU	GPR	-	-	2	4	5	1	-	CS1	-	-
mfsprsprg1	S	-	MU	GPR	-	-	2	4	5	1	-	CS1	-	-
mfspr_sprg3_ru	S	-	MU	GPR	-	-	2	4	5	1	-	CS1	-	-
mfsprsprg3	S	-	MU	GPR	-	-	2	4	5	1	-	CS1	-	-
mfsprspurr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprsr0	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprsr1	S	-	MF	GPR	-	-	2	12	13	1	-	CS0	-	-
mfsprtbt	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtbtl	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtbu_ru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtbu	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtbtu40	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtexasr	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N





Table A-1. Instruction Properties (Sheet 35 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfsprtexasru	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtfhbar	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtfiar	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtfrmbar	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtir	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprtrace	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtrig0	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprtrig1	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtrig2	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprtscr	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprttr	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprusprg0	S	-	MU	GPR	-	-	2	4	5	1	-	-	-	-
mfsprusprg1	S	-	MU	GPR	-	-	2	4	5	1	-	-	-	-
mfsprusrr0	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprusrr1	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprvrsave	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mfsprvtb	S	-	MF	GPR	-	-	2	12	13	1	-	-	-	N
mfsprwrt	S	-	MF	GPR	-	-	2	12	13	1	-	-	CS0	-
mftar	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mftb	S	-	SX	GPR	-	-	2	2	3	1	-	-	-	N
mftbu	S	-	SX	GPR	-	-	2	2	3	1	-	-	-	N
mfvscr	S	-	FX	VR	-	-	4	1	3	1	-	-	-	N
mfvsrd	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
mfvsrid	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
mfvsrwz	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 36 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mfixer	C2	1	FX	GPR	-	-	2	1	3	1	-	-	XER	-
		2	FX		-	-	4	1	3	1	-	-	-	-
miso	-	-	ST		-	-	2	2	3	1	-	-	-	-
modsd	-	-	DV	GPR	-	-	2/11	12	27	10	-	-	-	-
modsw	-	-	DV	GPR	-	-	2/11	12	20	10	-	-	-	-
modud	-	-	DV	GPR	-	-	2/11	12	27	10	-	-	-	-
moduw	-	-	DV	GPR	-	-	2/11	12	27	10	-	-	-	-
msgclr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
msgclrp	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
msgclru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
msgsnd	S	-	ST		-	-	2	2	3	1	-	-	-	N
msgsndp	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
msgsndu	S	-	ST		-	-	2	2	3	1	-	-	-	N
msgsync	S	-	ST		-	-	2	2	3	1	-	-	-	N
mtcrf	C2	1	FX		CR	XER	4	1	3	1	P	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
mtctr	-	-	FX	CTR	-	-	4	1	2	1	-	-	-	-
mtfsb0	S	-	FX		-	FPSCR	2	1	3	1	-	-	-	-
mtfsb0.	S	-	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N
mtfsb0.	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		CR	-	2	1	3	1	-	-	FPSCR	N
mtfsb1	-	-	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsb1	S	-	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsb1	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsb1.	S	-	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N



Table A-1. Instruction Properties (Sheet 37 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtfsb1.	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N
mtfsf	-	-	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsf	S	-	FX		-	FPSCR	2	1	3	1	-	-	-	-
mtfsf	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsf.	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		CR	-	2	1	3	1	-	-	FPSCR	N
mtfsfi	-	-	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsfi	S	-	FX		-	FPSCR	2	1	3	1	-	-	-	-
mtfsfi	C2	1	FX		-	FPSCR	2	1	3	1	-	-	-	-
		2	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
mtfsfi.	C2	1	FX		-	FPSCR	2	1	3	1	-	-	FPSCR	-
		2	FX		CR	FPSCR	2	1	3	1	-	-	FPSCR	N
mtgsr	S	-	SX		-	-	2	2	3	1	-	-	-	-
mthsprg1	-	-	FX		-	-	4	1	3	1	-	-	-	-
mtlr	-	-	FX	LR	-	-	4	1	2	1	-	-	-	-
mtmsr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtmsrd	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtmsrdee	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtmsree	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtocrf	-	-	FX		CR	-	4	1	3	1	-	-	-	-
mtspr_amor	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_amr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_asdr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dar	S	-	LD		-	-	2	4	6	1	-	SS0	-	N



Table A-1. Instruction Properties (Sheet 38 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtspr_dawr0	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dawr1	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dawrx0	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dawrx1	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dscr_ru	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dscr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_dsisr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_hdar	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_hdsisr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_hrmor	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_iamr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_l2qosr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_lpqr	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_lpldr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_mmcr1_ru	-	-	FX		-	-	4	1	3	1	-	-	-	-
mtspr_mmcr1	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_mmcr2	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N



Table A-1. Instruction Properties (Sheet 39 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtspr_mmcr_a_ru	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_mmcr_a	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_pidr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_ptcr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_sdar_ru	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_sdar	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_smfctrl	C2	1	SX		-	-	2	2	3	1	-	SS0	-	N
		2	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_tidr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_uamor	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_uamr	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtspr_urmor	S	-	LD		-	-	2	4	6	1	-	SS0	-	N
mtsprbescr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprbescr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtspr_bescrru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprbescrs	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtspr_bescrsu	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprcfar	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprciabr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprctrl_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprctrl	S	-	SX		-	-	2	2	3	1	-	SS0	-	N



**Table A-1. Instruction Properties** (Sheet 40 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtsprdec	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprdexcr_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprdexcrr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprdpdes	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprbbhr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprbbrr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprfscr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprg2	-	-	FX		-	-	4	1	3	1	-	-	-	-
mtsprhashkey	S	-	DF		-	-	2	12	13	1	-	SS0	-	N
mtsprhashpkey	S	-	DF		-	-	2	12	13	1	-	SS0	-	N
mtsprhdec	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprhdexcr_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhdxcr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprheir	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhfscr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhid	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhmeer	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhmer	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhsrg0	S	-	MU		-	-	2	4	5	1	-	SS1	-	N
mtsprhsrr0	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprhsrr1	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtspric	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprimc	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprldbar	S	-	SX		-	-	2	2	3	1	-	SS0	-	N



Table A-1. Instruction Properties (Sheet 41 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtsprmmcr0_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprmmcr0	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprmmcr2_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprmmcr3_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprmmcr3	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprmmcrc	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprppcr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprppir	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprpmc1_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc1	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc2_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc2	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc3_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc3	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc4_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc4	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc5_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc5	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc6_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmc6	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpmcr	S	-	SX		-	-	2	2	3	1	-	-	-	N



**Table A-1. Instruction Properties** (Sheet 42 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtsprpmsr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprppr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprppr32	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprpspb	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprpsscr_su	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprpsscr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprpurr	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprpvrr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprrpr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprrwmr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsiar_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsiar	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsier_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsier	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsiera	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsier_ru	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsierb	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprspro	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsprd	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsprg0	S	-	MU		-	-	2	4	5	1	-	SS1	-	N
mtsprsprg1	S	-	MU		-	-	2	4	5	1	-	SS1	-	N
mtsprsprg3_ru	S	-	FX		-	-	4	1	3	1	-	-	-	-
mtsprsprg3	S	-	MU		-	-	2	4	5	1	-	SS1	-	N



Table A-1. Instruction Properties (Sheet 43 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtsprspurr	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprsrr0	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprsrr1	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtb	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprtbl	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprtbu_ru	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprtbu	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprtbu40	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprtexasr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtexasu	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtfhar	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtfiar	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtfmr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtir	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtrace	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtrig0	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtrig1	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtrig2	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprtscr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprttr	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprusprg0	S	-	MU		-	-	2	4	5	1	-	-	-	N
mtsprusprg1	S	-	MU		-	-	2	4	5	1	-	-	-	N
mtsprusrr0	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprusrr1	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mtsprvrsave	S	-	SX		-	-	2	2	3	1	-	SS0	-	N



Table A-1. Instruction Properties (Sheet 44 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mtsprtb	S	-	SX		-	-	2	2	3	1	-	-	-	N
mtsprwort	S	-	SX		-	-	2	2	3	1	-	SS0	-	N
mttar	-	-	FX	TAR	-	-	4	1	2	1	-	-	-	-
mtvscr	S	-	FX		-	VSCR	4	1	3	1	-	-	-	-
mtvsrbm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrbmi	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrd	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
mtvsrdd	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
mtvsrdm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrhdm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrqm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrwa	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
mtvsrwm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
mtvsrws	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
mtvsrwz	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
mtxr	C4	1	FX		-	XER	4	1	3	1	-	-	-	-
		2	FX		-	XER	4	1	3	1	-	-	-	-
		3	FX		-	XER	4	1	3	1	-	-	-	-
		4	FX		-	XER	4	1	3	1	-	-	-	-
mulhd	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mulhd.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
mulhdu	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mulhdu.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
mulhw	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 45 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
mulhw.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
mulhwu	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mulhwu.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
mulld	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mulld.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
muldo	-	-	MU	GPR	-	XER	4	4	5	1	-	-	-	-
muldo.	C2	1	MU	GPR	-	XER	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
mulli	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mullw	-	-	MU	GPR	-	-	4	4	5	1	-	-	-	-
mullw.	C2	1	MU	GPR	-	-	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	-	-
mullwo	-	-	MU	GPR	-	XER	4	4	5	1	-	-	-	-
mullwo.	C2	1	MU	GPR	-	XER	4	4	5	1	-	-	-	-
		2	FX		CR	XER	4	1	3	1	-	-	XER	-
nand	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
nand.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
neg	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
neg.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
nego	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
nego.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
nop_ppr1	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop_ppr2	S	-	FX		-	-	4	1	3	1	-	-	-	N



**Table A-1. Instruction Properties** (Sheet 46 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
nop_ppr3	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop_ppr4	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop_ppr5	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop_ppr6	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop_ppr7	S	-	FX		-	-	4	1	3	1	-	-	-	N
nop0	-	-	FX		-	-	4	1	3	1	-	-	-	-
nop1	-	-	FX		-	-	4	1	3	1	-	-	-	-
nop2	-	-	FX		-	-	4	1	3	1	-	-	-	-
nop3	-	-	FX		-	-	4	1	3	1	-	-	-	-
nor	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
nor.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
or_nop	-	-	FX		-	-	4	1	3	1	-	-	-	-
or_nop	-	-	FX		-	-	4	1	3	1	-	-	-	-
or_nop	-	-	FX		-	-	4	1	3	1	-	-	-	-
or_nop	-	-	FX		-	-	4	1	3	1	-	-	-	-
or	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
or.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
orc	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
orc.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
ori	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
oris	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
paddi	-	-	SX	GPR	-	-	2	2	3	1	P	-	-	-
paste.	S	-	ST		CR	XER	2	2	3	1	-	-	-	-
pdepd	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
pextd	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
plbz	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-



Table A-1. Instruction Properties (Sheet 47 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
pld	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plfd	-	-	LD	FPR	-	-	2	4	6	1	P	-	-	-
plfs	-	-	LD	FPR	-	-	2	4	6	1	P	-	-	-
plha	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plhz	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plq	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plwa	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plwz	-	-	LD	GPR	-	-	2	4	6	1	P	-	-	-
plxsd	-	-	LD	VR	-	-	2	4	6	1	P	-	-	-
plxssp	-	-	LD	VR	-	-	2	4	6	1	P	-	-	-
plxv_vec	-	-	LD	VR	-	-	2	4	6	1	P	-	-	-
plxv_vsx	-	-	LD	VSR	-	-	2	4	6	1	P	-	-	-
plxvp	-	-	SX	VSR	-	-	2	2	3	1	-	-	-	-
plxvp	-	-	LD	VSR	-	-	2	4	6	1	P	-	-	-
pmx vbf16ger2	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	FPSCR	-
pmx vbf16ger2nn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vbf16ger2np	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vbf16ger2pn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vbf16ger2pp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf16ger2	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	FPSCR	-
pmx vf16ger2nn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-

Table A-1. Instruction Properties (Sheet 48 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
pmx vf16ger2np	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf16ger2pn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf16ger2pp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmxvf32ger	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	FPSCR	-
pmx vf32gernn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf32gernp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf32gerpn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf32gerpp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmxvf64ger	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	FPSCR	-
pmx vf64gernn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf64gernp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf64gerpn	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vf64gerpp	-	-	MM	ACC	-	FPSCR	2	4	10	1	P	-	-	-
pmx vi16ger2	-	-	MM	ACC	-	-	2	4	10	1	P	-	FPSCR	-
pmx vi16ger2pp	-	-	MM	ACC	-	-	2	4	10	1	P	-	-	-
pmx vi16ger2s	-	-	MM	ACC	-	VSCR	2	4	10	1	P	-	FPSCR	-
pmx vi16ger2spp	-	-	MM	ACC	-	VSCR	2	4	10	1	P	-	-	-





Table A-1. Instruction Properties (Sheet 49 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
pmxvi4ger8	-	-	MM	ACC	-	-	2	4	10	1	P	-	FPSCR	-
pmxvi4ger8pp	-	-	MM	ACC	-	-	2	4	10	1	P	-	-	-
pmxvi8ger4	-	-	MM	ACC	-	-	2	4	10	1	P	-	FPSCR	-
pmxvi8ger4pp	-	-	MM	ACC	-	-	2	4	10	1	P	-	-	-
pmxvi8ger4spp	-	-	MM	ACC	-	VSCR	2	4	10	1	P	-	-	-
pnop	-	-	FX		-	-	4	1	3	1	-	-	-	-
popcntb	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
popcntd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
popcntw	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
prtyd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
prtyw	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
pstb	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstd	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstfd	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstfs	-	-	ST		-	-	2	2	3	1	P	-	-	-
psth	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstq	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstw	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstxsd	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstxssp	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstxv_vec	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstxv_vsx	-	-	ST		-	-	2	2	3	1	P	-	-	-
pstxvp	-	-	ST		-	-	2	2	3	1	-	-	-	-
ptesync	S	-	ST		-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 50 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
rfebb0	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
rfebb1	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
rfid	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
rfscv	C2	1	SX		-	-	2	2	3	1	P	-	-	N
		2	SX		-	-	2	2	3	1	P	-	-	-
rlpcl	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rlpcl.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rldcr	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rldcr.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rldic	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rldic.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rldicl	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rldicl.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rldicr	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rldicr.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rldimi	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rldimi.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rlwimi	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rlwimi.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rlwinm	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rlwinm.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
rlwnm	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
rlwnm.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 51 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
sc	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
scv	C2	1	FX	CTR	-	-	2	1	2	1	-	CS0	-	-
		2	SX	LR	-	-	2	2	3	1	-	-	-	-
setb	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
setbc	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
setbcr	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
setnbc	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
setnbcr	-	-	FX	GPR	-	-	4	1	3	1	-	-	CR	-
slbfee	S	-	LD	GPR	CR	XER	2	4	6	1	-	CS0	-	N
slbia	S	-	LD		-	-	2	4	6	1	-	-	-	N
slbiag	S	-	ST		-	-	2	2	3	1	-	-	-	-
slbie	S	-	LD		-	-	2	4	6	1	-	-	-	N
slbieg	S	-	ST		-	-	2	2	3	1	-	-	-	-
slbmfee	S	-	LD	GPR	-	-	2	4	6	1	-	CS0	-	N
slbmfev	S	-	LD	GPR	-	-	2	4	6	1	-	CS0	-	N
slbmtc	C2	1	LD		-	-	2	4	6	1	P	SS0	-	N
		2	LD		-	-	2	4	6	1	P	SS0	-	N
slbsync	S	-	ST		-	-	2	2	3	1	-	-	-	-
sld	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
sld.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
slw	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
slw.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
spattn	S	-	FX		-	-	4	1	3	1	-	-	-	N
srad	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
srad.	S	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 52 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
sradi	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
sradi.	S	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
sraw	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
sraw.	S	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
srawi	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
srawi.	S	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
srd	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
srd.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
srw	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
srw.	-	-	F2	GPR	CR	XER	4	3	4	1	-	-	-	-
stb	-	-	ST		-	-	2	2	3	1	-	-	-	-
stb	-	-	ST		-	-	2	2	3	1	-	-	-	-
stbcix	S	-	ST		-	-	2	2	3	1	-	-	-	-
stbcx.	S	-	ST		CR	XER	2	2	3	1	-	-	-	-
stbu	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stbux	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stbx	-	-	ST		-	-	2	2	3	1	-	-	-	-
std	-	-	ST		-	-	2	2	3	1	-	-	-	-
stdat	C4	1	ST		-	-	2	2	3	1	-	-	-	N
		2	FX		-	-	4	1	3	1	-	-	-	N
		3	ST		-	-	2	2	3	1	-	-	-	N
		4	ST		-	-	2	2	3	1	-	-	-	N
stdbrx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stdcix	S	-	ST		-	-	2	2	3	1	-	-	-	-
stdcx.	S	-	ST		CR	XER	2	2	3	1	-	-	-	-
stdu	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 53 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
stdx	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stdx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfd	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfd	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfdp	C2	1	ST		-	-	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
stfdpx	C2	1	ST		-	-	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
stfdx	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stfdx	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stfiwx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfs	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfs	-	-	ST		-	-	2	2	3	1	-	-	-	-
stfsu	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stfsux	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stfsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
sth	-	-	ST		-	-	2	2	3	1	-	-	-	-
sth	-	-	ST		-	-	2	2	3	1	-	-	-	-
sthbrx	-	-	ST		-	-	2	2	3	1	-	-	-	-
sthcix	S	-	ST		-	-	2	2	3	1	-	-	-	-
sthcx.	S	-	ST		CR	XER	2	2	3	1	-	-	-	-
sthu	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
sthux	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
sthx	-	-	ST		-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 54 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
stmd	X	1	ST		-	-	2	2	3	1	-	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
		3	ST		-	-	2	2	3	1	-	-	-	-
		4	ST		-	-	2	2	3	1	-	-	-	-
stmw	X	1	ST		-	-	2	2	3	1	-	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
		3	ST		-	-	2	2	3	1	-	-	-	-
		4	ST		-	-	2	2	3	1	-	-	-	-
stop	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
stq	C2	1	ST		-	-	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
stqcx	C2	1	ST		CR	XER	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
stswi	X	1	ST		-	-	2	2	3	1	-	-	XER	-
		2	ST		-	-	2	2	3	1	-	-	XER	-
		3	ST		-	-	2	2	3	1	-	-	XER	-
		4	ST		-	-	2	2	3	1	-	-	XER	-



Table A-1. Instruction Properties (Sheet 55 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
stswx	X	1	SX		-	-	2	2	3	1	-	-	XER	N
		2	FX		-	-	4	1	3	1	-	-	-	-
		3	FX		-	-	4	1	3	1	-	-	-	-
		4	FX		-	-	4	1	3	1	-	-	-	-
		5	ST		-	-	2	2	3	1	-	-	XER	-
		6	ST		-	-	2	2	3	1	-	-	XER	-
		7	ST		-	-	2	2	3	1	-	-	XER	-
		8	ST		-	-	2	2	3	1	-	-	XER	-
		9	ST		-	-	2	2	3	1	-	-	XER	-
		10	ST		-	-	2	2	3	1	-	-	XER	-
		11	ST		-	-	2	2	3	1	-	-	XER	-
		12	ST		-	-	2	2	3	1	-	-	XER	-
stvebx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvehx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvewx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvxl	-	-	ST		-	-	2	2	3	1	-	-	-	-
stw	-	-	ST		-	-	2	2	3	1	-	-	-	-
stw	-	-	ST		-	-	2	2	3	1	-	-	-	-
stwat	C4	1	ST		-	-	2	2	3	1	-	-	-	N
		2	FX		-	-	4	1	3	1	-	-	-	N
		3	ST		-	-	2	2	3	1	-	-	-	N
		4	ST		-	-	2	2	3	1	-	-	-	N
stwbrx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stwcix	S	-	ST		-	-	2	2	3	1	-	-	-	-
stwcx.	S	-	ST		CR	XER	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 56 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
stwu	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stwux	-	-	ST	GPR	-	-	2	2	3	1	-	-	-	-
stwx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsd	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsdx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsibx_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsibx_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsihx_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsihx_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsiwx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxssp	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxsspx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxv_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxv_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvb16x_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvb16x_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvd2x	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvh8x_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvh8x_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvl_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvl_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvll_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvll_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvp	C2	1	ST		-	-	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 57 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
stxvpx	C2	1	ST		-	-	2	2	3	1	P	-	-	-
		2	ST		-	-	2	2	3	1	-	-	-	-
stxvrbx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvrdx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvrhx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvrwx	-	-	ST		-	-	2	2	3	1	-	-	-	-
stxvw4x	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvxv_vec	-	-	ST		-	-	2	2	3	1	-	-	-	-
stvxv_vsx	-	-	ST		-	-	2	2	3	1	-	-	-	-
subf	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
subf.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
subfc	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
subfc.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
subfco	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
subfco.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
subfe	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfe.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
subfeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
subfic	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
subfme	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfme.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
subfmeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfmeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
subfo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	-	-
subfo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 58 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
subfze	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfze.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
subfzeo	-	-	FX	GPR	-	XER	4	1	3	1	-	-	XER	-
subfzeo.	S	-	FX	GPR	CR	XER	4	1	3	1	-	-	XER	-
sync	C2	1	ST		-	-	2	2	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
tabort.	S	-	FX		CR	XER	4	1	3	1	-	-	-	-
tabortdc.	S	-	F2		CR	XER	4	3	4	1	-	-	-	-
tabortdci.	S	-	F2		CR	XER	4	3	4	1	-	-	-	-
tabortwc.	S	-	F2		CR	XER	4	3	4	1	-	-	-	-
tabortwci.	S	-	F2		CR	XER	4	3	4	1	-	-	-	-
tbegin	S	-	FX		CR	XER	4	1	3	1	-	-	-	-
tcheck	S	-	ST		CR	XER	2	2	3	1	-	-	-	-
td	-	-	F2		-	-	4	3	4	1	-	-	-	-
tdi	-	-	F2		-	-	4	3	4	1	-	-	-	-
tend.	S	-	ST		CR	XER	2	2	3	1	-	-	-	N
tlbie_h	S	-	ST		-	-	2	2	3	1	-	-	-	-
tlbie	S	-	ST		-	-	2	2	3	1	-	-	-	-
tlbiel_h	C2	1	LD		-	-	2	4	6	1	P	-	-	N
		2	LD		-	-	2	4	6	1	P	-	-	N
tlbiel	C2	1	LD		-	-	2	4	6	1	P	-	-	N
		2	LD		-	-	2	4	6	1	P	-	-	N
tlbsync	S	-	ST		-	-	2	2	3	1	-	-	-	-
trechkpt.	S	-	FX		CR	XER	4	1	3	1	-	-	-	N
treclaim.	S	-	FX		CR	XER	4	1	3	1	-	-	-	-
tsr.	S	-	FX		CR	XER	4	1	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 59 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
tw	-	-	F2		-	-	4	3	4	1	-	-	-	-
twi	-	-	F2		-	-	4	3	4	1	-	-	-	-
urfd	C2	1	FX		-	-	4	1	3	1	-	-	-	-
		2	FX		-	-	4	1	3	1	-	-	-	-
vabsdub	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vabsduh	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vabsduw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vaddcuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vaddcuw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vaddecuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vaddeuqm	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vaddfp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vaddsbs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vaddshs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vaddsws	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vaddubm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vaddubs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vaddudm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vadduhm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vadduhs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vadduqm	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vadduwm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vadduws	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vand	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vandc	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vavgsb	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 60 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vavgsh	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vavgsw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vavgub	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vavguh	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vavguw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vbpermd	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vbpermq	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vcfsx	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vcfuged	-	-	CY	VR	-	-	2	4	7	1	-	-	-	-
vcfux	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vcipher	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vcipherlast	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vclrlb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vclrrb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vclzb	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vclzd	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vclzdm	-	-	CY	VR	-	-	2	4	7	1	-	-	-	-
vclzh	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vclzlsbb	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vclzw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpbfp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpbfp.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpeqfp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpeqfp.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpequb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpequb.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 61 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vcmpequd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpequd.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpequh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpequh.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpequq	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpequq.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpequw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpequw.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgefp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpgefp.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtfp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpgtfp.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtsb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtsb.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtsd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtsd.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtsh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtsh.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtsq	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpgtsq.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtsw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtsw.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtub	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtub.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtud	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtud.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-

**Table A-1. Instruction Properties** (Sheet 62 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vcmpgtuh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtuh.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtuq	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vcmpgtuq.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpgtuw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpgtuw.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpneb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpneb.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpneh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpneh.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpnew	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpnew.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpnezb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpnezb.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpnezh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpnezh.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpnezw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vcmpnezw.	-	-	F2	VR	CR	-	4	3	4	1	-	-	-	-
vcmpsq	-	-	F2		CR	-	4	3	4	1	-	-	-	-
vcmpuq	-	-	F2		CR	-	4	3	4	1	-	-	-	-
vcntmbb	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vcntmbd	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vcntmbh	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vcntmbw	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vctsxs	-	-	BF	VR	-	VSCR	4	5	7	1	-	-	-	-
vctuxs	-	-	BF	VR	-	VSCR	4	5	7	1	-	-	-	-





Table A-1. Instruction Properties (Sheet 63 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vctzb	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vctzd	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vctzdm	-	-	CY	VR	-	-	2	4	7	1	-	-	-	-
vctzh	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vctzlsbb	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vctzw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vdivesd	-	-	DV	VR	-	-	2/22	26	75	21	-	-	-	-
vdivesq	-	-	DF	VR	-	-	1/13	22	61	12	-	-	-	-
vdivesw	-	-	DV	VR	-	-	2/34	38	83	33	-	-	-	-
vdriveud	-	-	DV	VR	-	-	2/22	26	75	21	-	-	-	-
vdriveuq	-	-	DF	VR	-	-	1/13	22	61	12	-	-	-	-
vdriveuw	-	-	DV	VR	-	-	2/34	38	83	33	-	-	-	-
vdivsd	-	-	DV	VR	-	-	2/22	26	43	21	-	-	-	-
vdivsq	-	-	DF	VR	-	-	1/13	22	61	12	-	-	-	-
vdivsw	-	-	DV	VR	-	-	2/34	38	54	33	-	-	-	-
vdivud	-	-	DV	VR	-	-	2/22	26	43	21	-	-	-	-
vdivuq	-	-	DF	VR	-	-	1/13	22	61	12	-	-	-	-
vdivuw	-	-	DV	VR	-	-	2/34	38	54	33	-	-	-	-
veqv	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vexpandbm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vexpanddm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vexpandhm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vexpandqm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vexpandwm	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vexpTEfp	C2	1	BF	VR	-	-	4	5	7	1	-	-	-	-
		2	BF	VR	-	-	4	5	7	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 64 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vextddvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextddvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextdubvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextdubvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextduhvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextduhvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextduwvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextduwvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextractbm	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vextractd	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextractdm	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vextracthm	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vextractqm	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vextractub	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextractuh	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextractuw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vextractwm	-	-	F2	GPR	-	-	4	3	4	1	-	-	-	-
vextsb2d	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextsb2w	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextsd2q	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextsh2d	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextsh2w	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextsw2d	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vextublx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vextubrx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vextuhlx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 65 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vextuhrx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vextuwlx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vextuwrx	-	-	PM	GPR	-	-	4	3	4	1	-	-	-	-
vgbbd	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vgnb	-	-	CY	GPR	-	-	2	4	7	1	-	-	-	-
vinsblx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsbrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsbvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsbvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsd	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsdlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsdrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsertb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsertd	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinserth	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsertw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinshlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinshrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinshvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinshvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinsw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinswlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinswrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinswvlx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vinswvrx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vlogefp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 66 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vmaddfp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vmaxfp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vmaxsb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxsd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxsh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxsw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxub	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxud	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxuh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmaxuw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmhaddshs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vmhraddshs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vminfp	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vminsbt	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminsd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminsh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminsw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminub	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminud	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminuh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vminuw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmladduhm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmodsd	-	-	DV	VR	-	-	2/22	26	47	21	-	-	-	-
vmodsq	-	-	DF	VR	-	-	1/16	25	68	15	-	-	-	-
vmodsw	-	-	DV	VR	-	-	2/34	38	60	33	-	-	-	-
vmodud	-	-	DV	VR	-	-	2/22	26	47	21	-	-	-	-



Table A-1. Instruction Properties (Sheet 67 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vmoduq	-	-	DF	VR	-	-	1/16	25	68	15	-	-	-	-
vmoduw	-	-	DV	VR	-	-	2/34	38	60	33	-	-	-	-
vmrgew	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmrghb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrghh	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrghw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrglb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrglh	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrglw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vmrgow	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vmsumcud	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsummbm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsumshm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsumshs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vmsumubm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsumudm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsumuhm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmsumuhs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vmul10cuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vmul10ecuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vmul10euq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vmul10uq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vmulesb	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulesd	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulesh	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulesw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 68 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vmuleub	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuleud	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuleuh	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuleuw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulhsd	-	-	MU	VR	-	-	4	4	5	1	-	-	-	-
vmulhsw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulhud	-	-	MU	VR	-	-	4	4	5	1	-	-	-	-
vmulhuw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulld	-	-	MU	VR	-	-	4	4	5	1	-	-	-	-
vmulosb	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulosd	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulosh	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulosw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuloub	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuloud	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulouh	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmulouw	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vmuluwm	-	-	vMU	VR	-	-	4	6	7	1	-	-	-	-
vnand	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vncipher	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vncipherlast	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vnegd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vnegw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vnmsubfp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vnor	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vor	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 69 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vorc	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vpdepd	-	-	CY	VR	-	-	2	4	7	1	-	-	-	-
vperm	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpermr	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpermxor	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpextd	-	-	CY	VR	-	-	2	4	7	1	-	-	-	-
vpkpx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpksdss	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpksdus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkshss	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkshus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkswss	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkswus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkudum	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpkudus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkuhum	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpkuhus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpkuwum	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpkuwus	-	-	PM	VR	-	VSCR	4	3	4	1	-	-	-	-
vpmsumb	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vpmsumd	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vpmsumh	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vpmsumw	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vpopcntb	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vpopcntd	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vpopcnth	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 70 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vpopcntw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vpptybd	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vpptybq	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vpptybw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vrefp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vrfim	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vrfin	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vrfip	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vrfiz	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vrlb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrld	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrlldmi	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrldnm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrlh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrlq	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vrlqmi	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vrlqnm	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vrlw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrlwmi	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrlwnm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vrsqrtefp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-
vsbox	-	-	CY	VR	-	-	4	4	7	1	-	-	-	-
vsel	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 71 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vshasigmad	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vshasigmaw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vsl	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vslb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsld	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsldbi	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsldoi	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vslh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vslo	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vslq	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vslv	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vslw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vspltb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsplth	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vspltisb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vspltish	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vspltisw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vspltiw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsr	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsrab	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsrad	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsrah	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsraq	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vsraw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-



**Table A-1. Instruction Properties** (Sheet 72 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vsrb	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsrd	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsrdbi	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsrh	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsro	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsrq	C2	1	F2	VR	-	-	4	3	4	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vsrv	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vsrw	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vstrtbl	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrtbl.	C2	1	FX		CR	-	4	1	3	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrtblr	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrtblr.	C2	1	FX		CR	-	4	1	3	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrihl	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrihl.	C2	1	FX		CR	-	4	1	3	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrihr	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vstrihr.	C2	1	FX		CR	-	4	1	3	1	-	-	-	-
		2	PM	VR	-	-	4	3	4	1	-	-	-	-
vsubcuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vsubcuw	-	-	F2	VR	-	-	4	3	4	1	-	-	-	-
vsubecuq	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vsubeuqm	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vsubfp	-	-	BF	VR	-	-	4	5	7	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 73 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
vsubssbs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsubshs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsubsws	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsububm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsububs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsubudm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsubuhm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsubuhs	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsubuqm	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
vsubuwm	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
vsubuws	-	-	F2	VR	-	VSCR	4	3	4	1	-	-	-	-
vsum2sws	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vsum4sbs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vsum4shs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vsum4ubs	-	-	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vsumsws	C2	1	PM	VR	-	-	4	3	4	1	-	-	-	-
		2	vMU	VR	-	VSCR	4	6	7	1	-	-	-	-
vupkhpx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupkhsb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupkhsh	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupkhsw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupklpx	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupklsb	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupklsh	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vupklsw	-	-	PM	VR	-	-	4	3	4	1	-	-	-	-
vxor	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 74 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
wait	-	-	FX		-	-	4	1	3	1	-	-	-	-
xor	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
xor.	-	-	FX	GPR	CR	XER	4	1	3	1	-	-	-	-
xori	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
xoris	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
xsabsdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xsabsqp	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
xsadddp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsaddqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xsaddqpo	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xsaddsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscmpeqdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xscmpeqqp	-	-	DX	VR	-	FPSCR	2	4	5	1	-	-	FPSCR	-
xscmpexpdp	-	-	F2		CR	FPSCR	4	3	4	1	-	-	-	-
xscmpexpqp	-	-	DX		CR	FPSCR	2	4	5	1	-	-	FPSCR	-
xscmpgedp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xscmpgeqp	-	-	DX	VR	-	FPSCR	2	4	5	1	-	-	FPSCR	-
xscmpgtdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xscmpgtqp	-	-	DX	VR	-	FPSCR	2	4	5	1	-	-	FPSCR	-
xscmpodp	-	-	F2		CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xscmpoqp	-	-	DX		CR	FPSCR	2	4	5	1	-	-	FPSCR	-
xscmpudp	-	-	F2		CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xscmpuqp	-	-	DX		CR	FPSCR	2	4	5	1	-	-	FPSCR	-
xscpsgndp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xscpsgnqp	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 75 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xscvdphp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvdppq	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvdpsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvdpspn	-	-	BF	VSR	-	-	4	5	7	1	-	-	-	-
xscvdpsxds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvdpsxws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvdpxuds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvdpxuws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvhpdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xscvqpdp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpdpo	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpsdz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpsqz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpswz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpudz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpuqz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvqpuwz	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvsdqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvspdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvspdpn	-	-	F2	VSR	-	-	4	3	4	1	-	-	-	-
xscvsqqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvsxddp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvsxdsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xscvudqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvuqqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xscvuxddp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 76 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xscvuxdsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsdvidp	-	-	BF	VSR	-	FPSCR	4/22	27	27	7	-	-	FPSCR	-
xsdivqp	-	-	DF	VR	-	FPSCR	1/50	57	59	49	-	-	FPSCR	-
xsdivqpo	-	-	DF	VR	-	FPSCR	1/50	57	59	49	-	-	FPSCR	-
xsdivsp	-	-	BF	VSR	-	FPSCR	4/17	22	22	5	-	-	FPSCR	-
xsievpdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xsiepxpq	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
xsmaddadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmaddasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmaddmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmaddmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmaddqp	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsmaddqpo	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsmaccdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmaccqp	-	-	DX	VR	-	FPSCR	2	4	5	1	-	-	FPSCR	-
xsmacd	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmajdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmincdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmincqp	-	-	DX	VR	-	FPSCR	2	4	5	1	-	-	FPSCR	-
xsmindp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmindjp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xsmsubadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmsubasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmsubmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmsubmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmsubqp	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 77 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xmsubqpo	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsmuldp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsmulqp	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsmulqpo	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsmulsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnabsdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xsnabsqp	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
xsnegdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xsnegqp	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
xsnmaddadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmaddasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmaddmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmaddmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmaddqdp	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsnmaddqpo	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsnmsubadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmsubasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmsubmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmsubmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsnmsubqdp	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 78 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xsn msubqpo	-	-	DF	VR	-	FPSCR	1/18	25	25	17	-	-	FPSCR	-
xsrdpi	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrdpic	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrdpim	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrdpip	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrdpiz	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsredp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsresp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrqpi	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xsrqpix	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xsrqpxp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xsrsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrqrtepd	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xsrqrtesp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xssqrtdp	-	-	BF	VSR	-	FPSCR	4/31	36	36	10	-	-	FPSCR	-
xssqrtpq	-	-	DF	VR	-	FPSCR	1/68	75	77	67	-	-	FPSCR	-
xssqrtpo	-	-	DF	VR	-	FPSCR	1/68	75	77	67	-	-	FPSCR	-
xssqrtsp	-	-	BF	VSR	-	FPSCR	4/21	26	26	5	-	-	FPSCR	-
xssubdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xssubqp	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xssubqpo	-	-	DF	VR	-	FPSCR	2	12	13	1	-	-	FPSCR	-
xssubsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xstdivdp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xstsqrtdp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xststdcdp	-	-	F2		CR	FPSCR	4	3	4	1	-	-	-	-



Table A-1. Instruction Properties (Sheet 79 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xststdcq	-	-	DX		CR	FPSCR	2	4	5	1	-	-	-	-
xststdcp	-	-	F2		CR	FPSCR	4	3	4	1	-	-	-	-
xsexpdp	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
xsexppq	-	-	FX	VR	-	-	4	1	3	1	-	-	-	-
xsexgdp	-	-	FX	GPR	-	-	4	1	3	1	-	-	-	-
xsexgqp	-	-	DX	VR	-	-	2	4	5	1	-	-	-	-
xvabsdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvabssp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvaddirp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvaddirp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvbf16ger2	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	FPSCR	-
xvbf16ger2nn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvbf16ger2np	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvbf16ger2pn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvbf16ger2pp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvcmppeqdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmppeqdp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmppeqsp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmppeqsp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgedp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgedp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgesp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgesp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgtdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-



**Table A-1. Instruction Properties** (Sheet 80 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xvcmpgtdp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgtsp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcmpgtsp.	-	-	F2	VSR	CR	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcpsgndp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvcpsgnsp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvcvbf16sp	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xvcvdpsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvdpsxds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvdpsxws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvdpxuds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvdpxuws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvhpsp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvcvspbf16	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspphp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspsxds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspsxws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspuxds	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvspuxws	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvsxddp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvsxdsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvsxwdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvsxwsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvuxddp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvuxdsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvcvuxwdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 81 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xvcvuxwsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xdivdp	-	-	BF	VSR	-	FPSCR	4/22	27	27	7	-	-	FPSCR	-
xdivsp	-	-	BF	VSR	-	FPSCR	4/19	24	24	8	-	-	FPSCR	-
xvf16ger2	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	FPSCR	-
xvf16ger2nn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf16ger2np	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf16ger2pn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf16ger2pp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf32ger	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	FPSCR	-
xvf32gernn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf32gernp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf32gerpn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf32gerpp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf64ger	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	FPSCR	-
xvf64gernn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf64gernp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf64gerpn	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvf64gerpp	-	-	MM	ACC	-	FPSCR	2	4	10	1	-	-	-	-
xvi16ger2	-	-	MM	ACC	-	-	2	4	10	1	-	-	FPSCR	-
xvi16ger2pp	-	-	MM	ACC	-	-	2	4	10	1	-	-	-	-
xvi16ger2s	-	-	MM	ACC	-	VSCR	2	4	10	1	-	-	FPSCR	-
xvi16ger2spp	-	-	MM	ACC	-	VSCR	2	4	10	1	-	-	-	-
xvi4ger8	-	-	MM	ACC	-	-	2	4	10	1	-	-	FPSCR	-
xvi4ger8pp	-	-	MM	ACC	-	-	2	4	10	1	-	-	-	-
xvi8ger4	-	-	MM	ACC	-	-	2	4	10	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 82 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xvi8ger4pp	-	-	MM	ACC	-	-	2	4	10	1	-	-	-	-
xvi8ger4spp	-	-	MM	ACC	-	VSCR	2	4	10	1	-	-	-	-
xviexpdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xviexpsp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvmaddadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmaddasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmaddmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmaddmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmaxdp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvmaxsp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvmindp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvminsp	-	-	F2	VSR	-	FPSCR	4	3	4	1	-	-	FPSCR	-
xvmsubadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmsubasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmsubmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmsubmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmuldp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvmulsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvnabsdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvnabssp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvnegdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvnegsp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvn maddadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn maddasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-



Table A-1. Instruction Properties (Sheet 83 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xvn maddmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn maddmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn msubadp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn msubasp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn msubmdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvn msubmsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrdpi	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrdpic	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrdpim	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrdpip	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrdpiz	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvredp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvresp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrspi	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrspic	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrspim	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrspip	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrspiz	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrsqrtdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvrsqrtesp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvsqrtdp	-	-	BF	VSR	-	FPSCR	4/31	36	36	10	-	-	FPSCR	-
xvsqrtp	-	-	BF	VSR	-	FPSCR	4/22	27	27	10	-	-	FPSCR	-

**Table A-1. Instruction Properties** (Sheet 84 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xvsubdp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvsubsp	-	-	BF	VSR	-	FPSCR	4	5	7	1	-	-	FPSCR	-
xvtdivdp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xvtdivsp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xvtlsbb	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xvtsqrtdp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xvtsqrtp	-	-	F2		CR	-	4	3	4	1	-	-	-	-
xvtstdcdp	-	-	F2	VSR	-	-	4	3	4	1	-	-	-	-
xvtstdcsp	-	-	F2	VSR	-	-	4	3	4	1	-	-	-	-
xvxexpdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvxexpsp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvxsigdp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xvxsigsp	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxblendvb	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxblendvd	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxblendvh	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxblendvw	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxbrd	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxbrh	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxbrq	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxbrw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxeval	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxextractuw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxgen pcvmb	-	-	DX	VSR	-	-	2	4	5	1	-	-	-	-
xxgen pcvmd	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-





Table A-1. Instruction Properties (Sheet 85 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xxgen pcvmh	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxgen pcvmw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxinsertw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxland	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlandc	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxleqv	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlnand	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlnor	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlor	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlorc	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxlxor	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxmfacc	C4	1	MM	ACC	-	-	2	4	10	1	P	-	-	-
		2	FX	ACC	-	-	2	3	3	1	-	-	-	-
		3	MM	ACC	-	-	2	4	10	1	P	-	-	-
		4	FX	ACC	-	-	2	3	3	1	-	-	-	-
xxmrghw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxmrglw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxmtacc	C2	1	MM	ACC	-	-	2	4	10	1	P	-	FPSCR	-
		2	FX		-	-	2	1	3	1	-	-	-	-
xxperm	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxpermdi_00	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxpermdi_01	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxpermdi_10	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-

**Table A-1. Instruction Properties** (Sheet 86 of 86)

Instruction	Single/ Cracked/ Expanded	Operation Number	Pipe Class	Main DST	CR DST	XER/ FPSCR/ VSCR DST	Ideal Max Operations Per Cycle	Latency (Min)	Latency (Max)	Pipe Busy Cycles (Min)	Dispatch Rule	Dispatch Interlock	CR / XER / FPSCR Source	Issue Next-to- Complete, "N"
xxpermди_11	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxpermr	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxpermx	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxsel	-	-	FX	VSR	-	-	4	1	3	1	-	-	-	-
xxsetaccz	-	-	MM	ACC	-	-	2	4	10	1	-	-	FPSCR	-
xxsldwi	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxsplti32dx0	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxsplti32dx0	-	-	PM	VR	-	-	4	3	4	1	P	-	-	-
xxsplti32dx1	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxsplti32dx1	-	-	PM	VR	-	-	4	3	4	1	P	-	-	-
xxspltiб	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-
xxspltidp	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxspltidp	-	-	PM	VR	-	-	4	3	4	1	P	-	-	-
xxspltiw	-	-	PM	VSR	-	-	4	3	4	1	P	-	-	-
xxspltiw	-	-	PM	VR	-	-	4	3	4	1	P	-	-	-
xxspltw	-	-	PM	VSR	-	-	4	3	4	1	-	-	-	-



## Appendix B. Transactional Memory Instruction Effects

Table B-1. lists the transactional memory (TM) instruction effects depending on the instruction state.

### Notes:

1. N: Non-transactional. Default, no transaction is executing.
2. T: Transactional. State is initiated by the execution of a **tbegin**. instruction in the non-transactional state.
3. S + Flag: Synthetic Suspend State.

No Flag: Suspend state is defined in the *Power ISA (Version 2.07B)* and *Power ISA (Version 3.0C)*.

Flag is set: Synthetic suspend state as defined in *Section 5.6.4.2 Synthetic Suspend State Details* on page 96. When entering synthetic suspend state, it is the hypervisor software's responsibility to set a flag in memory indicating synthetic suspend state has been entered.

4. The single letter N or S represents the following:
  - a. If the instruction execution does not result in a soft-patch interrupt, the N or S reflects the TS state resulting from the hardware execution of the instruction.
  - a. If the instruction execution does result in a soft-patch interrupt, the N or S reflects the final TS state the thread will be in after the hypervisor's soft-patch interrupt handler is executed.

Two letters separated by an arrow indicate the TS state transition attempted by software that results in the soft-patch interrupt being taken by the hypervisor.

5. **Blue** text indicates that the Power10 hardware differs from the *Power ISA (Version 3.0C)*.
6. Except for the **trechkpt** instruction executed in the non-transactional state, all TM Bad Thing detection is performed by the hardware as per the ISA.



Table B-1. Transactional Memory Instruction Effects (Sheet 1 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
<b>tbegin.</b>	<p>N Hardware will take a soft-patch interrupt independent of HV and PR. HSRR0 will have the next instruction address (<b>tbegin</b> instruction + 4). HSRR1 will have MSR when <b>tbegin</b> was executed in. HEIR has the <b>tbegin</b> image.</p> <p><b>Soft-patch handler:</b> check for HFAC and TM Unavailable interrupts (might need to generate (h)fac unavailable) Execute a dummy store conditional instruction (eg. <b>stwcx</b>) to clear any pending reservation. software sets: CR0 <math>\leftarrow</math> 0b101  0 (transaction failure handling value) TFIAR <math>\leftarrow</math> EA of the tbegin + 0x4,HV,PR TFHAR <math>\leftarrow</math> HSRR0 TEXASR(0:15) = '0101'x, TEXASR(16:32) = '0000'x, FS = 1, TFIAR_Exact = 1, ROT = R (taken from instruction in HEIR), S, TL = 1 HSRR1(TS) <math>\leftarrow</math> 0b00 (the TS bits were already this value) HSRR0 <math>\leftarrow</math> unchanged (already points to <b>tbegin</b> + 4) hfid</p>	<p>S Hardware will take a soft-patch interrupt independent of HV and PR. HSRR0 will have the next instruction address (<b>tbegin</b> instruction + 4). HSRR1 will have MSR when <b>tbegin</b> was executed in. HEIR has the <b>tbegin</b> image.</p> <p><b>Soft-patch handler:</b> check for HFAC and TM Unavailable interrupts (might need to generate (h)fac unavailable)  software sets: CR0 <math>\leftarrow</math> 0b0  01  0  TEXASR <math>\leftarrow</math> unchanged TFHAR <math>\leftarrow</math> unchanged TDOOMED <math>\leftarrow</math> unchanged MSR[TS] <math>\leftarrow</math> unchanged HSSR0 <math>\leftarrow</math> unchanged HSRR1[TS] <math>\leftarrow</math> 0b01 hfid</p>
<b>tend.</b>	<p>N CR0 <math>\leftarrow</math> 0b0  00  0</p>	<p>S TM Bad Thing</p>
Abort caused by <b>tabort</b> and conditional <b>tabort</b> variants	<p>N CR0 <math>\leftarrow</math> 0b0  00  0</p>	<p>S CR0 <math>\leftarrow</math> 0  01  0</p> <p>If TXR[FS] = 0 (<b>Note:</b> this should not be possible on the Power10 hardware because FS should always be '1' in the suspend state), then If RA = 0, then TXR <math>\leftarrow</math> 0x00000000  0b00  ROT  0b0  0x000001 else TXR <math>\leftarrow</math> RA(56:63)    0x000001</p> <p>TDOOMED <math>\leftarrow</math> 0b1 MSR[TS] <math>\leftarrow</math> unchanged TFHAR <math>\leftarrow</math> unchanged TFIAR <math>\leftarrow</math> CIA and TFIAR Exact bit set to 1 in TXR</p> <p><b>Disclaimer:</b> For conditional <b>taborts</b>, see the <i>Power ISA Versions 2.07B or 3.0C</i> for the complete setting of TXR.</p>
<b>tsuspend</b>	<p>N CR0 <math>\leftarrow</math> 0b0  00  0</p>	<p>S CR0 <math>\leftarrow</math> 0b0  01  0</p>

Table B-1. Transactional Memory Instruction Effects (Sheet 2 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
<b>tresume</b>	<p>N</p> <p>Hardware will take a soft-patch interrupt independent of HV and PR.</p> <p>HSRR0 will have the next instruction address (<b>tresume</b> instruction + 4).</p> <p>HSRR1 will have MSR when <b>tresume</b> was executed in.</p> <p>HEIR has the <b>tresume</b> image.</p> <p><b>Soft-patch handler:</b></p> <p>check for interrupts (might need to generate (h)fac unavailable)</p> <p>software sets:</p> <p>CR0 <math>\leftarrow</math> 0b0  00  0</p> <p>HSRR1(TS) <math>\leftarrow</math> 0b00 (the TS bits were already this value)</p> <p>HSRR0 <math>\leftarrow</math> unchanged</p> <p>hfid</p>	<p>N</p> <p>Hardware will take a soft-patch interrupt independent of HV and PR.</p> <p>HSRR0 will have the next instruction address (<b>tresume</b> instruction + 4).</p> <p>HSRR1 will have MSR when <b>tresume</b> was executed in.</p> <p>HEIR has the <b>tresume</b> image.</p> <p><b>Soft-patch handler:</b></p> <p>check for interrupts (might need to generate (h)fac unavailable)</p> <p>check flag</p> <p>execute treclaim S <math>\rightarrow</math> N</p> <p>restore pre tbegin state</p> <p>reset flag</p> <p>set HSRR0 to send to TFHAR</p> <p>CR0 <math>\leftarrow</math> 0b1  01  0</p> <p>HSSR0 <math>\leftarrow</math> TFHAR</p> <p>HSRR1[TS] <math>\leftarrow</math> 0b00</p> <p>hfid</p>
<b>treclaim</b> (HV  PR $\neq$ 10)	<p>N</p> <p>Hardware will take a soft-patch interrupt if HV = 0, PR = 0</p> <p>PR = 1 hardware will generate a privilege instruction interrupt.</p> <p>HSRR0 will have next instruction address (<b>treclaim</b> instruction + 4).</p> <p>HSRR1 will have MSR when <b>treclaim</b> was executed in.</p> <p>HEIR has the <b>treclaim</b> image.</p> <p><b>Soft-patch handler:</b></p> <p>check for interrupts (might need to generate (h)fac unavailable)</p> <p>Emulate TM Bad Thing</p>	<p>N</p> <p>Hardware will take a soft-patch interrupt if HV = 0, PR = 0</p> <p>PR = 1 hardware will generate a privilege instruction interrupt.</p> <p>HSRR0 will have next instruction address (<b>treclaim</b> instruction + 4).</p> <p>HSRR1 will have MSR when <b>treclaim</b> was executed in.</p> <p>HEIR has the <b>treclaim</b> image.</p> <p>Hardware will NOT restore any register checkpoint nor storage (no sideband <b>hwsync</b> required)</p> <p><b>Soft-patch handler:</b></p> <p>check for interrupts</p> <p>check flag</p> <p>treclaim</p> <p>restore pre-tbegin state</p> <p>reset flag</p> <p>HSRR0 <math>\leftarrow</math> unaltered by handler</p> <p>CR0 <math>\leftarrow</math> 0  01  0</p> <p>If TXR[FS] = 0, then</p> <p>If RA = 0, then TXR <math>\leftarrow</math> 0x00000001  0b00  ROT  0b0  0x000001</p> <p>else TXR <math>\leftarrow</math> RA(56:63)    0x000001  0b00  ROT  0b0  0x000001</p> <p>HSRR1[TS] <math>\leftarrow</math> 0b00</p> <p>hfid</p>



Table B-1. Transactional Memory Instruction Effects (Sheet 3 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
<b>treclaim</b> (HV  PR = 10)	N TM Bad Thing	N Hardware performs the instruction but only the CR0 and MSR[TS] bits are altered. Hardware will not restore any register checkpoint nor storage (no sideband <b>hwsync</b> required)  CR0 ← 0 01 10 MSR[TS] ← 0b00
<b>trechkpt.</b>	S  Hardware will take a soft-patch interrupt independent of HV value when PR = 0  For PR=1, the instruction takes a privileged instruction interrupt.  HSRR0 will have next instruction address (trechkpt instruction + 4).  HSRR1 will have MSR when trechkpt was executed in.  HEIR has the trechkpt image.  <b>Soft-patch handler:</b> If TXR[FS] = 0, then emulate TM Bad Thing;  Save pre-tbegin state to memory set flag CR0 ← 0 00 1 0 TXR ← unchanged TFHAR ← unchanged TFIAR ← unchanged HSRR1[TS] ← 0b01 HSRR0 ← unchanged hrfid	S For PR=1, the instruction takes a privileged instruction interrupt. TM Bad Thing per ISA
<b>tcheck</b>	N  CR field BF ← TDOOMED  0b00  0b0  <b>Note:</b> TDOOMED = 1 in NT mode ALWAYS. <b>Note:</b> tcheck can hard code the CR values instead of actually fetching the TDOOMED bit.	S CR field BF ← TDOOMED  0b01  0b0  <b>Note:</b> TDOOMED should always be a 0b1 if trechkpt emulation in progress flag is set. <b>Note:</b> tcheck can hard code the CR values instead of actually fetching the TDOOMED bit.
<b>hrfid</b> (Privileged instruction interrupt if HV, PR ≠ '10'b)	N → S:  Hardware will allow the transition from N → S regardless of MSR[TM] value on both current and incoming. MSR[TM] bit is ignored in the current state and incoming state.	No Change
<b>urfid</b> (Privileged instruction interrupt if S, HV, PR ≠ '110'b)	N → S:  Hardware will allow the transition from N → S regardless of MSR[TM] value on both current and incoming. MSR[TM] bit is ignored in the current state and incoming state.	No Change



Table B-1. Transactional Memory Instruction Effects (Sheet 4 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
rfid	No Change	<p>S → T: Hardware will take a soft-patch interrupt when HV = 0, PR = 0, and the incoming MSR[TM] = 1. Software must not execute when HV = 1. For PR = 1, the instruction takes a privileged instruction interrupt. HSRR0 will have next instruction address (rfid instruction + 4). HSRR1 will have MSR when rfid was executed in. HEIR has the rfid image.</p> <p><b>Soft-patch handler:</b> CFAR ← HSRR0-4 check flag execute treclaim S → N restore pre tbegn state reset flag CR0 ← 0b1 01 0 HSRR0 ← TFHAR HSRR1 ← SRR1 (set HV = 0 and ME = 1) HSRR1[TS] ← 0b00 hrfid</p>



Table B-1. Transactional Memory Instruction Effects (Sheet 5 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
<b>rfebb</b>	No Change	<p>S → T:  <b>Hardware will take a soft-patch interrupt if HV = 0 if the current MSR[TM] = 0b1.</b>  <b>Software must not execute when HV = 1.</b>  <b>PR has no effect on rfebb.</b>  <b>HSRR0 will have next instruction address (rfebb instruction + 4).</b>  <b>HSRR1 will have MSR when rfebb was executed in. HEIR has the rfebb image.</b></p> <p><b>Soft-patch handler:</b>  CFAR ← HSRR0-4  check flag  execute treclaim S → N  restore pre-tbegin state  reset flag  CR0 ← 0b1  01  0  HSRR0 ← TFHAR  HSRR1[TS] ← 0b00  BESCR[GE] ← S from rfebb image in HEIR  hrfid</p>
<b>rfscv</b>	No Change	<p>S → T:  <b>Hardware will take a soft-patch interrupt independent of the value if HV = 0 and the incoming MSR[TM] = 1.</b>  <b>Software must not execute when HV = 1.</b>  <b>For PR = 1, the instruction takes a privileged instruction interrupt.</b></p> <p><b>HSRR0 will have next instruction address (rfscv instruction + 4).</b>  <b>HSRR1 will have MSR when rfscv was executed in. HEIR has the rfscv image.</b></p> <p><b>Soft-patch handler:</b>  CFAR ← HSRR0-4  check flag  execute treclaim S → N  restore pre-tbegin state  reset flag  CR0 ← 0b1  01  0  HSRR0 ← TFHAR  HSRR1[0] ← CTR[0] I ((CTR[1] &amp; (<math>\neg</math>HSRR1[63]))  HSRR1[1] ← CTR[1] &amp; (<math>\neg</math>HSRR1[63]))  HSRR1[48] ← CTR[48] I CTR[49]  HSRR1[58] ← CTR[58] I CTR[49]  HSRR1[59] ← CTR[59] I CTR[49]  HSRR1[2 4 6:32 37 39 41 49 52:57 60:62] ← CTR[2 4 6:32 37 39 41 49 52:57 60:62]  HSRR1[TS] ← 0b00  hrfid</p>



Table B-1. Transactional Memory Instruction Effects (Sheet 6 of 6)

Instruction	Instruction State	
	Non-Transactional (N)	Synthetic Suspend (S + Flag)
<b>mtmsrd</b> with L = 0	No Change	<p>S → T: Hardware will take a soft-patch interrupt if HV = 0 (only if L = 0) and the incoming MSR[HV] = 1. Software must not execute when HV = 1. For PR = 1, the instruction takes a privileged instruction interrupt. HSRR0 will have next instruction address (mtmsrd instruction + 4). HSRR1 will have MSR when mtmsrd was executed in. HEIR has the mtmsrd image.</p> <p><b>Soft-patch handler:</b> check flag execute treclaim S → N restore pre tbegin state reset flag <math>CR0 \leftarrow 0b1 01 0</math> <math>HSSR0 \leftarrow TFHAR</math> <math>HSRR1[0] \leftarrow RS[0] \mid ((RS[1] \&amp; (\neg HSRR1[63]))</math> <math>HSRR1[1] \leftarrow RS[1] \&amp; (\neg HSRR1[63])</math> <math>HSRR1[5] \leftarrow RS[5] \&amp; (\neg(RS[1] \&amp; (\neg HSRR1[63])))</math> <math>HSRR1[2\ 4\ 6:50\ 52:62] \leftarrow RS[2\ 4\ 6:50\ 52:62]</math> <math>HSRR1[TS] \leftarrow 0b00</math> hrfid</p>





## Appendix C. Storage Exception Cases by Translation Type

The following tables summarize the storage exception cases for DSEG, ISEG, DSI, HDSI, HISI, ISI, and machine check.

*Table C-1. Exception Cases (DSEG) (Sheet 1 of 2)*

Case	DSISR	Description: Set to '1'b when...	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value	DAR Value	ASDR Value
1	Set bit 38 for a store. Other bits are set to '0'b	Search of the segment table and bolted entries fails to find a match when UPRT = 1 and HV = 0.			X	EA of the instruction access	EA of the data access	Not updated
2	Set bit 38 for a store. Other bits are set to '0'b	Search of the segment table and bolted entries fails to find a match when UPRT = 1, HV = 1, and LPIDR = 0.			X	EA of the instruction access	EA of the data access	Not updated
3	Set bit 38 for a store. Other bits are set to '0'b	Search of the bolted entries fails to find a match when UPRT = 1, HV = 1, and LPIDR ≠ 0.			X	EA of the instruction access	EA of the data access	Not updated
4	Set bit 38 for a store. Other bits are set to '0'b	Search of the SLB fails to find a match when UPRT = 0.			X	EA of the instruction access	EA of the data access	Not updated
5	Set bit 38 for a store. Other bits are set to '0'b	EA(2:11) ≠ '0000000000'b when DR = 1 (independent of HV).		X		EA of the instruction access	EA of the data access	Not updated
6	Set bit 38 for a store. Other bits are set to '0'b	EA(2:11) ≠ '0000000000'b when DR = 1 (independent of HV).	X			EA of the instruction access	EA of the data access	Not updated
7	Set bit 38 for a store. Other bits are set to '0'b	EA(2:11) ≠ '0000000000'b when DR = 0 when HV = 0 (guest real mode). <b>Note:</b> Quadrant bits EA(0:1) are ignored in guest real mode and always treated by hardware as if they are set to '00'b in guest real mode.		X		EA of the instruction access	EA of the data access	Not updated
8	Set bit 38 for a store. Other bits are set to '0'b	When HV = 0, PR = 0, LPIDR ≠ 0, DR = 1, and (quad 1 or quad 2)		X		EA of the instruction access	EA of the data access	Not updated
9	Set bit 38 for a store. Other bits are set to '0'b	When HV = 0, PR = 1, LPIDR ≠ 0, DR = 1, and (quad 1 or quad 2)		X		EA of the instruction access	EA of the data access	Not updated
10	Set bit 38 for a store. Other bits are set to '0'b	When HV = 1, PR = 1, LPIDR ≠ 0, DR = 1, and (quad 1 or quad 2).		X		EA of the instruction access	EA of the data access	Not updated



**Table C-1. Exception Cases (DSEG) (Sheet 2 of 2)**

Case	DSISR	Description: Set to '1'b when...	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value	DAR Value	ASDR Value
11	Set bit 38 for a store. Other bits are set to '0'b	When HV = 1, PR = 1, LPIDR = 0, DR = 1, and (quad 1 or quad 2).	X			EA of the instruction access	EA of the data access	Not updated
12	Set bit 38 for a store. Other bits are set to '0'b	When HV = 1, PR = 0, LPIDR ≠ 0, DR = 1, and quad 0.		X		EA of the instruction access	EA of the data access	Not updated
13	Set bit 38 for a store. Other bits are set to '0'b	When HV = X and PR = 1, LPIDR ≠ 0, DR = 1, and quad 3.		X				



Table C-2. Exception Cases (ISEG)

Case	SRR1	Description: Set to '1'b when...	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value	DAR Value	ASDR Value
1	Set per ISA	Search of the segment table and bolted entries fails to find a match when UPRT = 1 and HV = 0.			X	EA of the instruction access	Not updated	Not updated
2	Set per ISA	Search of the segment table and bolted entries fails to find a match when UPRT = 1, HV = 1, and LPIDR = 0.			X	EA of the instruction access	Not updated	Not updated
3	Set per ISA	Search of the bolted entries fails to find a match when UPRT = 1, HV = 1, and LPIDR ≠ 0.			X	EA of the instruction access	Not updated	Not updated
4	Set per ISA	Search of the SLB fails to find a match when UPRT = 0.			X	EA of the instruction access	Not updated	Not updated
5	Set per ISA	EA(2:11) ≠ '0000000000'b when IR = 1 (independent of HV).		X		EA of the instruction access	Not updated	Not updated
6	Set per ISA	EA(2:11) ≠ '0000000000'b when IR = 1 (independent of HV).	X			EA of the instruction access	Not updated	Not updated
7	Set per ISA	EA(2:11) ≠ '0000000000'b when IR = 0 when HV = 0 (guest real mode). <b>Note:</b> Quadrant bits EA(0:1) are ignored in guest real mode and always treated by hardware as if they are set to '00'b in guest real mode.		X		EA of the instruction access	Not updated	Not updated
8	Set per ISA	When HV = 0, PR = 0, LPIDR ≠ 0, IR = 1, and (quad 1 or quad 2)	X			EA of the instruction access	Not updated	Not updated
9	Set per ISA	When HV = 0, PR = 1, LPIDR ≠ 0, IR = 1, and (quad 1 or quad 2).		X		EA of the instruction access	Not updated	Not updated
10	Set per ISA	When HV = 1, PR = 1, LPIDR ≠ 0, IR = 1, and (quad 1 or quad 2)		X		EA of the instruction access	Not updated	Not updated
11	Set per ISA	When HV = 1, PR = 1, LPIDR = 0, IR = 1, and (quad 1 or quad 2)	X			EA of the instruction access	Not updated	Not updated
12	Set per ISA	When HV = 1, PR = 0, LPIDR ≠ 0, IR = 1, and quad 0.		X		EA of the instruction access	Not updated	Not updated
13	Set per ISA	When HV = X and PR = 1, LPIDR ≠ 0, IR = 1, and quad 3.		X				



Table C-3. Exception Cases (DSI) (Sheet 1 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
	32		Never							
1	33	PF	page fault occurs for data access for HPT when DR = 1 and either VPM = 0 or HVIIIPR = '10'b				X	EA of the data access when VPM = 0	Not updated	Not updated
3			Page fault occurs in process scoped (guest) table for data access when DR = 1 and effLPID ≠ 0			X		EA of the data access when no leaf is found in the process-scoped page tables.	Not updated	Not updated
4			Page fault occurs for data access for process scoped (host) table when DR = 1 and effLPID = 0		X			EA of the data access when no leaf is found in the process-scoped page tables.	Not updated	Not updated
2	34	ATT				X		EA of the data access	Not updated	Not updated
	35		Never							
1	36	Prot	Protection violation: This is PP violation for HPT when DR = 1 (DR = 0 cases are HDSI) and either VPM = 0 or HVIIIPR = '10'b while translating a data access.				X	EA of the data access	Not updated	Not updated
3			Protection violation: This is “read or write” EAA violation for radix for data access in process scoped table when DR = 1 and effLPID ≠ 0			X		EA of the data access	Not updated	Not updated
3a			This is a read violation for host tables that translate the process table entry when DR = 1 and effLPID = 0		X			EA of the data access	Not updated	Not updated
4			This is a read violation for process-scoped (host) table when DR = 1 and effLPID = 0		X			EA of the data access	Not updated	Not updated
1	37	CI	The access is due to an <b>lq</b> , <b>stq</b> , <b>lарx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 for HPT when DR = 1 (DR = 0 cases are HDSI) and either VPM = 0 or HVIIIPR = '10'b				X	EA of the data access	Not updated	Not updated

Table C-3. Exception Cases (DSI) (Sheet 2 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
3			The access is due to an <b>lq</b> , <b>stq</b> , <b>lарx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 in process scoped (guest) table when DR = 1 and effLPID ≠ 0 and HV = 0. <b>Note:</b> This case can occur for nested translation if both (1) the guest table and host are marked I = 1 and the guest will be detected first. Otherwise, the I-bit in the host PTE determines the effective I-bit value for the access.			X		EA of the data access	Not updated	Not updated
4			The access is due to an <b>lq</b> , <b>stq</b> , <b>lарx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 in process-scoped (host) table when DR = 1 and effLPID = 0 (HV = 1 and quadrants 0 and 3).		X			EA of the data access	Not updated	Not updated
5			hRA(8:12) ≠ '0000'b for data access DR = 1 and VPM = 0 or HVIIPR = 10				X	EA of the data access	Not updated	Not updated
7			hRA(8:12) ≠ '0000'b for data access when effLPID = 0 and HV = 1		X			EA of the data access	Not updated	Not updated
	38	Store	A store, <b>dcbz</b> , or AMO instruction resulted in the exception. [According to the <i>Power ISA (Version 3.1B)</i> , this bit is set when the exception is due to the explicit access caused by a store, <b>dcbz</b> , or AMO instruction; or to the failure of an implicit update of the C bit in the process-scoped PTE. For the instructions listed above, if either R or C needs to be updated, the Power10 core reports failure to update C, and thus sets bit 38 even if only R needs to be updated (because C is already '1').]		X	X	X			
	39:40		Never							
1	41	DAWR	A Data Address Watchpoint (DAWR) match occurs when DR = 1 and VPM = 0 (and HR = 0)				X	EA of the data access	Not updated	Not updated
2			A Data Address Watchpoint (DAWR) match occurs when DR = 1 and HVIIPR = '10'b (and HR = 0)				X	EA of the data access	Not updated	Not updated





Table C-3. Exception Cases (DSI) (Sheet 3 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
3			A Data Address Watchpoint (DAWR) match occurs for HR = 1 (regardless of DR value)		X			EA of the data access	Not updated	Not updated
4			A Data Address Watchpoint (DAWR) match occurs for HR = 1 (and DR = 1)		X			EA of the data access	Not updated	Not updated
5			A Data Address Watchpoint (DAWR) match occurs for DR = 0 and HVIPR = '10'b (regardless of HR value)	X				EA of the data access	Not updated	Not updated
1	42	VPCK	Virtual page class key (VPCK) violation (when VPM = '0'b and LPCR(KBV) = 0) when DR = 1 OR when hypervisor mode.				X	EA of the data access	Not updated	Not updated
1	43	SMF	Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0) and DR = 1, and the PTEG address resides in secure memory. This is the PTEG special case in the RFC and is independent of VPM.				X	EA of the data access	Not updated	Not updated
2			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, ^HV = 1, PR = 0, DR = 1 and VPM = 0, and the ARPN in the HPT PTE points to a data address that resides in secure memory.				X	EA of the data access	Not updated	Not updated
3			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0), DR = 1, and the ARPN in the HPT PTE points to a data address that resides in secure memory.				X	EA of the data access	Not updated	Not updated
9			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and DR = 1, and the RPDB in the PATE points to a (host) radix tree that resides in secure memory.		X			EA of the data access	Not updated	Not updated
6			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and DR = 1, and the NLB in the partition-scoped Radix PDE that maps the process table entry points to a PDE/PTE that resides in secure memory.		X			EA of the data access	Not updated	Not updated



Table C-3. Exception Cases (DSI) (Sheet 4 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
4			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and DR = 1 and the RPN in the partition-scoped Radix PTE that maps the process table entry points to an address that resides in secure memory.		X			EA of the data access	Not updated	Not updated
5			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and DR = 1 and the RPDB in the process table entry points a (host) radix tree that resides in secure memory.		X			EA of the data access	Not updated	Not updated
7			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and DR = 1 and the NLB in the process-scoped Radix PDE points to an PDE/PTE for a data address that resides in secure memory.		X			EA of the data access	Not updated	Not updated
8			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and DR = 1 and the RPN in the process-scoped Radix PTE points to an data address that resides in secure memory.		X			EA of the data access	Not updated	Not updated
10			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0) and DR = 0 and the data address attempts to access secure memory.	X				EA of the data access	Not updated	Not updated
1	44	RADIX	Unsupported radix tree configuration due to invalid RTS/RPDS values in the process table entry when HR = 1, DR = 1, and effLPID≠0			X		EA of the data access	Not updated	Not updated
2			Unsupported radix tree configuration due to invalid ME value in in the process-scoped radix tree when HR = 1, DR = 1, and effLPID≠0			X		EA of the data access	Not updated	Not updated
3			Unsupported radix tree configuration due to invalid RTS/RPDS in the process table entry when HR = 1, DR = 1, and effLPID = 0.		X			EA of the data access	Not updated	Not updated

Table C-3. Exception Cases (DSI) (Sheet 5 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
4			Unsupported radix tree configuration due to invalid ME value in the process scoped tree when HR = 1, DR = 1, HV = 1 and effLPID = 0.		X			EA of the data access	Not updated	Not updated
5			Unsupported radix tree configuration due to invalid ME value in the partition scoped tree which maps the process table entry when HR = 1, DR = 1, HV = 1 and effLPID = 0.		X			EA of the data access	Not updated	Not updated
	45	RC	<b>Note:</b> When HRIIGR = 0b00, all atomic R,C and TS bit updates will be reported as HDSI, not DSI.				Never	EA of the data access	Not updated	Not updated
1			Attempt to atomically set a R or C bit fails in the process scoped (guest) radix tree PTE which translates the data address when HR = 1, DR = 1, and effLPID ≠ 0.			X		EA of the data access	Not updated	Not updated
2			Attempt to atomically set a R or C bit fails in the process scoped (host) radix tree PTE which translates the process table entry when HR = 1, DR = 1, HV = 1, and effLPID = 0.		X			EA of the data access	Not updated	Not updated
3			Attempt to atomically set a R or C bit fails in the process scoped (host) radix tree PTE which translates the data address when HR = 1, DR = 1, HV = 1 and effLPID = 0.		X			EA of the data access	Not updated	Not updated
1	46	Guest_Tbl	The hVA of the process table entry or segment table entry cannot be translated when DR = 1 and either VPM = 0 or HVIPR = '10'b.				X	EA of the data access	Not updated	Not updated
2			The process table entry is invalid when DR = 1 and HRIIGR = 0b00 (independent of VPM or HVIPR).				X	EA of the data access	Not updated	Not updated
	47:59		Never							
1	60		A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when HR = 0, MSR(DR) = 1, and either VPM = 0 or HVIPR = '10'b.				X	EA of the data access	Not updated	Not updated



Table C-3. Exception Cases (DSI) (Sheet 6 of 7)

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
2			A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when MSR(DR) = 0 and HVIIIPR = '10'b.	X				EA of the data access	Not updated	Not updated
3			A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when HR = 1, DR = 1, and effLPID ≠ 0.			X		EA of the data access	Not updated	Not updated
4			A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when HR = 1, DR = 1, HV = 1, and effLPID = 0.		X			EA of the data access	Not updated	Not updated
1	61		An AMO is executed with an invalid function code (FC) when HR = 0, MSR(DR) = 1 and either VPM = 0 or HVIIIPR = '10'b.				X	EA of the data access	Not updated	Not updated
2			An AMO is executed with an invalid function code (FC) when MSR(DR) = 0 and HVIIIPR = '10'b.	X				EA of the data access	Not updated	Not updated
3			An AMO is executed with an invalid function code (FC) when HR = 1, DR = X (either 0 or 1), and effLPID ≠ 0.			X		EA of the data access	Not updated	Not updated
4			An AMO is executed with an invalid function code (FC) when HR = 1, DR = 1, HV = 1, and effLPID = 0.		X			EA of the data access	Not updated	Not updated
1	62	CIX	An attempt to execute a "cix" load or store instruction when MSR(DR) = 1 (independent of VPM, privileged instruction violation when HVIIIPR ≠ '10'b).				X	EA of the data access when VPM = 0 and either MSR(DR) = 1 or MSR(DR) = 0 and the page was previously accessed by a normal load/store or instruction fetch for which I = 0, G = 0 in hypervisor real mode.	Not updated	Not updated



**Table C-3. Exception Cases (DSI) (Sheet 7 of 7)**

Case	DSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
2			An attempt to execute a “cix” load or store instruction when specifying a G = 0 page as determined by the RMSC facility (hypervisor real mode only).				X	EA of the data access when either MSR(DR) = 1 or when MSR(DR) = 0 and the page was previously accessed by a normal load/store or instruction fetch for which I = 0, G = 0 in hypervisor real mode.	Not updated	Not updated
4			An attempt to execute a “cix” load or store instruction when MSR(DR) = 1 in the process scoped (host) table when DR = 1 and effLPID = 0.			X		EA of the data access when either MSR(DR) = 1 or when MSR(DR) = 0 and the page was previously accessed by a normal load/store or instruction fetch for which I = 0, G = 0 in hypervisor real mode.	Not updated	Not updated
5			An attempt to execute a “cix” load or store instruction when MSR(DR) = 1 in the process scoped (host) table when DR = 1 and effLPID = 0.		X			EA of the data access when either MSR(DR) = 1 or when MSR(DR) = 0 and the page was previously accessed by a normal load/store or instruction fetch for which I = 0, G = 0 in hypervisor real mode.	Not updated	Not updated



Table C-4. Exception Cases (HDSI) (Sheet 1 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
1	33	PF	Page fault occurs for data access when DR = 0 and HV = 0 for HPT (this is a VRM case and is always reported as HDSI).				X	Not updated	EA of data access.	VSID of data access.
2			Page fault occurs for data access when DR = 1, VPM = 1, and HV  PR ≠ '10'b for HPT.				X	Not updated	EA of data access.	VSID of data access.
5			Page fault occurs in the partition scoped (host) table when DR = 1 and effLPID ≠ 0.			X		Not updated	EA of the data access when no leaf is found in the partition scoped page tables.	gRA of the data access.
6			Page fault occurs in the partition scoped (host) table when DR = 0 and effLPID ≠ 0.			X		Not updated	EA of the data access when no leaf is found in the partition scoped page tables.	gRA of the data access.
	34:35		Never							
1	36	Prot	Protection violation: This is PP violation for the data access for HPT when DR = 0 and HV = 0 (this is a VRM case and is independent of VPM).				X	Not updated	EA of data access.	VSID of data access.
2			Protection violation: This is PP violation for the data access for HPT when DR = 1, VPM = 1, and HV  PR ≠ '10'b.				X	Not updated	EA of data access.	VSID of data access.
7			Protection violation: This is “read or write” EAA violation for the data access in the partition scoped (host) table for DR = 1 and effLPID ≠ 0.			X		Not updated	EA of the data access when partition-scoped PTE protection prevents the data access.	gRA of data access when partition-scoped PTE protection prevents the data access.
8			Protection violation: This is a “read or write” EAA violation in the partition scoped (host) table when attempting to access the guest tables when DR = 1 and effLPID ≠ 0. <b>Note:</b> There is no DR = 0 equivalent because there are no guest tables when DR = 0.			X		Not updated	EA of data access when the process-scoped PTE/PDE/ process table access fails or the process-scoped R/C update fails due to partition-scoped PTE (read or write) protection.	gRA of the table entry when the process-scoped PTE/PDE/process table access fails or the process-scoped R/C update fails due to partition-scoped PTE protection.

Table C-4. Exception Cases (HDSI) (Sheet 2 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
9			Protection violation: This is “read or write” EAA violation in the partition scoped (host) table when attempting to perform a data access when DR = 0 and effLPID ≠ 0.			X		Not updated	EA of the data access when partition-scoped PTE protection prevents the data access.	gRA of data access when partition-scoped PTE protection prevents the data access.
1	37	CI	The access is due to an <b>lq</b> , <b>stq</b> , <b>larx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 in the partition scoped (host) table when DR = 0 (virtual real mode).				X	Not updated	EA of data access in virtual real mode.	Hardware updates with all zeros.
2			The access is due to an <b>lq</b> , <b>stq</b> , <b>larx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 in the partition scoped (host) table when DR = 1 and VPM = 0b1 and HVIIPIR ≠ '10'b.				X	Not updated	EA of data access when VPM = 0b1.	Hardware updates with all zeros.
3			The access is due to an <b>lq</b> , <b>stq</b> , <b>larx</b> , or <b>stcx</b> instruction, which addresses storage that is I = 1 in the partition scoped (host) table when DR = 0 and effLPID ≠ 0. HV = 0 (guest real mode). <b>Note:</b> This case cannot occur for nested translation because either (1) the guest table is also marked I = 1 and that will be detected first, or (2) an attribute mismatch will be reported if the guest is I = 0 and the host is I = 1.				X	Not updated	EA of data access.	Hardware updates with all zeros.



Table C-4. Exception Cases (HDSI) (Sheet 3 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
	38	Store	A store, <b>dcbz</b> , or AMO instruction resulted in the exception. [According to the <i>Power ISA (Version 3.1B)</i> , this bit is set when the exception is due to the explicit access caused by a store, <b>dcbz</b> , or AMO instruction; or to the failure of an implicit update of the C bit in the partition-scoped PTE; or to an implicit update of the R and/or C bits in a process-scoped PTE that fails because the EAA field in the partition-scoped PTE that maps the process-scoped PTE prohibits the update. Because the Power10 core does not support atomic R or C bit updates by the hardware, the last of these three possibilities will not arise. Also, for the instructions listed above, if either R or C needs to be updated, the Power10 core reports failure to update C, and thus sets bit 38 even if only R needs to be updated (because C is already '1').]		X	X	X			
	39 - 40		Never.							
1	41	DAWR	A Data Address Watchpoint (DAWR) match occurs when DR = 0 and HV = 0 (and HR = 0).				X	Not updated	EA of the data access.	Hardware updates with all zeros.
2			A Data Address Watchpoint (DAWR) match occurs when DR = 1, VPM = 1 and HVIPR ≠ '10'b (and HR = 0).				X	Not updated	EA of the data access.	Hardware updates with all zeros.
1	42	VPCK	Virtual page class key(VPCK) violation (when VPM = '1' or LPCR(KBV) = 1) and not in hypervisor mode.				X	Not updated	EA of the data access.	VSID of data access when either VPM = 0b1 or LPCR(KBV) = 1
1	43	SMF	Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, ^HV = 1, PR = 0 and the PTEG address resides in secure memory. This is the PTEG special case in the RFC and is independent of VPM. The data-side version of this exception will probably never occur, because the instruction side should always occur first as the page table should never straddle the secure memory boundary.				X	Not updated	EA of the data access.	VSID of the data access.

**Table C-4. Exception Cases (HDSI) (Sheet 4 of 9)**

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
2			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0, PR = X, DR = 1, and VPM = 1 and the ARPN in the HPT PTE points to a data address that resides in secure memory.				X	Not updated	EA of the data access.	VSID of data access.
3			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, PR = 1, DR = 1, and VPM = 1 and the ARPN in the HPT PTE points to a data address that resides in secure memory (same as case directly below except DR & VRM differs)				X	Not updated	EA of the data access.	VSID of data access.
4			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0, PR = 0, and DR = 0 (VRM) and the ARPN in the HPT PTE points to a data address that resides in secure memory.				X	Not updated	EA of the data access.	VSID of data access.
5			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0 and the RPDB in the PATE points to a partition-scoped radix tree that resides in secure memory.				X	Not updated	EA of the data access.	gRA of the partition scoped radix tree.
6			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID≠0 and the NLB in the partition-scoped radix PDE that maps the process table entry points to a PDE/PTE that resides in secure memory.				X	Not updated	EA of the data access.	gRA of the partition scoped PDE/PTE which resides in secure memory.
7			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID≠0, DR = 1 and the RPN in the partition-scoped radix PTE that maps the process table entry points to an address that resides in secure memory.				X	Not updated	EA of the data access.	gRA of the process table entry which resides in secure memory.
8			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0, DR = 1 and the NLB in the partition-scoped radix PDE that maps the guest radix tree points to a PDE/PTE that resides in secure memory.				X	Not updated	EA of the data access.	gRA of the partition scoped PDE/PTE which resides in secure memory.





Table C-4. Exception Cases (HDSI) (Sheet 5 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
9			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0, DR = 1 and the RPN in the partition-scoped radix PTE that maps the guest radix tree points to an address that resides in secure memory.			X		Not updated	EA of the data access.	gRA of the guest radix tree which resides in secure memory.
10			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0, and DR = 1 and the NLB in the partition-scoped radix PDE points to a PDE/PTE for a data address that resides in secure memory.			X		Not updated	EA of the data access.	gRA of the data access which used attempted to access secure memory.
11			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0, DR = 1 and the RPN in the partition-scoped radix PTE points to a data address that resides in secure memory.			X		Not updated	EA of the data access.	gRA of the data address which resided in secure memory.
12			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and DR = 0 (Guest Real Mode) and the RPDB in the PATE points to a partition-scoped radix tree that resides in secure memory.	dupe of case 5		X		Not updated	EA of the data access.	gRA of the partition-scoped radix tree.
13			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and DR = 0 (Guest Real Mode) and the NLB in the partition-scoped radix PDE points to an PDE/PTE for a data address that resides in secure memory.			X		Not updated	EA of the data access.	gRA of the data access which used attempted to access secure memory.
14			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and DR = 0 (Guest Real Mode) and the RPN in the partition-scoped radix PTE for a data address that resides in secure memory.			X		Not updated	EA of the data access.	gRA of the data address which resided in secure memory.
1	44	RADIX	Unsupported radix tree configuration in the partition scoped tables for a data access when DR = 1 and effLPID ≠ 0. This is results from problems which occur with any of the size fields in the host radix trees.			X		Not updated	EA of the data access which used an unsupported radix tree configuration in the partition-scoped table.	gRA of the data access which used an unsupported radix tree configuration in the partition-scoped table.



Table C-4. Exception Cases (HDSI) (Sheet 6 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
2			Unsupported radix tree configuration in the partition scoped tables which are used to translate the process-scoped tables or the process table entry when DR = 1 and effLPID ≠ 0. This results from problems that occur with any of the size fields in the host radix trees.		X			Not updated	EA of the data access which used an unsupported radix tree configuration in the partition-scoped table.	gRA of the process scoped PDE/PTE or process table entry which used an unsupported radix tree configuration in the partition-scoped table.
3			Unsupported radix tree configuration in the partition-scoped (host) tables for a data access when DR = 0 and effLPID ≠ 0. The invalid RPDS/RTS will not occur as a DSi. They will always be reported as an ISI first (regardless of IR value). This results from problems that occur with any of the size fields in the host radix trees.		X	X		Not updated	EA of the data access which used an unsupported radix tree configuration in the partition-scoped table.	Hardware updates with all zeros.
3a			Unsupported radix tree configuration in the partition table entry for a data access regardless of DR value and effLPID = 0. This case is specifically when the RTS and RPDB values in the partition table entry are unsupported.		X	X		Not updated	EA of the instruction access which caused the read of the partition table entry which resulted in the unsupported MMU configuration error.	Hardware does NOT update.
5			Unsupported MMU configuration. This is specifically an unsupported combination of HR and UPRT when DR = 1 and effLPID ≠ 0.			X		Not updated	EA of the data access for other cases of unsupported MMU configuration or undefined.	Hardware updates with all zeros.
6			This case is now a checkstop in the Power10 core if the two copies of HR differ outside of hypervisor real mode.			X		Not updated	EA of the data access for other cases of unsupported MMU configuration or undefined.	Hardware updates with all zeros.
7			Unsupported MMU configuration. This is specifically an unsupported combinations of HR, GR, and UPRT when DR=0 and effLPID ≠ 0. Note that this case will never occur in the hardware even though it is defined in the Power ISA (Version 3.1B) because the interrupt will always occur on the instruction side first.			X		Not updated	EA of the data access for other cases of unsupported MMU configuration or undefined.	Hardware updates with all zeros.

Table C-4. Exception Cases (HDSI) (Sheet 7 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
8			HV = 0 (applies to both DR = 0 and DR = 1) and LPIDR = 0. This is the only case where SLR with LPIDR = 0 causes an HDSI instead of a DS1. The HDSI turns on HV which is thought to be the correction for this programming error.		X			Not updated	EA of the data access for other cases of unsupported MMU configuration or undefined.	Hardware updates with all zeros.
9			Occurs anytime other than when HV IPR = '10' and DR = 0 (that is, other than hypervisor or ultravisor real mode) if the LPCR[HR] value differs from the PATE[HR] value. This is a new interrupt cause for the Power10 core that is not documented in the <i>Power ISA (Version 3.1B)</i> (only documented in the User's Manual). Note that this will always be recorded on the instruction side (that is, HISI) first and never as an HDSI, because the instruction has to be translated first.		X	X	X	Not updated	Hardware does not NOT update.	Hardware does not update.
1	45	RC	An attempt to atomically set an R or C bit fails for HPT when DR = 1 (independent of HV and VPM) for data access.				X	Not updated	EA of the data access.	VSID of data access.
2			An attempt to atomically set an R or C bit fails for HPT when DR = 0, HV = 0 (independent of VPM) for data access.				X	Not updated	EA of the data access.	VSID of data access.
3			An attempt to atomically set an R or C bit fails in the partition-scoped tables for data access when DR = 1 and effLPID ≠ 0.			X		Not updated	EA of the data access when partition-scoped atomic R,C update fails.	gRA of the data access when partition-scoped atomic R,C update fails.
4			An attempt to atomically set an R or C bit fails in the partition-scoped tables which map the process-scoped PTE/PDE or the process table entry fails when DR = 1 and effLPID ≠ 0.			X		Not updated	EA of data access when the partition scoped PTE atomic R,C update (which maps the process scoped PTE/PDE or process table entry) fails.	gRA of process scoped PTE/PDE or process table entry when the atomic R,C update in the partition-scoped PTE fails.
5			An attempt to atomically set an R or C bit fails in the partition-scoped (host) table when DR = 0 and effLPID ≠ 0.			X		Not updated	EA of the data access when partition-scoped atomic R,C update fails.	gRA of the data access when partition-scoped atomic R,C update fails.





Table C-4. Exception Cases (HDSI) (Sheet 8 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
1	46	Guest _Tbl	The VA of the process table entry or segment table entry cannot be translated (page fault) when (VPM = 1 and HRIIGR = '00'). Note that the Power10 core will never report this case because the core only supports LPCR[UPRT] = 0 when LPCR[HR] = 0.				X	Not updated	VA lower bits of the process or segment table entry when VPM = 0b1.	VA upper bits of the process or segment table entry when VPM = '1'.
3			The gRA of a process-scoped PDE/PTE or process table entry could not be translated (page fault) by the partition-scoped tables when DR = 1 and effLPID ≠ 0.			X		Not updated	EA of the data access.	The gRA of process-scoped PTE/PDE or process table entry.
4			The process table entry cannot be translated when DR = 1 and HRIIGR = '11', HV = 1, and effLPID = 0		X			Not updated	EA of the data access.	The gRA of process-scoped PTE/PDE or process table entry.
	47:59		Never.							
1	60		A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when HR = 0, MSR(DR) = 1, VPM = 1, and HVIIIPR ≠ '10'.				X	Not updated	EA of the data access.	Hardware will update with all zeros.
2			A properly formed copy-paste sequence with illegal destination combination (foreign-to-foreign, local-to-local, or foreign-to-local) when HR = 0, MSR(DR) = 0 and HV = 0 (this is a VRM case and is independent of VPM).				X	Not updated	EA of the data access.	Hardware will update with all zeros.
1	61		An AMO is executed with an invalid function code (FC) when HR = 0, DR = 0 and HV = 0 (this is VRM case and is independent of VPM).				X	Not updated	EA of the data access.	Hardware will update with all zeros.
2			An AMO is executed with an invalid function code (FC) when HR = 0, DR = 1 and VPM = 1 and HVIIIPR ≠ '10'b.				X	Not updated	EA of the data access.	Hardware will update with all zeros.



Table C-4. Exception Cases (HDSI) (Sheet 9 of 9)

Case	HDSISR Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	DAR Value	HDAR Value	ASDR Value
3			AMO's with invalid function code (FC) is always reported as a DS1 when HR = 0 (regardless of HV, PR, DR or effLPID values) so crossing out this case on HDSI tab.		X			Not updated	EA of the data access.	Hardware will update with all zeros.
4			AMO's with invalid function code (FC) is always reported as a DS1 when HR = 0 (regardless of HV, PR, DR or effLPID values) so crossing out this case on HDSI tab.		X			Not updated	EA of the data access	Hardware will update with all zeros.
	62		Never							



Table C-5. Exception Cases (HSI) (Sheet 1 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
	32		Never							
1	33	PF	Page fault occurs for instruction access when IR = 0 and HV = 0 for HPT (this is VRM case and is always reported as HDSI).				X	EA of the instruction access.	Not updated	VSID of instruction access.
2			Page fault occurs for instruction access when IR = 1, VPM = 1, and HVIIIPR ≠ '10'b for HPT.				X	EA of the instruction access.	Not updated	
5			Page fault occurs in the partition scoped (host) table when IR = 1 and effLPID ≠ 0.			X		EA of the instruction access.	Not updated	gRA of instruction access.
6			Page fault occurs in the partition scoped (host) table when IR = 0 and effLPID ≠ 0.			X		EA of the instruction access when no leaf is found in the process scoped page tables.	Not updated	gRA of the instruction access.
	34		Never							
1	35	EX	Instruction fetch to No-execute segment (HPT when IR = 1) when VPM = 1 and HVIIIPR ≠ '10'b.				X	EA of the instruction address when No-execute bit is set in the STE/SLB when VPM = 1.	Not updated	VSID of instruction access.
2			Instruction fetch to No-execute page (HPT) when IR = 1 when VPM = 1 and HVIIIPR ≠ '10'b.				X	EA of the instruction address when No-execute bit is set in the PTE when VPM = 1.	Not updated	VSID of instruction access.
3			Instruction fetch to No-execute page (HPT) when IR = 0 and HV = 0 (this is VRM).				X	EA of the instruction address when No-execute bit is set in the PTE for IR = 0 (VRM) access.	Not updated	VSID of instruction access.



Table C-5. Exception Cases (HSI) (Sheet 2 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
5			Instruction fetch resulting in EAA “execute” violation (Radix) in the partition scoped tables when IR = 1 and effLPID ≠ 0.			X		EA of the instruction address when EAA “execute” violation.	Not updated	gRA of instruction access.
6			Instruction fetch resulting in EAA “execute” violation (Radix) in the partition scoped tables when IR = 0 and effLPID ≠ 0.			X		EA of the instruction address when EAA “execute” violation.	Not updated	gRA of instruction access.
7			Instruction fetch to G = 1 page in HPT when IR = 1, VPM = 1, and HVII PR ≠ ‘10’b.				X	EA of the instruction address when G = 1.	Not updated	gRA of instruction access.
8			Instruction fetch to G = 1 page in HPT when IR = 0 and HV = 0 (this is VRM).				X	EA of the instruction address when G = 1.	Not updated	gRA of instruction access.
11			Instruction fetch to G = 1 page in the process scoped (host) table when IR = 0 and effLPID ≠ 0.			X		EA of the instruction address when G = 1.	Not updated	gRA of instruction access.
1	36	Prot	Protection violation: This is PP violation for the instruction access for HPT when IR = 0 (this is VRM case and is independent of VPM).				X	EA of the instruction access.	Not updated	VSID of instruction access.
2			Protection violation: This is PP violation for the instruction access for HPT when IR = 1, VPM = 1 and HVII PR ≠ ‘10’b				X	EA of instruction access.	Not updated	VSID of instruction access.
5			Protection violation: This is “read or write” EAA violation in the partition scoped (host) table when attempting to access the guest tables when IR = 1 and effLPID ≠ 0.			X		EA of instruction access when the process-scoped PTE/PDE/process table access fails (read) or the process-scoped R/C update fails due to partition-scoped PTE (read or write) protection.	Not updated	gRA of the table entry when the process-scoped PTE/PDE/process table access fails (read) or the process-scoped R/C update fails due to partition-scoped PTE (write) protection.
	37:41		Never							

**Table C-5. Exception Cases (HISI) (Sheet 3 of 8)**

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
1	42	VPCK	VPCK (virtual page class key) violation when VPM = 1				X	EA of the instruction access.	Not updated	VSID of instruction access when HRIIGR = 0b00
1	43	SMF	Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, ^HV = 1, PR = 0 and the PTEG address resides in secure memory. This is the PTEG special case in the RFC and is independent of VPM.				X	EA of the instruction access.	Not updated	VSID of instruction access.
2			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0, PR = X, IR = 1 and VPM = 1 and the ARPN in the HPT PTE points to an instruction address that resides in secure memory.				X	EA of the instruction access.	Not updated	VSID of instruction access.
3			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, PR = 1, IR = 1 and VPM = 1 and the ARPN in the HPT PTE points to an instruction address that resides in secure memory (same as case directly below except IR & VRM differs)				X	EA of the instruction access.	Not updated	VSID of instruction access.
4			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0, PR = 0 and IR = 0 (VRM) and the ARPN in the HPT PTE points to an instruction address that resides in secure memory.				X	EA of the instruction access.	Not updated	VSID of instruction access.
5			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0 and the RPDB in the PATE points to a partition-scoped radix tree that resides in secure memory.			X		EA of the instruction access.	Not updated	gRA of the partition scoped radix tree.





Table C-5. Exception Cases (HSI) (Sheet 4 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
6			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID≠0 and IR = 1 and the NLB in the partition-scoped radix PDE that maps the process table entry points to a PDE/PTE that resides in secure memory.		X			EA of the instruction access.	Not updated	gRA of the partition scoped PDE/PTE, which resides in secure memory.
7			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID≠0 and IR = 1 and the RPN in the partition-scoped radix PTE that maps the process table entry points to an address that resides in secure memory.			X		EA of the instruction access.	Not updated	gRA of the process table entry, which resides in secure memory.
8			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID≠0 and IR = 1 and the NLB in the partition-scoped radix PDE that maps the guest radix tree points to a PDE/PTE that resides in secure memory.			X		EA of the instruction access.	Not updated	gRA of the partition scoped PDE/PTE, which resides in secure memory.
9			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0 and IR = 1 and the RPN in the partition-scoped radix PTE that maps the guest radix tree points to an address that resides in secure memory.			X		EA of the instruction access	Not updated	gRA of the guest radix tree, which resides in secure memory.
10			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0 and IR = 1 and the NLB in the partition-scoped radix PDE points to a PDE/PTE for a instruction address that resides in secure memory.			X		EA of the instruction access	Not updated	gRA of the instruction access, which used attempted to access secure memory.
11			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, effLPID ≠ 0 and the RPN in the partition-scoped radix PTE points to an instruction address that resides in secure memory.			X		EA of the instruction access	Not updated	gRA of the instruction address which resided in secure memory.

Table C-5. Exception Cases (HISI) (Sheet 5 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
12			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and IR = 0 (guest Real Mode) and the RPDB in the PATE points to a partition-scoped radix tree that resides in secure memory.		X			EA of the instruction access	Not updated	gRA of the partition scoped radix tree
13			Secure Memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and IR = 0 (Guest Real Mode) and the NLB in the partition-scoped radix PDE points to an PDE/PTE for an instruction address that resides in secure memory.		X			EA of the instruction access	Not updated	gRA of the instruction access, which used attempted to access secure memory.
14			Secure Memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 0 (effLPID ≠ 0) and IR = 0 (Guest Real Mode) and the RPN in the partition-scoped Radix PTE for an instruction address that resides in secure memory.		X			EA of the instruction access	Not updated	gRA of the instruction address which resided in secure memory.
1	44	RADIX	Unsupported radix tree configuration in the partition scoped tables for an instruction access when IR = 1 and effLPID ≠ 0. This is results from problems which occur with any of the size fields in the host radix trees.		X			EA of the instruction access which used an unsupported radix tree configuration in the partition-scoped table.	Not updated	gRA of the instruction access which used an unsupported radix tree configuration in the partition-scoped table.
2			Unsupported radix tree configuration in the partition scoped tables which are used to translate the process-scoped tables or the process table entry when IR = 1 and effLPID ≠ 0. This is results from problems which occur with any of the size fields in the host radix trees.		X			EA of the instruction access which used an unsupported radix tree configuration in the partition-scoped table.	Not updated	gRA of the process scoped PDE/PTE or process table entry which used an unsupported radix tree configuration in the partition-scoped table.





Table C-5. Exception Cases (HISI) (Sheet 6 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
3			Unsupported radix tree configuration in the partition scoped (host) tables for a instruction access when IR = 0 and effLPID ≠ 0. This is results from problems which occur with any of the size fields in the host radix trees.		X			EA of the instruction access which used an unsupported radix tree configuration in the partition-scoped table.	Not updated	gRA of the process scoped PDE/PTE or process table entry which used an unsupported radix tree configuration in the partition-scoped table.
3a			Unsupported radix tree configuration in the partition table entry for a instruction access regardless of IR value and effLPID = 0. This case is specifically when the RTS and RPDB values in the partition table entry are unsupported.		X	X		EA of the instruction access which caused the read of the partition table entry which resulted in the unsupported MMU configuration error.	Not updated	Hardware updates with all zeros.
5			Unsupported MMU configuration. This is specifically unsupported combinations of HR and UPRT when IR = 1 and effLPID≠0			X		EA of the instruction access for other cases of unsupported MMU configuration or undefined.	Not updated	Hardware updates with all zeros.
6			This case is now a checkstop in Power10 core if the two copies of HR differ outside of hypervisor real mode.			X		EA of the instruction access for other cases of unsupported MMU configuration or undefined.	Not updated	Hardware updates with all zeros.
7			Unsupported MMU configuration. This is specifically unsupported combinations of HR and UPRT when IR = 0 and effLPID≠0.			X		EA of the instruction access for other cases of unsupported MMU configuration or undefined.	Not updated	Hardware updates with all zeros.
8			HV = 0 (applies to both IR = 0 and IR = 1) and LPIDR = 0. This is the only case where SLR with LPIDR = 0 causes an HISI instead of a ISI. The HISI turns on HV which is thought to be the correction for this programming error.		X			EA of the instruction access for other cases of unsupported MMU configuration or undefined.	Not updated	Hardware updates with all zeros.

Table C-5. Exception Cases (HISI) (Sheet 7 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
9			Occurs anytime other than when HVIPR = '10'b and IR = 0 (that is, other than hypervisor or ultrvisor real mode) if the LPCR[HR] value differs from the PATE[HR] value. This is a new interrupt cause for Power10 core that is not documented in the Power ISA (only documented in the User's Manual).		X	X	X	EA of the instruction access for other cases of unsupported MMU configuration or undefined.	Not updated	Hardware updates with all zeros.
1	45	RC	Attempt to atomically set a R or C bit fails for HPT when IR = 1 (independent of HV and VPM) for instruction access				X	EA of the instruction access independent of VPM.	Undefined	VSID of instruction access.
2			Attempt to atomically set a R or C bit fails for HPT when IR = 0, HV = 0 (independent of VPM) for instruction access.				X			VSID of instruction access.
4			Attempt to atomically set a R or C bit fails in the partition scoped tables for instruction access when IR = 1 and effLPID ≠ 0.			X		EA of the instruction access when partition-scoped atomic R,C update fails	Not updated	gRA of the instruction access when partition-scoped atomic R,C update fails
5			Attempt to atomically set a R or C bit fails in the partition scoped tables which map the process scoped PTE/PDE or process table entry fails when IR = 1 and effLPID ≠ 0.			X		EA of instruction access when the partition scoped PTE atomic R,C update (which maps the process scoped PTE/PDE or process table entry) fails.	Not updated	gRA of process scoped PTE/PDE or process table entry when the atomic R,C update in the partition-scoped PTE fails.
6			Attempt to atomically set a R or C bit fails in the partition scoped tables fails when IR = 0 and effLPID ≠ 0 (that is, guest real mode)			X		EA of instruction access when the partition scoped PTE atomic R,C update (which maps the process scoped PTE/PDE or process table entry) fails.	Not updated	gRA of process scoped PTE/PDE or process table entry when the atomic R,C update in the partition-scoped PTE fails.





Table C-5. Exception Cases (HSI) (Sheet 8 of 8)

Case	HSRR1 Bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single-Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	HSRR0 Value	HDAR Value	ASDR Value
6			Attempt to atomically set a R or C bit fails in the partition scoped (host) table when IR = 0 and effLPID ≠ 0.			X		EA of the instruction access when partition-scoped atomic R,C update fails.	Not updated	gRA of the instruction access when partition-scoped atomic R,C update fails.
1	46	Guest_Tbl	The VA of the process table entry or segment table entry cannot be translated (page fault) when (VPM = '1'b and HRIIGR = 0b00). Note that the Power10 core will never report this case since the core only supports LPCR[UPRT]=0 when LPCR[HR]=0.				X	EA of the instruction access.	VA lower bits of the process or segment table entry when VPM = '1'b.	VA upper bits of the process or segment table entry when VPM = '1'.
3			The gRA of a process scoped PDE/PTE or process table entry could not be translated (page fault) by the partition-scoped tables.			X		EA of the instruction access.	Not updated	gRA of process scoped PTE/PDE or process table entry.
4			The process table entry cannot be translated when IR = 1 and HRIIGR = 0b11, HV = 1 and effLPID = 0.		X			EA of the instruction access.	Not updated	gRA of process scoped PTE/PDE or process table entry.
1	47	Write	Set to '1' if the operation that caused the exception was attempting to update storage; otherwise set to 0. This bit can be set as a modifier to bit 45 to indicate that a change bit must be set (rather than a reference bit). It can also be set as a modifier to bit 36, to indicate that write authority was required to complete the operation.			X	X			
	48:63		Never							



Table C-6. Exception Cases (ISI) (Sheet 1 of 4)

Case	SRR1 bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value
	32		Never					
1	33	PF	Page fault occurs for instruction access for HPT when IR = 1 and either VPM = 0 or HVIPR = '10'b			X		EA of the instruction address.
3			Page fault occurs in process-scoped (guest) table for instruction access when IR = 1 and effLPID ≠ 0			X		EA of the instruction address access when no leaf is found in the process scoped page tables.
4			Page fault occurs in process-scoped (host) table for instruction access when IR = 1 and effLPID = 0		X			EA of the instruction address access when no leaf is found in the process scoped page tables.
2	34	ATT				X		EA of the instruction address.
1	35	EX	Instruction fetch to No-execute segment (HPT) when IR = 1 and either VPM = 0 or HVIPR = '10'b				X	EA of the instruction address when No-execute bit is set in the STE/SLB when VPM = 0.
2			Instruction fetch to No-execute page (HPT) when IR = 1 and either VPM = 0 or HVIPR = '10'b				X	EA of the instruction address when No-execute bit is set in the PTE when VPM = 0.
4			Instruction fetch resulting in EAA "execute" violation (radix) in the process scoped tables when IR = 1 and effLPID ≠ 0			X		EA of the instruction address when EAA "execute" violation.
5			Instruction fetch resulting in EAA "execute" violation (radix) in the process scoped tables when IR = 1 and effLPID = 0		X			EA of the instruction address when EAA "execute" violation
6			Instruction fetch to G = 1 page in HPT when IR = 1 and either VPM = 0 or HVIPR = '10'b				X	EA of the instruction address when G = 1.
8			Instruction fetch to G = 1 page in the process scoped tables			X		EA of the instruction address when G = 1.
9			Instruction fetch to G = 1 page in the process scoped (host) table when IR = 1 and effLPID = 0		X			EA of the instruction address when G = 1
10			Instruction fetch of a prefixed instruction on an l=1 page independent of VPM bit (this is always an ISI, never an HISI).				X	EA of the prefixed instruction address.
11			Instruction fetch of a prefixed instruction on an l=1 page. This is always an ISI, never an HISI.		X			EA of the prefixed instruction address.
12			Instruction fetch of a prefixed instruction on an l=1 page. This is always an ISI, never an HISI.			X		EA of the prefixed instruction address.

Table C-6. Exception Cases (ISI) (Sheet 2 of 4)

Case	SRR1 bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value
1	36	Prot	Protection violation: This is PP (read) violation for an instruction fetch for HPT when VPM = 0. <b>Note:</b> There is no single level radix or radix-on-radix equivalent because only the "execute" bit matters which are reported in SRR1(35) above.				X	EA of the instruction address.
1a			Protection violation: This is PP (read) violation for an instruction fetch for HPT when VPM=1 and HV  PR='10'b. <b>Note:</b> There is no single level radix or radix-on-radix equivalent since only the "execute" bit matters which are reported in SRR1(35) above.				X	EA of the instruction address.
2			This is a PP (read) violation for the host PTE that maps the guest tables (process table and segment table) when VPM = 0.				X	EA of the instruction address.
3			This is a read violation for the host PTE that maps the process table when effLPID = 0 and HV = 1.		X			EA of the instruction address.
	37:41		Never					
1	42	VPCK	Virtual page class key (VPCK) violation when VPM = 0.				X	EA of the instruction address.
1	43	SMF	Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0) and IR = 1, and the PTEG address resides in secure memory. This is the PTEG special case in the RFC and is independent of VPM.				X	EA of the instruction access.
2			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, ^HV = 1, PR = 0, IR = 1 and VPM = 0 and the ARPN in the HPT PTE points to a instruction address that resides in secure memory.				X	EA of the instruction access.
3			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0), IR = 1 and the ARPN in the HPT PTE points to a instruction address that resides in secure memory.				X	EA of the instruction access.
9			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and IR = 1 and the RPDB in the PATE points to a (host) radix tree that resides in secure memory.		X			EA of the instruction access.
6			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and IR = 1 and the NLB in the partition-scoped radix PDE that maps the process table entry points to a PDE/PTE that resides in secure memory.		X			EA of the instruction access.



Table C-6. Exception Cases (ISI) (Sheet 3 of 4)

Case	SRR1 bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value
4			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and IR = 1 and the RPN in the partition-scoped Radix PTE that maps the process table entry points to an address that resides in secure memory.		X			EA of the instruction access.
5			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0, and DR = 1 and the RPDB in the process table entry points a (host) radix tree that resides in secure memory.		X			EA of the instruction access.
7			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and IR = 1 and the NLB in the process-scoped Radix PDE points to an PDE/PTE for a instruction address that resides in secure memory.		X			EA of the instruction access.
8			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, HV = 1, effLPID = 0 and IR = 1 and the RPN in the process-scoped Radix PTE points to an instruction address that resides in secure memory.		X			EA of the instruction access.
10			Secure memory violation (SMFCTRL[E] = 1 only) when S = 0, (HV = 1, PR = 0) and IR = 0 and the instruction address attempts to access secure memory.	X				EA of the instruction access.
1	44	RADIX	Unsupported radix tree configuration due to invalid RTS/RPDS values in the process table entry when HR = 1, IR = 1, and effLPID ≠ 0.			X		EA of the instruction access in the process scoped tables.
2			Unsupported radix tree configuration due to invalid ME value in in the process scoped radix tree when HR = 1, IR = 1 and effLPID ≠ 0.			X		EA of the instruction access in the process scoped tables.
3			Unsupported radix tree configuration due to invalid RTS/RPDS in the process table entry when HR = 1, IR = 1 and effLPID = 0.		X			EA of the instruction access in the process scoped tables.
4			Unsupported radix tree configuration due to invalid ME value in the process scoped tree when HR = 1, IR = 1, HV = 1 and effLPID = 0.		X			EA of the instruction access in the process scoped tables.
5			Unsupported radix tree configuration due to invalid ME value in the partition scoped tree which maps the process table entry when HR = 1, IR = 1, HV = 1, and effLPID = 0		X			EA of the instruction access in the process scoped tables.
	45	RC	<b>Note:</b> When HRIIGR = 0b00, all atomic R,C bit updates will be reported as HISI, not ISI.				Never	EA of the instruction address.

**Table C-6. Exception Cases (ISI) (Sheet 4 of 4)**

Case	SRR1 bit	GPRO Name	Description: Set to '1'b when...	Hypervisor Real Mode	Occurs for Single Level Radix (SLR)	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value
1			Attempt to atomically set a R or C bit fails in the process scoped (guest) radix tree PTE which translates the instruction address when HR = 1, IR = 1, and effLPID ≠ 0.			X		EA of the instruction access.
2			Attempt to atomically set a R or C bit fails in the process scoped (host) radix tree PTE which translates the process table entry when HR = 1, IR = 1, HV = 1, and effLPID = 0.		X			EA of the instruction access.
3			Attempt to atomically set a R or C bit fails in the process scoped (host) radix tree PTE which translates the instruction address when HR = 1, IR = 1, HV = 1, and effLPID = 0.		X			EA of the instruction access.
1	46	Guest_Tbl	The hVA of the process table entry or segment table entry cannot be translated when (IR = 1, VPM = 0, and HRIIGR = 0b00).				X	EA of the instruction address.
2			The process table entry is invalid when IR = 1 and HRIIGR = 0b00 (independent of VPM).				X	EA of the instruction address.
	47:63		Never					





Table C-7. Exception Cases (Machine Check)

Case	SRR1(42)	SRR1 (36, 43:45)	DSISR Bit Set	Description: Set to '1'b when...	Occurs for Single Level Radix	Occurs for Radix on Radix	Occurs for HPT	SRR0 Value	DAR Value	ASDR Value
1	1	0b0000	59	hRA(11) = '1'b when DR = 1 (independent of HV) for translation table look up		X		EA of the instruction access	Undefined	Undefined
2	1	0b0000	59	hRA(11) = '1'b when DR = 1 and effLPID = 0 for translation table look up	X			EA of the instruction access	Undefined	Undefined
3	0	0b1111		hRA(11) = '1'b when IR = 1 (independent of HV) for translation table look up		X		EA of the instruction access	Undefined	Undefined
4	0	0b1111		hRA(11) = '1'b when IR = 1 and effLPID = 0 for translation table look up	X			EA of the instruction access	Undefined	Undefined
5	0	0b0111		hRA(11) = '1'b when IR = 1 (independent of HV) for instruction fetch		X		EA of the instruction access	Undefined	Undefined
6	0	0b0111		hRA(11) = '1'b when IR = 1 and effLPID = 0 for instruction fetch	X			EA of the instruction access	Undefined	Undefined
7	0	0b0111		hRA(11) = '1'b when IR = 1 and effLPID ≠ 0 for instruction fetch	X			EA of the instruction access	Undefined	Undefined
8	0	0b0111		hRA(11) = '1'b when IR = 1 for instruction fetch			X	EA of the instruction access	Undefined	Undefined
9	1	0b0000	60	hRA(11) = '1'b for data access (DR = 1 and VPM = 1 and HV = 0) or DR = 0			X	EA of the data access	Undefined	Undefined
10	1	0b0000	60	hRA(11) = '1'b for data access when DR = 1 and effLPID ≠ 0		X		EA of the data access	Undefined	Undefined
11	1	0b0000	60	hRA(11) = '1'b for data access when DR = 0 and LPIDR ≠ 0. HV = 0 case (guest real mode).	X			EA of the data access	Undefined	Undefined
12	1	0b0000	60	hRA(11) = '1'b for data access when DR = 0 and HRM	X			EA of the data access	Undefined	Undefined



## Appendix D. **tlbie(l)** Encodings

*Table D-1.* on page 423 lists the **tlbie** encodings with GTSE = '1'. *Table D-2.* on page 433 lists the **tlbiel** encodings.

### Notes:

1. The red rows result in a privileged instruction interrupt.
2. HR was "R" in POWER9 hardware. In Power10 hardware, HR and R are separate values which resulted in doubling each of the 5 left-most columns for each value of R (but R was shown on the right half of the table) because it only comes into play now later. In DD 2.x, the effR value is computed as effR = HR when HV = 0 and effR = R when HV = 1.





Table D-1. TLBIE with GTSE = 1 (Sheet 1 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	0	0	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	0	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	1	0	NO	NO	
0	0	0	0	1									NO	0	0	YES	YES	
0	0	0	0	1									NO	1	0	YES	YES	
0	0	0	0	2									NO	0	0	YES	YES	
0	0	0	0	2									NO	1	0	YES	YES	
0	0	0	0	3	YES	YES	LPIDR	NO	AP = 110: 8 consecutive 4 KB pages aligned on 32 KB boundary AP = 111: 8 consecutive 64 KB pages aligned on 512 KB boundary	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	0	0	3	YES	YES	LPIDR	NO	AP = 110: 8 consecutive 4 KB pages aligned on 32 KB boundary AP = 111: 8 consecutive 64 KB pages aligned on 512 KB boundary	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	1	0	NO	NO	
0	0	0	1	0									NO	0	0	YES	YES	
0	0	0	1	0									NO	1	0	YES	YES	
0	0	0	1	1									NO	0	0	YES	YES	
0	0	0	1	1									NO	1	0	YES	YES	
0	0	0	1	2									NO	0	0	YES	YES	
0	0	0	1	2									NO	1	0	YES	YES	
0	0	0	1	3									NO	0	0	YES	YES	
0	0	0	1	3									NO	1	0	YES	YES	
0	0	1	0	0	YES	YES	RS	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	1	0	0	YES	YES	RS	NO	Partition-scoped entry with gRA/hEA match	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
0	0	1	0	1									NO	0	0	YES	YES	
0	0	1	0	1									NO	1	1	YES	YES	
0	0	1	0	2									NO	0	0	YES	YES	
0	0	1	0	2									NO	1	1	YES	YES	
0	0	1	0	3	YES	YES	RS	NO	AP = 110: 8 consecutive 4 KB pages aligned on 32 KB boundary AP = 111: 8 consecutive 64 KB pages aligned on 512 KB boundary	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	1	0	3									NO	1	1	NO	YES	X
0	0	1	1	0									NO	0	0	YES	YES	

Table D-1. TLBIE with GTSE = 1 (Sheet 2 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	0	1	1	0	YES	YES	RS	YES	Guest entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	1	1	YES	NO	X
0	0	1	1	1									NO	0	0	YES	YES	
0	0	1	1	1									NO	1	1	YES	YES	
0	0	1	1	2									NO	0	0	YES	YES	
0	0	1	1	2									NO	1	1	YES	YES	
0	0	1	1	3									NO	0	0	YES	YES	
0	0	1	1	3									NO	1	1	YES	YES	
0	1	0	0	0									NO	0	0	YES	YES	
0	1	0	0	0									NO	1	0	YES	YES	
0	1	0	0	1									NO	0	0	YES	YES	
0	1	0	0	1									NO	1	0	YES	YES	
0	1	0	0	2									NO	0	0	YES	YES	
0	1	0	0	2									NO	1	0	YES	YES	
0	1	0	0	3									NO	0	0	YES	YES	
0	1	0	0	3									NO	1	0	YES	YES	
0	1	0	1	0									NO	0	0	YES	YES	
0	1	0	1	0									NO	1	0	YES	YES	
0	1	0	1	1									NO	0	0	YES	YES	
0	1	0	1	1									NO	1	0	YES	YES	
0	1	0	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
0	1	0	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	1	0	NO	NO	
0	1	0	1	3									NO	0	0	YES	YES	
0	1	0	1	3									NO	1	0	YES	YES	
0	1	1	0	0									NO	0	0	YES	YES	
0	1	1	0	0									NO	1	1	YES	YES	
0	1	1	0	1									NO	0	0	YES	YES	
0	1	1	0	1									NO	1	1	YES	YES	
0	1	1	0	2									NO	0	0	YES	YES	
0	1	1	0	2									NO	1	1	YES	YES	
0	1	1	0	3									NO	0	0	YES	YES	
0	1	1	0	3									NO	1	1	YES	YES	
0	1	1	1	0									NO	0	0	YES	YES	





Table D-1. TLBIE with GTSE = 1 (Sheet 3 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	1	1	1	0	NO	YES	RS	YES	Guest entries with matching PID if LPID ≠ 0 Process-scoped host entries with matching PID if LPID = 0	NO	YES - All with matching LPID and PID	NO	NO	1	1	YES	NO	X
0	1	1	1	1									NO	0	0	YES	YES	
0	1	1	1	1	NO	YES	RS	YES	NO	YES - all process-scoped entries	NO	NO	NO	1	1	YES	NO	X
0	1	1	1	2	NO	YES	RS	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
0	1	1	1	2	NO	YES	RS	YES	Process-scoped entries with matching PID	YES - all process-scoped entries	YES - All with matching LPID and PID	YES - Process Table Caching	NO	1	1	NO	NO	
0	1	1	1	3									NO	0	0	YES	YES	
0	1	1	1	3									NO	1	1	YES	YES	
0	2	0	0	0	NO	YES	LPIDR	NO	Partition-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	0	0	NO	NO	
0	2	0	0	0	NO	YES	LPIDR	NO	Partition-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	1	0	NO	NO	
0	2	0	0	1									NO	0	0	YES	YES	
0	2	0	0	1									NO	1	0	YES	YES	
0	2	0	0	2									NO	0	0	YES	YES	
0	2	0	0	2									NO	1	0	YES	YES	
0	2	0	0	3									NO	0	0	YES	YES	
0	2	0	0	3									NO	1	0	YES	YES	
0	2	0	1	0									NO	0	0	YES	YES	
0	2	0	1	0									NO	1	0	YES	YES	
0	2	0	1	1									NO	0	0	YES	YES	
0	2	0	1	1									NO	1	0	YES	YES	
0	2	0	1	2	NO	YES	LPIDR	NO	NO (IS=2, effR = 0 and PRS = 1)	NO (effR = 0)	YES - All with matching LPID	YES - Process Table Entry	NO	0	0	NO	NO	
0	2	0	1	2	NO	YES	LPIDR	NO	NO (IS=2, effR = 0 and PRS = 1)	NO (effR = 0)	YES - All with matching LPID	YES - Process Table Entry	NO	1	0	NO	NO	
0	2	0	1	3									NO	0	0	YES	YES	
0	2	0	1	3									NO	1	0	YES	YES	
0	2	1	0	0	NO	YES	RS	NO	Partition-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	0	0	NO	NO	
0	2	1	0	0	NO	YES	RS	NO	Partition-scoped entries with matching LPID	NO	YES - All with matching LPID	NO	NO	1	1	NO	NO	
0	2	1	0	1									NO	0	0	YES	YES	

Table D-1. TLBIE with GTSE = 1 (Sheet 4 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	2	1	0	1	NO	YES	RS	NO	NO	YES - invalidate partition-scoped entries with matching LPID	NO	NO	NO	1	1	YES	NO	X
0	2	1	0	2	NO	YES	RS	NO	Partition-scoped entries with match- ing LPID	NO (effR = 0)	YES - All with matching LPID	YES - Partition Table Entry	NO	0	0	NO	NO	
0	2	1	0	2	NO	YES	RS	NO	Partition-scoped entries with match- ing LPID	YES - invalidate partition-scoped entries with matching LPID	YES - All with matching LPID	YES - Partition Table Entry	NO	1	1	NO	NO	
0	2	1	0	3									NO	0	0	YES	YES	
0	2	1	0	3									NO	1	1	YES	YES	
0	2	1	1	0									NO	0	0	YES	YES	
0	2	1	1	0	NO	YES	RS	NO	Guest entries with matching LPID if LPID ≠ 0 Process-scoped host entries with matching LPID if LPID = 0			NO	NO	1	1	YES	NO	X
0	2	1	1	1									NO	0	0	YES	YES	
0	2	1	1	1	NO	YES	RS	NO	NO	YES - all process- scoped entries with matching LPID	NO	NO	NO	1	1	YES	NO	X
0	2	1	1	2	NO	YES	RS	NO	NO (IS=2, effR = 0 and PRS = 1)	NO (effR = 0)	YES - All with matching LPID	YES - Process Table Entry	NO	0	0	NO	NO	
0	2	1	1	2	NO	YES	RS	NO	Process-scoped entries with matching PID	YES - all process- scoped entries with matching LPID	YES - All with matching LPID	YES - Process Table Entry	NO	1	1	NO	NO	
0	2	1	1	3									NO	0	0	YES	YES	
0	2	1	1	3									NO	1	1	YES	YES	
0	3	0	0	0	NO	YES	LPIDR	NO	Partition-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	0	0	NO	NO	
0	3	0	0	0	NO	YES	LPIDR	NO	Partition-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	1	0	NO	NO	
0	3	0	0	1									NO	0	0	YES	YES	
0	3	0	0	1									NO	1	0	YES	YES	
0	3	0	0	2									NO	0	0	YES	YES	
0	3	0	0	2									NO	1	0	YES	YES	
0	3	0	0	3									NO	0	0	YES	YES	
0	3	0	0	3									NO	1	0	YES	YES	
0	3	0	1	0									NO	0	0	YES	YES	
0	3	0	1	0									NO	1	0	YES	YES	
0	3	0	1	1									NO	0	0	YES	YES	
0	3	0	1	1									NO	1	0	YES	YES	
0	3	0	1	1									NO	1	0	YES	YES	





Table D-1. TLBIE with GTSE = 1 (Sheet 5 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	3	0	1	2	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	YES - Process Table Entry	NO	0	0	NO	NO	
0	3	0	1	2	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	NO (effR = 0)	YES - All with matching LPID	YES - Process Table Entry	NO	1	0	NO	NO	
0	3	0	1	3									NO	0	0	YES	YES	
0	3	0	1	3									NO	1	0	YES	YES	
0	3	1	0	0	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES (IS = 3, HV = 1, effR is ignored)	YES - regardless of LPID and effR for IS = 3 and HV = 1	NO	NO	0	0	NO	NO	
0	3	1	0	0	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	NO	YES - regardless of LPID	NO	NO	1	1	NO	NO	
0	3	1	0	1									NO	0	0	YES	YES	
0	3	1	0	1	NO	NO	RS	NO	NO	YES - invalidate all Partition-scoped entries regardless of LPID	NO	NO	NO	1	1	YES	NO	X
0	3	1	0	2	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES - invalidate all Partition-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	YES - regardless of LPID and effR for IS = 3 and HV = 1	YES - Partition Table Entry regardless of LPID	NO	0	0	NO	NO	
0	3	1	0	2	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES - invalidate all Partition-scoped entries regardless of LPID	YES - regardless of LPID	YES - Partition Table Entry regardless of LPID	NO	1	1	NO	NO	
0	3	1	0	3									NO	0	0	YES	YES	
0	3	1	0	3									NO	1	1	YES	YES	
0	3	1	1	0									NO	0	0	YES	YES	
0	3	1	1	0	NO	NO	RS		All process-scoped entries regardless of LPID	NO	YES - regardless of LPID	NO	NO	1	1	YES	NO	X
0	3	1	1	1									NO	0	0	YES	YES	
0	3	1	1	1	NO	NO	RS	NO	NO	YES - all process-scoped entries	NO	YES - Process Table Entry	NO	1	1	YES	NO	X
0	3	1	1	2	NO	NO	RS	NO	All process-scoped entries regardless of LPID	YES - all process-scoped entries (IS = 3, HV = 1, effR is ignored)	YES - regardless of LPID and effR for IS = 3 and HV = 1	YES - Process Table Entry	NO	0	0	NO	NO	
0	3	1	1	2	NO	NO	RS	NO	All process-scoped entries regardless of LPID	YES - all process-scoped entries	YES - regardless of LPID	YES - Process Table Entry	NO	1	1	NO	NO	
0	3	1	1	3									NO	0	0	YES	YES	
0	3	1	1	3									NO	1	1	YES	YES	
1	0	0	0	0									YES	0	1	NO	NO	
1	0	0	0	0									YES	1	1	NO	NO	
1	0	0	0	1									YES	0	1	YES	YES	

Table D-1. TLBIE with GTSE = 1 (Sheet 6 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	0	0	0	1									YES	1	1	YES	YES	
1	0	0	0	2									YES	0	1	YES	YES	
1	0	0	0	2									YES	1	1	YES	YES	
1	0	0	0	3									YES	0	1	YES	YES	
1	0	0	0	3									YES	1	1	YES	YES	
1	0	0	1	0	YES	YES	LPIDR	YES	Guest entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	0	1	NO	NO	
1	0	0	1	0	YES	YES	LPIDR	YES	Guest entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	0	1	1									NO	0	1	YES	YES	
1	0	0	1	1									NO	1	1	YES	YES	
1	0	0	1	2									NO	0	1	YES	YES	
1	0	0	1	2									NO	1	1	YES	YES	
1	0	0	1	3									NO	0	1	YES	YES	
1	0	0	1	3									NO	1	1	YES	YES	
1	0	1	0	0	YES	YES	RS	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
1	0	1	0	0	YES	YES	RS	NO	Partition-scoped entry with gRA/hEA match	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	1	0	1									NO	0	0	YES	YES	
1	0	1	0	1									NO	1	1	YES	YES	
1	0	1	0	2									NO	0	0	YES	YES	
1	0	1	0	2									NO	1	1	YES	YES	
1	0	1	0	3	YES	YES	RS	NO	AP = 110: 8 consecutive 4 KB pages aligned on 32 KB boundary AP = 111: 8 consecutive 64 KB pages aligned on 512 KB boundary	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	YES	NO	X
1	0	1	0	3									NO	1	1	YES	YES	
1	0	1	1	0									NO	0	0	NO	YES	X
1	0	1	1	0	YES	YES	RS	Yes	Guest entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	1	1	1									NO	0	0	YES	YES	
1	0	1	1	1									NO	1	1	YES	YES	
1	0	1	1	2									NO	0	0	YES	YES	
1	0	1	1	2									NO	1	1	YES	YES	





Table D-1. TLBIE with GTSE = 1 (Sheet 7 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?	
1	0	1	1	3									NO	0	0	YES	YES		
1	0	1	1	3									NO	1	1	YES	YES		
1	1	0	0	0									YES	0	1	YES	YES		
1	1	0	0	0									YES	1	1	YES	YES		
1	1	0	0	1									YES	0	1	YES	YES		
1	1	0	0	1									YES	1	1	YES	YES		
1	1	0	0	2									YES	0	1	YES	YES		
1	1	0	0	2									YES	1	1	YES	YES		
1	1	0	0	3									YES	0	1	YES	YES		
1	1	0	0	3									YES	1	1	YES	YES		
1	1	0	1	0	NO	YES	LPIDR	YES	Guest entries with matching PID (for a matching LPID)	NO	YES - All with matching LPID and PID		NO	0	1	NO	NO		
1	1	0	1	0	NO	YES	LPIDR	YES	Guest entries with matching PID (for a matching LPID)	NO	YES - All with matching LPID and PID		NO	1	1	NO	NO		
1	1	0	1	1	NO	YES	LPIDR	YES	NO	YES - all process-scoped entries		NO	NO	0	1	NO	NO		
1	1	0	1	1	NO	YES	LPIDR	YES	NO	YES - all process-scoped entries		NO	NO	1	1	NO	NO		
1	1	0	1	2	NO	YES	LPIDR	YES	Process-scoped entries with matching PID (for a matching LPID)	YES - all process-scoped entries	YES - All with matching LPID and PID	YES - Process Table Caching	NO	0	1	NO	NO		
1	1	0	1	2	NO	YES	LPIDR	YES	Process-scoped entries with matching PID (for a matching LPID)	YES - all process-scoped entries	YES - All with matching LPID and PID	YES - Process Table Caching	NO	1	1	NO	NO		
1	1	0	1	3									NO	0	1	YES	YES		
1	1	0	1	3									NO	1	1	YES	YES		
1	1	1	0	0									NO	0	0	YES	YES		
1	1	1	0	0									NO	1	1	YES	YES		
1	1	1	0	1									NO	0	0	YES	YES		
1	1	1	0	1									NO	1	1	YES	YES		
1	1	1	0	2									NO	0	0	YES	YES		
1	1	1	0	2									NO	1	1	YES	YES		
1	1	1	0	3									NO	0	0	YES	YES		
1	1	1	0	3									NO	1	1	YES	YES		
1	1	1	1	0									NO	0	0	NO	YES	X	
1	1	1	1	0	NO	YES	RS	YES	Guest entries with matching PID if LPID ≠ 0 Process-scoped host entries with matching PID if LPID = 0	NO	YES - All with matching LPID and PID		NO	1	1	NO	NO		
1	1	1	1	1	1								NO	0	0	NO	YES	X	
1	1	1	1	1	1	NO	YES	RS	YES	NO	YES - all process-scoped entries		NO	NO	1	1	NO	NO	

Table D-1. TLBIE with GTSE = 1 (Sheet 8 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	1	1	1	2	NO	YES	RS	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
1	1	1	1	2	NO	YES	RS	YES	Process-scoped entries with matching PID	YES - all process- scoped entries	YES - All with matching LPID and PID	YES - Process Table Caching	NO	1	1	NO	NO	
1	1	1	1	3									NO	0	0	YES	YES	
1	1	1	1	3									NO	1	1	YES	YES	
1	2	0	0	0									YES	0	1	NO	NO	
1	2	0	0	0									YES	1	1	NO	NO	
1	2	0	0	1									YES	0	1	NO	NO	
1	2	0	0	1									YES	1	1	NO	NO	
1	2	0	0	2									YES	0	1	NO	NO	
1	2	0	0	2									YES	1	1	NO	NO	
1	2	0	0	3									YES	0	1	YES	YES	
1	2	0	0	3									YES	1	1	YES	YES	
1	2	0	1	0	NO	YES	LPIDR	NO	Guest entries with matching LPID	NO	YES - All with matching LPID	NO	NO	0	1	NO	NO	
1	2	0	1	0	NO	YES	LPIDR	NO	Guest entries with matching LPID	NO	YES - All with matching LPID	NO	NO	1	1	NO	NO	
1	2	0	1	1	NO	YES	LPIDR	NO	NO	YES - all process- scoped entries with matching LPID	NO	NO	NO	0	1	NO	NO	
1	2	0	1	1	NO	YES	LPIDR	NO	NO	YES - all process- scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	0	1	2	NO	YES	LPIDR	NO	Process-scoped entries with matching LPID	YES - all process- scoped entries with matching LPID	YES - All with matching LPID	YES - Process Table Entry	NO	0	1	NO	NO	
1	2	0	1	2	NO	YES	LPIDR	NO	Process-scoped entries with matching LPID	YES - all process- scoped entries with matching LPID	YES - All with matching LPID	YES - Process Table Entry	NO	1	1	NO	NO	
1	2	0	1	3									NO	0	1	YES	YES	
1	2	0	1	3									NO	1	1	YES	YES	
1	2	1	0	0	NO	YES	RS	NO	Partition-scoped entries with match- ing LPID	NO (effR = 0)	YES - All with matching LPID	NO	NO	0	0	NO	NO	
1	2	1	0	0	NO	YES	RS	NO	Partition-scoped entries with match- ing LPID	NO	YES - All with matching LPID	NO	NO	1	1	NO	NO	
1	2	1	0	1									NO	0	0	NO	YES	X
1	2	1	0	1	NO	YES	RS	NO	NO	YES - invalidate Par- tition-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	1	0	2	NO	YES	RS	NO	Partition-scoped entries with match- ing LPID	NO (effR = 0)	YES - All with matching LPID	YES - Partition Table Entry	NO	0	0	NO	NO	





Table D-1. TLBIE with GTSE = 1 (Sheet 9 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	2	1	0	2	NO	YES	RS	NO	Partition-scoped entries with matching LPID	YES - invalidate Partition-scoped entries with matching LPID	YES - All with matching LPID	YES - Partition Table Entry	NO	1	1	NO	NO	
1	2	1	0	3									NO	0	0	YES	YES	
1	2	1	0	3									NO	1	1	YES	YES	
1	2	1	1	0									NO	0	0	NO	YES	X
1	2	1	1	0	NO	YES	RS	NO	Guest entries with matching LPID if LPID ≠ 0 Process-scoped host entries with matching LPID if LPID = 0	NO	YES - all with matching LPID	NO	NO	1	1	NO	NO	
1	2	1	1	1									NO	0	0	NO	YES	X
1	2	1	1	1	NO	YES	RS	NO	NO	YES - all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	1	1	2	NO	YES	RS	NO	NO (IS = 2, effR = 0 and PRS = 1)	NO (effR = 0)	YES - all with matching LPID	YES - Process Table Entry	NO	0	0	NO	NO	
1	2	1	1	2	NO	YES	RS	NO	Process-scoped entries with matching PID	YES - all process-scoped entries with matching LPID	YES - all with matching LPID	YES - Process Table Entry	NO	1	1	NO	NO	
1	2	1	1	3									NO	0	0	YES	YES	
1	2	1	1	3									NO	1	1	YES	YES	
1	3	0	0	0									YES	0	1	NO	NO	
1	3	0	0	0									YES	1	1	NO	NO	
1	3	0	0	1									YES	0	1	NO	NO	
1	3	0	0	1									YES	1	1	NO	NO	
1	3	0	0	2									YES	0	1	NO	NO	
1	3	0	0	2									YES	1	1	NO	NO	
1	3	0	0	3									YES	0	1	YES	YES	
1	3	0	0	3									YES	1	1	YES	YES	
1	3	0	1	0	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	NO	YES - all with matching LPID	NO	NO	0	1	NO	NO	
1	3	0	1	0	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	NO	YES - all with matching LPID	NO	NO	1	1	NO	NO	
1	3	0	1	1	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	YES - all process-scoped entries with matching LPID	NO	NO	NO	0	1	NO	NO	
1	3	0	1	1	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	YES - all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	3	0	1	2	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	YES - all process-scoped entries with matching LPID	YES - Process Table Entry	NO	NO	0	1	NO	NO	

Table D-1. TLBIE with GTSE = 1 (Sheet 10 of 10)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process / Partition Table Caching Invalidated? (effR is Ignored)	Privileged Instruction Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	3	0	1	2	NO	YES	LPIDR	NO	All process-scoped entries with matching LPID	YES - all process-scoped entries with matching LPID	YES - all with matching LPID	YES - Process Table Entry	NO	1	1	NO	NO	
1	3	0	1	3									NO	0	1	YES	YES	
1	3	0	1	3									NO	1	1	YES	YES	
1	3	1	0	0	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES (IS = 3, HV = 1, effR is ignored)	YES - regardless of LPID and effR for IS = 3 and HV = 1	NO	NO	0	0	NO	NO	
1	3	1	0	0	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	NO	YES - Regardless of LPID	NO	NO	1	1	NO	NO	
1	3	1	0	1									NO	0	0	NO	YES	X
1	3	1	0	1	NO	NO	RS	NO	NO	YES - invalidate all Partition-scoped entries regardless of LPID	NO	NO	NO	1	1	NO	NO	
1	3	1	0	2	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES - invalidate all Partition-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	YES (Regardless of LPID and effR for IS = 3 and HV = 1)	YES - Partition Table Entry regardless of LPID	NO	0	0	NO	NO	
1	3	1	0	2	NO	NO	RS	NO	All partition-scoped entries regardless of LPID	YES - invalidate all Partition-scoped entries regardless of LPID	YES - Regardless of LPID	YES - Partition Table Entry regardless of LPID	NO	1	1	NO	NO	
1	3	1	0	3									NO	0	0	YES	YES	
1	3	1	0	3									NO	1	1	YES	YES	
1	3	1	1	0									NO	0	0	NO	YES	X
1	3	1	1	0	NO	NO	RS	NO	All process-scoped entries regardless of LPID	NO	YES - Regardless of LPID	NO	NO	1	1	NO	NO	
1	3	1	1	1									NO	0	0	NO	YES	X
1	3	1	1	1	NO	NO	RS	NO	NO	YES - all process-scoped entries	NO	YES - Process Table Entry	NO	1	1	NO	NO	
1	3	1	1	2	NO	NO	RS	NO	All process-scoped entries regardless of LPID	YES - all process-scoped entries (IS = 3, HV = 1, effR is ignored)	YES (Regardless of LPID and effR for IS = 3 and HV = 1)	YES - Process Table Entry	NO	0	0	NO	NO	
1	3	1	1	2	NO	NO	RS	NO	All process-scoped entries regardless of LPID	YES - all process-scoped entries	YES - Regardless of LPID	YES - Process Table Entry	NO	1	1	NO	NO	
1	3	1	1	3									NO	0	0	YES	YES	
1	3	1	1	3									NO	1	1	YES	YES	





**Table D-2. *tibiael* Encodings (Sheet 1 of 11)**

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	0	0	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	0	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	1	0	NO	NO	
0	0	0	0	1									NO	0	0	YES	YES	
0	0	0	0	1									NO	1	0	YES	YES	
0	0	0	0	2									NO	0	0	YES	YES	
0	0	0	0	2									NO	1	0	YES	YES	
0	0	0	0	3									NO	0	0	YES	YES	
0	0	0	0	3									NO	1	0	YES	YES	
0	0	0	1	0									NO	0	0	YES	YES	
0	0	0	1	0									NO	1	0	YES	YES	
0	0	0	1	1									NO	0	0	YES	YES	
0	0	0	1	1									NO	1	0	YES	YES	
0	0	0	1	2									NO	0	0	YES	YES	
0	0	0	1	2									NO	1	0	YES	YES	
0	0	0	1	3									NO	0	0	YES	YES	
0	0	0	1	3									NO	1	0	YES	YES	
0	0	1	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
0	0	1	0	0	YES	YES	LPIDR	NO	Partition-scoped host entry with gRA/hEA match	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
0	0	1	0	1									NO	0	0	YES	YES	
0	0	1	0	1									NO	1	1	YES	YES	
0	0	1	0	2									NO	0	0	YES	YES	
0	0	1	0	2									NO	1	1	YES	YES	
0	0	1	0	3									NO	0	0	YES	YES	
0	0	1	0	3									NO	1	1	YES	YES	
0	0	1	1	0									NO	0	0	YES	YES	
0	0	1	1	0	YES	YES	LPIDR	YES	Process-scoped entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	1	1	YES	NO	X
0	0	1	1	1									NO	0	0	YES	YES	
0	0	1	1	1									NO	1	1	YES	YES	
0	0	1	1	2									NO	0	0	YES	YES	
0	0	1	1	2									NO	1	1	YES	YES	
0	0	1	1	3									NO	0	0	YES	YES	

**Table D-2. *tblel* Encodings (Sheet 2 of 11)**

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 <b>(red)</b>	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	0	1	1	3									NO	1	1	YES	YES	
0	1	0	0	0									NO	0	0	YES	YES	
0	1	0	0	0									NO	1	0	YES	YES	
0	1	0	0	1									NO	0	0	YES	YES	
0	1	0	0	1									NO	1	0	YES	YES	
0	1	0	0	2									NO	0	0	YES	YES	
0	1	0	0	2									NO	1	0	YES	YES	
0	1	0	0	3									NO	0	0	YES	YES	
0	1	0	0	3									NO	1	0	YES	YES	
0	1	0	1	0									NO	0	0	YES	YES	
0	1	0	1	0									NO	1	0	YES	YES	
0	1	0	1	1									NO	0	0	YES	YES	
0	1	0	1	1									NO	1	0	YES	YES	
0	1	0	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
0	1	0	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	1	0	NO	NO	
0	1	0	1	3									NO	0	0	YES	YES	
0	1	0	1	3									NO	1	0	YES	YES	
0	1	1	0	0									NO	0	0	YES	YES	
0	1	1	0	0									NO	1	1	YES	YES	
0	1	1	0	1									NO	0	0	YES	YES	
0	1	1	0	1									NO	1	1	YES	YES	
0	1	1	0	2									NO	0	0	YES	YES	
0	1	1	0	2									NO	1	1	YES	YES	
0	1	1	0	3									NO	0	0	YES	YES	
0	1	1	0	3									NO	1	1	YES	YES	
0	1	1	1	0									NO	0	0	YES	YES	
0	1	1	1	0	0	YES	LPIDR	YES	When SET = 0, invalidate all Process-scoped entries with matching PID (for a matching LPID)	NO	When SET = 0, then invalidate based on context tag for all threads.	NO	NO	1	1	YES	NO	X
0	1	1	1	1									NO	0	0	YES	YES	
0	1	1	1	1	NO	YES	LPIDR	YES	NO	When SET = 0, invalidate all process- scoped entries with matching LPID	NO	NO	NO	1	1	YES	NO	X





Table D-2. ***tblel*** Encodings (Sheet 3 of 11)

Version 1.0  
13 September 2021

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	1	1	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
0	1	1	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invalidate all process-scoped entries with matching LPID	When SET = 0, then invalidate based on context tag for all threads.	When SET = 0, then invalidate process table caching for all threads	NO	1	1	NO	NO	
0	1	1	1	3									NO	0	0	YES	YES	
0	1	1	1	3									NO	1	1	YES	YES	
0	2	0	0	0	NO	YES	LPIDR	NO	When SET = 0, then for all entries: process-scoped entries with matching PID if LPID ≠ 0 Partition-scoped host entries with matching PID if LPID = 0	NO (effR = 0)	When SET = 0, then invalidate based on context tag for all threads.	NO	NO	0	0	NO	NO	
0	2	0	0	0	NO	YES	LPIDR	NO	When SET = 0, then for all entries: process-scoped entries with matching PID if LPID ≠ 0 Process-scoped host entries with matching PID if LPID = 0	NO (effR = 0)	When SET = 0, then invalidate based on context tag for all threads.	NO	NO	1	0	NO	NO	
0	2	0	0	1									NO	0	0	YES	YES	
0	2	0	0	1									NO	1	0	YES	YES	
0	2	0	0	2									NO	0	0	YES	YES	
0	2	0	0	2									NO	1	0	YES	YES	
0	2	0	0	3									NO	0	0	YES	YES	
0	2	0	0	3									NO	1	0	YES	YES	
0	2	0	0	3									NO	0	0	YES	YES	
0	2	0	1	0									NO	0	0	YES	YES	
0	2	0	1	0									NO	1	0	YES	YES	
0	2	0	1	1									NO	0	0	YES	YES	
0	2	0	1	1									NO	1	0	YES	YES	
0	2	0	1	2	NO	YES	LPIDR	NO	NO (IS = 2, effR = 0, and PRS = 1)	NO (effR = 0)	YES	When SET = 0, then invalidate all process scoped entries	NO	0	0	NO	NO	
0	2	0	1	2	NO	YES	LPIDR	NO	NO (IS = 2, effR = 0, and PRS = 1)	NO (effR = 0)	YES	When SET = 0, then invalidate all process scoped entries	NO	1	0	NO	NO	
0	2	0	1	3									NO	0	0	YES	YES	
0	2	0	1	3									NO	1	0	YES	YES	
0	2	1	0	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO (effR = 0)	When SET = 0, then invalidate based on context tag for all threads.	NO	NO	0	0	NO	NO	
0	2	1	0	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO	When SET = 0, then invalidate based on context tag for all threads.	NO	NO	1	1	NO	NO	
0	2	1	0	1									NO	0	0	YES	YES	

**Table D-2. *tblel* Encodings (Sheet 4 of 11)**

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	2	1	0	1	NO	YES	LPIDR	NO	NO	When SET = 0, invali- date all partition-scoped entries with matching LPID	NO	NO	NO	1	1	YES	NO	X
0	2	1	0	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO (effR = 0)	When SET = 0, then invalidate based on context tag for all threads.	When SET = 0, then invalidate all partition scoped entries	NO	0	0	NO	NO	
0	2	1	0	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	When SET = 0, invali- date all partition-scoped entries with matching LPID	When SET = 0, invalidate based on context tag for all threads.	When SET = 0, invalidate all partition-scoped entries	NO	1	1	NO	NO	
0	2	1	0	3									NO	0	0	YES	YES	
0	2	1	0	3									NO	1	1	YES	YES	
0	2	1	1	0									NO	0	0	YES	YES	
0	2	1	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO			NO	1	1	YES	NO	X
0	2	1	1	1									NO	0	0	YES	YES	
0	2	1	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invalidate all process- scoped entries with matching LPID	NO	NO	NO	1	1	YES	NO	X
0	2	1	1	2	NO	YES	LPIDR	NO	NO (IS=2, effR = 0 and PRS = 1)	NO (effR = 0)	YES	When SET = 0, then invalidate all process-scoped entries	NO	0	0	NO	NO	
0	2	1	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invalidate all process- scoped entries with matching LPID	YES	When SET = 0, then invalidate all process-scoped entries	NO	1	1	NO	NO	
0	2	1	1	3									NO	0	0	YES	YES	
0	2	1	1	3									NO	1	1	YES	YES	
0	3	0	0	0	NO	YES	LPIDR	NO	If SET = 0, then invali- date partition-scoped entries with matching LPID	NO (effR = 0)	If SET = 0, then invalidate all entries with matching context tag	NO	NO	0	0	NO	NO	
0	3	0	0	0	NO	YES	LPIDR	NO	If SET = 0, then invali- date partition-scoped entries with matching LPID	NO (effR = 0)	If SET = 0, then invalidate all entries with matching context tag	NO	NO	1	0	NO	NO	
0	3	0	0	1									NO	0	0	YES	YES	
0	3	0	0	1									NO	1	0	YES	YES	
0	3	0	0	2									NO	0	0	YES	YES	
0	3	0	0	2									NO	1	0	YES	YES	
0	3	0	0	3									NO	0	0	YES	YES	
0	3	0	0	3									NO	1	0	YES	YES	
0	3	0	1	0									NO	0	0	YES	YES	
0	3	0	1	0									NO	0	0	YES	YES	
0	3	0	1	0									NO	1	0	YES	YES	



Table D-2. ***tibiae*** Encodings (Sheet 5 of 11)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 <b>(red)</b>	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	3	0	1	1									NO	0	0	YES	YES	
0	3	0	1	1									NO	1	0	YES	YES	
0	3	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO (effR = 0)	YES	When SET = 0, invalidate all process table caching with matching LPID	NO	0	0	NO	NO	
0	3	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all Process-Scoped entries with matching LPID	NO (effR = 0)	YES	When SET = 0, invalidate all process table caching with matching LPID	NO	1	0	NO	NO	
0	3	0	1	3									NO	0	0	YES	YES	
0	3	0	1	3									NO	1	0	YES	YES	
0	3	1	0	0	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all partition-scoped entries (IS = 3, HV = 1, effR is ignored)	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	NO	NO	0	0	NO	NO	
0	3	1	0	0	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	NO	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	NO	NO	1	1	NO	NO	
0	3	1	0	1									NO	0	0	YES	YES	
0	3	1	0	1	NO	YES	LPIDR	NO	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	NO	NO	NO	1	1	YES	NO	X
0	3	1	0	2	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all partition-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	When SET = 0, invalidate all partition table caching regardless of LPID	NO	0	0	NO	NO	
0	3	1	0	2	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	When SET = 0, invalidate all partition table caching regardless of LPID	NO	1	1	NO	NO	
0	3	1	0	3									NO	0	0	YES	YES	
0	3	1	0	3									NO	1	1	YES	YES	
0	3	1	1	0									NO	0	0	YES	YES	
0	3	1	1	0					When SET = 0, invalidate all process-scoped entries regardless of LPID				NO	1	1	YES	NO	X
0	3	1	1	1									NO	0	0	YES	YES	
0	3	1	1	1	NO	NO	LPIDR	NO	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	NO	NO	NO	1	1	YES	NO	X

**Table D-2. *tblel* Encodings (Sheet 6 of 11)**

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
0	3	1	1	2	NO	NO	NA	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	When SET = 0, invalidate all process-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	YES	When SET = 0, invalidate all process table caching regardless of LPID	NO	0	0	NO	NO	
0	3	1	1	2	NO	NO	NA	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	When SET = 0, invalidate all process-scoped entries regardless of LPID	YES	When SET = 0, invalidate all process table caching regardless of LPID	NO	1	1	NO	NO	
0	3	1	1	3									NO	0	0	YES	YES	
0	3	1	1	3									NO	1	1	YES	YES	
1	0	0	0	0									YES	0	1	NO	NO	
1	0	0	0	0									YES	1	1	NO	NO	
1	0	0	0	1									YES	0	1	YES	YES	
1	0	0	0	1									YES	1	1	YES	YES	
1	0	0	0	2									YES	0	1	YES	YES	
1	0	0	0	2									YES	1	1	YES	YES	
1	0	0	0	3									YES	0	1	YES	YES	
1	0	0	0	3									YES	1	1	YES	YES	
1	0	0	1	0	YES	YES	LPIDR	YES	Process-scoped entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	0	1	NO	NO	
1	0	0	1	0	YES	YES	LPIDR	YES	Process-scoped entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	0	1	1									NO	0	1	YES	YES	
1	0	0	1	1									NO	1	1	YES	YES	
1	0	0	1	2									NO	0	1	YES	YES	
1	0	0	1	2									NO	1	1	YES	YES	
1	0	0	1	3									NO	0	1	YES	YES	
1	0	0	1	3									NO	1	1	YES	YES	
1	0	1	0	0	YES	YES	LPIDR	NO	VA match	NO (effR = 0)	YES - matching address, PID, and LPID	NO	NO	0	0	NO	NO	
1	0	1	0	0	YES	YES	LPIDR	NO	Partition-scoped host entry with gRA/hEA match	NO	YES - matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	1	0	1									NO	0	0	YES	YES	
1	0	1	0	1									NO	1	1	YES	YES	
1	0	1	0	2									NO	0	0	YES	YES	
1	0	1	0	2									NO	1	1	YES	YES	
1	0	1	0	3									NO	0	0	YES	YES	





Table D-2. ***tblel*** Encodings (Sheet 7 of 11)

Version 1.0  
13 September 2021

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	0	1	0	3									NO	1	1	YES	YES	
1	0	1	1	0									NO	0	0	NO	YES	X
1	0	1	1	0	YES	YES	LPIDR	Yes	Process-scoped entry with gEA match if LPID ≠ 0 Process-scoped host entry with hEA match if LPID = 0	NO	YES -matching address, PID, and LPID	NO	NO	1	1	NO	NO	
1	0	1	1	1									NO	0	0	YES	YES	
1	0	1	1	1									NO	1	1	YES	YES	
1	0	1	1	2									NO	0	0	YES	YES	
1	0	1	1	2									NO	1	1	YES	YES	
1	0	1	1	3									NO	0	0	YES	YES	
1	0	1	1	3									NO	1	1	YES	YES	
1	1	0	0	0									YES	0	1	YES	YES	
1	1	0	0	0									YES	1	1	YES	YES	
1	1	0	0	1									YES	0	1	YES	YES	
1	1	0	0	1									YES	1	1	YES	YES	
1	1	0	0	2									YES	0	1	YES	YES	
1	1	0	0	2									YES	1	1	YES	YES	
1	1	0	0	3									YES	0	1	YES	YES	
1	1	0	0	3									YES	1	1	YES	YES	
1	1	0	1	0	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	When SET = 0, invalidate based on context tag for all threads.	NO	NO	0	1	NO	NO	
1	1	0	1	0	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	When SET = 0, invalidate based on context tag for all threads.	NO	NO	1	1	NO	NO	
1	1	0	1	1	NO	YES	LPIDR	YES	NO	When SET = 0, invali- date all process-scoped entries with matching LPID	NO	NO	NO	0	1	NO	NO	
1	1	0	1	1	NO	YES	LPIDR	YES	NO	When SET = 0, invali- date all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	1	0	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	When SET = 0, invali- date based on context tag for all threads.	When SET = 0, invali- date process table caching for all threads	NO	0	1	NO	NO	
1	1	0	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	When SET = 0, invali- date based on context tag for all threads.	When SET = 0, invali- date process table caching for all threads	NO	1	1	NO	NO	
1	1	0	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	When SET = 0, invali- date based on context tag for all threads.	When SET = 0, invali- date process table caching for all threads	NO	1	1	NO	NO	
1	1	0	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	When SET = 0, invali- date based on context tag for all threads.	When SET = 0, invali- date process table caching for all threads	NO	1	1	NO	NO	

Table D-2. ***tblel*** Encodings (Sheet 8 of 11)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 <b>(red)</b>	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	1	0	1	3									NO	0	1	YES	YES	
1	1	0	1	3									NO	1	1	YES	YES	
1	1	1	0	0									NO	0	0	YES	YES	
1	1	1	0	0									NO	1	1	YES	YES	
1	1	1	0	1									NO	0	0	YES	YES	
1	1	1	0	1									NO	1	1	YES	YES	
1	1	1	0	2									NO	0	0	YES	YES	
1	1	1	0	2									NO	1	1	YES	YES	
1	1	1	0	3									NO	0	0	YES	YES	
1	1	1	0	3									NO	1	1	YES	YES	
1	1	1	1	0									NO	0	0	NO	YES	X
1	1	1	1	0	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	When SET = 0, invalidate based on context tag for all threads.	NO	NO	1	1	NO	NO	
1	1	1	1	1									NO	0	0	NO	YES	X
1	1	1	1	1	NO	YES	LPIDR	YES	NO	When SET = 0, invalidate all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	1	1	1	2	NO	YES	LPIDR	YES	NO (PRS = 1 and effR = 0)	NO (effR = 0)	NO (PRS = 1 and effR = 0)	NO (PRS = 1 and effR = 0)	NO	0	0	NO	NO	
1	1	1	1	2	NO	YES	LPIDR	YES	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invalidate all process-scoped entries with matching LPID	When SET = 0, invalidate based on context tag for all threads.	When SET = 0, invalidate process table caching for all threads	NO	1	1	NO	NO	
1	1	1	1	3									NO	0	0	YES	YES	
1	1	1	1	3									NO	1	1	YES	YES	
1	2	0	0	0									YES	0	1	NO	NO	
1	2	0	0	0									YES	1	1	NO	NO	
1	2	0	0	1									YES	0	1	NO	NO	
1	2	0	0	1									YES	1	1	NO	NO	
1	2	0	0	2									YES	0	1	NO	NO	
1	2	0	0	2									YES	1	1	NO	NO	
1	2	0	0	3									YES	0	1	YES	YES	
1	2	0	0	3									YES	1	1	YES	YES	
1	2	0	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	YES	NO	NO	0	1	NO	NO	





Table D-2. **tibiae** Encodings (Sheet 9 of 11)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 <b>(red)</b>	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	2	0	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	YES	NO	NO	1	1	NO	NO	
1	2	0	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invalidate all process-scoped entries with matching LPID	NO	NO	NO	0	1	NO	NO	
1	2	0	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invalidate all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invalidate all process-scoped entries with matching LPID	YES	When SET = 0, then invalidate all process-scoped entries	NO	0	1	NO	NO	
1	2	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invalidate all process-scoped entries with matching LPID	YES	When SET = 0, then invalidate all process-scoped entries	NO	1	1	NO	NO	
1	2	0	1	3									NO	0	1	YES	YES	
1	2	0	1	3									NO	1	1	YES	YES	
1	2	1	0	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO (effR = 0)	When SET = 0, invalidate based on context tag for all threads.	NO	NO	0	0	NO	NO	
1	2	1	0	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO	When SET = 0, invalidate based on context tag for all threads.	NO	NO	1	1	NO	NO	
1	2	1	0	1									NO	0	0	NO	YES	X
1	2	1	0	1	NO	YES	LPIDR	NO	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	1	0	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	NO (effR = 0)	When SET = 0, invalidate based on context tag for all threads.	When SET = 0, invalidate all partition-scoped entries	NO	0	0	NO	NO	
1	2	1	0	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all partition-scoped entries with matching LPID	When SET = 0, invalidate all partition-scoped entries with matching LPID	When SET = 0, invalidate based on context tag for all threads.	When SET = 0, invalidate all partition scoped entries	NO	1	1	NO	NO	
1	2	1	0	3									NO	0	0	YES	YES	
1	2	1	0	3									NO	1	1	YES	YES	
1	2	1	1	0									NO	0	0	NO	YES	X
1	2	1	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	YES	NO	NO	1	1	NO	NO	
1	2	1	1	1									NO	0	0	NO	YES	X

**Table D-2. *tblel* Encodings (Sheet 10 of 11)**

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0, PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	2	1	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invali- date all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	2	1	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO (effR = 0)	YES	When SET = 0, invalidate all process-scoped entries	NO	0	0	NO	NO	
1	2	1	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	YES	When SET = 0, invalidate all process-scoped entries	NO	1	1	NO	NO	
1	2	1	1	3									NO	0	0	YES	YES	
1	2	1	1	3									NO	1	1	YES	YES	
1	3	0	0	0									YES	0	1	NO	NO	
1	3	0	0	0									YES	1	1	NO	NO	
1	3	0	0	1									YES	0	1	NO	NO	
1	3	0	0	1									YES	1	1	NO	NO	
1	3	0	0	2									YES	0	1	NO	NO	
1	3	0	0	2									YES	1	1	NO	NO	
1	3	0	0	3									YES	0	1	YES	YES	
1	3	0	0	3									YES	1	1	YES	YES	
1	3	0	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	YES	NO	NO	0	1	NO	NO	
1	3	0	1	0	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	NO	YES	NO	NO	1	1	NO	NO	
1	3	0	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invali- date all process-scoped entries with matching LPID	NO	NO	NO	0	1	NO	NO	
1	3	0	1	1	NO	YES	LPIDR	NO	NO	When SET = 0, invali- date all process-scoped entries with matching LPID	NO	NO	NO	1	1	NO	NO	
1	3	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	YES	When SET = 0, invali- date all process table caching with matching LPID	NO	0	1	NO	NO	
1	3	0	1	2	NO	YES	LPIDR	NO	When SET = 0, invalidate all process-scoped entries with matching PID (for a matching LPID)	When SET = 0, invali- date all process-scoped entries with matching LPID	YES	When SET = 0, invali- date all process table caching with matching LPID	NO	1	1	NO	NO	
1	3	0	1	3									NO	0	1	YES	YES	
1	3	0	1	3									NO	1	1	YES	YES	



Table D-2. **tblel** Encodings (Sheet 11 of 11)

HR (was R in POWER9)	IS	HV	PRS	RIC	VA Match Required?	LPID Match Required?	LPID Taken from RS or LPIDR	PID Match Required?	TLB Entry/Entries Invalidated	PWC Invalidated?	ERAT and L1 Caches Invalidated?	Process/Partition Table Cach- ing Invalidated? (effR is ignored)	Privileged Instruc- tion Interrupt? HV = 0,PRS = 0 (red)	R	effR	POWER9 Invalid Form Machine Check Case?	Power10 Invalid Form Machine Check Case?	Change from POWER9 to Power10?
1	3	1	0	0	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all Partition-Scoped entries (IS = 3, HV = 1, effR is ignored)	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	NO	NO	0	0	NO	NO	
1	3	1	0	0	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	NO	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	NO	NO	1	1	NO	NO	
1	3	1	0	1									NO	0	0	NO	YES	X
1	3	1	0	1	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	NO	NO	NO	NO	1	1	NO	NO	
1	3	1	0	2	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all partition-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	When SET = 0, invalidate all partition table caching regardless of LPID	NO	0	0	NO	NO	
1	3	1	0	2	NO	NO	NA	NO	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all partition-scoped entries regardless of LPID	When SET = 0, invalidate all entries (regardless of LPID and effR for IS = 3 and HV = 1)	When SET = 0, invalidate all partition table caching regardless of LPID	NO	1	1	NO	NO	
1	3	1	0	3									NO	0	0	YES	YES	
1	3	1	0	3									NO	1	1	YES	YES	
1	3	1	1	0									NO	0	0	NO	YES	X
1	3	1	1	0	NO	NO	NA	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	NO	YES	NO	NO	1	1	NO	NO	
1	3	1	1	1							YES		NO	0	0	NO	YES	X
1	3	1	1	1	NO	NO	NA	NO	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	NO	NO	NO	1	1	NO	NO	
1	3	1	1	2	NO	NO	NA	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	When SET = 0, invalidate all process-scoped entries regardless of LPID (IS = 3, HV = 1, effR is ignored)	YES	When SET = 0, invalidate all process table caching regardless of LPID	NO	0	0	NO	NO	
1	3	1	1	2	NO	NO	NA	NO	When SET = 0, invalidate all process-scoped entries regardless of LPID	When SET = 0, invalidate all process-scoped entries regardless of LPID	YES	When SET = 0, invalidate all process table caching regardless of LPID	NO	1	1	NO	NO	
1	3	1	1	3									NO	0	0	YES	YES	
1	3	1	1	3									NO	1	1	YES	YES	



## Appendix E. Power10 Performance Monitor and Instrumentation

### E.1 Core Performance Monitoring Facilities

The Power10 chip has built-in features for monitoring and collecting data for performance analysis. Collectively, the features are referred to as instrumentation. Performance instrumentation is divided into two broad categories: the performance monitor and the trace facilities. This section provides a brief overview of the essential core performance monitor functions and facilities available to each thread.

#### E.1.1 Essential Performance Monitor Functions

The Power10 performance monitor performs the following functions:

- Counts instructions completed and cycles gated by the run latch in individual (dedicated) 32-bit counters. The counting of these events can be enabled by software under several conditions such as problem or supervisor state.
- Counts up to four concurrent software-selected events in individual 32-bit counters. The counting of events can be enabled by software under several conditions such as problem or supervisor state, run or wait state.
- One event per counter can be selected for monitoring at a given time. The event to be monitored is selected by setting the appropriate value into the Monitor Mode Control Register (MMCR) event selection fields for that counter. The event counted can be the number of cycles that the event occurs or the number of occurrences of the event depending on the particular event selected. Performance monitor counting can be enabled or disabled under the machine states mentioned previously, which are selected using control bit fields in the MMCRs and the state bits in other Special Purpose Registers (SPRs).
- Generates performance monitor exceptions, alerts, and interrupts. The performance monitor can generate a maskable interrupt when an event counter overflows or when a software-specified bit of the Time Base Register transitions to '1'. The exception enable bits in the MMCRs allow software to enable and specify the behavior that will trigger the interrupt or exception. An enabled exception causes an indicator bit to be set in the MMCRs. This bit can only be reset by software. When enabled for external interrupts, a performance monitor alert causes a performance monitor interrupt to occur. When running in a partitioned environment, the operating system may be swapped out while a performance monitor exception alert is pending. The hypervisor preserves the value of MMCRs across the partition swap. When the operating system is re-dispatched, the alert is still pending.
- Freezes the contents of the event counters until a selected event or condition occurs and then begins counting (triggering). The event counters can also be incremented until a selected event or condition occurs, and then freeze counting.
- Chooses an instruction for detailed monitoring, which is called sampling or marking an instruction. The Power10 uninstrumentation supports setting mask values for matching particular instructions or types of instructions, which are then eligible to be sampled. The performance monitor includes events for counting sampled instructions at each stage of the pipeline and in certain other situations. Instruction sampling is a useful facility for gathering both detailed and statistical information for particular instructions. Profiling and sampling are common approaches to associate expensive performance events in a processor to instruction and data addresses. Profiling allows the identification of hotspots in code and data, finds performance-sensitive areas, and identifies problem instructions, data areas, or both. Profiling is commonly achieved by identifying a particular instruction and collecting detailed information about that instruction (instruction sampling).



- Performs thresholding. The Power10 processor monitors the pipeline stage progression of sampled instructions and can detect when the stage-to-stage cycle count for a selected start/stop pair of pipeline stages exceeds a specified threshold value. The threshold value can also be used to detect sampled loads whose latency exceeds the threshold value.
- Allows for software-driven marking using Probe NOPs.
- The Power10 processor provides for special no-operation instructions called Probe NOP which are reserved exclusively for performance monitor use. Software can insert Probe NOP instructions at various points in the program and configure the performance monitor sampling facility to count only Probe NOP events, and/or to mark or sample only probe no-ops.

### E.1.2 Definitions and Terminology

**Branch History Rolling Buffer (BHRB)** - A buffer that contains a history of branch addresses that have been taken.

**Continuous Sampling** - Sampling every instruction executed and continuously collecting instruction and data addresses.

**Eligible Instruction** - Instructions that are eligible for random sampling.

**Event-Based Sampling** - Sampling an instruction based on the occurrence of an event, such as a cache miss or branch mis-predict. Randomly mark an instruction after an event has happened (as compared to marking an instruction and hoping that an event happens to that instruction).

**Instruction Completion Table (ICT)** - A table tracking in-flight instructions. Long delays in instruction fetching can cause the ICT to be empty for the currently tracked thread.

**Lightweight Profiling** - User-level interrupts and reduced latency of PMU interrupts.

**Marked Events** - Events attributed to a marked instruction. By convention, marked event names are often annotated with the Mark bit in the VHDL.

**Marked Instruction** - The instruction that has been randomly picked for detailed data collection. This instruction is also referred to as a sampled instruction.

**Next-to-Complete (NTC)** - The program order next-to-complete instruction.

**Performance Monitor Unit (PMU)** - The PMU is a programmable component of each microprocessor core on the chip. It collects and filters information collected from various aspects of the chip and attributes the events to the threads within the core.

**Random Instruction Sampling** - Randomly picking one instruction on which to collect detailed performance data, including instruction and data addresses. This is also referred to as marking.

**Run Latch** - A mechanism used by hypervisors and operating systems to flag cycles when the processor is not idle for a hardware thread. Run cycles are the cycles the thread is not idle. Run instructions are the instructions executed by the thread when it is not idle.

**Sampling** - Collecting performance data from a single instruction.

**Simultaneous Multithreading (SMT)** - SMT enables a number of hardware threads to run concurrently on the core. When multiple threads are running concurrently on the core, resources on the core are shared by the threads on the core.



**Software Driven Marking** - Probe NOP is inserted into an instruction stream to mark the following instruction.

### E.1.3 Essential Performance Monitor Facilities

The Power10 processor instrumentation facilities and the associated Power10 components include several SPRs associated with performance monitoring, instruction matching, instruction sampling, and tracing. Unless otherwise noted, the special purpose registers described can be read in problem and supervisor state and written in supervisor state by using the **mfsp**r and **mts**p instructions, respectively. The Machine State Register (MSR) is read and written by the **mfms**r and **mtm**sr instructions.

### E.1.4 Performance Monitor Special Purpose Registers and Fields

A high-level overview of the performance-monitor-related registers and fields is provided in this section. Additional details can be found in *Section E.2 Performance Monitor Registers* on page 451.

#### Performance Monitor Counter Registers (**PMCx**)

- These registers increment each time (or cycle, depending on the selected event) an event occurs while the counter is enabled. These registers also have the control function for the counter overflow condition. Each thread has six Performance Monitor Counters (PMCs). With four threads per core, each Power10 core has 24 thread-level Performance Monitor Counters (PMCs). Each PMC is 32 bits wide. By connecting adjacent PMCs, PMC1 - 4 can also be used as a  $32 \cdot N$  ( $N = 1 - 4$ ) bit counter.
  - PMC1-4 are programmable.
  - PMC5 is a dedicated counter for Run instructions. Run instructions are completed PowerPC instructions gated by the run latch.
  - PMC6 is a dedicated counter for Run cycles. Run cycles are processor cycles gated by the run latch.

#### Performance Monitor Mode Control Registers (**MMCRx**)

- The Performance Monitor is configured and controlled through the Monitor Mode Control Registers (MMCRs). These registers include both counting control and event select bit fields.
  - MMCR0 - Partition resource controls basic operation (start/stop/freeze) of the performance monitor.
  - MMCR1 - Partition resource controls what to count
  - MMCR2 - Per PMC controls for basic operation
  - MMCR3 - Per PMC reload\_src controls and per thread L2/L3 event select
  - MMCR4 - Partition resource includes indicator bits for feedback between the hardware and software and configuration fields for special features of the performance monitor.



### Sample Address/Event Registers (**SxxR**)

- These registers can only be updated when performance monitor exceptions are enabled. This protects the contents from change until software can read them. The values written to these registers by the hardware depend on the processing state and on the type of instruction that is being sampled.
  - Sampled Instruction Address Register (SIAR) - a 64-bit register containing the instruction address relating to a sampled instruction.
  - Sampled Data Address Register (SDAR) - a 64-bit register containing the data address relating to a sampled or marked instruction.
  - Sampled Instruction Event Register (SIER) - a 64-bit register storing information relating to a sampled instruction.
  - Sampled Instruction Event Register 2 (SIER2) - a 64-bit register storing information relating to a sampled instruction.
  - Sampled Instruction Event Register 3 (SIER3) - a 64-bit register storing information relating to a sampled instruction.

Machine State Register fields related to performance monitoring are as follows:

- Machine State Register [EE] (**MSR[EE]**)
  - This register bit is used to enable/disable the external interrupt. The performance monitor interrupt is considered an external interrupt.
- Machine State Register [PMM] (**MSR[PMM]**)
  - This register bit is used to enable/disable performance monitor activity controlled by the process mark bit.
- Machine State Register [PR] (**MSR[PR]**)
  - This register bit is used to establish problem/supervisor mode and the performance monitor counting activity controlled by this bit.
- Machine State Register [SE] (**MSR[SE]**)
  - This register bit is used to enable/disable a trace interrupt after each instruction is completed.
- Machine State Register [BE] (**MSR[BE]**)
  - This register bit is used to enable/disable a trace interrupt after a branch instruction is completed.
- Control Register (**CNTL[31]**)
  - This register bit is used by operating systems to indicate an idle or run state. The performance monitor can use this bit to avoid counting events during idle periods. This bit is commonly called the *Run Latch*.
- Timebase [47,51,55,63] (**Timebase[47, 51, 55, 63]**)
  - These register bits are used for time-based events. Most performance monitor events are cycle-based; they count based on processor cycles. The Timebase register is used to maintain time-of-day and can be used by the performance monitor to count time intervals.
- Machine Status Save/Restore Register (**SRRO, SRR1**)
  - These registers are used to save the machine status during interrupts.
- Core Monitor Mode Control Register (**MMCRC**)
  - A core-level SPR with per-thread freeze controls.



Table E-1 describes the SPR address bits and the widths for these registers.

Table E-1. Performance Monitor Related Special Purpose Registers

Register Name	SPR Address Bits			
	Decimal (U,P)	5 - 9 0 <sup>1</sup> - 4	Width (bits)	Function
MMCR0	779,795	b'11000' b'n1011'	32	Performance Monitor Mode Control Register 0
MMCR1	782,798	b'11000' b'n1110'	64	Performance Monitor Mode Control Register 1
MMCR2	769,785	b'11000 b'n0001'	64	Performance Monitor Mode Control Register 2
MMCR3	738,754	b'10111 b'n0010'	64	Performance Monitor Mode Control Register 3
MMCRA	770,786	b'11000' b'n0010'	64	Performance Monitor Mode Control Register A
MMCRC	851	b'11010' b'10011'	64	Core Mode Monitor Control Register
PMC1	771,787	b'11000' b'n0011'	32	Performance Monitor Counter Register 1
PMC2	772,788	b'11000' b'n0100'	32	Performance Monitor Counter Register 2
PMC3	773,789	b'11000' b'n0101'	32	Performance Monitor Counter Register 3
PMC4	774,790	b'11000' b'n0110'	32	Performance Monitor Counter Register 4
PMC5	775,791	b'11000' b'n0111'	32	Performance Monitor Counter Register 5
PMC6	776,792	b'11000' b'n1000'	32	Performance Monitor Counter Register 6
SIER	768,784	b'11000' b'n0000'	64	Sampled Instruction Event Register
SIER2	736,752	b'10111' b'n0000'	64	Sampled Instruction Event Register 2
SIER3	737,753	b'10111' b'n0001'	64	Sampled Instruction Event Register 3
SIAR	780,796	b'11000' b'n01100'	64	Sampled Instruction Address Register
SDAR	781,797	b'11000' b'n1101'	64	Sampled Data Address Register
MSR[61]	Use <b>mtmsr</b> , <b>mfmsr</b> instructions (privileged mode only)		64	Machine State Register [Performance Monitor Mark]
MSR[48]			64	Machine State Register [External Interrupt]
MSR[49]			64	Machine State Register [problem/supervisor state]
MSR[1]			64	Machine State Register [Reserved]
MSR[53]			64	Machine State Register [Single Step Trace Enable]
MSR[54]			64	Machine State Register [Branch Trace Enable]
CTRL[63]	136,152	b'00100 b'n1000	64	Control Register [Run latch, thread control]
TBL	284	b'01000 b'n1100	64	Timebase bits used for performance monitor timebase events

- When n = 1, use privileged mode **mtspr** instruction SPR address bits.  
When n = 0, use user mode **mfsp** instruction SPR address bits.

For **mfsp**, the instruction is privileged mode if and only if SPR[0] = 1.



Table E-2. Performance Monitor Counter (PMC) Properties

PMC	Programmable by Partition	Programmable by Hypervisor	Partition Access	Hypervisor Access	Exception on Overflow	Interrupt Destination	Width (bits)
PMC1	Yes	Yes	Read/Write	Read/Write	Yes	Partition	32
PMC2	Yes	Yes	Read/Write	Read/Write	Yes	Partition	32
PMC3	Yes	Yes	Read/Write	Read/Write	Yes	Partition	32
PMC4	Yes	Yes	Read/Write	Read/Write	Yes	Partition	32
PMC5	No (completed instruction count gated by run latch only)		Read/Write	Read/Write	Yes	Partition	32
PMC6	No (cycle count gated by run latch only)		Read/Write	Read/Write	Yes	Partition	32

## E.2 Performance Monitor Registers

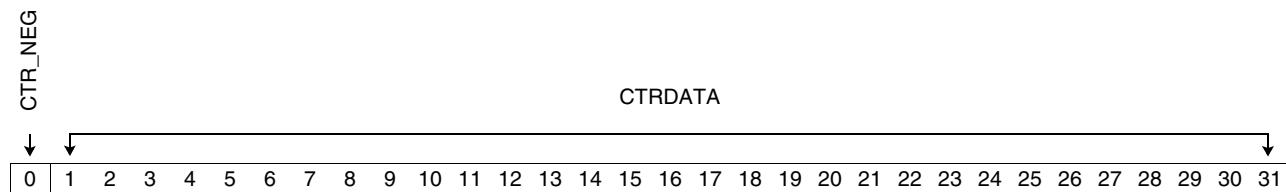
This section describes the following performance monitor related registers:

- Performance Monitor Counters (PMC1 - 6)
- Core Monitor Mode Control Register (MMCRC)
- Performance Monitor Control Register 0 (MMCR0)
- Performance Monitor Mode Control Register 1 (MMCR1)
- Performance Monitor Mode Control Register 2 (MMCR2)
- Performance Monitor Mode Control Register 3 (MMCR3)
- Monitor Mode Control Register A (MMCRA)
- Sampled Instruction Event Register (SIER)
- Sampled Instruction Event Register 2 (SIER2)
- Sampled Instruction Event Register 3 (SIER3)
- Sampled Instruction Address Register (SIAR)
- Sampled Data Address Register (SDAR)

### E.2.1 Performance Monitor Counters (PMC1 - 6)

The six performance monitor counters, PMC1 through PMC6, are 32-bit registers that count events. PMC1 - PMC4 are referred to as “programmable” counters because the events that can be counted can be specified by software. The codes that identify the events that are counted are selected by specifying the appropriate code in the PMCxSEL event select fields in MMCR1[32:63].

Chaining of two or more counters is accomplished by setting a PMCxSEL to select an event for counting defined as the overflow (most-significant-bit going to ‘1’) of a chained counter. The counted events are described in *Appendix E.5 Core PMU Events* on page 493. PMC5 and PMC6 are not programmable. PMC5 counts instructions completed, and PMC6 counts cycles.

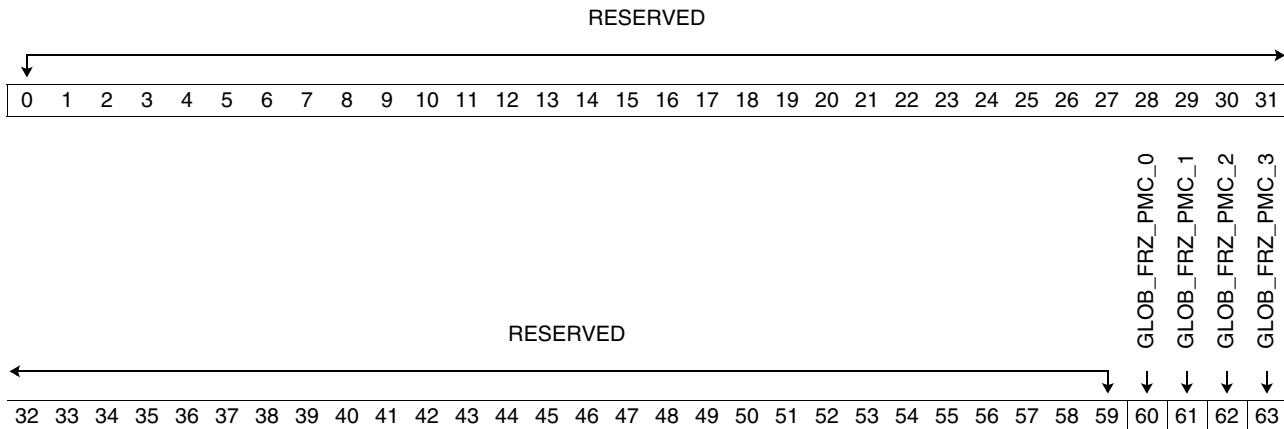


Bits	Field Name	Description
0	CTR_NEG	Counter negative bit. When an adjacent PMC uses overflow counting, this becomes counter data.
1:31	CTRDATA	Counter data.



### E.2.2 Core Monitor Mode Control Register (MMCRC)

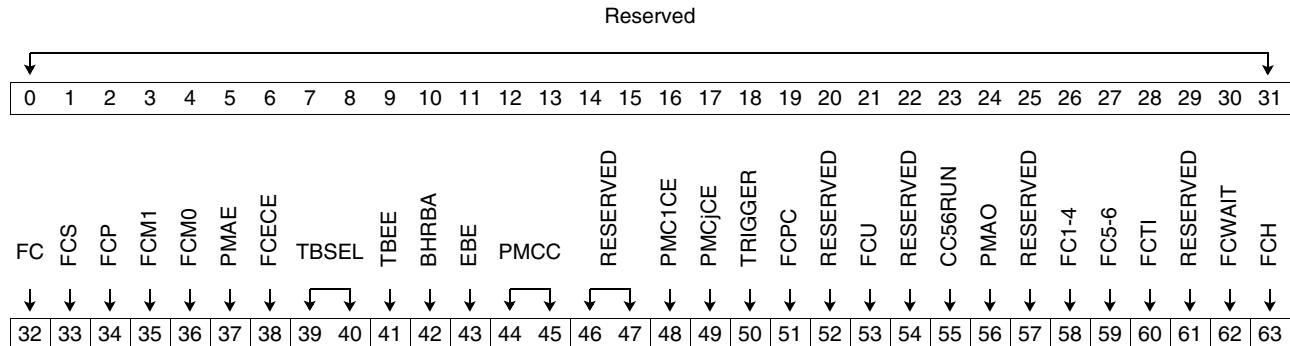
This SPR is not replicated on a per-split core basis. This is a core-level SPR (hypervisor access only) that is used for a variety of purposes.



Bits	Field Name	Description
0:59	RESERVED	Reserved.
60	GLOB_FRZ_PMC_0	Global freeze for thread-level PMU T0. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6. In 4LPAR mode, this maps to LPAR0.
61	GLOB_FRZ_PMC_1	Global freeze for thread-level PMU T1. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6. In 4LPAR mode, this maps to LPAR1.
62	GLOB_FRZ_PMC_2	Global freeze for thread level PMU T2. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6. In 4LPAR mode, this maps to LPAR2.
63	GLOB_FRZ_PMC_3	Global freeze for thread level PMU T3. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4 otherwise, it controls PMCs 1 - 6. In 4LPAR mode, this maps to LPAR3.

### E.2.3 Performance Monitor Control Register 0 (MMCR0)

The 64-bit Performance Monitor Mode Control Register 0 (MMCR0) controls the basic operation (start/stop/freeze) of the performance monitor.



Bits	Field Name	Description
0:31	Reserved	Reserved.
32	FC	<p>Freeze counters.</p> <p>0 The PMCs are incremented (if permitted by other MMCR bits).</p> <p>1 The PMCs are not incremented.</p> <p>The processor sets this bit to '1' when an enabled condition or event occurs and the "freeze counter on an enabled condition or an event bit" is '1' (MMCR0[FCECE] = '1').</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
33	FCS	<p>Freeze counters and the <u>BHRB</u> are in privileged state.</p> <p>0 The PMCs are incremented, and entries are written into the BHRB (if permitted by other MMCR bits).</p> <p>1 The PMCs are not incremented, and entries are not written into the BHRB if MSR[HV, PR] = '00'.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
34	FCP	<p>Freeze counters and the BHRB are in problem state.</p> <p>If value of MMCR0[FCPC] = '0', and FCP is equal to:</p> <p>0 The PMCs are incremented, and entries are written into the BHRB if permitted by other MMCR bits.</p> <p>1 The PMCs are not incremented, and entries are not written into the BHRB</p> <p>If MSR[S, HV, PR] = 'XX1'.</p> <p>If value of MMCR0[FCPC] = '1'</p> <p>0 The PMCs are not incremented and entries are not written into BHRB if MSR[S, HV, PR] = 'X01'</p> <p>1 The PMCs are not incremented and BHRB entries are not written if MSR[S, HV, PR] = 'X11'</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>1. To freeze the counters in problem state regardless of MSR[HV], MMCR0[FCPC] must be set to '0' and MMCR0[FCP] must be set to '1'.</li> <li>2. When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</li> </ul>
35	FCM1	<p>Freeze counters when the performance monitor Mark bit (MSR[PMM]) is set to '1'.</p> <p>0 The PMCs are incremented.</p> <p>1 The PMCs are not incremented when MSR[PMM] = '1'.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
36	FCM0	<p>Freeze counters when the performance monitor Mark bit (MSR[PMM]) is set to '0'.</p> <p>0 The PMCs are incremented.</p> <p>1 The PMCs are not incremented when MSR[PMM] = '0'.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>



Bits	Field Name	Description
37	PMAE	<p>Performance monitor alert enable.</p> <p>0 Performance monitor alerts are disabled.</p> <p>1 Performance monitor alerts are enabled until a performance monitor alert occurs, at which time MMCR0[PMAE] is set to '0' and MMCR0[PMA0] is set to '1'. When MMCR0[EBE] = '1', setting <u>BESCR[PME]</u> to '0' or '1' sets PMAE to '0' or '1' respectively.</p> <p>For implementations that do not provide a performance monitor interrupt, software can set PMAE to '1' and then poll the bit to determine whether an enabled condition or event has occurred.</p> <p><b>Note:</b> Software can set this bit and MMCR0[PMAO] to '0' to prevent performance monitor exceptions. Software can set this bit to '1' and then poll the bit to determine whether an enabled condition or event has occurred. This is especially useful for software that runs with MSR[EE] = '0'. In earlier versions of the architecture that lacked the concept of performance monitor alerts, this bit was called the Performance Monitor Exception Enable (PMXE).</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
38	FCECE	<p>Freeze counters on enabled condition or event.</p> <p>0 The PMCs are incremented (if permitted by other MMCR bits).</p> <p>1 The PMCs are incremented (if permitted by other MMCR bits) until an enabled condition or event occurs when MMCR0[TRIGGER] = '0', at which time MMCR0[FC] is set to '1'. If the enabled condition or event occurs when MMCR0[TRIGGER] = '1', the FCECE bit is treated as if it were '0'.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
39: 40	TBSEL	<p>Timebase selector.</p> <p>00 Timebase bit 47 is selected.</p> <p>01 Timebase bit 51 is selected.</p> <p>10 Timebase bit 55 is selected.</p> <p>11 Timebase bit 63 is selected.</p> <p>When the selected timebase transitions from '0' to '1', the timebase event is enabled (MMCR0[TBEE] = '1'), and the performance monitor interrupt is enabled: a performance monitor interrupt occurs and the performance monitor interrupt is disabled (MMRC0[PMAE] is negative).</p> <p>In multiprocessor systems with the timebase registers synchronized among the processors, timebase transition events can be used to correlate the performance monitor data obtained by the several processors provided that software has specified the same TBSEL value for all of the processors in the system. Even with multi-LPAR support, LPAR0 timebase bits are used so that all partitions are counting events over the same periods and throwing alerts at the same time.</p> <p><b>Note:</b> This bit is the default for timebase selection. However, other bits can be selected in the timebase unit.</p>
41	TBEE	<p>Timebase exception enable.</p> <p>0 Disable timebase transition events.</p> <p>1 Enable timebase transition events.</p>
42	BHRBA	<p>BHRB access.</p> <p>This field controls whether the BHRB instructions are available in problem state. If an attempt is made to execute a BHRB instruction in problem state when the BHRB instructions are not available, a Facility Unavailable interrupt occurs.</p> <p>0 <b>mfbhrb</b> and <b>clrbhrb</b> are not available in problem state.</p> <p>1 <b>mfbhrb</b> and <b>clrbhrb</b> are available in problem state.</p>

Bits	Field Name	Description								
43	EBE	<p>Performance monitor event-based branch enable.</p> <p>This field controls whether performance monitor event-based branches are enabled.</p> <p>0 Performance monitor event-based branches are disabled. 1 Performance monitor event-based branches are enabled.</p> <p><b>Note:</b> Enabling event-based branches gives problem state programs visibility to and control of MMCR0[PMAE] and MMCR0[PMAO]. This enables problem state programs to recognize when performance monitor event-based exceptions have occurred and to re-enable performance monitor event-based exceptions. To enable a problem state program to use the event-based branch facility for performance monitor events, software first initializes the performance monitor registers to values appropriate to the program, sets MMCR0[PMAE] and MMCR0[PMAO] to '0', sets BESCR[PMEO] to '0', and sets MMCR0[EBE] to '1'. MMCR0[PMCC] should also be set to '10' or '11' to give problem state programs write access to the PMCs. If the event-based branch facility has not been enabled in the Facility Status and Control Register (FSCR) and Hypervisor Facility Status and Control Register (HFSCR), it must be enabled in these registers as well.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>								
44:45	PMCC	<p>PMC control.</p> <p>This field controls the number of counters that are included in the performance monitor and the set of registers that are available to be read and written in problem state. If an attempt is made to read a register that is unavailable to be read, or an attempt is made to write a register that is unavailable to be written to in problem state, a facility unavailable interrupt occurs. The values and their meanings are described as follows:</p> <table> <tr> <td>00</td><td>PMCs 1 - 6 are included in the performance monitor, and SPRs 768 - 782 are available to be read in problem state but are not available to be written in problem state.</td></tr> <tr> <td>01</td><td>PMCs 1 - 6 are included in the performance monitor, but SPRs 768 - 782 are not available to be read or written in problem state.</td></tr> <tr> <td>10</td><td>PMCs 1 - 6 are included in the performance monitor. Selected fields of MMCR0, MMCR2, MMCRA, and all bits of PMCs 1 - 6 are available to be read and written in problem state. SIER, SDAR, and SIAR are read-only in problem state. All other SPRs in the range of 768 - 782 are not available to be read or written in problem state.</td></tr> <tr> <td>11</td><td>PMCs 5 - 6 are not included in the performance monitor. Group A SPRs (MMCR2, MMCRA, PMC 1 - 6, MMCR0) except for PMCs 5 - 6 can be read and written in problem state. Group B (SIER, SIAR, SDAR, MMCR1), except for MMCR1, can be read in problem state.</td></tr> </table> <p>If an attempt is made when in problem state, to read or write to PMCs 5 - 6 or to read from MMCR1, a Facility Unavailable interrupt occurs. When an SPR is made available by the PMCC field, it is available only if it has not been made unavailable by the HFSCR.</p> <p>When PMCs 5 and 6 are not part of the performance monitor (for example, when PMCC = '11'), counter negative conditions in PMCs 5 and 6 do not result in performance monitor alerts or exceptions and do not result in performance monitor interrupts</p>	00	PMCs 1 - 6 are included in the performance monitor, and SPRs 768 - 782 are available to be read in problem state but are not available to be written in problem state.	01	PMCs 1 - 6 are included in the performance monitor, but SPRs 768 - 782 are not available to be read or written in problem state.	10	PMCs 1 - 6 are included in the performance monitor. Selected fields of MMCR0, MMCR2, MMCRA, and all bits of PMCs 1 - 6 are available to be read and written in problem state. SIER, SDAR, and SIAR are read-only in problem state. All other SPRs in the range of 768 - 782 are not available to be read or written in problem state.	11	PMCs 5 - 6 are not included in the performance monitor. Group A SPRs (MMCR2, MMCRA, PMC 1 - 6, MMCR0) except for PMCs 5 - 6 can be read and written in problem state. Group B (SIER, SIAR, SDAR, MMCR1), except for MMCR1, can be read in problem state.
00	PMCs 1 - 6 are included in the performance monitor, and SPRs 768 - 782 are available to be read in problem state but are not available to be written in problem state.									
01	PMCs 1 - 6 are included in the performance monitor, but SPRs 768 - 782 are not available to be read or written in problem state.									
10	PMCs 1 - 6 are included in the performance monitor. Selected fields of MMCR0, MMCR2, MMCRA, and all bits of PMCs 1 - 6 are available to be read and written in problem state. SIER, SDAR, and SIAR are read-only in problem state. All other SPRs in the range of 768 - 782 are not available to be read or written in problem state.									
11	PMCs 5 - 6 are not included in the performance monitor. Group A SPRs (MMCR2, MMCRA, PMC 1 - 6, MMCR0) except for PMCs 5 - 6 can be read and written in problem state. Group B (SIER, SIAR, SDAR, MMCR1), except for MMCR1, can be read in problem state.									
46:47	RESERVED	Reserved.								
48	PMC1CE	<p>PMC1 condition enable.</p> <p>This bit determines whether the counter negative condition due to a negative value in PMC1 is enabled.</p> <p>0 Disable PMC1 counter negative condition. 1 Enable PMC1 counter negative condition.</p>								
49	PMCjCE	<p>PMCj condition enable.</p> <p>This bit determines whether the counter negative condition, due to a negative value in PMCj (<math>2 \leq j</math>), is enabled.</p> <p>0 Disable PMCj (<math>2 \leq j</math>) counter negative condition. 1 Enable PMCj (<math>2 \leq j</math>) counter negative condition.</p> <p><b>Note:</b> The following notation is used in the subsequent definitions: When MMCR0[PMCC] = '11', "PMCs" refers to PMCs 1 - 4, and "PMCj" or PMCjCE refers to "PMCj" or PMCjCE, respectively, where <math>j = 2 - 4</math>; otherwise, "PMCs" refers to PMCs 1 - 6 and "PMCj" or PMCjCE refers to "PMCj" or PMCjCE, respectively, where <math>j = 2 - 6</math>.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 2 - 4. Otherwise, it controls PMCs 2 - 6.</p>								



Bits	Field Name	Description
50	TRIGGER	<p>TRIGGER enable.</p> <p>0 The PMCs are incremented (if permitted by other MMCR bits).</p> <p>1 PMC1 is incremented (if permitted by other MMCR bits). The PMCjs are not incremented until PMC1 is negative or an enabled condition or an event occurs, at which time the PMCjs resume incrementing (if permitted by other MMCR bits). MMCR0[TRIGGER] is set to '0'.</p> <p><b>Note:</b> See the description of the FCECE bit regarding the interaction between TRIGGER and FCECE.</p> <p><b>Note:</b> Resume counting in the PMCjs when PMC1 becomes negative, without causing a performance monitor interrupt. Then freeze all PMCs (and optionally cause a performance monitor interrupt) when a PMCj becomes negative. The PMCjs then reflect the events that occurred between the time PMC1 became negative and the time a PMCj becomes negative. This use requires the following MMCR0 bit settings:</p> <ul style="list-style-type: none"> <li>• TRIGGER = '1'</li> <li>• PMC1CE = '0'</li> <li>• PMCjCE = '1'</li> <li>• TBEE = '0'</li> <li>• FCECE = '1'</li> <li>• PMAE = '1' (if a performance monitor interrupt is required)</li> </ul> <p>Resume counting in the PMCjs when PMC1 becomes negative and cause a performance monitor interrupt without freezing any PMCs. The PMCjs then reflect the events that occurred between the time PMC1 became negative and the time the interrupt handler reads them. This use requires the following MMCR0 bit settings:</p> <ul style="list-style-type: none"> <li>• TRIGGER = '1'</li> <li>• PMC1CE = '1'</li> <li>• TBEE = '0'</li> <li>• FCECE = '0'</li> <li>• PMAE = '1'</li> </ul> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
51	FCPC	<p>Freeze counters and the BHRB in the problem state condition.</p> <p>This bit controls the operation of bit 34 (FCP). See the definition of bit 34 for details.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
52	RESERVED	Reserved.
53	FCU	<p>Freeze Counters and BHRB in ultrvisor state.</p> <p>0 The PMCs are incremented and entries are written into the BHRB if permitted by other MMCR bits).</p> <p>1 The PMCs are not incremented and entries are not written into the BHRB if MSR[S, HV, PR] = '110'.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>
54	RESERVED	Reserved.
55	CC56RUN	<p>Freeze counters 5 and 6 in the wait state.</p> <p>When MMCR0[PMCC] = '11', the setting of this bit has no effect; otherwise, it is defined as follows:</p> <p>0 PMCs 5 and 6 are incremented only when CTRL[RUN] = '1' (if permitted by other MMCR bits). Software is expected to set CTRL[RUN] = '0' when it is in a wait state; for example, when there is no process ready to run.</p> <p>1 PMCs 5 and 6 are incremented regardless of the state of CTRL[RUN].</p>
56	PMAO	<p>Performance monitor alert has occurred.</p> <p>0 A performance monitor event has not occurred since the last time software set this bit to '0'.</p> <p>1 A performance monitor event has occurred since the last time software set this bit to '0'.</p> <p>This bit is set to '1' by the processor when a performance monitor event occurs. This bit can be set to '0' only by the <b>mtspr</b> instruction.</p> <p>When MMCR0[EBE] = '1', setting BESCR[PME0] to '0' or '1' sets PMAO to '0' or '1', respectively.</p> <p>Software can set this bit to '1' to simulate the occurrence of a performance monitor event.</p> <p>Software should set this bit to '0' after handling the performance monitor event.</p> <p><b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</p>



Bits	Field Name	Description
57	RESERVED	Reserved.
58	FC1-4	Freeze counters 1 - 4. 0 PMC 1 - 4 are incremented (if permitted by other MMCR bits). 1 PMC 1 - 4 are not incremented.
59	FC5-6	Freeze counters 5 - 6. 0 PMC5 - 6 are incremented (if permitted by other MMCR bits). 1 PMC5 - 6 are not incremented. <b>Note:</b> When MMCR0[PMCC] = '11', this bit has no effect.
60	FCTI	Freeze counters. 0 The PMCs are incremented. 1 The PMCs are not incremented. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.
61	RESERVED	Reserved.
62	FCWAIT	Freeze counters in the wait state. 0 The PMCs are incremented (if permitted by other MMCR bits). 1 The PMCs are not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a wait state; for example, when there is no process ready to run. <b>Notes:</b> <ol style="list-style-type: none"><li>When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise, it controls PMCs 1 - 6.</li><li>PM_CYC counts cycles regardless of the run latch.</li></ol>
63	FCH	Freeze counters and BHRB in a hypervisor state. 0 The PMCs are incremented (if permitted by other MMCR bits) and BHRB entries are written (if permitted by MMCRA bits). 1 The PMCs are not incremented and the BHRB entries are not written if MSR[S, HV, PR] = '010'. <b>Note:</b> When MMCR0[PMCC] = '11', this bit affects PMCs 1 - 4; otherwise it controls PMCs 1 - 6.



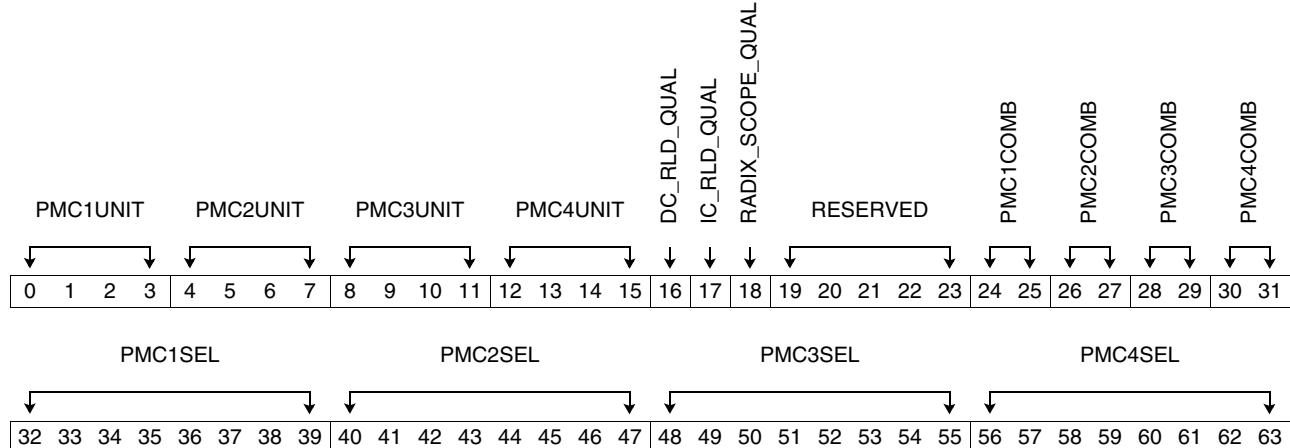
Table E-3 shows the MMCR0 freeze logic based on the MSR state.

Table E-3. MMCR0 Freeze Logic per MSR State, FCU, FCH, FCS, FCPC, and FCP

FCU	FCH	FCS	FCPC	FCP	MSR[S]	MSR[HV]	MSR[PR]	Freeze Condition Description
x	x	x	0	1	x	x	1	PMCs freeze when in problem state (irrespective of HV) MSR[S, HV, PR] = XX1
x	x	x	1	0	x	0	1	PMCs freeze when in true problem state MSR[S, HV, PR] = X01
x	x	x	1	1	x	1	1	PMCs freeze in adjunct state MSR[S, HV, PR] = X11
x	x	1	x	x	x	0	0	PMCs freeze in privileged state MSR[S, HV, PR] = X00
x	1	x	x	x	0	1	0	PMCs freeze in hypervisor state MSR[S, HV, PR] = X10
1	x	x	x	x	1	1	0	PMCs freeze in ultravisor state MSR[S, HV, PR] = 110

### E.2.4 Performance Monitor Mode Control Register 1 (MMCR1)

The 64-bit Performance Monitor Mode Control Register 1 (MMCR1) enables software to specify the events that are counted by the PMCs.



Bits	Field Name	Description
0:3	PMC1UNIT	PMC1 events unit selector. 0000 MMU4 0001 MMU5 0010 <u>ISU0</u> 0011 ISU1 0100 <u>IFU0</u> 0101 IFU1 0110 <u>L2/L3 Bus0</u> 0111 Reserved 1000 MMU0 1001 MMU1 1010 MMU2 1011 MMU3 1100 LSU0 1101 LSU1 1110 IFU2 1111 LSU2 Refer to <i>Table E-5</i> on page 461.
4:7	PMC2UNIT	PMC2 events unit selector (same encoding as PMC1UNIT). Refer to <i>Table E-5</i> on page 461.
8:11	PMC3UNIT	PMC3 events unit selector (same encoding as PMC1UNIT). Refer to <i>Table E-5</i> on page 461.
12:15	PMC4UNIT	PMC4 events unit selector (same encoding as PMC1UNIT). Refer to <i>Table E-5</i> on page 461.
16	DC_RLD_QUAL	This bit defines the data-cache reload qualifier. 0 Counts only demand reloads to the L1 data cache. 1 Counts all reloads to the L1 data cache (demand and prefetch).
17	IC_RLD_QUAL	This bit defines the instruction-cache reload qualifier. 0 Counts only demand reloads to the L1 instruction cache. 1 Counts all reloads to the L1 instruction cache (demand and prefetch).
18	RADIX_SCOPE_QUAL	0 Parent scope or HPT. 1 Process scope (always radix).



Bits	Field Name	Description
19:23	RESERVED	Reserved.
24:25	PMC1COMB	This bit controls the event bus combinatorial function for PMC1: 00 Selects event bus bit 0. 10 Selects event bus bit 1. 01 AND the event bus bits 0 and 1. 11 ADD the event bus bits 0 and 1.
26:27	PMC2COMB	This bit controls the event bus combinatorial function for PMC2 (see PMC1COMB bit description).
28:29	PMC3COMB	This bit controls the event bus combinatorial function for PMC3 (see PMC1COMB).
30:31	PMC4COMB	This bit controls the event bus combinatorial function for PMC4 (see PMC1COMB).
32:39	PMC1SEL	PMC1 event selector bits 0 - 7. The value in this bit field combined with MMCR1[0:31] determines which event will be counted. x'10' PMC1 counts PMC4 overflows. x'24' PMC1 counts PMC5 overflows. When counting overflows, freeze conditions for both counters must be identical. See <i>Table E-4</i> on page 461 for selection of event types. PMCxSEL(0:1) = '10' selects event bus events. PMCxSEL(0:2) = '111' for compatibility direct events. PMCxSEL(0) = '0' for other direct events. When PMCxSEL(7) is high, the falling edges of an event are counted rather than the event itself. Changes from nonzero to '0' are the "edge" for event bus events.
40:47	PMC2SEL	PMC2 event selector bits 0 - 7. The value in this bit field combined with MMCR1[0:31] determines which event will be counted. x'10' PMC2 counts PMC1 overflows. When counting overflows, freeze conditions for both counters must be identical. See <i>Table E-4</i> on page 461 for selection of event types. PMCxSEL(0:1) = '10' selects event bus events. PMCxSEL(0:2) = '111' for compatibility direct events. PMCxSEL(0) = '0' for other direct events. When PMCxSEL(7) is high, the falling edges of an event are counted rather than the event itself. Changes from nonzero to '0' are the "edge" for event bus events.
48:55	PMC3SEL	PMC3 event selector bits 0 - 7. The value in this bit field combined with MMCR1[0:31] determines which event will be counted. x'10' PMC3 counts PMC2 overflows. x'24' PMC3 counts PMC6 overflows. When counting overflows, freeze conditions for both counters must be identical. See <i>Table E-4</i> on page 461 for selection of event types. PMCxSEL(0:1) = '10' selects event bus events. PMCxSEL(0:2) = '111' for compatibility direct events. PMCxSEL(0) = '0' for other direct events. When PMCxSEL(7) is high, falling edges of an event are counted rather than the event itself. Changes from non-zero to 0 are the "edge" for event bus events.
56:63	PMC4SEL	PMC4 event selector bit 0 - 7. The value in this bit field combined with MMCR1[0:31] determines which event will be counted. x'10' PMC4 counts PMC3 overflows. When counting overflows, freeze conditions for both counters must be identical. See <i>Table E-4</i> on page 461 for selection of event types. PMCxSEL(0:1) = '10' selects event bus events. PMCxSEL(0:2) = '111' for compatibility direct events. PMCxSEL(0) = '0' for other direct events. When PMCxSEL(7) is high, the falling edges of an event are counted rather than the event itself. Changes from nonzero to '0' are the "edge" for event bus events.

**Note:** Event counting on PMC1 - 4 suspends counting for one cycle during SPR writes to MMCR1 to avoid a spurious or phantom count during the transition.

*Table E-4* lists the PMC event selector bit values versus the event type.

*Table E-4. MMCR1[PMCxSEL] Selection of Direct Events versus Event Bus Events*

PMCxSEL(0:3)	Event Type
0000	
0001	
0010	
0011	
0100	Direct Events
0101	
0110	
0111	
1000	
1001	
1010	Event Bus Events
1011	
1100	
1101	Reserved
1110	
1111	Compatibility Direct Events

*Table E-5* lists the PMC events unit selector values versus the physical event bus selection.

*Table E-5. MMCR1[PMCxUNIT] Selection of Physical Event Buses (Sheet 1 of 2)*

PMCxUNIT(0:3)	Physical Event Bus Selected
0000	
0001	mu_pc_event_bus
0010	
0011	sd_pc_pmu_event_bus
0100	
0101	if_pc_event_bus
0110	i2_pc_pmon_i23event_bus i2_pc_pmon_i23event_bus_tid
0111	reserved
1000	
1001	
1010	mu_pc_event_bus
1011	

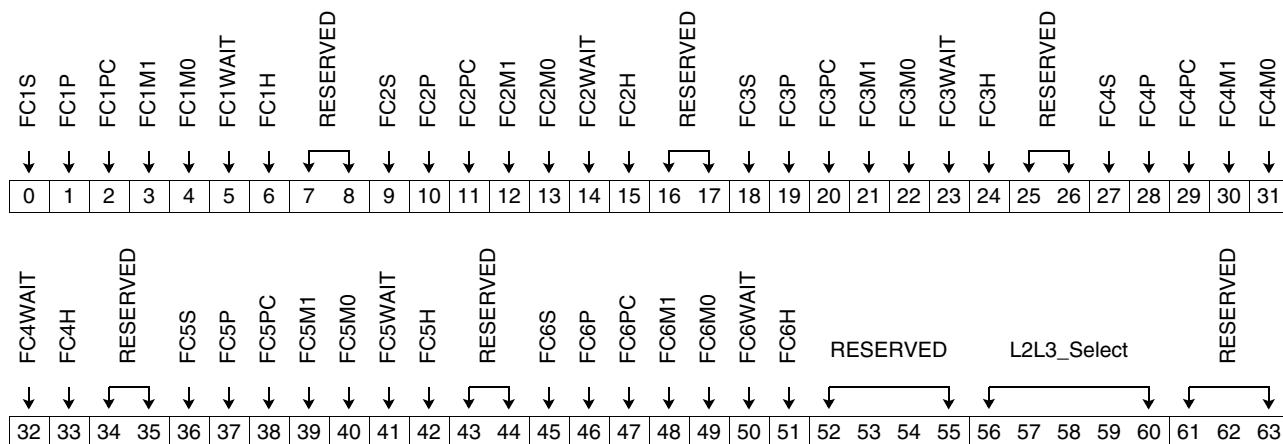
Table E-5. MMCR1[PMCxUNIT] Selection of Physical Event Buses (Sheet 2 of 2)

PMCxUNIT(0:3)	Physical Event Bus Selected
1100	
1101	ls_pc_pmu_event_bus
1110	if_pc_event_bus
1111	ls_pc_pmu_event_bus

### E.2.5 Performance Monitor Mode Control Register 2 (MMCR2)

The 64-bit Performance Monitor Mode Control Register 2 (MMCR2) contains 9-bit fields that control the operation of PMCs 1 - 6.

A single bit in each Cn field of MMCR2 is described in the following bit description table. When MMCR0[PMCC] = '11', fields FC1 - FC4 control the operation of PMC1 - PMC4, respectively and fields FC5 and FC6 are meaningless; otherwise, fields FC1 - FC6 control the operation of PMC1 - PMC6, respectively. The bit definitions of each FCn field are as follows, where n = 1,...6.



Bits	Name	Description
0	FC1S	Freeze PMC1 in privileged state. 0 PMC1 is incremented if permitted by other MMCR bits. 1 PMC1 is not incremented if MSR[HV, PR] = '00'.
1	FC1P	Freeze PMC1 in problem state if MSR[HV] = '0'. 0 PMC1 is incremented if permitted by other MMCR bits. 1 If FC1PC = '0', then PMC1 is not incremented if MSR[PR] = '1'.
2	FC1PC	Freeze PMC1 in problem state if MSR[HV] = '1'. 0 PMC1 is incremented if permitted by other MMCR bits. 1 If FC1P = '0', PMC1 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC1P = '1', PMC1 is not incremented if MSR[HV, PR] = '11' (adjunct state).
3	FC1M1	Freeze PMC1 while Mark = '1'. 0 PMC1 is incremented if permitted by other MMCR bits. 1 PMC1 is not incremented if MSR[PMM] = '1'.



Bits	Name	Description
4	FC1M0	Freeze PMC1 while Mark = '0'. 0 PMC1 is incremented if permitted by other MMCR bits. 1 PMC1 is not incremented if MSR[PMM] = '0'.
5	FC1WAIT	Freeze PMC1 in wait state. 0 PMC1 is incremented if permitted by other MMCR bits. 1 PMC1 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a "wait state"; that is, when there is no process ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
6	FC1H	Freeze PMC1 in hypervisor state. 0 PMC1 is incremented if permitted by other MMCR bits. 1 PMC1 is not incremented if MSR[HV, PR] = '10'.
7:8	RESERVED	Reserved.
9	FC2S	Freeze PMC2 in privileged state. 0 PMC2 is incremented if permitted by other MMCR bits. 1 PMC2 is not incremented if MSR[HV, PR] = '00'.
10	FC2P	Freeze PMC2 in problem state if MSR[PR] = '1'. 0 PMC2 is incremented if permitted by other MMCR bits. 1 If FC2PC = '0', then PMC2 is not incremented if MSR[PR] = '1'.
11	FC2PC	Freeze PMC2 in problem state if MSR[HV] = '1'. 0 PMC2 is incremented if permitted by other MMCR bits. 1 If FC2P = '0', PMC2 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC2P = '1', PMC2 is not incremented if MSR[HV, PR] = '11' (adjunct state).
12	FC2M1	Freeze PMC2 while Mark = '1'. 0 PMC2 is incremented if permitted by other MMCR bits. 1 PMC2 is not incremented if MSR[PMM] = '1'.
13	FC2M0	Freeze PMC2 while Mark = '0'. 0 PMC2 is incremented if permitted by other MMCR bits. 1 PMC2 is not incremented if MSR[PMM] = '0'.
14	FC2WAIT	Freeze PMC2 in wait state. 0 PMC2 is incremented if permitted by other MMCR bits. 1 PMC2 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a "wait state"; that is, when there is no process ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
15	FC2H	Freeze PMC2 in hypervisor state. 0 PMC2 is incremented if permitted by other MMCR bits. 1 PMC2 is not incremented if MSR[HV, PR] = '10'.
16:17	RESERVED	Reserved
18	FC3S	Freeze PMC3 in privileged state. 0 PMC3 is incremented if permitted by other MMCR bits. 1 PMC3 is not incremented if MSR[HV, PR] = '00'.
19	FC3P	Freeze PMC3 in problem state if MSR[PR] = '1'. 0 PMC3 is incremented if permitted by other MMCR bits. 1 If FC3PC = '0', then PMC3 is not incremented if MSR[PR] = '1'.
20	FC3PC	Freeze PMC3 in problem state if MSR[HV] = '1'. 0 PMC3 is incremented if permitted by other MMCR bits. 1 If FC3P = '0', PMC3 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC3P = '1', PMC3 is not incremented if MSR[HV, PR] = '11' (adjunct state).



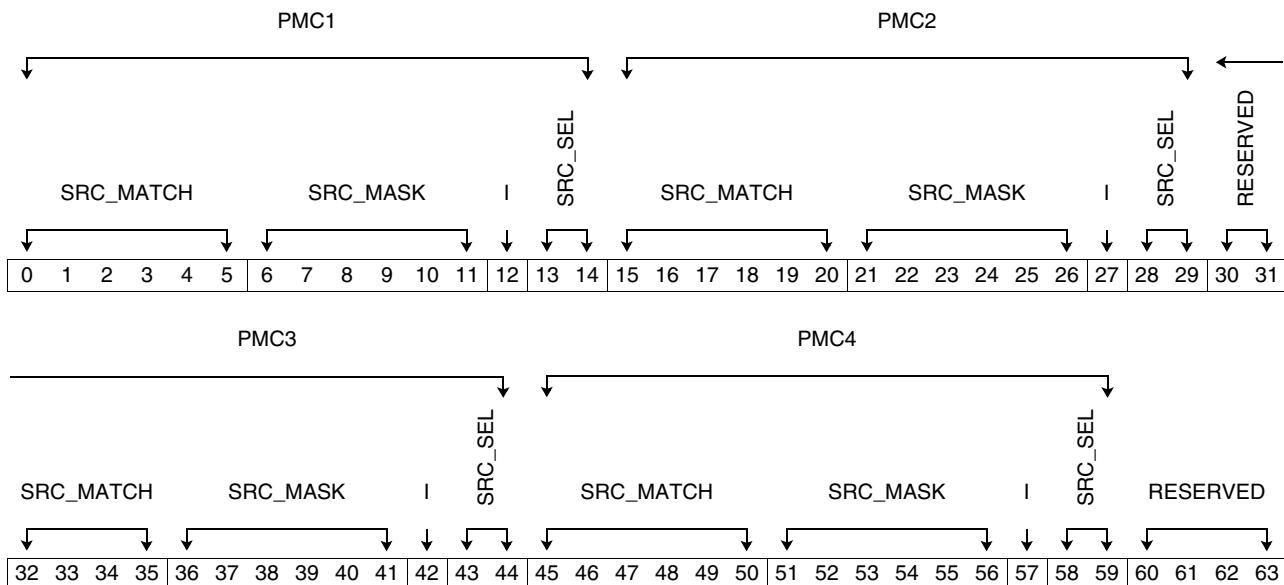
Bits	Name	Description
21	FC3M1	Freeze PMC3 while Mark = '1'. 0      PMC3 is incremented if permitted by other MMCR bits. 1      PMC3 is not incremented if MSR[PMM] = '1'.
22	FC3M0	Freeze PMC3 while Mark = 0 0      PMC3 is incremented if permitted by other MMCR bits. 1      PMC3 is not incremented if MSR[PMM] = '0'.
23	FC3WAIT	Freeze PMC3 in wait state. 0      PMC3 is incremented if permitted by other MMCR bits. 1      PMC3 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a "wait state"; that is, when no process is ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
24	FC3H	Freeze PMC3 in hypervisor state. 0      PMC3 is incremented if permitted by other MMCR bits. 1      PMC3 is not incremented if MSR[HV, PR] = '10'.
25:26	RESERVED	Reserved
27	FC4S	Freeze PMC4 in privileged state. 0      PMC4 is incremented if permitted by other MMCR bits. 1      PMC4 is not incremented if MSR[HV, PR] = '00'.
28	FC4P	Freeze PMC4 in Problem State if MSR[PR] = '1'. 0      PMC4 is incremented if permitted by other MMCR bits. 1      If FC4PC = '0', then PMC4 is not incremented if MSR[PR] = '1'.
29	FC4PC	Freeze PMC4 in Problem State if MSR[HV] = '1'. 0      PMC4 is incremented if permitted by other MMCR bits. 1      If FC4P = '0', PMC4 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC4P = '1,' PMC4 is not incremented if MSR[HV, PR] = '11' (adjunct state).
30	FC4M1	Freeze PMC4 while Mark = '1'. 0      PMC4 is incremented if permitted by other MMCR bits. 1      PMC4 is not incremented if MSR[PMM] = '1'.
31	FC4M0	Freeze PMC4 while Mark = 0 0      PMC4 is incremented (if permitted by other MMCR bits). 1      PMC4 is not incremented if MSR[PMM] = '0'.
32	FC4WAIT	Freeze PMC4 in wait state. 0      PMC4 is incremented if permitted by other MMCR bits. 1      PMC4 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = 0 when it is in a "wait state"; that is, when there is no process ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
33	FC4H	Freeze PMC4 in hypervisor state. 0      PMC4 is incremented if permitted by other MMCR bits. 1      PMC4 is not incremented if MSR[HV, PR] = '10'.
34:35	RESERVED	Reserved
36	FC5S	Freeze PMC5 in privileged state. 0      PMC5 is incremented if permitted by other MMCR bits. 1      PMC5 is not incremented if MSR[HV, PR] = '00'.
37	FC5P	Freeze PMC5 in problem state if MSR[PR] = '1'. 0      PMC5 is incremented if permitted by other MMCR bits. 1      If FC5PC = '0', then PMC5 is not incremented if MSR[PR] = '1'.



Bits	Name	Description
38	FC5PC	Freeze PMC5 in Problem State if MSR[HV] = '1'. 0      PMCn is incremented if permitted by other MMCR bits. 1      If FC5P = '0', PMC5 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC5P = '1', PMC5 is not incremented if MSR[HV, PR] = '11' (adjunct state).
39	FC5M1	Freeze PMC5 while Mark = '1'. 0      PMC5 is incremented if permitted by other MMCR bits. 1      PMC5 is not incremented if MSR[PMM] = '1'.
40	FC5M0	Freeze PMC5 while Mark = '0'. 0      PMC5 is incremented if permitted by other MMCR bits. 1      PMC5 is not incremented if MSR[PMM] = '0'.
41	FC5WAIT	Freeze PMC5 in wait state. 0      PMC5 is incremented if permitted by other MMCR bits. 1      PMC5 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a "wait state"; that is, when no process is ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
42	FC5H	Freeze PMC5 in hypervisor state. 0      PMC5 is incremented if permitted by other MMCR bits. 1      PMC5 is not incremented if MSR[HV, PR] = '10'.
43:44	RESERVED	Reserved
45	FC6S	Freeze PMC6 in privileged state. 0      PMC6 is incremented if permitted by other MMCR bits. 1      PMC6 is not incremented if MSR[HV, PR] = '00'.
46	FC6P	Freeze PMC6 in Problem State if MSR[PR] = '1'. 0      PMC6 is incremented if permitted by other MMCR bits. 1      If FC6PC = '0', then PMC6 is not incremented if MSR[PR] = '1'.
47	FC6PC	Freeze PMC6 in Problem State if MSR[HV] = '1'. 0      PMC6 is incremented if permitted by other MMCR bits. 1      If FC6P = '0', PMC6 is not incremented if MSR[HV, PR] = '01' (true problem state). If PC6P = '1', PMC6 is not incremented if MSR[HV, PR] = '11' (adjunct state).
48	FC6M1	Freeze PMC6 while Mark = '1'. 0      PMC6 is incremented if permitted by other MMCR bits. 1      PMC6 is not incremented if MSR[PMM] = '1'.
49	FC6M0	Freeze PMC6 while Mark = '0'. 0      PMC6 is incremented if permitted by other MMCR bits. 1      PMC6 is not incremented if MSR[PMM] = '0'.
50	FC6WAIT	Freeze PMC6 in wait state. 0      PMC6 is incremented if permitted by other MMCR bits. 1      PMC6 is not incremented if CTRL[RUN] = '0'. Software is expected to set CTRL[RUN] = '0' when it is in a "wait state"; that is, when no process is ready to run. <b>Note:</b> PM_CYC counts cycles regardless of the run latch (ref. CTRL).
51	FC6H	Freeze PMC6 in hypervisor state. 0      PMC6 is incremented if permitted by other MMCR bits. 1      PMC6 is not incremented if MSR[HV, PR] = '10'.
52:55	RESERVED	Reserved
56:60	L2L3_Select	A 5-bit selector for L2 and L3 events. This field takes effect when MMCR1[UNITxSEL] is set to '0110'. See <i>Table E-5 MMCR1[PMCxUNIT] Selection of Physical Event Buses</i> on page 461.
61:63	RESERVED	Reserved.

### E.2.6 Performance Monitor Mode Control Register 3 (MMCR3)

The 64-bit Performance Monitor Mode Control Register 3 (MMCR3) enables software to specify the reload source controls for PMC1 - 4.



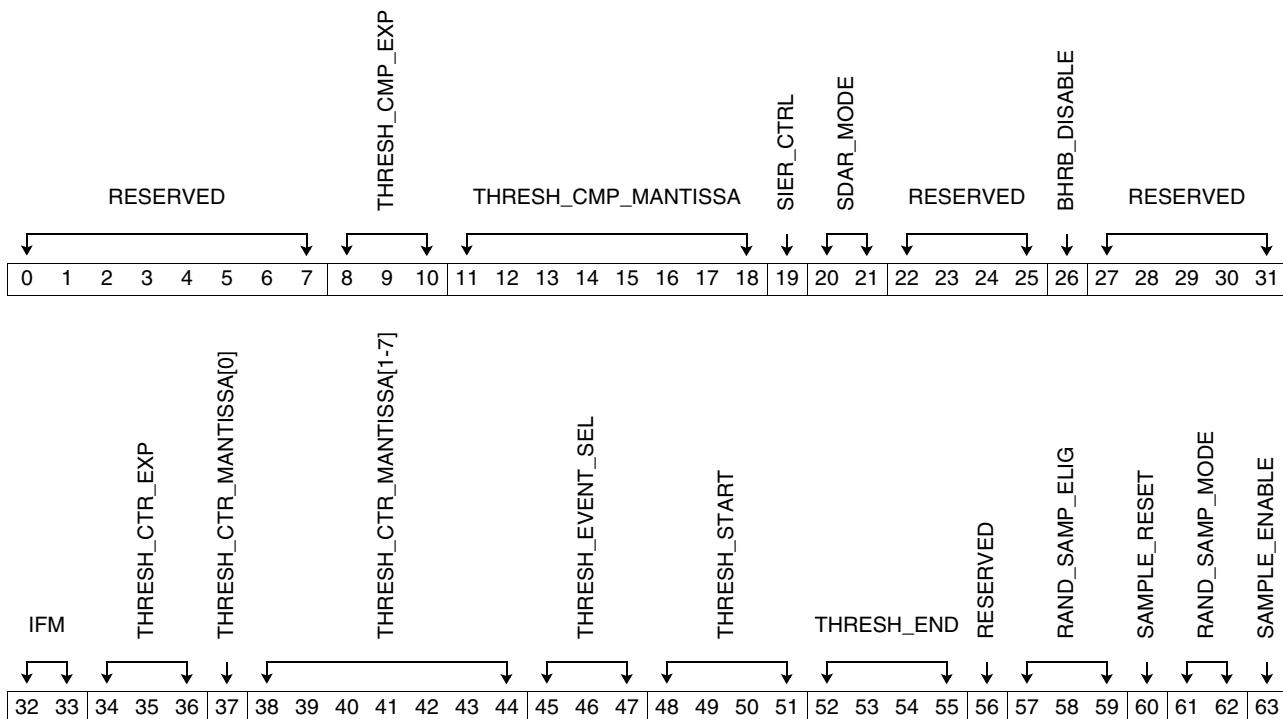
Bits	Field Name	PMCx	Description
0:5	SRC_MATCH	PMC1	Compare values in SRC_MATCH to reload_src.
6:11	SRC_MASK		Ignores bits set to '1' in compare of SRC_MATCH to reload_src.
12	I		Invert output of applying SRC_MATCH and SRC_MASK. Event = I XOR AND ((SRC_MATCH XNOR RELOAD_ARC) OR SRC_MASK)
13:14	SRC_SEL		00 Instruction reload 01 Instruction page table reload 10 Data reload 11 Data page table reload
15:20	SRC_MATCH	PMC2	Compare values in SRC_MATCH to reload_src.
21:26	SRC_MASK		Ignores bits set to '1' in compare of SRC_MATCH to reload_src.
27	I		Invert output of applying SRC_MATCH and SRC_MASK. Event = I XOR AND ((SRC_MATCH XNOR RELOAD_ARC) OR SRC_MASK)
28:29	SRC_SEL		00 Instruction reload 01 Instruction page table reload 10 Data reload 11 Data page table reload



Bits	Field Name	PMCx	Description
30:31	RESERVED	PMC3	Reserved.
32:35	SRC_MATCH		Compare values in SRC_MATCH to reload_src.
36:41	SRC_MASK		Ignores bits set to '1' in compare of SRC_MATCH to reload_src.
42	I		Invert output of applying SRC_MATCH and SRC_MASK. Event = I XOR AND ((SRC_MATCH XNOR RELOAD_ARC) OR SRC_MASK)
43:44	SRC_SEL		00 Instruction reload 01 Instruction page table reload 10 Data reload 11 Data page table reload
45:50	SRC_MATCH		Compare values in SRC_MATCH to reload_src.
51:56	SRC_MASK		Ignores bits set to '1' in compare of SRC_MATCH to reload_src.
57	I		Invert output of applying SRC_MATCH and SRC_MASK. Event = I XOR AND ((SRC_MATCH XNOR RELOAD_ARC) OR SRC_MASK)
58:59	SRC_SEL	PMC4	00 Instruction reload 01 Instruction page table reload 10 Data reload 11 Data page table reload
60:63	RESERVED		Reserved.

### E.2.7 Monitor Mode Control Register A (MMCRA)

The 64-bit Monitor Mode Control Register A (MMCRA) contains bits for controlling the sampling process, BHRB filtering, and threshold events.



Bits	Field Name	Description																												
0:7	RESERVED	Reserved.																												
8:10	THRESH_CMP_EXP	The 3-bit exponent portion of threshold compare floating point.																												
11:18	THRESH_CMP_MANTISSA	<p>The 8-bit mantissa portion of threshold compare floating point value.  <b>Note:</b> The THRESH_CMP_Mantissa upper two bits must be non-zero when THRESH_CMP_EXP is non-zero. This is due to the way the floating-point threshold counter is defined, operating as an event up-counter only.</p> <p>The threshold counter (THRESH_CTR_EXP &amp; THRESH_CTR_MANTISSA) strictly adheres to the following ranges of values:</p> <table border="1"> <thead> <tr> <th>CTR_EXP</th> <th>CTR_MANTISSA</th> <th>Thresh Event Actual Count</th> <th>Multi</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0 to 255 (0x00 to 0xFF)</td> <td>0 to 255 (0x00 - 0xFF)</td> <td></td> </tr> <tr> <td>1</td> <td>64 to 255 (0x40 to 0xFF)</td> <td>256 to 1023 (0x100-0x3FF)</td> <td>x16</td> </tr> <tr> <td>2</td> <td>128 to 255 (0x40 to 0xFF)</td> <td>1024 to 4095 (0x400-0xFFFF)</td> <td>x16</td> </tr> <tr> <td>3</td> <td>128 to 255 (0x40 to 0xFF)</td> <td>4096 to 16383x64</td> <td></td> </tr> <tr> <td>4</td> <td>128 to 255 (0x40 to 0xFF)</td> <td>16384 to 65535 x256</td> <td></td> </tr> <tr> <td>5</td> <td>128 to 255 (0x40 to 0xFF)</td> <td>65536 to 262143 x1024</td> <td></td> </tr> </tbody> </table> <p>The hardware starts counting a new threshold sequence from a THRESH_CTR_Mantissa = '00000000' (THRESH_CTR_EXP = 0) and when it has reached '11111111' it rolls over to '01000000' and the exponent is incremented by 1. See <i>Section E.4.2 Thresholding Operation</i> on page 488 for a full description of thresholding.</p>	CTR_EXP	CTR_MANTISSA	Thresh Event Actual Count	Multi	0	0 to 255 (0x00 to 0xFF)	0 to 255 (0x00 - 0xFF)		1	64 to 255 (0x40 to 0xFF)	256 to 1023 (0x100-0x3FF)	x16	2	128 to 255 (0x40 to 0xFF)	1024 to 4095 (0x400-0xFFFF)	x16	3	128 to 255 (0x40 to 0xFF)	4096 to 16383x64		4	128 to 255 (0x40 to 0xFF)	16384 to 65535 x256		5	128 to 255 (0x40 to 0xFF)	65536 to 262143 x1024	
CTR_EXP	CTR_MANTISSA	Thresh Event Actual Count	Multi																											
0	0 to 255 (0x00 to 0xFF)	0 to 255 (0x00 - 0xFF)																												
1	64 to 255 (0x40 to 0xFF)	256 to 1023 (0x100-0x3FF)	x16																											
2	128 to 255 (0x40 to 0xFF)	1024 to 4095 (0x400-0xFFFF)	x16																											
3	128 to 255 (0x40 to 0xFF)	4096 to 16383x64																												
4	128 to 255 (0x40 to 0xFF)	16384 to 65535 x256																												
5	128 to 255 (0x40 to 0xFF)	65536 to 262143 x1024																												
19	SIER_CTRL	When this bit is set to '1', SIER bits 16:28 are changed to read data RA bits 44:56.																												



Bits	Field Name	Description
20:21	SDAR_MODE	Continuous sampling. These bits specify how the SDAR should be updated in continuous sampling mode. 00 No updates. 01 Continuous sampling mode; update SDAR on a TLB miss. 10 Continuous sampling mode; update SDAR on a data-cache miss. 11 Continuous sampling mode; update SDAR on a store issue.
22:25	RESERVED	Reserved.
26	BHRB_DISABLE	When this bit is asserted, the branch history rolling buffer function is disabled. 0 BHRB is enabled if permitted by other MMCR bits. 1 BHRB is disabled regardless of other MMCR bits.
27:31	RESERVED	Reserved.
32:33	IFM	BHRB instruction filtering mode. 00 All taken branch instructions are entered into the BHRB unless prevented by other filtering fields. 01 Do not enter jump instructions or return instructions into the BHRB.
34:36	THRESH_CTR_EXP	The 3-bit exponent portion of the threshold floating-point counter.
37	THRESH_CTR_MANTISSA[0]	Bit 0 of the 8-bit mantissa portion of the threshold floating-point counter.
38:44	THRESH_CTR_MANTISSA[1-7]	Bits 1 - 7 of the 8-bit mantissa portion of threshold floating-point counter.
45:47	THRESH_EVENT_SEL	Selection for an event specified for threshold counting. PMC1 - 4 event counting, selected as shown in <i>Table E-6 Threshold Event Selection (MMCRA[45:47])</i> on page 470 and adheres to possible freeze conditions set up in the MMCR0, MMCR2, and MMCRC registers.
48:51	THRESH_START	Threshold start event. See <i>Table E-7 Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55])</i> on page 470.
52:55	THRESH_END	Threshold end event. See <i>Table E-7 Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55])</i> on page 470.
56	RESERVED	Reserved.
57:59	RAND_SAMP_ELIG	Eligibility criteria. See <i>Table E-8 Random Sampling Eligibility Criteria</i> on page 472.
60	SAMPLE_RESET	When this bit is set to '1', the sampling state machine is forced from the start state into the idle state. If the state machine is not in the start state when this bit is set, it has no effect.
61:62	RAND_SAMP_MODE	Random sampling mode (SM). 00 Random instruction sampling (RIS). Instructions that meet the criterion specified in the RAND_SAMP_ELIG field for random instruction sampling are randomly selected and sampled. 01 Random load/store sampling (LSS). Events that meet the criterion specified in the RAND_SAMP_ELIG field for random load/store sampling are randomly selected for sampling. 10 Random branch facility sampling (BFS). Events that meet the criterion specified in the RAND_SAMP_ELIG field for random branch facility sampling are randomly selected for sampling. 11 Reserved.
63	SAMPLE_ENABLE	0 Continuous sampling. 1 Random sampling.



*Table E-6. Threshold Event Selection (MMCRA[45:47])*

MMCRA[45:47] THRESH_EVENT_SEL	Description
000	Do not count; disable thresholding.
001	Count the number of cycles that the CTRL run latch is set (does not depend on freeze conditions).
010	Count the number of completed Instructions while the CTRL run latch is set (does not depend on freeze conditions).
011	Reserved.
100	Count PMC1 events (depends on freeze conditions).
101	Count PMC2 events (depends on freeze conditions).
110	Count PMC3 events (depends on freeze conditions).
111	Count PMC4 events (depends on freeze conditions).

*Table E-7 lists the threshold start/stop event selection.*

*Table E-7. Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55])*

MMCRA[48] MMCRA[52]	MMCRA[49:51] MMCRA[53:55]	Description
0	000	Reserved.
0	001	PM_MRK_INST_DECODED Sampled instruction is decoded.
0	010	PM_MRK_INST_DISP Sampled instruction is dispatched.
0	011	PM_MRK_INST_ISSUED Sampled instruction is issued.
0	100	PM_MRK_INST_FIN Sampled instruction is finished.
0	101	PM_MRK_INST_CMPL Sampled instruction has completed.
0	110	PM_MRK_LD_MISS_L1 Sampled instruction L1 load cache miss.
0	111	PM_MRK_L1_RELOAD_VALID Sampled instruction L1 reload valid.
1	000	Event selected in MMCR1 for PMC1 occurred.
1	001	Event selected in MMCR1 for PMC2 occurred.
1	010	Event selected in MMCR1 for PMC3 occurred.
1	011	Event selected in MMCR1 for PMC4 occurred.
1	100	PM_MRK_NTF_FIN Sampled group is next-to-finish.



Table E-7. Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55])

MMCRA[48] MMCRA[52]	MMCRA[49:51] MMCRA[53:55]	Description
1	101	PM_MRK_L2_RC_DISP <u>RC</u> machine dispatched for the sampled instruction.
1	110	PM_MRK_ST_DONE_L2 RC machine is done for the sampled instruction.
1	111	Reserved.



Table E-8 describes the random sampling eligibility criteria.

*Table E-8. Random Sampling Eligibility Criteria*

MMCRA[57:59] RAND_SAMP_ELIG	MMCRA[61:62] RAND_SAMP_MODE	MMCRA[63] SAMPLE_ENABLE	Eligibility Criteria
000	00	1	All instructions are eligible.
001	00	1	Load/store. Any operation that is routed to the LSU (for example, load or store).
010	00	1	Probe NOP.
011	00	1	<b>Larx/stcx</b>
100	00	1	Reserved.
101	00	1	Load sampling. Sample only load instructions.
110	00	1	Long latency operation ( <b>div/sqrt/mul</b> ).
111	00	1	Store sampling. Sample only store instructions.
000	01	1	Load misses.
001	01	1	Reserved.
010	01	1	Reserved.
011	01	1	Reserved.
100	01	1	Load Hit Store (LHS). Only for cases where a load is able to forward data from the store queue and finish. Excludes cases where a load-hit-store in the store queue, but the load only partially overlaps with the store.
110	01	1	Reserved.
111	01	1	Reserved.
000	10	1	Branch mispredicts.
001	10	1	Branch mispredicts (Direction, CR, or CTR).
010	10	1	Branch mispredicts (TA).
011	10	1	Taken branches.
100	10	1	Nonrepeating branches.
101	10	1	All branches that require prediction.
110	10	1	Reserved.
111	10	1	Reserved.

### E.2.8 Sampled Instruction Event Register (SIER)

The 64-bit Sampled Instruction Event Register stores information that pertains to a sampled-marked instruction.

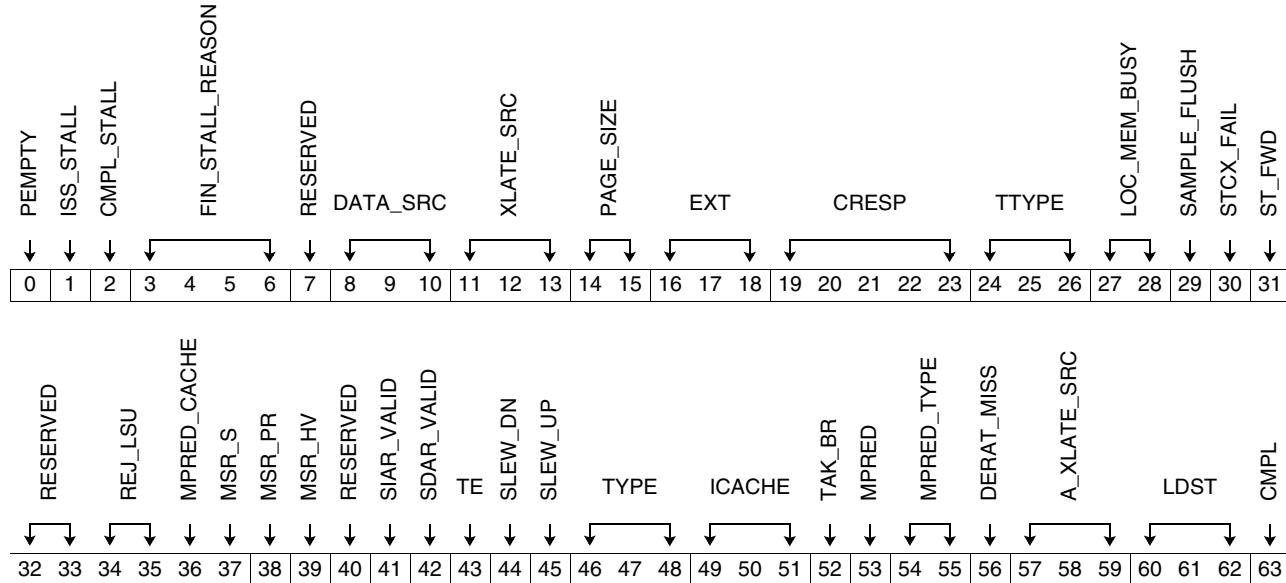


Table E-9. Sampled Instruction Event Register (SIER)

Bits	Field Name	Description
0	PEMPTY	Pipeline was empty (first instruction after GCT no slot).
1	ISS_STALL	Sampled instruction issue stall. The marked instruction was issued after it became NTC.
2	CMPL_STALL	Sampled instruction group completion stall. The marked instruction became NTC some time before completion.
3:6	FIN_STALL_REASON	Sampled instruction group finish stall reason. 0000 Reserved. 0001 Marked finish before NTC 0010 Reserved 0011 Reserved 0100 LSU D-cache miss 0101 LSU load finish 0110 LSU store forward 0111 LSU store 1000 FXU multi-cycle 1001 BRU finish mispredict 1010 VSU multi-cycle 1011 Reserved 1100 FXU other 1101 VSU other 1110 BRU other 1111 Simple FX executed in LSU
7	RESERVED	Reserved.
8:10	DATA_SRC	Sampled instruction microarchitecture-dependent data source information for loads and extended store information. (Encoding defined in Table E-10 on page 476.)



*Table E-9. Sampled Instruction Event Register (SIER)*

Bits	Field Name	Description
11:13	XLATE_SRC	Sampled instruction microarchitecture-dependent data source information included with data translation source load information. (Encoding defined in <i>Table E-10</i> on page 476.)
14:15	PAGE_SIZE	Sampled instruction suffered an <u>ERAT</u> miss for the page size. 00 4 KB 01 64 KB 10 16 MB 11 16 GB
16:18	EXT	Sampled Instruction data/store/translation extension information. (Encoding defined in <i>Table E-10</i> on page 476.)
19:23	CRESP	Combined response.
24:26	TTYPE	SMP interconnect event ttype.
27:28	LOC_MEM_BUSY	Local memory busy.
29	SAMPLE_FLUSH	The sampled instruction was flushed.
30	STCX_FAIL	Sampled <b>stcx</b> failed.
31	ST_FWD	Store forward.
32:33	RESERVED	Reserved.
34:35	REJ_LSU	Sampled instruction suffered a reject. 00 No reject 01 LMQ full 10 TIQ full 11 Reserved
36	MPRED_CCACHE	<u>C-cache</u> misprediction.
37	MSR_S	Sampled ultrvisor state (MSR[S] = '1').
38	MSR_PR	Sampled problem state (MSR[PR] = '1').
39	MSR_HV	Sampled hypervisor state (MSR[HV] = '1').
40	RESERVED	Reserved
41	SIAR_VALID	SIAR is valid for a sampled instruction.
42	SDAR_VALID	SDAR is valid for a sampled instruction.
43	TE	Threshold exceeded.
44	SLEW_DN	Frequency was slewed down for any reason.
45	SLEW_UP	Frequency was slewed up for any reason.
46:48	TYPE	Type of sampled instruction. 000 Reserved. 001 Sampled instruction is a load. 010 Sampled instruction is a store (includes syncs and msgsnd). 011 Sampled instruction is a branch. 100 Sampled instruction is a floating-point instruction. 101 Sampled instruction is a fixed-point instruction. 110 Sampled instruction is an IFU but nonbranch. 111 Sampled instruction is a <b>lарx</b> or <b>stcx</b> .

Table E-9. Sampled Instruction Event Register (SIER)

Bits	Field Name	Description																
49:51	ICACHE	<p>000 Reserved          001 Sampled instruction hit in the I-cache          010 Sampled instruction hit in the L2 cache.          011 Sampled instruction hit in the L3 cache.          100 Sampled instruction is an L3 miss.          101 Reserved.          110 Reserved.          111 Reserved.</p> <p><b>Note:</b> This field is not valid when random event sampling is enabled MMCRA[63] = '1' and MMCRA[61:62] ≠ '00'.</p>																
52	TAK_BR	Sampled instruction is a taken branch.																
53	MPRED	Sampled branch instruction is mispredicted.																
54:55	MPRED_TYPE	<p>Sampled instruction branch mispredicted information type.</p> <table> <tr><td>00</td><td>Reserved.</td></tr> <tr><td>01</td><td>Branch mispredicted due to direction.</td></tr> <tr><td>10</td><td>Branch mispredicted due to target address.</td></tr> <tr><td>11</td><td>Reserved.</td></tr> </table>	00	Reserved.	01	Branch mispredicted due to direction.	10	Branch mispredicted due to target address.	11	Reserved.								
00	Reserved.																	
01	Branch mispredicted due to direction.																	
10	Branch mispredicted due to target address.																	
11	Reserved.																	
56	DERAT_MISS	Sampled instruction incurred a D-ERAT miss.																
57:59	A_XLATE_SRC	<p>Sampled instruction data translation information.</p> <table> <tr><td>000</td><td>Reserved.</td></tr> <tr><td>001</td><td>TLB hit.</td></tr> <tr><td>010</td><td>Data translation hit in the L2 cache.</td></tr> <tr><td>011</td><td>Data translation hit in the L3 cache.</td></tr> <tr><td>100</td><td>Data translation resolved from memory.</td></tr> <tr><td>101</td><td>Data translation resolved from on-chip cache (other than local L2/L3 cache).</td></tr> <tr><td>110</td><td>Data translation resolved from off-chip cache.</td></tr> <tr><td>111</td><td>Reserved.</td></tr> </table>	000	Reserved.	001	TLB hit.	010	Data translation hit in the L2 cache.	011	Data translation hit in the L3 cache.	100	Data translation resolved from memory.	101	Data translation resolved from on-chip cache (other than local L2/L3 cache).	110	Data translation resolved from off-chip cache.	111	Reserved.
000	Reserved.																	
001	TLB hit.																	
010	Data translation hit in the L2 cache.																	
011	Data translation hit in the L3 cache.																	
100	Data translation resolved from memory.																	
101	Data translation resolved from on-chip cache (other than local L2/L3 cache).																	
110	Data translation resolved from off-chip cache.																	
111	Reserved.																	
60:62	LDST	<p>Sampled load/store instruction information.</p> <table> <tr><td>000</td><td>Reserved.</td></tr> <tr><td>001</td><td>Load hit the L1 D-cache.</td></tr> <tr><td>010</td><td>Load hit in the L2 cache.</td></tr> <tr><td>011</td><td>Load hit in the L3 cache.</td></tr> <tr><td>100</td><td>Load resolved from memory.</td></tr> <tr><td>101</td><td>Load resolved from on-chip cache (other than local L2/L3 cache).</td></tr> <tr><td>110</td><td>Load resolved from off-chip cache.</td></tr> <tr><td>111</td><td>Store missed L1 and sent to cache/memory subsystem.</td></tr> </table>	000	Reserved.	001	Load hit the L1 D-cache.	010	Load hit in the L2 cache.	011	Load hit in the L3 cache.	100	Load resolved from memory.	101	Load resolved from on-chip cache (other than local L2/L3 cache).	110	Load resolved from off-chip cache.	111	Store missed L1 and sent to cache/memory subsystem.
000	Reserved.																	
001	Load hit the L1 D-cache.																	
010	Load hit in the L2 cache.																	
011	Load hit in the L3 cache.																	
100	Load resolved from memory.																	
101	Load resolved from on-chip cache (other than local L2/L3 cache).																	
110	Load resolved from off-chip cache.																	
111	Store missed L1 and sent to cache/memory subsystem.																	
63	CMPL	Sampled instruction completed.																



Table E-10 shows the implementation-dependent extension to data source encodes.

*Table E-10. Implementation-Dependent Extension to Data Source Encodes*

Architected Bits SIER[LDST] SIER[A_XLATE_SRC]	Implementation- Dependent Bits Encoding SIER[DATA_SRC] SIER[XLATE_SRC]	Performance Monitor Reload Bus Source Encoding (Binary)	Implementation-Dependent Bits Description
L2 Hit			
010	000	000000	Private L2 cache for this core sourced data (or NCU loads) without dispatch conflicts
010	001	000001	Private L2 cache for this core sourced data in the Mepf state without dispatch conflicts. (L3 prefetch brought line in Me)
010	010	000010	Private L2 cache for this core sourced data that had a dispatch conflict on load-hit-store.
010	011	000011	Private L2 cache for this core sourced data that had a dispatch conflict other than load-hit-store.
010	100		Reserved
010	101		Reserved
010	110		Reserved
010	111		Reserved
L3 Hit			
011	000	000100	Private L3 cache for this core sourced data without dispatch conflicts
011	001	000101	Private L3 cache for this core sourced data in the Mepf state without dispatch conflicts. (L3 prefetch brought line in Me)
011	010	000111	Private L3 cache for this core sourced data that had a dispatch conflict.
011	011		Reserved
011	100		Reserved
011	101		Reserved
011	110		Reserved
011	111		Reserved
Memory			
100	000		Reserved
100	001	100101	Local (on-chip) memory controller
100	010	100110	Local OpenCapp Cache
100	011	100111	Local OpenCapp memory
100	100	110101	Remote (same group, off-chip) memory controller
100	101	11011X	Remote OpenCapp Cache or memory
100	110	111101	Distant (outside group, off-chip) memory controller
100	111	11111X	Distant OpenCapp Cache or memory
On-Chip Cache			

Table E-10. Implementation-Dependent Extension to Data Source Encodes

Architected Bits SIER[LDST] SIER[A_XLATE_SRC]	Implementation- Dependent Bits Encoding SIER[DATA_SRC] SIER[XLATE_SRC]	Performance Monitor Reload Bus Source Encoding (Binary)	Implementation-Dependent Bits Description
101	000	100000	Data sourced from another core's L2 cache on the same chip in the same regent in valid, but non-exclusive state
101	001	100001	Data sourced from another core's L2 on the same chip in the same regent in exclusive state
101	010	100010	Data sourced from another core's L3 on the same chip in the same regent in valid, but non-exclusive state
101	011	100011	Data sourced from another core's L3 on the same chip in the same regent in exclusive state
101	100	101000	Data sourced from another core's L2 on the same chip outside the regent domain in valid, but non-exclusive state
101	101	101001	Data sourced from another core's L2 on the same chip outside the regent domain in exclusive state
101	110	101010	Data sourced from another core's L3 on the same chip outside the regent domain in valid, but non-exclusive state
101	111	101011	Data sourced from another core's L3 on the same chip outside the regent domain in exclusive state
Off-Chip Cache			
110	000	1100X0	Data sourced from a remote (same group, off-chip) core's L2 or L3 in valid, but non-exclusive state
110	001	1100X1	Data sourced from a remote (same group, off-chip) core's L2 or L3 in exclusive state
110	010	1110X0	Data sourced from a distant (outside group, off-chip) core's L2 or L3 in valid, but non-exclusive state
110	011	1110X1	Data sourced from a distant (outside group, off-chip) core's L2 or L3 in exclusive state
110	100		Reserved
110	101		Reserved
110	110		Reserved
110	111		Reserved
Store Information (only applicable for SIER[LDST])			
111	000		Store did not require an RC dispatch.
111	001		Store completed in L2 cache without intervention.
111	010		Reserved
111	011		Reserved
111	100		Reserved
111	101		Reserved
111	110		Reserved
111	111		Reserved



Table E-11 shows the implementation-dependent extension bits for data source encodes (SIER[EXT]).

*Table E-11. Implementation-Dependent Extension Bits for Data Source Encodes (SIER[EXT])*

Encoding	Case	Description	Prediction
000	Non-Fabric operation	Private L2/L3 hit for this core sourced data (or NCU load)	NA
001	Fabric Initial/Final Pump = Chip	Initial and final pump scope and data sourced across this scope	Correct
010	Fabric Initial/Final Pump = Group	Initial and final pump scope and data sourced across this scope.	Correct
011	Fabric Initial/Final Pump = System	Initial and final pump scope and data sourced across this scope	Correct
100	Fabric Final Pump = Group	Final pump scope to get data sourced ended up larger than initial pump scope (start too small)	Mispredict
101	Fabric Final Pump = Group	Final pump scope got data from source that was at smaller scope (should have started smaller)	Mispredict
110	Fabric Final Pump = System	Final pump scope to get data sourced ended up larger than initial pump scope. (start too small)	Mispredict
111	Fabric Final Pump = System	Final pump scope got data from source that was at smaller scope	Mispredict

0 - Hit-mod case occurs when L2 = M (that is, true M data)  
 1 - Hit-mod case occurs when L2 = M, MuMe, S (that is, early data state)

### E.2.9 Sampled Instruction Event Register 2 (SIER2)

A 64-bit register that stores information that pertains to a sampled instruction.

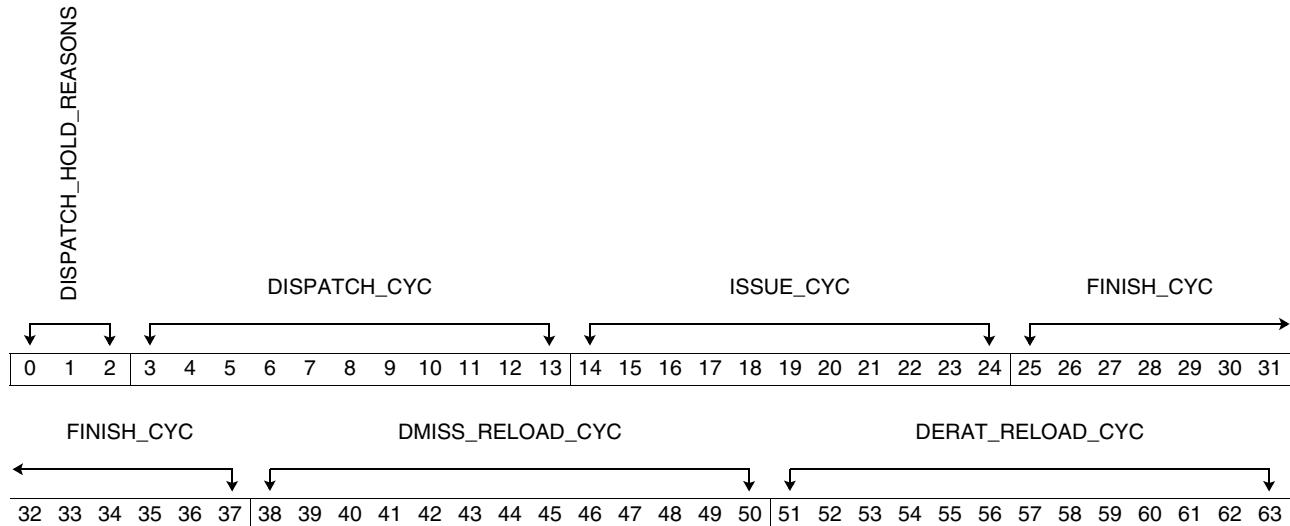


Table E-12. Sampled Instruction Event Register 2 (SIER2)

Bits	Name	Description
0:2	DISPATCH_HOLD_REASON	The 3-bit reason why the marked instruction was held at dispatch. If more than one reason occurred to this marked instruction, the most recent one will be shown. 000 No hold. 001 Synchronizing instruction hold other. 010 HALT. 011 Wait on ICT empty, SRQ empty, or EAT sync with ICT (for example, <code>tlbie</code> ). 100 Wait on scoreboard (in the Power10 chip, this will include VSCR and FPSCR, which have their own scoreboard). 101 Issue queue full (ISQ). 110 STF mapper or SRB full: GPR (includes Count, Link, TAR), VSR, VMR, FPR. 111 XVFC mapper or SRB full.
3:13	DISPATCH_CYC	The 11-bit saturating counter of the number of cycles that elapsed between decode and dispatch for the sampled instruction. A value of '1111111111' indicates 2047 or more cycles. This counter will always be 0 when random event sampling is used.
14:24	ISSUE_CYC	The 11-bit saturating counter of the number of cycles that elapsed between dispatch and issue for the sampled instruction. A value of '1111111111' indicates 2047 or more cycles. This counter will always be 0 when random event sampling is used. <b>Note:</b> The <code>mtspr</code> and <code>mfspr</code> instructions do not issue and do not finish. Issue and finish cycles will not be reported for these instructions.
25:37	FINISH_CYC	The 13-bit saturating counter of the number of cycles that elapsed between issue (in the case of random instruction sampling) or the time the instruction is marked (in the case of random event sampling) and finish for the sampled instruction. A value of '111111111111' indicates 8191 or more cycles. <b>Note:</b> The <code>mtspr</code> and <code>mfspr</code> instructions do not issue and do not finish. Issue and finish cycles will not be reported for these instructions.
38:50	DMISS_RELOAD_CYC	The 13-bit saturating counter of the reload latency (from miss-to-reload) for the sampled instruction. A value of '111111111111' indicates 8191 or more cycles. <b>Note:</b> This latency includes cycles to satisfy a translation miss (SIER2[DERAT_RELOAD_CYC]).



Table E-12. Sampled Instruction Event Register 2 (SIER2)

Bits	Name	Description
51:63	DERAT_RELOAD_CYC	The 13-bit saturating counter of the DERAT reload latency (from the time the original L1 miss was detected to when the DERAT is reloaded with the missing address) for the marked instruction. A value of '111111111111' indicates 8191 or more cycles. This value is only valid if SIER[DERAT_MISS] is set. If SIER[DERAT_MISS] is clear, this value will match SIER2[DMISS_RELOAD_CYC].

### E.2.10 Sampled Instruction Event Register 3 (SIER3)

The Sampled Instruction Event Register 3 is a 64-bit register that stores information that pertains to a sampled instruction.

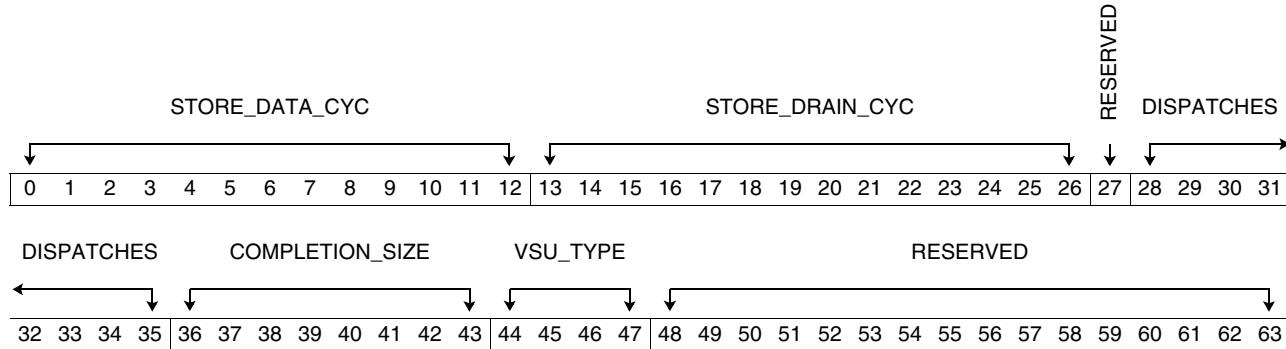


Table E-13. Sampled Instruction Event Register 3 (SIER3)

Bits	Name	Description
0:12	STORE_DATA_CYC	The 13-bit saturating counter of the number of cycles that a marked store is waiting for data to become available. A value of '111111111111' indicates 8191 or more cycles.
13:26	STORE_DRAIN_CYC	The 14-bit saturating counter of the number of cycles that it takes a store to completely drain from the core, from the cycle the store completes in the core to the cycle the corresponding L2 machine is released. A value of '111111111111' indicates 16383 or more cycles.
27	RESERVED	Reserved.
28:35	DISPATCHES	The number of instructions that were dispatched from the cycle after the marked instruction was dispatched until it was either flushed or completed. If SIER[SAMPLE_FLUSH] is set, all these instructions will be flushed.
36:43	COMPLETION_SIZE	The number of instructions that completed with this marked instruction. The count includes the marked instruction.
44:47	VSU_TYPE	4-bit VSU instruction type reported at issue time. 0000 Permute 0001 Matrix Multiply (MM) 0010 DX 0011 CY 0100 FX Divide 0101 DFU 0110 GPR Mult 0111 BFU (plain floating point, scalar) 1000 ALU2 (2 cycle) 1001 FX (1 cycle) 1010 Reserved 1011 Branch 1100 Load 1101 Reserved 1110 Store
48:63	RESERVED	Reserved.

### E.2.11 Sampled Instruction Address Register (SIAR)



Table E-14. PMU-Related Bits in the Sampled Instruction Address Register (SIAR)

Bits	Field Name	Description
0:63	SampIA	Sampled Instruction Address.

### E.2.12 Sampled Data Address Register (SDAR)

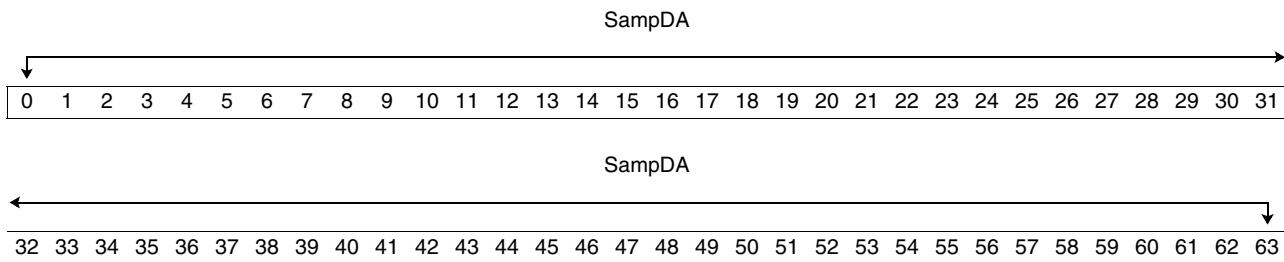


Table E-15. PMU-Related Bits in the Sampled Data Address Register (SDAR)

Bits	Field Name	Description
0:63	SampDA	Sampled Data Address



## E.3 Power10 Sampling Support

Sampling or profiling is a common approach to associate expensive performance events in a processor to instruction and data addresses. Profiling enables the identification of hotspots in code and data, finds performance-sensitive areas, and identifies problem instructions, data areas, or both. Sampling is commonly achieved by identifying a particular instruction and collecting detailed information about that instruction (instruction sampling or marking).

The Power ISA provides five SPRs to retain information about the sampled instructions. The Sampled Instruction Address Register (SIAR) captures the effective address of the sampled instruction. The Sampled Data Address Register (SDAR) captures the effective address of the sampled instruction's data operand, if any. The Sampled Instruction Event Registers (SIER, SIERA, SIERB) capture event information for the sampled instruction.

The SIAR indicates which instruction was responsible for the events indicated in the SIERs and for any marked event that a configured PMC overflowed on, and the SDAR adds the data address when applicable. SIER, SIER2, and SIER3, each record up to 64 bits of information pertaining to the marked instruction during its lifetime in the pipeline. These registers enable the collection of a wealth of pertinent information for the marked instruction. Events that can be attributed to a sampled instruction are called *marked events* (PM\_MRK\_\*). By profiling on marked events, it is possible to uniquely identify which instruction caused a particular event.

Two bits in the SIER indicate when the SIAR and SDAR apply to the same instruction: SIER[SIAR\_Valid] and SIER[SDAR\_Valid]; see *Appendix E.2.8 Sampled Instruction Event Register (SIER)* on page 473. The SIAR and SDAR are never cleared and SDAR is not always recaptured when a new instruction is sampled. Therefore, the indicator bits are needed to show that these registers are not for a previous sampled instruction. The sampled registers can only be updated by the processor when performance monitor exceptions are enabled. Performance monitor exceptions toggle the enable bit, locking the contents of the sampled registers. This makes it possible to profile code not enabled for interrupts.

The Power10 processor supports the following three sampling modes:

- Continuous Sampling collects information for every instruction completion and select data cache operations. Continuous sampling is useful for profiling on execution frequency and cache-line accesses, among other things.
- Random Instruction Sampling (RIS) marks one instruction at a time and tracks its execution through the processor pipeline.
- Random Event Sampling (RES) marks an instruction after an event has happened to an instruction. This mode improves sampling rates for some very important performance-sensitive events, such as branch mispredicts.

### E.3.1 Sampled Instruction Address Register

The SIAR is a 64-bit register that contains the effective address of the sampled instruction. When continuous sampling is enabled by MMCR0[PMAE] = '1' and MMCRA[SAMPLE\_ENABLE] = '0', all instructions are sampled. When a performance monitor alert occurs, the SIAR contains the effective address of the instruction that was being executed, possibly out of order, at or around the time that the performance monitor alert occurred.



When random sampling is enabled by MMCRO[PMAE] = '1' and MMCRA[SAMPLE\_ENABLE] = '1', only instructions specified by the MMCRA list fields are sampled. When a performance monitor alert occurs, the SIAR contains the effective address of the last sampled instruction that had completed when the performance monitor alert occurred. The contents of SIAR can be altered by the processor hardware, if and only if, MMCRO[PMAE] = '1'. Therefore, after the performance monitor alert occurs, the contents of SIAR are not altered by the hardware until software sets MMCRO[PMAE] to '1'. After software sets MMCRO[PMAE] = '1', the contents of SIAR are undefined until the next performance monitor alert occurs.

### E.3.2 Sampled Data Address Register

The SDAR is a 64-bit register that contains the effective address of the sampled data when a performance monitor alert occurs. When a performance monitor alert occurs, the SDAR is set to the effective address of the storage operand of an instruction that was being executed, possibly out-of-order, at or around the time that the performance monitor alert occurred. This storage operand is called the "sampled data". The sampled data can be, but is not required to be, the storage operand (if any) of the sampled instruction. The contents of SDAR can be altered by the processor hardware, if and only if, MMCRO[PMAE] = '1'. Therefore, after the performance monitor alert occurs, the contents of SDAR are not altered by the hardware until software sets MMCRO[PMAE] to '1'. After software sets MMCRO[PMAE] = '1', the contents of SDAR are undefined until the next performance monitor alert occurs.

### E.3.3 Continuous Sampling

The Power10 processor supports continuous sampling where the SIAR and SDAR are continuously loaded, as long as MMCRO[PMAE] = '1', MMCRA[63] = '0', and MSR[BE, SE] = '00'.

- The SIAR is loaded at completion time with the effective address of the youngest instruction.
- The SDAR is loaded at execution time based on the mode selected by MMCRA[SDAR\_MODE]. See *Table E-16*. The SIAR and SDAR do not have to correlate to the same instruction in continuous sampling mode.
- SIER[SIAR\_VALID] and SIER[SDAR\_VALID] are cleared to '0' when in continuous sampling mode.

*Table E-16. SDAR Modes for Continuous Sampling*

MMCRA[20:21] SDAR_MODE	Mode Description
00	Continuous Sampling will not update the SDAR.
01	Continuous Sampling updates SDAR on TLB miss.
10	Continuous Sampling updates SDAR on D-cache miss.
11	Continuous Sampling updates SDAR on Store issue.

### E.3.4 Random Instruction Sampling (RIS)

The Power10 processor supports instruction-based sampling, where an instruction is randomly picked during group formation based on eligibility criteria, selected by MMCRA[RAND\_SAMP\_ELIG] and MMCRA[RAND\_SAMP\_MODE]. The SIAR is loaded with the exact instruction address of the marked instruction at completion time. If applicable, the SDAR is loaded with the data effective address by the load store unit (LSU).

As shown in *Table E-17*, random sampling is enabled when MMCRA[63] = '1'. RIS is selected when MMCRA[61:61] = '00'. Additional criteria are selected in MMCRA[57:59].

*Table E-17. Random Instruction Sampling Modes*

MMCRA[57:59] RAND_SAMP_ELIG	MMCRA[61:62] RAND_SAMP_MODE	MMCRA[63] SAMPLE_ENABLE	Eligibility Criteria
000	00	1	All instructions are eligible.
001	00	1	All Load/Store instructions are eligible.
010	00	1	All Probe NOPs are eligible.
011	00	1	LARX/STCX instructions are eligible.
101	00	1	Only load instructions are eligible.
110	00	1	Long latency operation (div/sqrt/mul)
111	00	1	Only store instructions are eligible.

A marked instruction can generate a number of marked events which are configurable to be counted on any of PMC1 - 4. This can be achieved as follows:

- Configure MMCRA to pick random instruction sampling with eligibility criteria specified.
- Configure a relevant marked event in one of the PMCs and load an appropriate near-overflow value into that PMC; for example ( $2^{31} - 1$ ).
- Upon PMC counter overflow, an interrupt occurs. The SIAR, SDAR, and SIER contain information about the marked instruction which caused an event that caused the counter to overflow.

#### E.3.4.1 Probe NOP

The architecture provides two special no-operation (NOP) instructions, called Probe NOPs, defined as **AND 0,0,0** and **AND 1,1,1**. This form of AND is reserved exclusively for performance monitor use. Software can insert Probe NOP instructions at various points in the program and configure the performance monitor sampling facility to mark or sample only Probe NOPs. The sampled Probe NOPs can then be configured to count in a PMC and, upon overflow, cause a synchronous PMU interrupt. This provides the capability to freeze the state of the thread after the completion of a Probe NOP, so that interrupt handlers can examine general purpose registers, SPRs, memory locations to get stack spills global variables etc.

Below is an example on specifying a Probe NOP

Instrumentation point could sample the loop upper bounds values (NX,NY) and a value VAL.

```
for (i=0;i < NX;i++) {
    for(j =0;j < NY;j++) {
        <expression VAL>
        probeNop; // to read NX,NY,VAL
    }
}
```

### E.3.5 Random Event Sampling (RES)

While RIS is effective at providing coverage of a wide variety of interesting events for a single instruction, RES can provide higher sample rates for specific events that might have a higher performance cost.

For example, when loads that miss the L1 cache are of interest, RIS can be set up to mark only loads. Typically, 95 - 98% of all loads hit the L1 cache, so the sampling rate for load misses might still be low. To support dynamic optimizers, which require much higher sampling rates to make optimization decisions, RES allows the random marking of only load misses (filtering out all the L1 hits). The SIAR is updated at completion time with the exact instruction address of the marked instruction that suffered the L1 miss, and the SDAR is updated with the data effective address of marked instruction.

As shown in *Table E-18*, random sampling is enabled by MMCRA[63] = '1'. RES for the load unit is selected by MMCRA[61:62] = '01'. RES for the branch unit is selected by MMCRA[61:62] = '10'. Additional criteria are selected in MMCRA[57:59].

*Table E-18. Random Event Sampling Modes*

MMCRA[57:59] RAND_SAMP_ELIG	MMCRA[61:62] RAND_SAMP_MODE	MMCRA[63] SAMP_ENABLE	Eligibility Criteria
000	01	1	Load Misses
100	01	1	Load Hit Store (LHS)
000	10	1	Branch Mispredicts
001	10	1	Branch Mispredicts (Direction, CR or CTR)
010	10	1	Branch Mispredicts (TA)
011	10	1	Branches Resolved Taken
100	10	1	Non-repeating Branches
101	10	1	All Branches that require prediction



## E.4 Thresholding

The Power10 processor has a threshold counter facility, which is a distinct floating-point counter facility separate from the primary PMCs. The threshold counter facility can count the number of events occurring between a separate set of designated start/end events from the core, cache, and memory subsystem. As an example, application developers can sample on-cache misses and identify code where cache misses are most frequent. While this is useful, some situations require the ability to profile on-cache misses that can take more than an expected number of cycles to resolve. Sampling (profiling) is a performance-analysis technique described in *Appendix E.3 Power10 Sampling Support* on page 483. Thresholding in this case can be used to identify marked instructions that take more than an expected range of cycles between a threshold start event [marked cache miss] and a threshold end event [marked cache reload]. In this way, software can specify a threshold of such delay and focus only on cache misses that exceed this value. The same technique can be used to threshold on pipeline stages such as decode, dispatch, issue, finish, completion, and also for stores to flag those with high delays in the memory subsystem for completing the store operation.

The Power10 processor has expanded on this capability in these ways:

- Ability to count number of events between a start and end event.
- Ability to specify a threshold to compare against the count of events between a start and end event.
- Ability to specify any event programmable in PMC1 - PMC4 apart from cycles to count between start and end events.

Hardware supports the following capabilities:

- 11-bit software-accessible floating-point counter with mantissa and exponent
- 11-bit software-accessible floating-point threshold with mantissa and exponent
- 4-bit software-accessible threshold start event
- 4-bit software-accessible threshold end event
- 3-bit software-accessible bits to specify events to threshold on
- 10-bit hidden auto pre-scaler

### E.4.1 Floating-Point Counter

The Power10 processor implements a threshold floating-point counter that can count the number of events elapsed between a threshold start event and a threshold end event. Hardware provides an 8-bit mantissa and a 3-bit exponent for both a counter and a compare facility.

The floating-point counter starts counting from zero, beginning after an occurrence of the specified threshold start event and ending at the occurrence of the specified threshold end event, both of which are set in the MMCRA. At the start, the auto-prescalar is set at divide-by-1 of the event specified by the event selector. Each time the 8-bit counter mantissa overflows, the exponent is incremented by 1, and the counter mantissa continues counting from the new value '01000000'. This ensures that a subsequent read of the counter (exponent and mantissa) is a "base 4" floating-point number. Each time the exponent increases, a new divide factor is set in the auto prescalar according to *Table E-19* on page 488.

To get a fixed-point value from the counter, read the counter mantissa and shift it left two bits for the number of times specified by the counter exponent.



To write the threshold in MMCRA, take an 18-bit number (N), shift it right 2 bit positions, and increment the exponent (E) by 1 until the upper 10 bits of N are zero. Write E to the threshold exponent and write the lower 8 bits of N to the threshold mantissa. Note that it is invalid to write a mantissa with the upper two bits of mantissa being zero, unless the exponent is also zero. The maximum threshold that can be written is 261,120.

*Table E-19. Floating-Point Counter Values*

Exponent	Prescaler Divide-by	Minimum Count Value	Maximum Count Value
0	1	0	255
1	4	256	1020
2	16	1024	4080
3	64	4096	16320
4	256	16320	65280
5	1024	65536	261120

#### E.4.2 Thresholding Operation

Software programs the MMCRA to specify the following:

1. Enable sampling, with any eligibility required, using MMCRA[57:63]. These bits enable the sampling state machine, which is a prerequisite for thresholding.
2. Enable a threshold start event (MMCRA[48:51]) and Threshold Stop event (MMCRA[52:55]).
3. If required, software can also specify a threshold value to compare against, as specified in the MMCRA.
4. Enable thresholding by specifying an event to threshold in (MMCRA[45:47]). The event can be cycles during which the run latch is set, instructions completed while the run latch is set, or any event programmable in PMC1 - PMC4 or as specified in the MMCRA (dependent on all freeze conditions including the run latch).

When a threshold start event is encountered, the logic starts incrementing the threshold counter from zero. The counter stops counting if either a threshold end condition occurs or if the counter hits the maximum value. Subsequent start conditions reset the threshold counter and the threshold event counting resumes. If the threshold value specified in the compare fields of the MMCRA register matches the actual threshold counter value, the SIER bit for threshold exceeded is set. Software can also specify an event called Threshold Exceeded to count the number of occurrences.

As an example, if software intends to capture the latency of a cache miss, it can select the start event to be Marked Cache Miss and the stop event to be Marked Cache Reload. Software can setup a PMC to count Marked Cache Reloads, and the interrupt handler can capture the instruction and data effective address from SIAR/SDAR and look at MMCRA to capture the cycles between miss and reload.

The software can also select events called threshold\_exceeded and threshold\_not\_exceeded to profile on for the cases of interest.

A threshold end event takes precedence over a threshold start event. For example, if the threshold state machine is idle and both a start and end event occur on the same cycle, then the threshold state machine will ignore the start event while recognizing the end event, and remain in idle state. If running and both a start and end event occur on the same cycle, the threshold state machine goes to idle state. Note that whenever a

---

threshold start event occurs (with no end event), the threshold counter gets reset to zero. Because a threshold start is restarting another threshold event counting period, a threshold met and threshold exceeded events will not occur when threshold start is active.

All of the start/end threshold events are provided directly to the performance monitor, which means that the thresholding facility is available regardless of the configuration of the event bus. The events, which can be used for threshold start/end measurement, occur for a marked instruction moving through the pipeline in this order.

**For All Instructions:**

- Marked instruction decoded
- Marked instruction dispatched
- Marked instruction issued
- Marked instruction finished
- Marked instruction next to finish
- Marked instruction complete

**For Loads:**

- Marked instruction decoded
- Marked instruction dispatched
- Marked instruction issued
- Marked load miss L1
- L2 RC machine dispatched for marked instruction
- L2 RC machine done for marked instruction
- Marked load reload
- Marked instruction finished
- Marked instruction next to finish
- Marked instruction complete

**For Stores:**

- Marked instruction decoded
- Marked instruction dispatched
- Marked instruction issued
- Marked instruction finished
- Marked instruction next to finish
- Marked instruction complete
- L2 RC machine dispatched for marked instruction
- L2 RC machine done for marked instruction



The PMU also provides the following events to be used along with the thresholding facility:

- **PM\_THRESH\_MET**

This event is active in the cycle subsequent to the threshold counter becoming equal to or greater than the threshold compare value (MMCRA[THRESH\_CMP\_EXP] and MMCRA[THRESH\_CMP\_MANTISSA]).

- **PM\_THRESH\_NOT\_MET**

A threshold end event occurs, but the threshold compare value is not met. This is useful for cases like instructions completing between a load miss and a cache reload. (Lower values are worse than higher values.)

- **Threshold Exceeded Events**

The threshold exceeded events are active for one cycle as the threshold counter crosses from less than to equal or greater than the compare value. These events provide coarse grain thresholding capability on multiple ranges without needing to specify a high precision compare value. Up to four events (PMC1 - PMC4) can be specified. These include the following events:

- PM\_THRESH\_CTR\_EXC\_32
- PM\_THRESH\_CTR\_EXC\_64
- PM\_THRESH\_CTR\_EXC\_128
- PM\_THRESH\_CTR\_EXC\_256
- PM\_THRESH\_CTR\_EXC\_512
- PM\_THRESH\_CTR\_EXC\_1024
- PM\_THRESH\_CTR\_EXC\_2048
- PM\_THRESH\_CTR\_EXC\_4096

### E.4.3 Examples of the Ability to Change Events to Threshold

Any event configurable in PMC1 - PMC4 can be used to count between a start and stop event in Power10 thresholding. A few examples are shown below as to how users can configure different events with different start and stop pairs to measure different metrics

#### E.4.3.1 Loads

In this use case, the threshold start event is a marked load L1 cache miss and the threshold end event is a marked L1 cache reload.

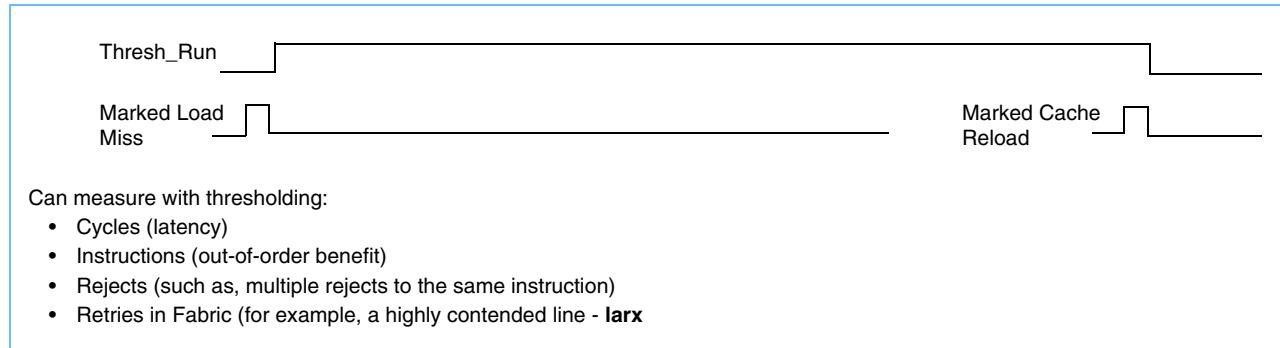
By selecting to count cycles between these two events, the number of cycles elapsed between a miss and reload will be counted, which yields the load L1 miss latency. Users can also count instructions completed by this thread while this miss was pending, which indicates an idea of out-of-order benefits. If, between the start and stop event, the thread was able to complete many instructions, it is an indication that the pipeline was able to hide the latency of the cache miss. If no instructions were completed, then the pipeline was waiting on the cache miss to finish and was unable to hide the latency.

There are other useful events to analyze with threshold counting. Examples include identifying and quantifying load L1 cache misses that further suffer from multiple resource misses or rejections. The start event can be changed to marked instruction issued and the stop event can be marked cache reload to count other types of rejects, such as issue rejects, ERAT misses, LMQ rejects, store-hit-load collisions, and so on.

The Power10 processor also has expanded the capability to tie fabric-level events to thread-level data. Users can configure to count fabric retries and threshold on those that take too long, this also shows up as an added latency.

A marked load miss is the threshold start event and a cache reload is the stop event. *Figure E-1* shows how to measure cycles between these events. The cycles can be from the cache latency, rejects, and so on.

*Figure E-1. Marked Load Events*



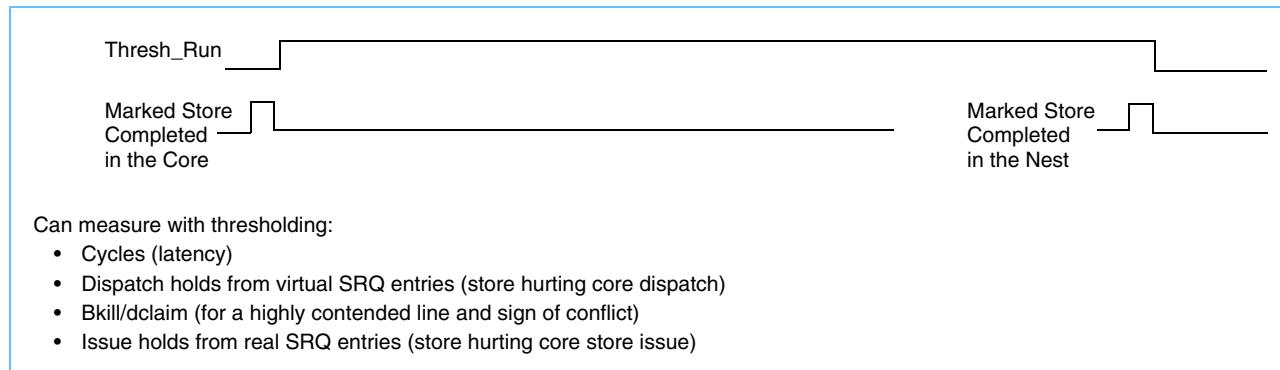
#### E.4.3.2 Stores

Characterizing long-latency stores is difficult, because, from a core-stall perspective, stores complete in a disconnected fashion from the instruction execution in the core, and the core can progress towards completing younger instructions. However, if there are many stores that take too long to complete in the L2 cache, this can cause back pressure and the core can run out of virtual and real SRQ entries.

The Power10 thresholding facility provides the capability to identify stores that take too long to complete in the nest. When combined with events that signify that the core had store-related stalls while this store was pending, this helps identify stores that cause stalls from a dispatch and issue perspective in the core. This is important because stores can take hundreds of cycles in the L2, but might not impact performance.

A marked store (completed in the core) is the threshold start event and the marked store (completed in the Nest) is the stop event. *Figure E-2* shows how to measure cycles between these events.

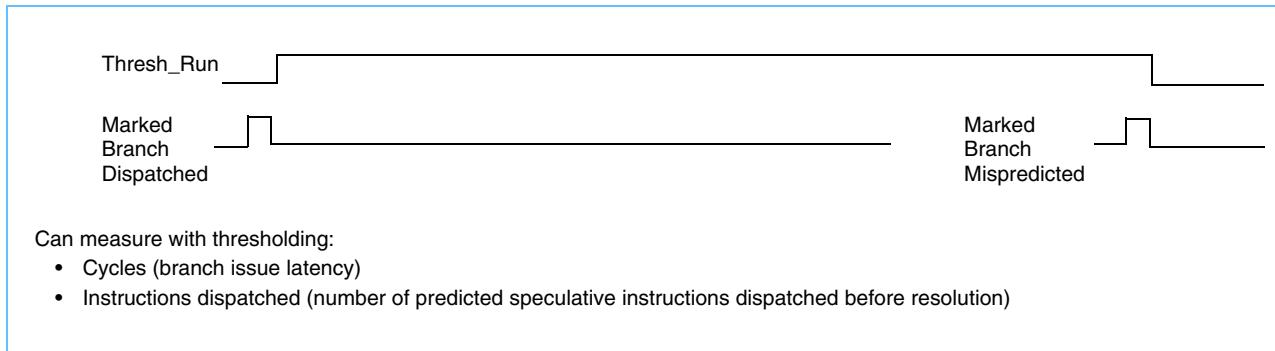
*Figure E-2. Marked Store Events*



#### E.4.3.3 Branches

A marked branch dispatched is the threshold start event and a marked branch mispredicted is the stop event. *Figure 3-4* shows how to measure cycles between these events.

*Figure E-3. Branches*



#### 19.2.1 Threshold Event Selection

See *Table E-6 Threshold Event Selection (MMCRA[45:47])* on page 470 for a description of the threshold event selection field when used to select an event specified for threshold counting.

#### 19.2.2 Threshold Start/Stop Event Selection

See *Table E-7 Threshold Start/Stop Events Selection (MMCRA[48:51] or [52:55])* on page 470 for a description of the threshold start and threshold stop event conditions.



## E.5 Core PMU Events

The Power10 core supports in excess of 1000 events that collect and filter information from across the core. These events provide insight into how the program instructions flow in the Power10 pipeline and assist in understanding any performance bottlenecks seen in the core.

The following sections provide a list of all the PMU events supported on the Power10 chip for each category and briefly describes which performance bottlenecks are characterized by the events. The event tables show the PMC that counts the event, an event code that uniquely identifies an event, the event name, and the event description. The event code value is used by software tools (for example, *PERF*) to program the various MMCR registers to read the events on the PMCs.

The events lists are organized as follows:

- *Appendix E.5.1 IFU Events* on page 494
- *Appendix E.5.2 Branch Events* on page 496
- *Appendix E.5.3 ISU Events* on page 499
- *Appendix E.5.4 VSU Events* on page 503
- *Appendix E.5.5 LSU Events* on page 504
- *Appendix E.5.6 L2 Events* on page 511
- *Appendix E.5.7 L3 Events* on page 518
- *Appendix E.5.8 CPI Stack Events* on page 525
- *Appendix E.5.9 Marked Events* on page 531
- *Appendix E.5.10 MMU Events* on page 536
- *Appendix E.5.11 PMC Events* on page 542
- *Appendix E.5.12 Reload Source Events* on page 545
- *Appendix E.5.13 Radix Events* on page 548
- *Appendix E.5.14 Metrics* on page 558



### E.5.1 IFU Events

Table E-20 lists the Power10 events related to the instruction cache and other fetch-related (IFU) events.

Table E-20. IFU Events (Sheet 1 of 2)

PMC	Event Code	Event Name	Description
PMC1	10018	PM_IC_DEMAND_CYC	Cycles in which an instruction reload is pending to satisfy a demand miss.
PMC1	1005A	PM_FLUSH_MPRED	A flush occurred due to a mispredicted branch. Includes target and direction.
PMC2	2F04C	PM_ICBI_FIN	An <b>icbi</b> instruction finished.
PMC2	200FD	PM_L1_ICACHE_MISS	Demand instruction cache miss.
PMC3	30068	PM_L1_ICACHE_RELOADED_PREF	Counts all instruction cache prefetch reloads (includes demand turned into prefetch).
PMC4	40012	PM_L1_ICACHE_RELOADED_ALL	Counts all instruction cache reloads (includes demand, prefetch, prefetch turned into demand and demand turned into prefetch).
PMC4	4D02E	PM_NO_FETCH_CYC	Cycles in which no instructions were fetched by the instruction fetch unit (IFU), for any reason.
PMC4	45058	PM_IC_MISS_CMPL	Non-speculative instruction cache miss, counted at completion.
Any PMC	0000004080	PM_INST_FROM_L1	An instruction fetch hit in the L1 cache. Each fetch group contains eight instructions. The same line can hit four times if 32 sequential instructions are fetched.
Any PMC	0000004880	PM_NO_FETCH_BANK_CONFLICT_CYC	Cycles in which no instructions are fetched because of an interleave conflict. The ifar logic will detect an interleave conflict and kill the data that was read that cycle.
Any PMC	0000004084	PM_NO_FETCH_EAT_FULL_CYC	Cycles in which no instructions are fetched because there is no room in EAT.
Any PMC	0000004884	PM_NO_FETCH_IBUF_FULL_CYC	Cycles in which no instructions are fetched because there is no room in the instruction buffers.
Any PMC	0000004088	PM_NO_FETCH_THROTTLE_CYC	Cycles in which no instructions are fetched for this thread because it is being throttled.
Any PMC	0000004888	PM_FETCH_CYC	Cycles in which instructions are successfully fetched from the instruction cache.
Any PMC	000000408C	PM_NO_FETCH_THROTTLE_POWMAN_CYC	Cycles in which the thread is throttled because of power management.
Any PMC	000000488C	PM_NO_FETCH_THROTTLE_OTHER_CYC	Cycles in which the thread is throttled for a reason other than power management, relative priority, and dynamic priority. This includes throttles for speculation for power or performance, independently decided by the IFAR.
Any PMC	0000004090	PM_NO_FETCH_THROTTLE_REL_PRIO_CYC	Cycles in which the thread is throttled because of relative priority. This is thread arbitration based on the Relative Priority Register (RPR).
Any PMC	0000004890	PM_NO_FETCH_THROTTLE_DYN_PRIO_CYC	Cycles in which the thread is throttled because of dynamic priority. This is the generic thread arbitration without relative thread priority. A thread can also be throttled if an instruction cache reload for a different thread occurs in the same cycle.
Any PMC	0000004894	PM_DECODE_THROTTLE_IIF_CYC	Cycles in which decode was held for a thread due to an instruction-in-flight throttle (WOF). This event will increment regardless of other decode holds being present.



Table E-20. IFU Events (Sheet 2 of 2)

PMC	Event Code	Event Name	Description
Any PMC	0000004098	PM_DECODE_HOLD_NO_ITAGS	Cycles in which the IFU was not able to decode and transmit one or more instructions because all itags were in use. This means that the ICT is full for this thread. This event will only increment when no other hold is present.
Any PMC	0000004898	PM_DECODE_THROTTLE_IPC_CYC	Cycles in which decode is throttled because M over N throttle is active (WOF). This event will increment regardless of other decode holds being present.
Any PMC	000000409C	PM_IC_INVALIDATE	A line in the instruction cache has been invalidated by an ICBI.
Any PMC	000000489C	PM_IC_RELOAD_PRIVATE	An instruction cache line was brought in private for a specific thread. Most lines are brought in shared for all four threads.
Any PMC	00000040A0	PM_IC_PREF_REQ	Instruction prefetch requests.
Any PMC	00000040B0	PM_BCQ_FULL_CYC	Cycles in which all 12 entries of the BCQ are full.
Any PMC	00000050A8	PM_Ieadir_Hit_IDIR_MISS	A fetch hit in the IEADIR but missed in the IDIR. This is usually due to IEADIR aliasing, but could be due to an IDIR invalidate that did not invalidate the corresponding IEADIR entry.
Any PMC	00000058A8	PM_ICACHE_MISS_DUE_TO_CTXTTAG	A fetch missed the instruction cache due to a context tag miscompare. All other fields matched. This is similar to PM_IEA_TRACKING_TABLE_WRITE but is not restricted to context tags restricted to IEA sharing and does not require IEA sharing to be enabled.
Any PMC	000000E088	PM_IEA_ALIAS_TABLE_WRITE	IFU EA sharing was detected and the sharing information was written to the IFU Alias Table.
Any PMC	000000E888	PM_IEA_ALIAS_TABLE_HIT	Instruction cache reload hit on the Alias Table and was written to the I-cache as an IEA Shared entry.
Any PMC	000000E08C	PM_IEA_TRACKING_TABLE_WRITE	Instruction cache miss occurred where I-cache EA sharing might be possible. That is, all IDIR fields matched except the context tag and the incumbent context tag is compatible with IEA sharing.
Any PMC	000000E88C	PM_IEA_ICACHE_SHARED_HIT	Instruction cache hit occurred on an entry that is marked as IEA sharing. This does not include I-cache reload bypasses.
Any PMC	00000048A0	PM_FUSED_BACK_TO_BACK	Back-to-back two-cycle execution. Counted at decode time.
Any PMC	00000040A4	PM_FUSED_DESTRUCTIVE	The first and second instruction have the same target register. Counted at decode time.
Any PMC	00000048A4	PM_FUSED_RESULT	Two fused PowerPC (PPC set) instructions: the first PPC instruction, in age order, is the result of the second PPC instruction. Counted at decode time.
Any PMC	00000040A8	PM_FUSED_TOGETHER	Produce two results together. Example: instruction having a fusible compare: fabs f3,f1; fcmpl cr5,f2,f3. Counted at decode time.
Any PMC	00000048A8	PM_FUSED_LOADCOMPARE	Fusion of a load and a compare immediate referencing the load's data. Example: ldx r3,r1,r2; cmpi cr4,r3,immediate. Counted at decode time.
Any PMC	00000040AC	PM_FUSED_LOAD_LOAD	Fusion of two displacement loads (meeting the requirements for load-load fusion).



## E.5.2 Branch Events

*Table E-21* lists the branch events. Branch predictions and mispredictions from different predictors are provided, such as the local/global count cache, link stack, TAGE, TOP, local/global BHT. Also provided are events that count taken versus not-taken branches, and conditional versus unconditional.

*Table E-21. Branch Events* (Sheet 1 of 3)

PMC	Event Code	Event Name	Description
PMC1	10068	PM_BR_FIN	A branch instruction finished. Includes predicted, mispredicted, and unconditional.
PMC2	2F04A	PM_BR_FIN	A branch instruction finished. Includes predicted, mispredicted, and unconditional.
PMC2	200FA	PM_BR_TAKEN_CMPL	Branch taken instruction completed.
PMC4	4D05E	PM_BR_CMPL	A branch completed. All branches are included.
PMC4	4E058	PM_BR_COND_CMPL	A conditional branch completed.
PMC4	400F6	PM_BR_MPRED_CMPL	A mispredicted branch completed. Includes direction and target.
Any PMC	00000048B0	PM_BR_FIN_FROM_BCQ	Branches that were issued with sources not ready (CR = 0), counted at finish time. Branches go to the BCQ when their sources are not ready.
Any PMC	00000040B4	PM_BR_TKN_FIN	A taken branch (conditional or unconditional) finished.
Any PMC	00000048B4	PM_BR_TKN_UNCOND_FIN	An unconditional branch finished. All unconditional branches are taken.
Any PMC	00000040B8	PM_PRED_BR_TKN_COND_DIR	A conditional branch finished with correctly predicted direction. Resolved taken.
Any PMC	00000048B8	PM_PRED_BR_NTKN_COND_DIR	A conditional branch finished with correctly predicted direction. Resolved not taken.
Any PMC	00000040BC	PM_MPRED_BR_TKN_COND_DIR	A conditional branch finished with mispredicted direction. Resolved taken.
Any PMC	00000048BC	PM_MPRED_BR_NTKN_COND_DIR	A conditional branch finished with mispredicted direction. Resolved not taken.
Any PMC	0000005080	PM_PRED_BR_TKN_COND_TGT_DIR	A conditional branch finished with correctly predicted target and direction. Resolved taken.
Any PMC	0000005880	PM_PRED_BR_NTKN_COND_TGT_DIR	A conditional branch finished with correctly predicted target and direction. Resolved not taken.
Any PMC	0000005884	PM_MPRED_BR_NTKN_COND_TGT_DIR	A conditional branch finished with mispredicted target and direction. Resolved not taken.
Any PMC	0000005088	PM_BR_PRED_TKN_COND_DIR_LBHT_LSEL	A conditional branch finished with correctly predicted direction using the local branch history table selected by the local selector. Resolved taken.
Any PMC	0000005888	PM_BR_PRED_TKN_COND_DIR_LBHT_GSEL	A conditional branch finished with correctly predicted direction using the local branch history table selected with the global selector. Resolved taken.
Any PMC	000000508C	PM_BR_PRED_TKN_COND_DIR_GBHT	A conditional branch finished with correctly predicted direction using the global branch history table. Resolved taken.
Any PMC	000000588C	PM_BR_PRED_TKN_COND_DIR_TAGE	A conditional branch finished with correctly predicted direction using a TAGE override. Resolved taken.
Any PMC	0000005090	PM_BR_PRED_TKN_COND_DIR_TOP	A conditional branch finished with correctly predicted direction using a TOP override to the BHT. Resolved taken.
Any PMC	0000005890	PM_BR_PRED_TKN_COND_TGT_COUNT_LCC	A conditional branch finished with correctly predicted target using the local count cache. Resolved taken.



Table E-21. Branch Events (Sheet 2 of 3)

PMC	Event Code	Event Name	Description
Any PMC	0000005094	PM_BR_PRED_TKN_COND_TGT_COUNT_GCC	A conditional branch finished with correctly predicted target using the global count cache. Resolved taken.
Any PMC	0000005894	PM_BR_PRED_TKN_COND_TGT_COUNT_TIP	A conditional branch finished with correctly predicted target using the count TIP override to the count cache. Resolved taken.
Any PMC	0000005098	PM_BR_PRED_TKN_TGT_LINK	A conditional or unconditional branch finished with correctly predicted target using the Link register ( <code>bclr[I]</code> ) and predicted by Link Stack [BH(1) = 0]. Resolved taken.
Any PMC	0000005898	PM_BR_PRED_NTKN_COND_DIR_LBHT_LSEL	A conditional branch finished with correctly predicted direction using the local branch history table selected by the local selector. Resolved not taken.
Any PMC	000000509C	PM_BR_PRED_NTKN_COND_DIR_LBHT_GSEL	A conditional branch finished with correctly predicted direction using the local branch history table selected with the global selector. Resolved not taken.
Any PMC	000000589C	PM_BR_PRED_NTKN_COND_DIR_GBHT	A conditional branch finished with correctly predicted direction using the global branch history table. Resolved not taken.
Any PMC	00000050A0	PM_BR_PRED_NTKN_COND_DIR_TAGE	A conditional branch finished with correctly predicted direction using a TAGE override. Resolved not taken.
Any PMC	00000058A0	PM_BR_PRED_NTKN_COND_DIR_TOP	A conditional branch finished with correctly predicted direction using a TOP override to the BHT. Resolved not taken.
Any PMC	00000050AC	PM_BR_MPRED_TKN_COND_DIR_LBHT_LSEL	A conditional branch finished with mispredicted direction using the local branch history table selected by the local selector. Resolved taken.
Any PMC	00000058AC	PM_BR_MPRED_TKN_COND_DIR_LBHT_GSEL	A conditional branch finished with mispredicted direction using the local branch history table selected with the global selector. Resolved taken.
Any PMC	00000050B0	PM_BR_MPRED_TKN_COND_DIR_GBHT	A conditional branch finished with mispredicted direction using the global branch history table. Resolved taken.
Any PMC	00000058B0	PM_BR_MPRED_TKN_COND_DIR_TAGE	A conditional branch finished with mispredicted direction using a TAGE override. Resolved taken
Any PMC	00000050B4	PM_BR_MPRED_TKN_COND_DIR_TOP	A conditional branch finished with mispredicted direction using a TOP override to the BHT. Resolved taken.
Any PMC	00000058B4	PM_BR_MPRED_TKN_COND_TGT_COUNT_LCC	A conditional branch finished with mispredicted target using the local count cache. Resolved taken.
Any PMC	00000050B8	PM_BR_MPRED_TKN_COND_TGT_COUNT_GCC	A conditional branch finished with mispredicted target using the global count cache. Resolved taken.
Any PMC	00000058B8	PM_BR_MPRED_TKN_COND_TGT_COUNT_TIP	A conditional branch finished with mispredicted target using the count TIP override to the count cache. Resolved taken.
Any PMC	00000050BC	PM_BR_MPRED_TKN_TGT_LINK	A conditional or unconditional branch finished with mispredicted target using the Link register. Resolved taken.
Any PMC	00000058BC	PM_BR_MPRED_NTKN_COND_DIR_LBHT_LSEL	A conditional branch finished with mispredicted direction using the local branch history table selected by the local selector. Resolved not taken.
Any PMC	000000E080	PM_BR_MPRED_NTKN_COND_DIR_LBHT_GSEL	A conditional branch finished with mispredicted direction using the local branch history table selected with the global selector. Resolved not taken.
Any PMC	000000E880	PM_BR_MPRED_NTKN_COND_DIR_GBHT	A conditional branch finished with mispredicted direction using the global branch history table. Resolved not taken.
Any PMC	000000E084	PM_BR_MPRED_NTKN_COND_DIR_TAGE	A conditional branch finished with mispredicted direction using a TAGE override. Resolved not taken.
Any PMC	000000E884	PM_BR_MPRED_NTKN_COND_DIR_TOP	A conditional branch finished with mispredicted direction using a TOP override to the BHT. Resolved not taken.



*Table E-21. Branch Events (Sheet 3 of 3)*

PMC	Event Code	Event Name	Description
Any PMC	000000E890	PM_BR_BTAC_INV_TGT	BTAC predicts a target that is different from what the BHT and count cache predict. The BTAC entry is invalidated. Reported at fetch time.
Any PMC	000000E094	PM_BR_BTAC_INV_DIR	BTAC predicts taken for a branch that the BHT predicts not taken, so that BTAC entry is invalidated. Reported at fetch time.
Any PMC	000000E894	PM_BR_PRED_COND_BTAC	A conditional branch finished with correctly predicted target or direction using the BTAC. Reported at fetch time.
Any PMC	000000E098	PM_MPRED_BR_FIN	A conditional branch mispredicted its direction or target address. Counted at finish time.
Any PMC	000000E898	PM_BR_PRED_TKN_SWHINT	A software hinted branch finished and the branch resolved taken and the hint was correct.
Any PMC	000000E09C	PM_BR_PRED_NTKN_SWHINT	A software hinted branch finished and the branch resolved not taken and the hint was correct.
Any PMC	000000E89C	PM_BR_MPRED_TKN_SWHINT	A software hinted branch finished and the branch resolved taken and the hint was incorrect.
Any PMC	000000E0A0	PM_BR_MPRED_NTKN_SWHINT	A software hinted branch finished and the branch resolved not taken and the hint was incorrect.
Any PMC	000000E0A4	PM_BACK_BRANCH	Branch whose target address is lower than the program counter.

### E.5.3 ISU Events

Table E-22 lists the Power10 events related to instruction dispatch and completion.

Table E-22. ISU Events (Sheet 1 of 4)

PMC	Event Code	Event Name	Description
PMC1	10028	PM_NTC_FLUSH	The instruction was flushed after becoming next-to-complete (NTC).
PMC1	1F056	PM_DISP_SS0_2_INSTR_CYC	Cycles in which Superslice 0 dispatches either 1 or 2 instructions.
PMC1	1F058	PM_DISP_HELD_CYC	Cycles dispatch is held.
PMC1	1F05A	PM_DISP_HELD_SYNC_CYC	Cycles dispatch is held because of a synchronizing instruction that requires the ICT to be empty before dispatch.
PMC1	10066	PM_ADJUNCT_CYC	Cycles in which the thread is in an Adjunct state. MSR[S, HV, PR] bits = '011'.
PMC1	100F2	PM_1PLUS_PPC_CMPL	Cycles in which at least one instruction is completed by this thread.
PMC1	1000E	PM_MMA_ISSUED	MMA instruction issued.
PMC2	2E010	PM_ADJUNCT_INST_CMPL	PowerPC instruction completed while the thread was in an Adjunct state.
PMC2	24050	PM_IOPS_DISP	Internal operations dispatched. PM_IOPS_DISP or PM_INST_DISP will show the average number of internal operations per PowerPC instruction.
PMC2	2405A	PM_NTC_FIN	Cycles in which the oldest instruction in the pipeline (NTC) finishes. Note that instructions can finish out-of-order; therefore, not all the instructions that finish have a next-to-complete status.
PMC2	2405E	PM_ISSUE_CANCEL	An instruction issued and the issue was later cancelled. Only one cancel per PowerPC instruction.
PMC2	2F054	PM_DISP_SS1_2_INSTR_CYC	Cycles in which Superslice 1 dispatches either 1 or 2 instructions.
PMC2	2F056	PM_DISP_SS1_4_INSTR_CYC	Cycles in which Superslice 1 dispatches either 3 or 4 instructions.
PMC2	20066	PM_DISP_HELD_OTHER_CYC	Cycles dispatch is held for any other reason.
PMC2	20068	PM_DISP_HELD_HALT_CYC	Cycles dispatch is held because of power management.
PMC2	2006A	PM_DISP_HELD_STF_MAPPER_CYC	Cycles dispatch is held because the STF mapper/SRB was full. Includes GPR (count, link, tar), VSR, VMR, FPR.
PMC2	200F2	PM_INST_DISP	PowerPC instruction dispatched.
PMC2	2C012	PM_MMA_AVAIL_NACTIVE	MMA is powered up and available, but has no active MMA instructions to execute.
PMC3	30002	PM_INST_CMPL	PowerPC instruction completed.
PMC3	30012	PM_FLUSH_COMPLETION	The instruction that was next to complete (oldest in the pipeline) did not complete because it suffered a flush.
PMC3	3F04E	PM_PROBLEM_CYC	Cycles in which the MSR register shows the thread in problem state.
PMC3	3405A	PM_PRIVILEGED_INST_CMPL	PowerPC instruction completed while the thread was in privileged state.
PMC3	3F054	PM_DISP_SS0_4_INSTR_CYC	Cycles in which Superslice 0 dispatches either 3 or 4 instructions
PMC3	3F056	PM_DISP_SS0_8_INSTR_CYC	Cycles in which Superslice 0 dispatches either 5, 6, 7, or 8 instructions.



Table E-22. ISU Events (Sheet 2 of 4)

PMC	Event Code	Event Name	Description
PMC3	3F05E	PM_DISP_HELD_ISSQ_FULL_CYC	Cycles dispatch is held due to issue queue full. Includes issue queue and branch queue.
PMC3	30060	PM_DISP_HELD_XVFC_MAPPER_CYC	Cycles dispatch is held because the XVFC mapper/SRB was full.
PMC3	30064	PM_NTF_ISSUE_HOLD_CYC	Cycles in which the oldest instruction in the pipeline is being held at issue.
PMC3	300F2	PM_INST_DISP	PowerPC instruction dispatched.
PMC3	300F4	PM_RUN_INST_CMPL_CONC	PowerPC instruction completed by this thread when all threads in the core had the run-latch set.
PMC3	30006	PM_MMA_ACTIVE_NAVAIL	MMA instructions are being held and not executed as the MMA is not powered up and available.
PMC4	40006	PM_ISSUE_CANCEL	Cycles in which an instruction or group of instructions were cancelled after being issued. This event increments once per occurrence, regardless of how many instructions are included in the issue group.
PMC4	40008	PM_NTC_ALL_FIN	Cycles in which both instructions in the ICT entry pair show as finished. These are the cycles between finish and completion for the oldest pair of instructions in the pipeline.
PMC4	4F056	PM_DISP_SS1_8_INSTR_CYC	Cycles in which Superslice 0 dispatches either 5, 6, 7, or 8 instructions.
PMC4	40060	PM_DISP_HELD_SCOREBOARD_CYC	Cycles dispatch is held while waiting on the scoreboard. This event combines VSCR and FPSCR together.
PMC4	40062	PM_DISP_HELD_RENAME_CYC	Cycles dispatch is held because the mapper/SRB was full. Includes GPR (count, link, tar), VSR, VMR, FPR, and XVFC.
PMC4	400F2	PM_1PLUS_PPC_DISP	Cycles at least one instruction dispatched.
PMC4	400F8	PM_FLUSH	Flush (any type).
PMC4	4E040	PM_SP_MMA_CMPL	Single-precision MMA instruction completed.
PMC4	4E042	PM_DP_MMA_CMPL	Double-precision MMA instruction completed.
PMC4	4E044	PM_HP_MMA_CMPL	FP16 half-precision MMA instruction completed.
PMC4	4E046	PM_4INT_MMA_CMPL	Int4 mixed-precision MMA instruction completed.
PMC4	4E048	PM_8INT_MMA_CMPL	Int8 mixed-precision MMA instruction completed.
PMC4	4E04A	PM_16INT_MMA_CMPL	Int16 mixed-precision MMA instruction completed.
PMC4	4E04C	PM_BF16_MMA_CMPL	Bfloat16 half-precision MMA instruction completed.
PMC4	4E04E	PM_MOVE_TO_MMA_CMPL	VSR-to-accumulator move MMA instruction completed ( <b>xxmtacc</b> and <b>xxsetaccz</b> ). This event implies priming instructions.
PMC4	4F040	PM_MOVE_FROM_MMA_CMPL	Accumulator-to-VSR move MMA instruction completed ( <b>xxmfacc</b> ). This event implies de-priming instructions.
PMC4	4F042	PM_MUL_MMA_CMPL	All multiply-only MMA instruction completed. This event might imply priming instructions.
PMC4	44058	PM_LWSYNC_CMPL	A lightweight synchronizing or barrier instruction (LWSYNC) completed.
PMC4	4405A	PM_HWSYNC_CMPL	A heavyweight synchronizing or barrier instruction (HWSYNC) completed.
Any PMC	00000050A4	PM_SHL_CREATED	A store-hit-load table entry was created as a result of an SHL flush.
Any PMC	00000058A4	PM_SHL_ST_DEPENDENCY	A fetched instruction hit in the store-hit-load table.

Table E-22. ISU Events (Sheet 3 of 4)

PMC	Event Code	Event Name	Description
Any PMC	0000002080	PM_EE_OFF_EXT_INT_CYC	Cycles in which MSR[EE] is off and external interrupts are active.
Any PMC	0000002880	PM_ISU_FLUSH	All flushes initiated by the instruction sequencing unit (ISU). Excludes LSU NTC+1 flushes.
Any PMC	0000002084	PM_ISU_FLUSH_DISP	Dispatch flushes occur when one thread is causing other threads to stall.
Any PMC	0000002884	PM_ISU_FLUSH_BALANCE	A balance flush occurred. Balance flushes are triggered when excessive L3 or TLB misses occur.
Any PMC	0000002088	PM_ISU_FLUSH_PARTIAL	A flush occurred only to the odd ITAG of a pair. This type of flush requires an additional 10 cycles to process. More pairs of instructions can be included in this count.
Any PMC	0000002888	PM_ISU_FLUSH_DISP_SRQ_EMPTY	Dispatch flush while waiting for the SRQ to become empty: TLBIE, SLBIEG, SLBIET, EIEIO, TLBSYNC.
Any PMC	000000288C	PM_ISU_FLUSH_LWSYNC	A flush to a lightweight synchronizing or barrier instruction ( <b>lwsync</b> ) instruction that had trouble. This event includes NTC flushes and NTC + 1 flushes.
Any PMC	0000002090	PM_ISU_FLUSH_ISYNC	A flush to an <b>isync</b> instruction that had trouble. This event includes NTC flushes and NTC + 1 flushes.
Any PMC	0000002094	PM_ISU_FLUSH_HWSYNC	A flush to a heavyweight synchronizing or barrier instruction ( <b>hwsync</b> ) instruction that had trouble. This event includes NTC flushes and NTC + 1 flushes.
Any PMC	0000002894	PM_ISU_FLUSH_MMA_OFF_CYC	If the MMA engine is off when an MMA instruction is issued, the instruction will issue and finish as a NOP. It will then get flushed when it becomes NTC and it will not be refetched until the MMA engine is on. This event counts the number of cycles between the flush and the MMA engine turning on.
Any PMC	00000028AC	PM_ISU_FLUSH_DISP_STF_REBAL	The sliced target file (STF) is the register file for GPRs, VSRs, LR, CTR, and TAR. This event indicates that an execution unit attempted to write to a slice that was full. In this case, the instruction gets flushed and the slices get rebalanced.
Any PMC	00000020B8	PM_FUNCTION_CALL_DISP	A <b>bl</b> instruction dispatched. Only one reported per cycle. It is possible to dispatch 8 instructions per cycle in single-thread mode.
Any PMC	00000028B8	PM_FUNCTION_RETURN_DISP	A <b>blr</b> instruction dispatched. Only one reported per cycle. It is possible to dispatch 8 instructions per cycle in single-thread mode.
Any PMC	00000020BC	PM_0CYC_CONST_DISP	A <b>xxor</b> instruction dispatched.
Any PMC	0000003080	PM_ISSUE_HOLD_STAGS_CYC	Cycles in which one or more instructions are being held at issue while waiting for store tags to become available. These instructions can also be held simultaneously for other conditions. Other non-store instructions are allowed to issue while the store is being held.
Any PMC	0000003880	PM_ISSUE_HOLD_LTAGS_CYC	Cycles in which one or more instructions are being held at issue while waiting for load tags to become available. These instructions can also be held simultaneously for other conditions. Other non-load instructions are allowed to issue while the load is being held.
Any PMC	0000003084	PM_ISSUE_KILL_DL_MISS	An instruction was primed to issue but was killed before being written because one of the sources takes a data load miss (that is, dependent data from a load does not get valid data).
Any PMC	0000003884	PM_ISSUE_KILL_RESOURCE	An instruction was primed to issue but was killed before being written because a resource is unavailable. Includes BFU, FX-DIV, DFU, BRU, and SFX instructions that would collide at finish with a store AGEN.
Any PMC	0000003088	PM_ISSUE_KILL_THROTTLE	An instruction was primed to issue but was killed before being written because power throttling was enabled.



Table E-22. ISU Events (Sheet 4 of 4)

PMC	Event Code	Event Name	Description
Any PMC	00000030B4	PM_DISP_HELD_OUT_OF_LTAGS_CYC	Cycles in which dispatch is held because the LRQ is full. No LTAGS are available. There are twice as many LTAGS as there are LRQ entries. The signal should be on if less than 8 tags, and should be off if greater than 24 tags, and can be on if in-between. In ST/SMT2 mode, if less than 24 tags, the signal can come on; and if less than 8 it will be on. In SMT4 mode, if less than 12 tags, it can come on, if less than 4 tags it will be on.
Any PMC	00000038B4	PM_DISP_HELD_OUT_OF_STAGS_CYC	Cycles in which dispatch is held because the SRQ is full. No STAGS are available. There are twice as many STAGS as there are SRQ entries. The signal should be on if less than 8 tags, should be off if greater than 24 tags, and could be on if in-between. In ST/SMT2 mode, if less than 24 tags, signal can come on, if less than 8 tags it will be on. In SMT4 mode, if less than 12 tags, it can come on, if less than 4 tags it will be on.
Any PMC	00000030B8	PM_DISP_CLB_HELD_BALANCE_CYC	Dispatch/CLB hold as cause of a balance flush.
Any PMC	00000038B8	PM_DISP_CLB_HELD_SRQ_EMPTY_CYC	Dispatch hold: waiting for the SRQ to become empty: TLBIE, SLBIEG, SLBIET, EIEIO, and TLBSYNC. This should exclude cycles when it is only waiting for the ICT to become empty.
Any PMC	00000030BC	PM_DISP_PARTIAL	Dispatches in which the instruction sequencing unit (ISU) dispatches instructions at either half rate or quarter rate. This can happen when some of the queues (or mapper subblocks) are either full, or close to full.
Any PMC	00000038BC	PM_ISYNC_CMPL	An isync completion count per thread.
Any PMC	0000003898	PM_MMA_VSR_CONFLICT_FLUSH	A VSR operation references an ACC that has been primed. This is indicative of a context switch.
Any PMC	000000309C	PM_MMA_ACC_CONFLICT_FLUSH	An MMA instruction references an ACC that has not been primed. This is indicative of faulty software.
Any PMC	000000399C	PM_MRK_MMA_ACC_VSR_CONFLICT	Either a marked MMA instruction references an ACC that has not been primed or a marked VSR operation references an ACC that has been primed.
Any PMC	00000030A0	PM_MMA_IN_USE_CYC	Cycles in which the MMA engine is actively being used by this thread. This is measured by counting the cycles in which at least one ACC register is primed for MMA use.
Any PMC	00000038A0	PM_MMA_ON_CYC	Cycles in which this core's MMA engine is enabled.

## E.5.4 VSU Events

Table E-23 lists the Power10 events that cover Vector-Scalar (VSU) instructions.

Table E-23. VSU Events

PMC	Event Code	Event Name	Description
PMC1	10016	PM_VSU0_ISSUE	VSU instruction issued to VSU pipe 0.
PMC1	100F4	PM_FLOP_CMPL	Floating-point operations completed. Includes any type. It counts once for each 1, 2, 4, or 8 flop instruction. Use PM_1 2 4 8_FLOP_CMPL events to count flops.
PMC2	2D012	PM_VSU1_ISSUE	VSU instruction issued to VSU pipe 1.
PMC2	2505C	PM_VSU_ISSUE	At least one VSU instruction was issued to one of the VSU pipes. Up to 4 per cycle. Includes fixed-point operations.
PMC3	3F044	PM_VSU2_ISSUE	VSU instruction issued to VSU pipe 2.
PMC3	3D058	PM_SCALAR_FSQRT_FDIV_ISSUE	Scalar versions of four floating point operations: <b>fdiv,fsqrt (xvdivdp, xvdivsp, xvsqrtdp, xvsqrtsp)</b> .
PMC4	40004	PM_FXU_ISSUE	A fixed-point instruction was issued to the VSU.
PMC4	4001C	PM_VSU_FIN	VSU instruction finished.
PMC4	4D020	PM_VSU3_ISSUE	VSU instruction was issued to VSU pipe 3.
PMC4	4D04E	PM_VECTOR_FSQRT_FDIV_ISSUE	Vector versions of <b>fdiv or fsqrt: fdiv, fdivs, fsqrt, fsqrts</b> (or a record variant of any one).
PMC4	4405E	PM_ANY_FLOP_CMPL	Duplicate of PM_FLOP_CMPL. A floating-point (FLOP) instruction of any type completed. It counts once for each 1, 2, 4, or 8 FLOP instruction. Use PM_1 2 4 8_FLOP_CMPL events to count FLOPs.
PMC4	45050	PM_1FLOP_CMPL	One floating-point instruction completed ( <b>fadd, fmul, fsub, fcmp, fsel, fabs, fnabs, fres, fsqrte, fneg</b> ).
PMC4	45052	PM_4FLOP_CMPL	Four floating-point instruction completed ( <b>fadd, fmul, fsub, fcmp, fsel, fabs, fnabs, fres, fsqrte, fneg</b> )
PMC4	45054	PM_FMA_CMPL	Two floating-point instruction completed (FMA class of instructions: <b>fmadd, fnmadd, fmsub, fnmsub</b> ). Scalar instructions only.
PMC4	45056	PM_SCALAR_FLOP_CMPL	Scalar floating-point instruction completed.
PMC4	4505A	PM_SP_FLOP_CMPL	Single-precision floating-point instruction completed.
PMC4	4505C	PM_MATH_FLOP_CMPL	Math floating-point instruction completed.
PMC4	4D050	PM_VSU_NON_FLOP_CMPL	Non-floating-point VSU instruction completed.
PMC4	4D052	PM_2FLOP_CMPL	Double-precision vector version of <b>fmul, fsub, fcmp, fsel, fabs, fnabs, fres, fsqrte, fneg</b> completed.
PMC4	4D054	PM_8FLOP_CMPL	Four double-precision vector instruction completed.
PMC4	4D056	PM_NON_FMA_FLOP_CMPL	Non-FMA instruction completed.
PMC4	4D058	PM_VECTOR_FLOP_CMPL	Vector floating-point instruction completed.
PMC4	4D05A	PM_NON_MATH_FLOP_CMPL	Non-math instruction completed.
PMC4	4D05C	PM_DPP_FLOP_CMPL	Double-precision or quad-precision instruction completed.
PMC4	4E05A	PM_PREFIXED_CMPL	Prefixed instruction completed.



### E.5.5 LSU Events

Table E-24 lists the Power10 events related to the Load-Store Unit (LSU).

Table E-24. LSU Events (Sheet 1 of 7)

PMC	Event Code	Event Name	Description
PMC1	1000C	PM_LSU_LD0_FIN	LSU finished an internal operation in LD0 port.
PMC1	10012	PM_LSU_ST0_FIN	LSU finished an internal operation in ST0 port.
PMC1	10014	PM_LSU_ST4_FIN	LSU finished an internal operation in ST4 port.
PMC1	10056	PM_MEM_READ	Reads from memory from this thread (includes data/instr/xlate/l1prefetch/inst prefetch). This event count should be divided by two because the event is sourced from 2:1 clock domain.
PMC1	1505E	PM_LD_HIT_L1	Load finished without experiencing an L1 miss.
PMC1	10062	PM_LD_L3MISS_PEND_CYC	Cycles in which an L3 miss was pending for this thread.
PMC1	1006A	PM_FX_LSU_FIN	Simple fixed-point instruction issued to the store unit. Measured at finish time.
PMC1	100FC	PM_LD_REF_L1	All L1 D-cache load references counted at finish, gated by reject. In Power9 and earlier, this event counted only cacheable loads; however, in Power10, both cacheable and non-cacheable loads are included.
PMC1	1E058	PM_STCX_FAIL_FIN	Conditional store instruction ( <b>stcx</b> ) failed. The <b>lax</b> and <b>stcx</b> instructions are used to acquire a lock.
PMC1	1002C	PM_LD_PREFETCH_CACHE_LINE_MISS	The L1 cache was reloaded with a line that fulfills a prefetch request.
PMC2	2000E	PM_LSU_LD1_FIN	LSU finished an internal operation in the LD1 port.
PMC2	20016	PM_ST_FIN	Store finish count. Includes speculative activity.
PMC2	20018	PM_ST_FWD	Store forwards that finished.
PMC2	2D010	PM_LSU_ST1_FIN	LSU finished an internal operation in the ST1 port.
PMC2	2C058	PM_MEM_PREF	Memory prefetch for this thread. Includes instruction and data. This event count should be divided by two, because the event is sourced from a 2:1 clock domain.
PMC2	2E05A	PM_LD_REJECT_TIQ	Internal LSU reject from LRQ. Rejects cause the load to go back to the LRQ, but it stays contained within the LSU once it is issued. This event counts the number of times the LRQ attempts to relaunch an instruction after a reject. Any load can suffer multiple rejects.
PMC2	200F0	PM_ST_CMPL	Stores completed from S2Q (2nd-level store queue). This event includes regular stores, stcx, and cache-inhibited stores. The following operations are excluded (pteupdate, snoop tlbie complete, store atomics, miso, load atomic payloads, tlbie, tlbsync, slbieg, isync, msgsnd, slbiag, cpabort, copy, tcheck, tend, stsync, dcbst, icbi, dcdf, hwsync, lwsync, ptesync, eieio, msgsync).
PMC2	2E014	PM_STCX_FIN	Conditional store instruction ( <b>stcx</b> ) finished. The <b>lax</b> and <b>stcx</b> instructions are used to acquire a lock.
PMC3	3000E	PM_ST_REJECT_TIQ	TIQ reject for stores.
PMC3	3001A	PM_LSU_ST2_FIN	LSU finished an internal operation in the ST2 port.
PMC3	3011C	PM_MRK_LD_REJECT_LMQ_FULL	A marked LSU reject due to an LMQ full (up to 4 per cycle).



Table E-24. LSU Events (Sheet 2 of 7)

PMC	Event Code	Event Name	Description
PMC3	3F042	PM_HOT_LOCK_COLLISION	A thread [x] larx is sent to the L1 miss pipe to generate a request to the L2. However, the thread [x] larx is blocked because a thread [y] larx stcx sequence to the same cache line is already in progress. The thread [x] larx will be sent once the thread [y] stcx has drained to the L2. This collision pulses once for the detection described above and is meant as a count event.
PMC3	3F04A	PM_LSU_ST5_FIN	LSU finished an internal operation in the ST2 port.
PMC3	3C05E	PM_MEM_RWITM	Memory read with intent to modify for this thread. This event count should be divided by two because the event is sourced from a 2:1 clock domain.
PMC3	3E054	PM_LD_MISS_L1	Load missed L1, counted at finish time. LMQ merges are not included in this count. That is, if a load instruction misses on an address that is already allocated on the LMQ, this event will not increment for that load. Note that this count is per slice; thus, if a load spans multiple slices, this event will increment multiple times for a single load.
PMC3	30066	PM_LSU_FIN	LSU finished an internal operation (up to 4 per cycle).
PMC3	300F0	PM_ST_MISS_L1	Store missed L1.
PMC3	300F6	PM_LD_DEMAND_MISS_L1	The L1 cache was reloaded with a line that fulfills a demand miss request. Counted at reload time, before finish.
PMC3	3C058	PM_LARX_FIN	Load and reserve instruction ( <b>larx</b> ) finished. The <b>larx</b> and <b>stcx</b> instructions are used to acquire a lock.
PMC3	30058	PM_TLBIE_FIN	The <b>tlbie</b> instruction finished in the LSU. Two <b>tlbie</b> instructions can finish each cycle. All will be counted.
PMC4	4C01E	PM_LSU_ST3_FIN	LSU finished an internal operation in the ST3 port.
PMC4	4003E	PM_LD_CMPL	Load instruction completed.
PMC4	44054	PM_VECTOR_LD_CMPL	Vector load instruction completed.
PMC4	44056	PM_VECTOR_ST_CMPL	Vector store instruction completed.
PMC4	400F0	PM_LD_DEMAND_MISS_L1_FIN	Load missed L1, counted at finish time.
PMC4	4E050	PM_STCX_PASS_FIN	Conditional store instruction ( <b>stcx</b> ) passed. The <b>larx</b> and <b>stcx</b> instructions are used to acquire a lock.
Any PMC	000000388C	PM_SHL_HIT	A dependency was created when a dispatched store matched the address for a load in the store-hit-load table.
Any PMC	0000003890	PM_LHS_HIT	A dependency was created when a dispatched load matched the address for a store instruction in the load-hit-store table.
Any PMC	0000003094	PM_LHS_CREATED	New entry added to the load-hit-store table.
Any PMC	000000C080	PM_LD0_8B_FIN	A 64-bit or smaller load finished in the LD0 load execution unit.
Any PMC	000000C880	PM_LD1_8B_FIN	A 64-bit or smaller load finished in the LD1 load execution unit.
Any PMC	000000C084	PM_LD0_16B_FIN	A 128-bit load finished in the LD0 load execution unit.
Any PMC	000000C884	PM_LD1_16B_FIN	A 128-bit load finished in the LD1 load execution unit.
Any PMC	000000C088	PM_LD0_32B_FIN	A 256-bit load finished in the LD0 load execution unit.
Any PMC	000000C888	PM_LD1_32B_FIN	A 256-bit load finished in the LD1 load execution unit.
Any PMC	000000C08C	PM_LD0_VECTOR_FIN	Any vector load operation finished in the LD0 execution unit. Excludes load quad, larx quad, fused load double, and load float double-pair.
Any PMC	000000C88C	PM_LD1_VECTOR_FIN	Any vector load operation finished in the LD1 execution unit. Excludes load quad, larx quad, fused load double, and load float double-pair.



Table E-24. LSU Events (Sheet 3 of 7)

PMC	Event Code	Event Name	Description
Any PMC	000000C090	PM_LD0_UNALIGNED_FIN	Load instructions in the LD0 port that are either unaligned, or treated as unaligned and require an additional recycle through the pipeline using the load gather buffer. This typically adds about 10 cycles to the latency of the instruction. This includes loads that cross the 128-byte boundary, octword loads that are not aligned, and a special forward progress case of a load that does not hit in the L1 and crosses the 32-byte boundary and is launched NTC. Counted at finish time.
Any PMC	000000C890	PM_LD1_UNALIGNED_FIN	Load instructions in LD1 port that are either unaligned, or treated as unaligned and require an additional recycle through the pipeline using the load gather buffer. This typically adds about 10 cycles to the latency of the instruction. This includes loads that cross the 128-byte boundary, octword loads that are not aligned, and a special forward progress case of a load that does not hit in the L1 and crosses the 32-byte boundary and is launched NTC. Counted at finish time.
Any PMC	000000C094	PM_ST0_8B_FIN	An 8-byte or smaller store finished in the ST0 store execution unit.
Any PMC	000000C894	PM_ST1_8B_FIN	An 8-byte or smaller store finished in the ST1 store execution unit.
Any PMC	000000C098	PM_ST0_16B_FIN	A 16-byte store finished in the ST0 store execution unit.
Any PMC	000000C898	PM_ST1_16B_FIN	A 16-byte store finished in the ST1 store execution unit.
Any PMC	000000C09C	PM_ST0_32B_FIN	A 32-byte store finished in the ST0 store execution unit.
Any PMC	000000C89C	PM_ST1_32B_FIN	A 32-byte store finished in the ST1 store execution unit.
Any PMC	000000C0A0	PM_ST0_VECTOR_FIN	Any vector store operation finished in the ST0 store execution unit. Excludes store quad, stcx quad, store float double-pair, and fused store double.
Any PMC	000000C8A0	PM_ST1_VECTOR_FIN	Any vector store operation finished in the ST1 store execution unit. Excludes store quad, stcx quad, store float double-pair, and fused store double.
Any PMC	000000C0A4	PM_ST0_UNALIGNED_FIN	Store instructions in the ST0 port that are either unaligned, or treated as unaligned and require an additional recycle through the pipeline. This typically adds about 10 cycles to the latency of the instruction. This only includes stores that cross the 128-byte boundary. Counted at finish time.
Any PMC	000000C8A4	PM_ST1_UNALIGNED_FIN	Store instructions in ST1 port that are either unaligned, or treated as unaligned and require an additional recycle through the pipeline. This typically adds about 10 cycles to the latency of the instruction. This only includes stores that cross the 128-byte boundary. Counted at finish time.
Any PMC	000000C0A8	PM_FALSE_LHS	False load-hit-store (LHS) match detected. Load and store have a partial effective address (EA) match but they are actually independent of each other. The other cases of false LHS cause flushes and they are instrumented with separate PMU events.
Any PMC	000000C8A8	PM_LD0_LHS_REJECT	On the LD0 load execution unit, the real address of a load matched the real address of an earlier store and it is not possible to store forward because not all bytes required by the load are available on the store entry that was hit. In this case, the load must wait for the store to drain or for the store data to issue.
Any PMC	000000C0AC	PM_LD1_LHS_REJECT	On the LD1 load execution unit, the real address of a load matched the real address of an earlier store and it is not possible to store forward because not all bytes required by the load are available on the store entry that was hit. In this case, the load must wait for the store to drain or for the store data to issue.
Any PMC	000000C8AC	PM_ST0_STORE_REJECT	Store reject on the ST0 store execution unit. All internal store rejects cause the instruction to go back to the SRQ and go to sleep until woken up, to try again after the condition has been met.
Any PMC	000000C0B0	PM_ST1_STORE_REJECT	Store reject on the ST1 store execution unit. All internal store rejects cause the instruction to go back to the SRQ and go to sleep until woken up, to try again after the condition has been met.



Table E-24. LSU Events (Sheet 4 of 7)

PMC	Event Code	Event Name	Description
Any PMC	000000C8B0	PM_ST_DRAIN_MERGE	Two stores drain together. Use PM_ST_CMPL as a divider. If one of the stores crosses a cache-line boundary, a store drain can be counted for each half. In this case, if there are three stores they will count as two merges.
Any PMC	000000C0B4	PM_LMQ_MERGE	A load-hit-reload occurred. A load launched in the LSU and hit an existing LMQ entry which is fetching data for the same cache line.
Any PMC	000000C8B4	PM_STCX_CMPL	STCX data was sent to nest that is, total count of stcx. <b>Stcx</b> instructions are not allowed to gather before the final drain-out to the L1 and L2 cache, so this count is always accurate.
Any PMC	000000C0B8	PM_NCST_CMPL	Counts when an I = 1 store operation is sent to the nest. This event counts at drain time and it can include up to 3 stores that have gathered for the same cache line before drain.
Any PMC	000000C8B8	PM_STCX_SUCCESS_CMPL	<b>Stcx</b> instructions that completed successfully. Specifically, counts only when a pass status is returned from the nest.
Any PMC	000000C0BC	PM_DC_RELOAD_COLLISIONS	A load reading the L1 cache has a bank collision with another load reading the same bank, or due to a cache-line reload writing to that bank of the L1 cache.
Any PMC	000000C8BC	PM_DC_STORE_WRITE_COLLISIONS	A store writing the L1 cache at the same time as a reload or dkill writing the L1 cache that results in a bank collision.
Any PMC	000000D080	PM_LSU_SET_MPRED	Set prediction (set-p) miss. The entry was not found in the set prediction table. The set-p table contains a hash of the EA and it helps access the L1 cache.
Any PMC	000000D880	PM_DERAT_HIT	A load or store instruction missed the data cache and hit in the primary ERAT. There is no secondary ERAT.
Any PMC	000000D084	PM_IERAT_HIT	An instruction fetch missed in the instruction cache and hit in the primary ERAT. There is no secondary ERAT.
Any PMC	000000D884	PM_TIQ_BYPASS	A translation request bypasses the TIQ either because the TIQ is completely empty at the time of the request.
Any PMC	000000D088	PM_TIQ_ALLOC_CYC	Cycles when one or more operations on that thread allocated one or more TIQ entries.
Any PMC	000000D888	PM_TIQ_HALF_FULL_CYC	Cycles in which the TIQ has 4 or less active entries.
Any PMC	000000D08C	PM_TIQ_ERAT_MISS_EMB_FULL_RESPIN	A load or store missed in the ERAT and found no space in the EMB.
Any PMC	000000D88C	PM_EMB_FULL_CYC	Cycles in which the ERAT miss buffer is full. When the ERAT miss buffer is full the operation will go to the TIQ.
Any PMC	000000D090	PM_LSU_FLUSH_CYC	LSU flushes: includes all LSU flushes. This event only counts one flush per cycle but the hardware can flush up to 5 instructions per cycle. In general, multiple flushes for the same thread are consolidated into a single flush operation.
Any PMC	000000D890	PM_LSU_FLUSH_CI	Load was not initially issued to the LSU as a cache inhibited (non-cacheable) load, but it was later determined to be cache inhibited.
Any PMC	000000D094	PM_LSU_FLUSH_ALL_WAYS_LOCKED	Non-speculative stores have locked all ways in the data cache and a newly-issued older store is not able to allocate a way for a given congruence class.
Any PMC	000000D894	PM_LSU_FLUSH_LHL	If a load hits on an older load of the same address that has been snooped, or two loads of the same thread and address execute out-of-order and a store of another thread occurs to that address, the younger load will be flushed. The LSU only checks the out-of-orderness of the two loads and ignores the address in detecting this condition.
Any PMC	000000D098	PM_LSU_FLUSH_SAME_ICT_GRP	This flush happens if an ICT pair contains a store followed by a load to the same address with at least one byte of overlap between the two.



Table E-24. LSU Events (Sheet 5 of 7)

PMC	Event Code	Event Name	Description
Any PMC	000000D898	PM_LSU_REJECT_LHS	Effective address (EA) alias reject: no EA match but real addresses do match.
Any PMC	000000D09C	PM_LSU_FLUSH_SPECIAL	LSU workaround flush. These flushes are setup with programmable scan-only latches to perform various actions when the flush macro receives a trigger from the debug macros. These actions include things like flushing the next operation encountered for a particular thread, or flushing the next operation that is the next-to-complete (NTC) operation that is encountered on a particular slice. The kind of flush that the workaround is setup to perform is highly variable.
Any PMC	000000D89C	PM_LSU_FLUSH_SHL	The instruction was flushed because of a sequential load/store consistency issue. If a load or store hits on an older load that has either been snooped (for loads) or has stale data (for stores), a flush will occur.
Any PMC	000000D0A0	PM_LSU_FLUSH_SAO	A load-hit-load condition with strong address ordering (SAO) will have address compare disabled and will flush.
Any PMC	000000D8A0	PM_LSU_FLUSH_LARX_STCX	A <b>lарx</b> is flushed because an older <b>lарx</b> has an LMQ reservation for the same thread. A <b>stcx</b> is flushed because an older <b>stcx</b> is in the STQ. The flush happens when the older <b>lарx/stcx</b> relaunches.
Any PMC	000000D0A4	PM_LSU_FLUSH_OTHER	Other LSU flushes, including: <ul style="list-style-type: none"> <li>Sync - a sync acknowledge from the L2 caused a search of the LRQ for the oldest snooped load. This will either signal a precise flush of the oldest snooped load, or a flush next PPC.</li> <li>Data Valid Flush Next - there are several cases of this, one example is a store and reload are lined up such that a store-hit-reload scenario exists and the CDF has already launched and has received bad/stale data.</li> <li>Bad Data Valid (DVAL) Flush Next - there might be a few cases of this, one example is a larxa (D-cache hit) return data and DVAL signal, but cannot allocate to the LMQ (either the LMQ is full, or some other reason). Already signaled DVAL but cannot watch it for snoop_hit_larx. Need to take the "bad dval" back and flush all younger operations).</li> </ul>
Any PMC	000000F094	PM_LD0_SETP_HIT_EADIR_MISS	A load in the LD0 execution unit matched the setp effective address (EA) hash function for one of the ways in the L1 cache. But the full EA does not match the L1 directory, so the request is sent to the L1 miss pipe.
Any PMC	000000F898	PM_SAME_EA_DIFF_CTXTAG_RADIR_HIT	A load or store did not get a full effective address (EA) plus context match in the EA directory, so it is sent to the L1 miss pipe. In particular, a full EA match, but a context miss occurred. In the L1 miss pipe, the real address (RA) directory lookup finds that the real address is in the L1, but allocated in the EA directory with a different context.
Any PMC	000000F09C	PM_CTXT_MP4_ALLOC	A new entry is created in the 4-entry context alias table due to a load or store that the L1 missed due to same EA/different context, but whose real address (RA) was found in the L1 during a (RA) directory lookup in the miss pipe.
Any PMC	000000F89C	PM_CTXT_ALIAS_HIT CONTRIB	A load or a store in the miss pipe hits in the real address (RA) directory and also on one of the 4 entries of the context alias table. The index and way getting an RA directory hit is written with a context alias tag to allow future L1 hits.
Any PMC	000000F0A0	PM_START_NEW_RENAME	Upon creation of a new context alias table entry, a 4K walk is initiated to set the alias tag valid for the 31 other congruence classes. The walk is only initiated after it is confirmed that at least 2 other congruence classes detected a context alias situation.
Any PMC	000000F8A0	PM_STORE_ALLOCATE	A store misses the L1 effective address (EA) directory and allocates an L1 index and way with its address for use in tracking the store address in the load-store unit (LSU) during the SRQ lifetime of the store.

Table E-24. LSU Events (Sheet 6 of 7)

PMC	Event Code	Event Name	Description
Any PMC	000000F0A4	PM_LOAD_ALLOC_DEPRA_FOR_ALL_WAYS_LOCKED	A load miss allocates a load miss queue (LMQ) entry; however, it is not able to allocate an L1 index and way for load hazard detection in the LRQ, because all 8 ways are locked, either by stores in the SRQ or other loads in the LMQ. The current load is instead marked deprecated. CDF will occur but it will be marked deprecated in the LRQ and ordering hazard checking will be pessimistic.
Any PMC	000000F8A4	PM_STORE_REJECT_FOR_ALL_WAYS_LOCKED	A store misses the L1 effective address (EA) directory, but cannot allocate an L1 index and way with its address due to all 8 ways being locked; either by stores already in the SRQ or by loads in the LMQ. The store is rejected and relaunched immediately.
Any PMC	000000F0B0	PM_DERAT_HIT_4K	A load or store instruction missed the data cache and hit in the primary ERAT. Page size = 4 KB.
Any PMC	000000F8B0	PM_IERAT_HIT_4K	An instruction fetch missed in the instruction cache and hit in the primary ERAT. Page size = 4 KB.
Any PMC	000000F0B4	PM_DERAT_HIT_64K	A load or store instruction missed the data cache and hit in the primary ERAT. Page size = 64 KB.
Any PMC	000000F8B4	PM_IERAT_HIT_64K	An instruction fetch missed in the instruction cache and hit in the primary ERAT. Page size = 64 KB.
Any PMC	000000F0B8	PM_DERAT_HIT_2M	A load or store instruction missed the data cache and hit in the primary ERAT. Page size = 2 MB.
Any PMC	000000F8B8	PM_IERAT_HIT_2M	An instruction fetch missed in the instruction cache and hit in the primary ERAT. Page size = 2 MB.
Any PMC	000000F0BC	PM_DERAT_HIT_1G	A load or store instruction missed the data cache and hit in the primary ERAT. Page size = 1 GB.
Any PMC	000000F8BC	PM_IERAT_HIT_1G	An instruction fetch missed in the instruction cache and hit in the primary ERAT. Page size = 1 GB.
Any PMC	0000003894	PM_LARX_HIT_LARX_HIT	A dependency was created for a <b>lарx</b> instruction that was dispatched and matched an older <b>lарx</b> in the larx-hit-larx table.
Any PMC	0000003098	PM_LARX_HIT_LARX_CREATED	New entry added to the larx-hit-larx table. The <b>lарx</b> and <b>stcx</b> instructions are used to acquire a lock.
Any PMC	000000D8A4	PM_DC_PREF_HW_ALLOC	Prefetch stream allocated by the hardware prefetch mechanism.
Any PMC	000000D0A8	PM_DC_PREF_SW_ALLOC	Prefetch stream allocated by software prefetching.
Any PMC	000000D8A8	PM_DC_PREF_STRIDED_ALLOC	Strided prefetch stream allocated by either the software or hardware mechanisms.
Any PMC	000000D0AC	PM_DC_PREF_CONS_ALLOC	Prefetch stream allocated in the conservative phase by either the hardware prefetch mechanism or software prefetch. The sum of this pair subtracted from the total number of allocs will give the total allocs in normal phase.
Any PMC	000000D8AC	PM_DC_PREF_XCONS_ALLOC	Prefetch stream allocated in the ultra conservative phase by either the hardware prefetch mechanism or software prefetch.
Any PMC	000000D0B0	PM_DC_PREF_CONF	A demand load referenced a line in an active prefetch stream. The stream could have been allocated through the hardware prefetch mechanism or through software. Includes forward and backward streams.
Any PMC	000000D8B0	PM_DC_PREF_FUZZY_CONF	A demand load referenced a line in an active fuzzy prefetch stream. The stream could have been allocated through the hardware prefetch mechanism or through software. Fuzzy stream confirm (out-of-order effects or prefetch cannot keep up).
Any PMC	000000D0B4	PM_DC_PREF_STRIDED_CONF	A demand load referenced a line in an active strided prefetch stream. The stream could have been allocated through the hardware prefetch mechanism or through software.



*Table E-24. LSU Events (Sheet 7 of 7)*

PMC	Event Code	Event Name	Description
Any PMC	000000D8B4	PM_DC_PREF DEALLOC_NO_CONF	A demand load referenced a line in an active fuzzy prefetch stream. The stream could have been allocated through the hardware prefetch mechanism or through software. Fuzzy stream confirm (out-of-order effects or prefetch cannot keep up).
Any PMC	000000D0B8	PM_L1_SW_PREF	Software L1 prefetches, including software transient prefetches.
Any PMC	000000D0BC	PM_L3_SW_PREF	L3 load prefetch, sourced from a software prefetch stream, was sent to the nest.
Any PMC	000000F080	PM_SNOOP_TLBIE_MY_LPAR_CYC	TLBIE snoops executed in the LSU.
Any PMC	000000F880	PM_SNOOP_TLBIE_CYC	Cycles in which TLBIE snoops are executed in the LSU.
Any PMC	000000F084	PM_SNOOP_TLBIE_CACHE_WALK_CYC	TLBIE snoop cycles in which the data cache is being walked.
Any PMC	000000F884	PM_SNOOP_TLBIE_WAIT_ST_CYC	TLBIE snoop cycles in which older stores are still draining.
Any PMC	000000F088	PM_SNOOP_TLBIE_WAIT_LD_CYC	TLBIE snoop cycles in which older loads are still draining.
Any PMC	000000F888	PM_SNOOP_TLBIE_WAIT_IFU_CYC	TLBIE snoop cycles in which the load-store unit is waiting for the instruction cache to be walked.
Any PMC	000000F08C	PM_SNOOP_TLBIE_WAIT_MMU_CYC	TLBIE snoop cycles in which the load-store unit is waiting for the MMU to finish invalidation.

## E.5.6 L2 Events

Table E-25 lists the L2 events.

*Table E-25. L2 Events (Sheet 1 of 7)*

PMC	Event Code	Event Name	Description
Any PMC	000000016080	PM_L2_LD	All successful D-side load dispatches for this thread (L2 miss + L2 hits). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000016880	PM_L2_ST	All successful D-side store dispatches for this thread (L2 miss + L2 hits). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000026080	PM_L2_LD_MISS	All successful D-Side Load dispatches for this thread that missed in the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000026880	PM_L2_ST_MISS	All successful D-Side Store dispatches for this thread that missed in the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000036080	PM_L2_INST	All successful I-side-instruction-fetch (for example, i-demand, i-prefetch) dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000036880	PM_L2_INST_MISS	All successful instruction (demand and prefetch) dispatches for this thread that missed in the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000046080	PM_L2_ALL_MISS	All successful instruction and data load and store (demand and prefetch) dispatches for this thread that missed in the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	000000046880	PM_L2_ISIDE_MRU_TOUCH	I-side L2 MRU touch commands sent to the L2 for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000016080	PM_L2_CASTOUT_MOD	A line in an exclusive (M, Mu, Me) state is evicted from the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000016880	PM_L2_CASTOUT_SHR	A line in a shared (Tx, Sx) state is evicted from the L2. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000026080	PM_L2_IC_INV	Instruction cache invalidates sent over the reload bus to the core. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000026880	PM_L2_DC_INV	Data cache invalidates sent over the reload bus to the core. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000036080	PM_L2_ISIDE_DSIDE	All successful D-side-load or I-side-instruction-fetch dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000036880	PM_L2_ISIDE_DSIDE_HIT	All successful D-side-load or I-side-instruction-fetch dispatches for this thread that were L2 hits. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	010000046080	PM_L2_ST	All successful D-side store dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-25. L2 Events (Sheet 2 of 7)

PMC	Event Code	Event Name	Description
Any PMC	010000046880	PM_L2_ST_HIT	All successful D-side store dispatches for this thread that were L2 hits. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000016080	PM_L2_ISIDE_DSIDE_ATTEMPT	All D-side-load or I-side-instruction-fetch dispatch attempts for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000016880	PM_L2_ISIDE_DSIDE_FAIL_ADDR	All D-side-load or I-side-instruction-fetch dispatch attempts for this thread that failed due to an address collision conflicts with an L2 machine already working on this line (for example, Id-hit-stq or read-claim/castout/snoop machines). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000026080	PM_L2_ISIDE_DSIDE_FAIL_OTHER	All D-side-load or I-side-instruction-fetch dispatch attempts for this thread that failed due to reasons other than an address collision conflicts with an L2 machine (for example, read-claim/snoop machine not available). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000036080	PM_L2_ST_ATTEMPT	All D-side store dispatch attempts for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000036880	PM_L2_ST_DISP_FAIL_ADDR	All D-side store dispatch attempts for this thread that failed due to address collision with L2 machine already working on this line (for example, Id-hits-tq or read-claim/castout/snoop machines). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	020000046080	PM_L2_ST_DISP_FAIL_OTHER	All D-side store dispatch attempts for this thread that failed due to reason other than address collision (for example, read-claim/snoop machine not available). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	030000016080	PM_L2_SN_M_WR_DONE	Snoop dispatched for a write and was M (true M); for DMA cache inject (cacheinj) this will pulse if rty/push is required (will not pulse if cacheinj is accepted). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	030000016880	PM_CO_DISP_FAIL	CO dispatch failed due to all CO machines being busy. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	030000026080	PM_L2_CO_TM_SC_FOOTPRINT	L2 did a cleanifdirty CO to the L3 (that is, created an SC line in the L3) OR L2 TM_store hit dirty HPC line and L3 indicated SC line formed in L3 on RDR bus. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	030000036080	PM_L2_RC_ST_DONE	Read-claim machine did store to line that was in Tx or Sx (tagged or shared state). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	030000036880	PM_L2_SN_SX_I_DONE	Snoop dispatched and went from Sx to Ix. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	030000046080	PM_L2_SN_M_RD_DONE	Snoop dispatched for a read and was M (true M). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	030000046880	PM_L2_SN_M_WR_DONE	Snoop dispatched for a write and was M (true M); for DMA cacheinj this will pulse if rty/push is required (will not pulse if cacheinj is accepted). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	040000016080	PM_L2_LOC_GUESS_CORRECT	L2 guess local (LNS) and guess was correct (that is, data local). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-25. L2 Events (Sheet 3 of 7)

PMC	Event Code	Event Name	Description
Any PMC	040000016880	PM_L2_LOC_GUESS_WRONG	L2 guess local (LNS) and guess was not correct (that is, data not on-chip). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	040000026080	PM_L2_GRP_GUESS_CORRECT	L2 guess group (GS or NNS) and guess was correct (data intra-group and not on-chip). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	040000026880	PM_L2_GRP_GUESS_WRONG	L2 guess group (GS or NNS) and guess was not correct (that is, data is on-chip or beyond-group). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	040000036080	PM_L2_SYS_GUESS_CORRECT	L2 guess system (VGS or RNS) and guess was correct (that is, data is “beyond-group”). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	040000036880	PM_L2_SYS_GUESS_WRONG	L2 guess system (VGS or RNS) and guess was not correct (that is, data not “beyond-group”). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	040000046080	PM_L2_CHIP_PUMP	RC requests that were local (LNS, aka chip) pump attempts. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	040000046880	PM_L2_GROUP_PUMP	RC requests that were on group (GS or NNS, also known as “nodal”) pump attempts. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	050000016080	PM_L2_ST_ALL	Total number of store operations. Does not include pte_update, copy/paste, barriers, and so on. This signal can then be used in conjunction with PM_L2_ST_GATHER_ALL to calculate the percentage of total stores gathered. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	050000016880	PM_ISIDE_ATTEMPT	All I-side-instruction-fetch dispatch attempts for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	050000026080	PM_ISIDE_FAIL_ADDR	All I-side-instruction-fetch dispatch attempts for this thread that failed due to an address collision conflict with an L2 machine already working on this line (for example, load-hit-stq or RC/CO/SN machines). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	050000026880	PM_ISIDE_FAIL_OTHER	All I-side-instruction-fetch dispatch attempts for this thread that failed due to reasons other than an address collision conflict with an L2 machine (for example, no available RC/CO machines). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	050000036080	PM_L2_RTY_ST	RC retries on the Power bus for any store from the core (excludes DCBFs). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	050000036880	PM_L2_RTY_LD	RC retries on the Power bus for any load from the core (excludes DCBFs). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	050000046080	PM_L2_ST_GATHER_ALL	Total number of store operations that gathered that can include st, and stqw. Note that <b>stcx</b> , <b>stcxfk</b> , <b>stcxfnk</b> , and <b>dcbz</b> will never gather. This signal can then be used in conjunction with PM_L2_ST_ALL to calculate the number of store operations that were gathered. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.



Table E-25. L2 Events (Sheet 4 of 7)

PMC	Event Code	Event Name	Description
Any PMC	050000046880	PM_L2_SYS_PUMP	RC requests that were system pump attempts. Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	060000016080	PM_L2_RC0_BUSY	RC mach 0 busy. Used by PMU to sample average RC lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000016880	PM_L2_RC_USAGE	Continuous 16 cycle (2:1) window where this signal rotates through sampling each RC machine busy. The PMU uses this wave to then do 16-cycle count to sample the total number of machs running. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000026080	PM_L2_RC0_BUSY	RC mach 0 busy. Used by the PMU to sample average RC lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000026880	PM_L2_CO_USAGE	Continuous 16-cycle (2:1) window where this signals rotates through sampling each CO machine busy. The PMU uses this wave to then do 16-cycle count to sample the total number of machs running. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000036080	PM_L2_CO0_BUSY	CO mach 0 busy. Used by the PMU to sample average CO lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000036880	PM_L2_SN_USAGE	Continuous 16-cycle (2:1) window where this signal rotates through sampling each snoop machine busy. The PMU uses this wave to then do a 16-cycle count to sample the total number of machs running. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	060000046080	PM_L2_CO0_BUSY	CO mach 0 busy. Used by the PMU to sample average CO lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	070000016080	PM_L2_ST CAUSED_ TM_FAIL	Non-TM store caused a TM transaction in any thread to fail. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000016080	PM_L2_SN0_BUSY	SN mach 0 busy. Used by the PMU to sample average snoop lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000016880	PM_L2_L1PF_READ	Valid when the first beat of data comes in for an L1 prefetch where data came from memory or the L4 cache. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000026080	PM_L2_SN0_BUSY	SN mach 0 busy. Used by the PMU to sample average snoop lifetime (mach0 used as sample point). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000026880	PM_L2_ISIDE_READ	Valid when the first beat of data comes in for an I-side fetch where data came from memory or the L4 cache. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-25. L2 Events (Sheet 5 of 7)

PMC	Event Code	Event Name	Description
Any PMC	080000036080	PM_L2_DSIDE_READ	Valid when the first beat of data comes in for a D-side fetch where data came exclusively from memory or the L4 cache) (that is, the total memory accesses by RCs). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000036880	PM_DUALMCDATA_REQ_USE_LPC_DATA	The L2 RC machine made a dual memory controller data request to the Power bus and the RC was sent LPC data that was usable. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	080000046080	PM_DUALMCDATA_REQ_USE_INTV_DATA	The L2 RC machine made a dual memory controller data request to the Power bus and the RC was sent both intervention data (which it used) and LPC data. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	090000016080	PM_L2_ISIDE_DSIDE_ST_ATTEMPT	All D-side load, D-side store, or I-side instruction fetch dispatch attempts for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	090000026080	PM_L2_ISIDE_DSIDE_ST_SUCCESS	All D-side load, D-side store, or I-side-instruction-fetch successful dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	090000036080	PM_L2_RC_CACHE_READ	RC cache read request. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	090000046080	PM_L2_CO_CACHE_READ	CO cache read request. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000016080	PM_ST_DATA_FROM_L2	Store data line hit in the local L2 cache. Includes cache-line states Sx, Tx, Mx. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000016880	PM_ST_DATA_FROM_L3	Store data line hit in the local L3 cache. Includes cache-line states Tx and Mx. If the cache line is in the Sx state, the RC machine will send a RWITM command. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000026080	PM_ST_DATA_FROM_L21_L31	Store data line missed in the local chiplet and was found in a neighbor L2 or L3 (on-chip L2/L3 intervention). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000026880	PM_ST_DATA_FROM_LMEM	Store data line missed in the local chiplet and was found in local memory. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000036080	PM_ST_DATA_FROM_RL2L3	Store data line missed in the local chiplet and was found in a different chip's L2 or L3 (off chip, within the group). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000036880	PM_ST_DATA_FROM_RMEM	Store data line missed in the local chiplet and was found in remote memory (off chip, within the group). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0C0000046080	PM_ST_DATA_FROM_DL2L3	Store data line missed in the local chiplet and was found in a different group's L2 or L3 (off-group L2/L3 intervention). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



*Table E-25. L2 Events (Sheet 6 of 7)*

PMC	Event Code	Event Name	Description
Any PMC	0C0000046880	PM_ST_DATA_FROM_DMEM	Store data line missed in the local chiplet and was found in distant memory (off group). Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000016080	PM_L2_LD_DISP	All successful D-side load or I-side instruction fetch dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000016880	PM_L2_ST_DISP	All successful D-side store dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000026080	PM_L2_LD_HIT	All successful D-side load or I-side instruction fetch dispatches for this thread that were L2 hits. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000026880	PM_L2_ST_HIT	All successful D-side store dispatches for this thread that were L2 hits. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000036080	PM_L2_INST	All successful I-side instruction etch (for example, i-dem, i-pref) dispatches for this thread. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000036880	PM_L2_RTY_LD	RC encounters retries on the Power bus for any load from core (excludes DCBFs). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	0F0000046080	PM_L2_INST_MISS	All successful I-side instruction fetch (for example, i-dem, i-pref) dispatches for this thread that were an L2 miss. Because the event happens in a 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	0F0000046880	PM_L2_RTY_ST	RC encounters retries on the Power bus for any store from core (excludes DCBFs). Because the event is time-sliced across all 4 threads, the event count should be multiplied by 4.
Any PMC	0B0000016080	PM_L2_TLBIE_SLBIE_START	NCU master received a TLBIE/SLBIEG/SLBIAG operation from the core. Event count should be multiplied by 2, because the data is coming from a 2:1 clock domain and the data is time sliced across all 4 threads.
Any PMC	0B0000016880	PM_L2_TLBIE_SLBIE_DELAY	Cycles when a TLBIE/SLBIEG/SLBIAG command was held in a hottemp condition by the NCU master. Multiply this count by 1000 to obtain the total number of cycles. This can be divided by PM_L2_TLBIE_SLBIE_SENT to obtain the average time a TLBIE/SLBIEG/SLBIAG command was held. Event count should be multiplied by 2, because the data is coming from a 2:1 clock domain and the data is time sliced across all 4 threads.



Table E-25. L2 Events (Sheet 7 of 7)

PMC	Event Code	Event Name	Description
Any PMC	0B0000026080	PM_L2_SNP_TLBIE_SLBIE_START	The NCU snooper snooped a TLBIE/SLBIEG/SLBIAG operation that targets this thread's LPAR and has sent it to the core. Event count should be multiplied by 2, because the data is coming from a 2:1 clock domain and the data is time sliced across all 4 threads.
Any PMC	0B0000026880	PM_L2_SNP_TLBIE_SLBIE_DELAY	Cycles when a TLBIE/SLBIEG/SLBIAG that targets this thread's LPAR was in flight while in a hottemp condition. Multiply this count by 1000 to obtain the total number of cycles. This can be divided by PM_L2_SNP_TLBIE_SLBIE_START to obtain the overall efficiency. <b>Note:</b> 'inflight' means SnpTLB has been sent to the core (that is, it does not include when SnpTLB is in the NCU waiting to be launched serially behind a different SnpTLB). The NCU Snooper gets in a 'hottemp' delay window when it detects it is above its TLBIE/SLBIE threshold for process SnpTLBIE/SLBIE with this core. Event count should be multiplied by 2 because the data is coming from a 2:1 clock domain and the data is time sliced across all 4 threads.
Any PMC	0B0000036080	PM_NCU_SNP_TLBIE_CYC	Cycles in which the NCU had a TLBIE snoop in flight to core. Event count should be multiplied by 2, because the data is coming from a 2:1 clock domain and the data is time sliced across all 4 threads.



### E.5.7 L3 Events

Table E-26 lists the L3 events.

*Table E-26. L3 Events (Sheet 1 of 7)*

PMC	Event Code	Event Name	Description
PMC2	20058	PM_LD_CACHE_INHIBITED	A data line was brought in from the nest to fulfill a request initiated by a non-cacheable load instruction. The data will not be written to the L1 cache.
PMC3	3E05E	PM_L3_CO_MEFP	L3 castouts in Mepf state for this core.
Any PMC	100000016080	PM_L3_PF_MISS_L3	L3 prefetch (L3PF) missed in the L3 cache. The L3 cache received an L3PF request from the core (or possibly the L2 if enabled), for a line that was not already in the L3 cache. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000016880	PM_L3_CO_MEFP	L3 CO of line in Mep state (includes cast through to memory). The L3 cache made a castout of the line in the Mepf state, indicating that the line was pulled into the L3 cache via a fabric prefetch, but evicted without being read by the core or L2 cache. The Mepf state indicates that a line was brought in to satisfy an L3 prefetch request. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000026080	PM_L3_CO_MEM	L3 CO to memory. L3 castout a line to memory and it was accepted (CRESP = good) by the memory controller. Lossy - increments only once when two COs receive CRESP = good simultaneously, thus the event might undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000026880	PM_L3_CO_L31	L3 CO to L3.1 Lossy. L3 castout a line as an LCO (L3.1) and it was accepted (CRESP = good) by a destination L3. Lossy - increments only once when two LCOs receive CRESP = good simultaneously, thus the event might undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000036080	PM_L3_PF_ON_CHIP_CACHE	L3 PF from on-chip cache. L3PF successfully read data and that data came from an on-chip cache. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000036880	PM_L3_PF_OFF_CHIP_CACHE	L3 PF from off-chip cache. L3PF successfully read data and that data came from an off-chip cache. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000046080	PM_L3_PF_ON_CHIP_MEM	L3 PF from on-chip memory. L3PF successfully read data and that data came from an on-chip memory. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	100000046880	PM_L3_PF_OFF_CHIP_MEM	L3 PF from off-chip memory. L3PF successfully read data and that data came from off-chip memory. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	110000026080	PM_L3_CI_HIT	L3 castins hit L3. L3 inserted a line due to an L2 castout, an incoming LCO, or a successful L3PF and the line was already in the L3 (a "hit"). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-26. L3 Events (Sheet 2 of 7)

PMC	Event Code	Event Name	Description
Any PMC	110000026880	PM_L3_CI_MISS	L3 castins miss L3. L3 inserted a line due to an L2 castout, an incoming LCO or a successful L3PF and the line was new to the L3 (a "miss"). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	110000036080	PM_L3_L2_CO_HIT	L2 CO hits L3. L3 inserted a line due to an L2 castout and the line was already in the L3 (a "hit"). PM_L3_CI_HIT also increments when this does. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	110000036880	PM_L3_L2_CO_MISS	L2 CO misses L3. L3 inserted a line due to an L2 castout and the line was new to the L3 (a "miss"). PM_L3_CI_MISS also increments when this does. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	110000046080	PM_L3_LAT_CI_HIT	L3 lateral castins hit. L3 inserted a line due to an incoming LCO and the line was already in the L3 (a "hit"). PM_L3_CI_HIT also increments when this does. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	110000046880	PM_L3_LAT_CI_MISS	L3 lateral castins miss. L3 inserted a line due to an incoming LCO and the line was new to the L3 (a "miss"). PM_L3_CI_MISS also increments when this does. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000016080	PM_L3_HIT	L3 hits. Any L2 read that hits in the L3, including data load and store, instruction load, or translate load. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000016880	PM_L3_MISS	L3 misses. Any L2 read to the L3 that misses in the L3, including data load and store, instruction load, or translate load. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000026080	PM_L3_LD_HIT	L3 hits for loads, but not stores. Any L2 load (but not store) that hits in the L3, including data load, instruction load, or translate load. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000026880	PM_L3_LD_MISS	L3 misses for loads, but not stores. Any L2 load (but not store) to the L3 that misses in the L3, including data load, instruction load, or translate load. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000036080	PM_L3_CO_LCO	Incoming LCOs that cause L3 castouts. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000036880	PM_L3_CINJ	L3 castin of cache inject. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	120000046880	PM_L3_TRANS_PF	L3 transient prefetch received from L2. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	130000046080	PM_RD_FORMING_SC	Does not occur. Deprecated.
Any PMC	130000046880	PM_RD_CLEARING_SC	Core TM load hits line in L3 in TM_SC state and causes it to be invalidated. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000016880	PM_L3_WI_USAGE	Lifetime, sample of write inject machine 0 valid. Increments while write inject machine 0 is valid. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-26. L3 Events (Sheet 3 of 7)

PMC	Event Code	Event Name	Description
Any PMC	140000026080	PM_L3_PF_HIT_L3	L3 PF hit in L3 (abandoned). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000026880	PM_RD_HIT_PF	L3 RD machine hit L3 PF machine. L3 RD machine, which is running an L2 read, including data load and store, instruction load or translate load, hit an active L3 prefetch machine. Thus an L2 read occurred before the previously queued L3 prefetch request completed. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000036080	PM_L3_CO	L3 castout occurred (does not include castthrough). Counts L3 castout dispatches, thus includes castout requests that are subsequently dropped, such as CP_Me. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000036880	PM_SN_INVL	Snoop of L3 on detects a store to a line in the Sx state and invalidates the line. Lossy - increments only once for all such snoops that occur on the same cycle. Up to 4 can happen in a cycle but only one increment occurs. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000046080	PM_SN_HIT	Snoop of L3 on any port hits L3. Lossy - increments only once for all such snoops that occur on the same cycle. Up to 4 can happen in a cycle but only one increment occurs. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	140000046880	PM_SN_MISS	Snoop of L3 on any port either misses L3 or collides with a machine working on the line. Lossy - increments only once for all such snoops that occur on the same cycle. Up to 4 can happen in a cycle but only one increment occurs. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000016080	PM_L3_P0_LCO_NO_DATA	Dataless L3 LCO sent port 0. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000016880	PM_L3_P1_LCO_NO_DATA	Dataless L3 LCO sent port 1. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000026080	PM_L3_P0_LCO_DATA	LCO sent with data port 0. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000026880	PM_L3_P1_LCO_DATA	LCO sent with data port 1. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000036080	PM_L3_P0_CO_MEM	L3 CO to memory from L3 CO machine 0-7. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000036880	PM_L3_P1_CO_MEM	L3 CO to memory from L3 CO machine 8-15. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000046080	PM_L3_P0_CO_L31	L3 CO to L3.1 (LCO) from L3 CO machine 0-7. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	150000046880	PM_L3_P1_CO_L31	L3 CO to L3.1 (LCO) from L3 CO machine 8-15. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-26. L3 Events (Sheet 4 of 7)

PMC	Event Code	Event Name	Description
Any PMC	160000016080	PM_L3_SN_USAGE	Rotating sample of 16 snoop valids. Increments while selected L3 snoop machine is valid. Every 48 clocks, switches among the 16 snoop machines. Thus, indicates overall utilization of the snoop machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000016880	PM_L3_CI_USAGE	Rotating sample of 16 CI or CO actives. Increments while selected L3 CI or CO machine is active. Every 48 clocks, switches among the 16 CI/CO machine pairs. Thus, indicates overall utilization of the CI/CO machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000026080	PM_L3_PF_USAGE	Rotating sample of 48 PF actives. Increments when selected L3 PF machine is active. Every 48 clocks, switches among the 48 PF machines. Thus, indicates overall utilization of the PF machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000026880	PM_L3_RD_USAGE	Rotating sample of 16 RD actives. Increments when selected L3 RD machine is active. Every 48 clocks, switches among the 16 RD machines. Thus, indicates overall utilization of the RD machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000036080	PM_L3_SN0_BUSY	Lifetime, sample of snooper machine 0 valid. Increments when L3 snoop machine 0 is valid. Can increment on the same clock as PM_L3_SN_USAGE. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000036880	PM_L3_CO0_BUSY	Lifetime, sample of CO machine 0 valid. Increments when L3 CO machine 0 is valid. Can increment on the same clock as PM_L3_CI_USAGE. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000046080	PM_L3_SN0_BUSY	Lifetime, sample of snooper machine 0 valid. Increments when L3 snoop machine 0 is valid. Can increment on the same clock as PM_L3_SN_USAGE. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	160000046880	PM_L3_CO0_BUSY	Lifetime, sample of CO machine 0 valid. Increments when L3 CO machine 0 is valid. Can increment on the same clock as PM_L3_CI_USAGE. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000016080	PM_L3_P0_PF_RTY	L3 PF received retry on fabric CRESP port 0, every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000016880	PM_L3_P1_PF_RTY	L3 PF received retry on fabric CRESP port 1, every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000026080	PM_L3_P2_PF_RTY	L3 PF received retry on fabric CRESP port 2, every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000026880	PM_L3_P3_PF_RTY	L3 PF received retry on fabric CRESP port 3, every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000036080	PM_L3_P0_CO_RTY	L3 CO received retry on fabric CRESP port 0 (memory only), every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



Table E-26. L3 Events (Sheet 5 of 7)

PMC	Event Code	Event Name	Description
Any PMC	170000036880	PM_L3_P1_CO_RTY	L3 CO received retry on fabric CRESP port 1 (memory only), every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000046080	PM_L3_P2_CO_RTY	L3 CO received retry on fabric CRESP port 2 (memory only), every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	170000046880	PM_L3_P3_CO_RTY	L3 CO received retry on fabric CRESP port 3 (memory only), every retry counted. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000016080	PM_L3_P0_NODE_PUMP	L3 PF sent with nodal scope from PF machine 0-23, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000016880	PM_L3_P1_NODE_PUMP	L3 PF sent with nodal scope from PF machine 24-47, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000026080	PM_L3_P0_GRP_PUMP	L3 PF sent with grp or nn scope from PF machine 0-23, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000026880	PM_L3_P1_GRP_PUMP	L3 PF sent with grp or nn scope from PF machine 24-47, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000036080	PM_L3_P0_SYS_PUMP	L3 PF sent with sys vg or rn scope from PF machine 0-23, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	180000036880	PM_L3_P1_SYS_PUMP	L3 PF sent with sys vg or rn scope from PF machine 24-47, counts even retried requests. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000016080	PM_L3_LOC_GUESS_CORRECT	L3 prefetch scope predictor selected LNS and was correct. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000016880	PM_L3_GRP_GUESS_CORRECT	L3 prefetch scope predictor selected GS or NNS and was correct. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000026080	PM_L3_SYS_GUESS_CORRECT	L3 prefetch scope predictor selected VGS or RNS and was correct. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000026880	PM_L3_LOC_GUESS_WRONG	L3 prefetch scope predictor selected LNS, but was wrong. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000036080	PM_L3_GRP_GUESS_WRONG_LOW	L3 prefetch scope predictor selected GS or NNS, but was wrong because scope was LNS. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.

Table E-26. L3 Events (Sheet 6 of 7)

PMC	Event Code	Event Name	Description
Any PMC	190000036880	PM_L3_GRP_GUESS_WRONG_HIGH	L3 prefetch scope predictor selected GS or NNS, but was wrong because scope was VGS or RNS. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	190000046080	PM_L3_SYS_GUESS_WRONG	L3 prefetch scope predictor selected VGS or RNS, but was wrong. Increments when data received for L3 PF machine 0 or 24. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000016080	PM_L3_P0_LCO_RTY	L3 initiated LCO received retry on fabric CRESP port 0 (can try 4 times). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000016880	PM_L3_P1_LCO_RTY	L3 initiated LCO received retry on fabric CRESP port 1 (can try 4 times). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000026080	PM_L3_P2_LCO_RTY	L3 initiated LCO received retry on fabric CRESP port 2 (can try 4 times). Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000026880	PM_L3_P3_LCO_RTY	L3 initiated LCO received retry on fabric CRESP port 3 (can try 4 times). Since the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000036080	PM_L3_PF0_BUSY	Lifetime, sample of L3 PF machine 0 busy. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000036880	PM_L3_RD0_BUSY	Lifetime, sample of L3 RD machine 0 busy. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000046080	PM_L3_PF0_BUSY	Lifetime, sample of L3 PF machine 0 busy. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1A0000046880	PM_L3_RD0_BUSY	Lifetime, sample of L3 RD machine 0 busy. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1B0000016080	PM_L3_WI0_BUSY	Rotating sample of 16 WI valid. Increments when selected L3 WI machine is valid. Every 48 clocks, switches among the 16 WI machines. Thus, indicates overall utilization of the WI machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1B0000026080	PM_L3_WI0_BUSY	Rotating sample of 16 WI valid. Increments when selected L3 WI machine is valid. Every 48 clocks, switches among the 16 WI machines. Thus, indicates overall utilization of the WI machines. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1B0000026880	PM_L3_CO_L31	L3 CO to L3.1 lossy. L3 castout a line as an LCO (L3.1) and it was accepted (CRESP = good) by a destination L3. Lossy - increments only once when two LCO receive CRESP = good simultaneously, thus event might undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.



*Table E-26. L3 Events (Sheet 7 of 7)*

PMC	Event Code	Event Name	Description
Any PMC	1B0000036080	PM_L3_PF_MPRED_DUALMCDATA	L3 prefetch request predicted dual MC data but a cache responded (bad prediction). Lossy - increments only once when multiple L3 PFs receive CRESP = good simultaneously; thus event might undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1B0000036880	PM_L3_CO_L31_INST_XLAT	L3 cast out an instruction or translation line to a neighbor L3 (LCO). Lossy - increments only once when multiple L3 COs receive CRESP = good simultaneously; thus may undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.
Any PMC	1B0000046080	PM_L3_PF_CPRED_DUALMCDATA	L3 prefetch request predicted dual MC data and LPC responded (good prediction). Lossy - increments only once when multiple L3 PFs receive CRESP = good simultaneously, thus event might undercount. Because the event happens in the 2:1 clock domain and is time-sliced across all 4 threads, the event count should be multiplied by 2.

### E.5.8 CPI Stack Events

The simplest way to visualize processor performance is to consider that during every cycle the thread might complete one or more instructions or none. Identifying the cycles when no instructions complete or a suboptimal number of instructions complete is the basic step in tuning. Tuning can involve environmental changes, such as using more efficient page sizes for an application, or an actual program, or library changes that more efficiently flow through the processor cores.

The Power10 processor is designed to track and complete instructions individually; therefore, the PMU can identify unique conditions when an individual instruction is not completed. The Power10 PMU provides functionality for a hardware-based CPI stall measurement facility that can hierarchically account for completion stalls, execution stalls, and front-end stalls on a per-thread basis. The Power10 PMU CPI analysis uses a set of counters to count the cycles for an instruction to complete or that do not complete. For every cycle the processor is executing instructions, the performance monitor identifies the reason the oldest instruction in the pipeline is not completing. A CPI breakdown contains multiple levels.

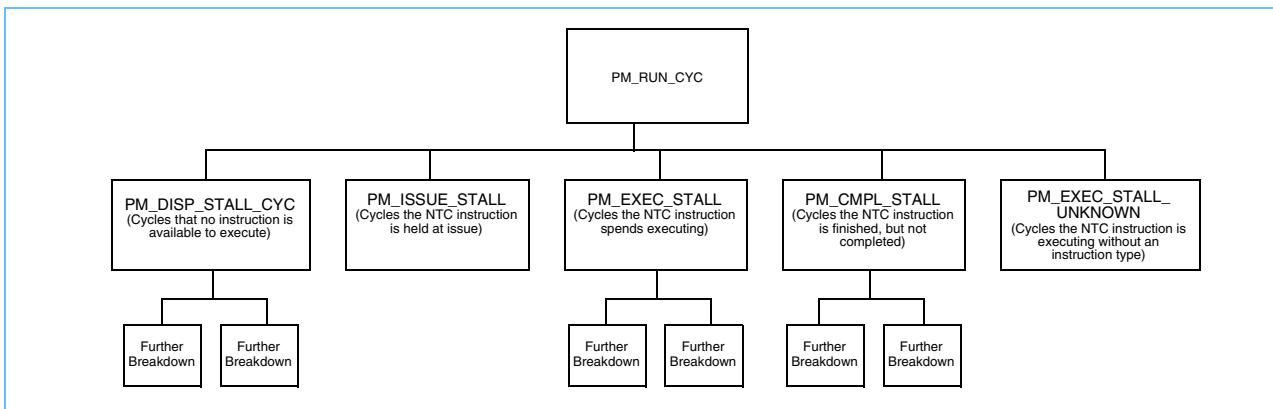
The first level identifies the first-order categorization of why the next instruction to complete did not finish in that cycle. The lower levels of the analysis, when present, provide more granularity of the root cause. Because the CPI stack monitors events that stall completion, it is important to note that it is most sensitive to long latency stall events while many other sources of performance disruption, such as those that occur within the pipeline, might not be visible. For this reason, it is suggested to augment analysis of the CPI stack with the analysis of other performance events and/or events captured in SIER/SIER2/SIER3 registers during instruction sampling.

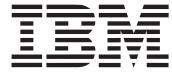
The CPI breakdown is best visualized as a tree. Each level of the tree brings further categorization of hardware performance. The sum of the cycles of a lower level of the tree should match the cycles of the higher-level component.

At the most basic level, the CPI stack (on a per-thread level) can be broken down into five main categories:

- ICT empty (no instructions have been dispatched for this thread).
- Issue hold (instructions that have been dispatched are awaiting issue to an execution unit).
- Execution stall (the instruction is being executed or waiting to satisfy a request).
- Completion (the instruction finished but cannot complete because the thread is blocked).
- Unknown (the instruction finished too quickly after the previous instruction and the instruction type is not reported)

Figure E-4. CPI Breakdown as a Tree





The ratio of any category to run cycles defines the fraction of cycles that the category represents. Tuning begins by identifying the most significant categories and drilling down to further levels for more specific reasons on why instructions are not completing. When specific categories require further correlation to the program code, profiling can be employed on the related marked events.

On the Power10 processor, the CPI breakdown tree is a maximum of six levels deep. In some cases, the cycle accreditation of the components of a leaf is directly measured, and sometimes it is computed from other elements of the leaf.

*Table E-27. CPI Stack (Tree Format) (Sheet 1 of 2)*

PM_CYC	PM_DISP_STALL_CYC (pmc1)	PM_DISP_STALL_FLUSH (pmc3)
		PM_DISP_STALL_FETCH (pmc2)
		PM_DISP_STALL_TRANSLATION (pmc1) PM_DISP_STALL_IERAT_ONLY_MISS (pmc2) PM_DISP_STALL_ITLB_MISS (pmc3)
		PM_DISP_STALL_IC_MISS (pmc2) PM_DISP_STALL_IC_L2 (pmc1) PM_DISP_STALL_IC_L3 (pmc3) PM_DISP_STALL_IC_L3MISS (pmc4)
		PM_DISP_STALL_BR_MPRED_IC_MISS (pmc3) PM_DISP_STALL_BR_MPRED_IC_L2 (pmc1) PM_DISP_STALL_BR_MPRED_IC_L3 (pmc2) PM_DISP_STALL_BR_MPRED_IC_L3MISS (pmc4)
		PM_DISP_STALL_BR_MPRED (pmc4)
		PM_DISP_STALL_HELD_SYNC_CYC (pmc4) PM_DISP_STALL_HELD_SCOREBOARD_CYC (pmc3) PM_DISP_STALL_HELD_ISSQ_FULL_CYC (pmc2)
		PM_DISP_STALL_HELD_CYC (pmc4) PM_DISP_STALL_HELD_STF_MAPPER_CYC (pmc1) PM_DISP_STALL_HELD_XVFC_MAPPER_CYC (pmc2)
		PM_DISP_STALL_HELD_HALT_CYC (pmc1) PM_DISP_STALL_HELD_OTHER_CYC (pmc1)
		PM_ISSUE_STALL (pmc2)

Table E-27. CPI Stack (Tree Format) (Sheet 2 of 2)

		PM_EXEC_STALL_NTC_FLUSH (pmc2)		
		PM_EXEC_STALL_FIN_AT_DISP (pmc1)		
		PM_EXEC_STALL_BRU (pmc4)		
		PM_EXEC_STALL_SIMPLE_FX (pmc3)		
		PM_EXEC_STALL_VSU (pmc2)		
PM_CYC	PM_EXEC_STALL (pmc3)	PM_EXEC_STALL_TRANSLATION (pmc1)	PM_EXEC_STALL_DERAT_ONLY_MISS (pmc4)	
			PM_EXEC_STALL_DERAT_DTLB_MISS (pmc3)	
		PM_EXEC_STALL_LOAD (pmc4)	PM_EXEC_STALL_DMIS-S_L2L3 (pmc1)	PM_EXEC_STALL_DMISS_L2L3_CONFLICT (pmc4)
				PM_EXEC_STALL_DMISS_L2L3_NOCONFLICT (pmc3)
			PM_EXEC_STALL_DMIS-S_L3MISS (pmc2)	PM_EXEC_STALL_DMISS_L21_L31 (pmc1)
				PM_EXEC_STALL_DMISS_LMEM (pmc3)
				PM_EXEC_STALL_DMISS_OFF_CHIP (pmc2)
				PM_EXEC_STALL_DMISS_OFF_NODE (pmc4)
				PM_EXEC_STALL_TLBIEL (pmc4)
				PM_EXEC_STALL_LOAD_FINISH (pmc3)
		PM_EXEC_STALL_STORE (pmc3)	PM_EXEC_STALL_STORE_PIPE (pmc1)	
			PM_EXEC_STALL_STORE_MISS (pmc3)	
			PM_EXEC_STALL_TLBIE (pmc2)	
			PM_EXEC_STALL_PTESYNC (pmc4)	
PM_CMPL	PM_CMPL_STALL (pmc4)	PM_CMPL_STALL_EXCEPTION (pmc3)		
		PM_CMPL_STALL_MEM_ECC (pmc3)		
		PM_CMPL_STALL_STCX (pmc2)		
		PM_CMPL_STALL_LWSYNC (pmc1)		
		PM_CMPL_STALL_HWSYNC (pmc4)		
		PM_CMPL_STALL_SPECIAL (pmc2)		
		PM_EXEC_STALL_UNKNOWN (pmc4)		



Table E-28 lists the Power10 events related to CPI Stack events.

*Table E-28. CPI Stack Events (Sheet 1 of 3)*

PMC	Event Code	Event Name	Description
PMC1	10004	PM_EXEC_STALL_TRANSLATION	Cycles in which the oldest instruction in the pipeline suffered a TLB miss or ERAT miss and waited for it to resolve.
PMC1	10006	PM_DISP_STALL_HELD_OTHER_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch for any other reason
PMC1	10038	PM_DISP_STALL_TRANSLATION	Cycles when dispatch was stalled for this thread because the MMU was handling a translation miss.
PMC1	1003A	PM_DISP_STALL_BR_MPRED_IC_L2	Cycles when dispatch was stalled while the instruction was fetched from the local L2 cache after suffering a branch mispredict.
PMC1	1003C	PM_EXEC_STALL_DMISS_L2L3	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from either the local L2 or local L3 cache.
PMC1	10058	PM_EXEC_STALL_FIN_AT_DISP	Cycles in which the oldest instruction in the pipeline finished at dispatch and did not require execution in the LSU, BRU, or VSU.
PMC1	1D05E	PM_DISP_STALL_HELD_HALT_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch because of power management
PMC1	1E050	PM_DISP_STALL_HELD_STF_MAPPER_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch because the STF mapper/SRB was full. Includes GPR (count, link, tar), VSR, VMR, FPR
PMC1	1E054	PM_EXEC_STALL_DMISS_L21_L31	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from another core's L2 or L3 cache on the same chip.
PMC1	1E056	PM_EXEC_STALL_STORE_PIPE	Cycles in which the oldest instruction in the pipeline was executing in the store unit. This does not include cycles spent handling store misses, PTE-SYNC instructions or TLBIE instructions.
PMC1	1E05A	PM_CMPL_STALL_LWSYNC	Cycles in which the oldest instruction in the pipeline was a <b>lwsync</b> waiting to complete.
PMC1	10064	PM_DISP_STALL_IC_L2	Cycles when dispatch was stalled while the instruction was fetched from the local L2 cache.
PMC1	100F8	PM_DISP_STALL_CYC	Cycles the ICT has no itags assigned to this thread (no instructions were dispatched during these cycles).
PMC2	20004	PM_ISSUE_STALL	Cycles in which the oldest instruction in the pipeline was dispatched but not issued yet.
PMC2	20006	PM_DISP_STALL_HELD_ISSQ_FULL_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch due to issue queue full. Includes issue queue and branch queue.
PMC2	2C010	PM_EXEC_STALL_LSU	Cycles in which the oldest instruction in the pipeline was executing in the load store unit. This does not include simple fixed-point instructions.
PMC2	2C014	PM_CMPL_STALL_SPECIAL	Cycles in which the oldest instruction in the pipeline required special handling before completing.
PMC2	2C016	PM_DISP_STALL_IERAT_ONLY_MISS	Cycles when dispatch was stalled while waiting to resolve an instruction ERAT miss.
PMC2	2C018	PM_EXEC_STALL_DMISS_L3MISS	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from a source beyond the local L2 or local L3 cache.
PMC2	2C01C	PM_EXEC_STALL_DMISS_OFF_CHIP	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from a remote chip.
PMC2	2C01E	PM_DISP_STALL_BR_MPRED_IC_L3	Cycles when dispatch was stalled while the instruction was fetched from the local L3 cache after suffering a branch mispredict.
PMC2	2D018	PM_EXEC_STALL_VSU	Cycles in which the oldest instruction in the pipeline was executing in the VSU (includes FXU, VSU, CRU).

Table E-28. CPI Stack Events (Sheet 2 of 3)

PMC	Event Code	Event Name	Description
PMC2	2D01A	PM_DISP_STALL_IC_MISS	Cycles when dispatch was stalled for this thread due to an instruction cache miss.
PMC2	2D01C	PM_CMPL_STALL_STCX	Cycles in which the oldest instruction in the pipeline was a <b>stcx</b> waiting for resolution from the nest before completing.
PMC2	2E018	PM_DISP_STALL_FETCH	Cycles when dispatch was stalled for this thread because fetch was being held.
PMC2	2E01A	PM_DISP_STALL_HELD_XVFC_MAPPER_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch because the XVFC mapper/SRB was full.
PMC2	2E01C	PM_EXEC_STALL_TLBIE	Cycles in which the oldest instruction in the pipeline was a <b>tlbie</b> instruction executing in the load store unit.
PMC2	2E01E	PM_EXEC_STALL_NTC_FLUSH	Cycles in which the oldest instruction in the pipeline was executing in any unit before it was flushed. Note that if the flush of the oldest instruction happens after finish, the cycles from dispatch to issue will be included in PM_DISP_STALL and the cycles from issue to finish will be included in PM_EXEC_STALL and its corresponding children. This event will also count cycles when the previous next-to-finish (NTF) instruction is still completing and the new NTF instruction is stalled at dispatch.
PMC3	30004	PM_DISP_STALL_FLUSH	Cycles when dispatch was stalled because of a flush that happened to an instruction(s) that was not yet next-to-complete (NTC). PM_EXEC_STALL_NTC_FLUSH only includes instructions that were flushed after becoming NTC.
PMC3	30008	PM_EXEC_STALL	Cycles in which the oldest instruction in the pipeline was waiting to finish in one of the execution units (BRU, LSU, VSU). Only cycles between issue and finish are counted in this category.
PMC3	3000A	PM_DISP_STALL_ITLB_MISS	Cycles when dispatch was stalled while waiting to resolve an instruction TLB miss.
PMC3	30014	PM_EXEC_STALL_STORE	Cycles in which the oldest instruction in the pipeline was a store instruction executing in the load store unit.
PMC3	30016	PM_EXEC_STALL_DERAT_DTLB_MISS	Cycles in which the oldest instruction in the pipeline suffered a TLB miss and waited for it resolve.
PMC3	30018	PM_DISP_STALL_HELD_SCOREBOARD_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch while waiting on the scoreboard. This event combines VSCR and FPSCR together.
PMC3	30026	PM_EXEC_STALL_STORE_MISS	Cycles in which the oldest instruction in the pipeline was a store whose cache line was not resident in the L1 cache and was waiting for allocation of the missing line into the L1.
PMC3	30028	PM_CMPL_STALL_MEM_ECC	Cycles in which the oldest instruction in the pipeline was waiting for the non-speculative finish of either a <b>stcx</b> waiting for its result or a load waiting for non-critical sectors of data and ECC.
PMC3	30036	PM_EXEC_STALL_SIMPLE_FX	Cycles in which the oldest instruction in the pipeline was a simple fixed-point instruction executing in the load store unit.
PMC3	30038	PM_EXEC_STALL_DMISS_LMEM	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from the local memory, local OpenCAPP cache, or local OpenCAPP memory.
PMC3	3003A	PM_CMPL_STALL_EXCEPTION	Cycles in which the oldest instruction in the pipeline was not allowed to complete because it was interrupted by ANY exception, which has to be serviced before the instruction can complete.
PMC3	34054	PM_EXEC_STALL_DMISS_L2L3_NOCONFLICT	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from the local L2 or local L3 cache, without a dispatch conflict.



Table E-28. CPI Stack Events (Sheet 3 of 3)

PMC	Event Code	Event Name	Description
PMC3	34056	PM_EXEC_STALL_LOAD_FINISH	Cycles in which the oldest instruction in the pipeline was finishing a load after its data was reloaded from a data source beyond the local L1; cycles in which the LSU was processing an L1 hit; cycles in which the next-to-finish (NTF) instruction merged with another load in the LMQ; cycles in which the NTF instruction is waiting for a data reload for a load miss, but the data comes back with a non-NTF instruction.
PMC3	34058	PM_DISP_STALL_BR_MPRED_ICMISS	Cycles when dispatch was stalled after a mispredicted branch resulted in an instruction cache miss.
PMC3	3D05C	PM_DISP_STALL_HELD_RENAME_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch because the mapper/SRB was full. Includes GPR (count, link, tar), VSR, VMR, FPR, and XVFC
PMC3	3E052	PM_DISP_STALL_IC_L3	Cycles when dispatch was stalled while the instruction was fetched from the local L3 cache.
PMC4	4C010	PM_DISP_STALL_BR_MPRED_IC_L3MISS	Cycles when dispatch was stalled while the instruction was fetched from sources beyond the local L3 cache after suffering a mispredicted branch.
PMC4	4C012	PM_EXEC_STALL_DERAT_ONLY_MISS	Cycles in which the oldest instruction in the pipeline suffered an ERAT miss and waited for it resolve.
PMC4	4C016	PM_EXEC_STALL_DMISS_L2L3_CONFLICT	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from the local L2 or local L3 cache, with a dispatch conflict.
PMC4	4C018	PM_CMPL_STALL	Cycles in which the oldest instruction in the pipeline cannot complete because the thread was blocked for any reason.
PMC4	4C01A	PM_EXEC_STALL_DMISS_OFF_NODE	Cycles in which the oldest instruction in the pipeline was waiting for a load miss to resolve from a distant chip.
PMC4	4D014	PM_EXEC_STALL_LOAD	Cycles in which the oldest instruction in the pipeline was a load instruction executing in the Load Store Unit.
PMC4	4D016	PM_EXEC_STALL_PTESYNC	Cycles in which the oldest instruction in the pipeline was a <b>ptesync</b> instruction executing in the load store unit.
PMC4	4D018	PM_EXEC_STALL_BRU	Cycles in which the oldest instruction in the pipeline was executing in the Branch unit.
PMC4	4D01A	PM_CMPL_STALL_HWSYNC	Cycles in which the oldest instruction in the pipeline was a <b>hwsync</b> waiting for response from the L2 cache before completing.
PMC4	4D01C	PM_EXEC_STALL_TLBIEL	Cycles in which the oldest instruction in the pipeline was a <b>tlbiel</b> instruction executing in the load store unit. <b>TlbIEL</b> instructions have lower overhead than <b>tlbie</b> instructions because they do not get set to the nest.
PMC4	4D01E	PM_DISP_STALL_BR_MPRED	Cycles when dispatch was stalled for this thread due to a mispredicted branch.
PMC4	4E010	PM_DISP_STALL_IC_L3MISS	Cycles when dispatch was stalled while the instruction was fetched from any source beyond the local L3 cache.
PMC4	4E012	PM_EXEC_STALL_UNKNOWN	Cycles in which the oldest instruction in the pipeline completed without an ntf_type pulse. The ntf_pulse was missed by the ISU because the next-to-finish (NTF) instruction finishes and completions came too close together.
PMC4	4E01A	PM_DISP_STALL_HELD_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch for any reason.
PMC4	4003C	PM_DISP_STALL_HELD_SYNC_CYC	Cycles in which the next-to-complete (NTC) instruction is held at dispatch because of a synchronizing instruction that requires the ICT to be empty before dispatch.

## E.5.9 Marked Events

Table E-29 lists the Power10 events related to marked/sampled instructions. For a marked event to occur, MMCRA[MARK] must be set.

Table E-29. Marked Events (Sheet 1 of 5)

PMC	Event Code	Event Name	Description
PMC1	10132	PM_MRK_INST_ISSUED	Marked instruction issued. Note that stores are always issued twice, the address is issued to the LSU and the data is issued to the VSU. Also, issues can sometimes be killed/cancelled and cause multiple sequential issues for the same instruction.
PMC1	10134	PM_MRK_ST_DONE_L2	Marked store completed in the L2 cache.
PMC1	1013E	PM_MRK_LD_MISS_EXPOSED_CYC	Marked load exposed miss (use edge detect to count number).
PMC1	1C040	PM_XFER_FROM_SRC_PMC1	The processor's L1 data cache was reloaded from the source specified in MMCR3[0:12]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.
PMC1	1C142	PM_MRK_XFER_FROM_SRC_PMC1	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[0:12]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.
PMC1	1C144	PM_MRK_XFER_FROM_SRC_CYC_PMC1	Cycles taken for a marked demand miss to reload a line from the source specified in MMCR3[0:12].
PMC1	1C146	PM_MRK_DATA_FLUSHED_FROM_SRC	Marked demand loads that attempted a reload, but were flushed.
PMC1	1C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC	Cycles spent attempting to reload a line from any source for a marked demand miss that was later flushed.
PMC1	1C14A	PM_MRK_SYS_PUMP_MPRED_RTY	Final pump scope (system) ended up larger than the initial pump scope (chip/group) for a marked instruction. Includes instructions and data lines.
PMC1	1015E	PM_MRK_FAB_RSP_RD_T_INTV	Sampled read got a T intervention.
PMC1	1D156	PM_MRK_LD_MISS_L1_CYC	Marked load latency. The latency measured for this event counts between the original launch of a load L1 miss to the relaunch of that same load L1 miss once data is back from the nest and the load is going to finish.
PMC1	1D15C	PM_MRK_DTLB_MISS_1G	Marked data TLB reload (after a miss) page size 1 GB. Implies radix translation was used. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1E152	PM_MRK_DERAT_MISS_16M	Data ERAT miss (data TLB access) page size 16 MB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1E15E	PM_MRK_L2_TM_REQ_ABORT	TM abort.
PMC1	1F150	PM_MRK_ST_L2_CYC	Cycles from L2 RC dispatch to L2 RC completion.
PMC1	1F152	PM_MRK_FAB_RSP_BKILL_CYC	Cycles L2 RC took for a bkill.
PMC1	101E0	PM_MRK_INST_DISP	The thread has dispatched a randomly sampled marked instruction.
PMC1	101E2	PM_MRK_BR_TAKEN_CMPL	Marked branch taken instruction completed.



Table E-29. Marked Events (Sheet 2 of 5)

PMC	Event Code	Event Name	Description
PMC1	101E4	PM_MRK_L1_ICACHE_MISS	Marked instruction suffered an instruction cache miss.
PMC1	101EA	PM_MRK_L1_RELOAD_VALID	Marked demand reload.
PMC2	20108	PM_MRK_TLBIE_CYC	Latency of <b>tlbie</b> instructions from the cycle they become next-to-finish (NTF) until they complete in the nest.
PMC2	20112	PM_MRK_NTF_FIN	The marked instruction became the oldest in the pipeline before it finished. It excludes instructions that finish at dispatch.
PMC2	20114	PM_MRK_L2_RC_DISP	Marked instruction RC dispatched in the L2 cache.
PMC2	2011C	PM_MRK_NTF_CYC	Cycles in which the marked instruction is the oldest in the pipeline (next-to-finish or next-to-complete).
PMC2	2D12C	PM_MRK_DATA_GRP_PUMP_MPRED_RTY	Final pump scope (group) ended up larger than initial pump scope (chip) for a marked demand load.
PMC2	2D12E	PM_MRK_DTLB_HIT	The PTE required by the instruction was resident in the TLB (data TLB access) for the marked instruction. When MMCR1[16] = 0, this event counts only demand hits. When MMCR1[16] = 1, this event includes demand and prefetch. Applies to both HPT and RPT.
PMC2	20130	PM_MRK_INST_DECODED	An instruction was marked at decode time. Random instruction sampling (RIS) only.
PMC2	20132	PM_MRK_DFU_ISSUE	The marked instruction was a decimal floating-point operation issued to the VSU. Measured at issue time.
PMC2	20134	PM_MRK_FXU_ISSUE	The marked instruction was a fixed-point operation issued to the VSU. Measured at issue time.
PMC2	20138	PM_MRK_ST_NEST	A store has been sampled/marketed and is at the point of execution where it has completed in the core and can no longer be flushed. At this point the store is sent to the L2 cache.
PMC2	2013A	PM_MRK_BRU_FIN	Marked branch instruction finished.
PMC2	2013C	PM_MRK_FX LSU FIN	The marked instruction was simple fixed-point that was issued to the store unit. Measured at finish time.
PMC2	2003E	PM_PTESYNC_FIN	A <b>ptesync</b> instruction finished in the store unit. Only one <b>ptesync</b> can finish at a time.
PMC2	2C142	PM_MRK_XFER_FROM_SRC_PMC2	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[15:27]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.
PMC2	2C144	PM_MRK_XFER_FROM_SRC_CYC_PMC2	Cycles taken for a marked demand miss to reload a line from the source specified in MMCR3[15:27].
PMC2	2C146	PM_MRK_DATA_FLUSHED_FROM_SRC	Marked demand loads that attempted a reload, but were flushed.
PMC2	2C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC	Cycles spent attempting to reload a line from any source for a marked demand miss that was later flushed.
PMC2	2015E	PM_MRK_FAB_RSP_RWITM_RTY	Sampled store did a <b>rwitm</b> and got a rty.
PMC2	24156	PM_MRK_STCX_FIN	Marked conditional store instruction ( <b>stcx</b> ) finished. <b>Larx</b> and <b>stcx</b> are instructions used to acquire a lock



Table E-29. Marked Events (Sheet 3 of 5)

PMC	Event Code	Event Name	Description
PMC2	24158	PM_MRK_INST	An instruction was marked. Includes both random instruction sampling (RIS) at decode time and random event sampling (RES) at the time the configured event happens.
PMC2	2415C	PM_MRK_BR_CMPL	A marked branch completed. All branches are included.
PMC2	2D150	PM_MRK_DERAT_MISS_4K	Data ERAT miss (data TLB access) page size 4 KB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2D154	PM_MRK_DERAT_MISS_64K	Data ERAT miss (data TLB access) page size 64 KB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2D156	PM_MRK_DTLB_MISS_16M	Marked data TLB reload (after a miss) page size 16 MB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2D15E	PM_MRK_DTLB_MISS_16G	Marked data TLB reload (after a miss) page size 16 GB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2F152	PM_MRK_FAB_RSP_DCLAIM_CYC	Cycles L2 RC took for a dclaim.
PMC2	201E0	PM_MRK_DATA_FROM_MEMORY	The processor's data cache was reloaded from local, remote, or distant memory due to a demand miss for a marked load.
PMC2	201E2	PM_MRK_LD_MISS_L1	Marked demand data load miss counted at finish time.
PMC2	201E4	PM_MRK_DATA_FROM_L3MISS	The processor's data cache was reloaded from a source other than the local core's L1, L2, or L3 cache due to a demand miss for a marked load.
PMC3	3012A	PM_MRK_L2_RC_DONE	L2 RC machine completed the transaction for the marked instruction.
PMC3	3012C	PM_MRK_ST_FWD	Marked store forward.
PMC3	3012E	PM_MRK_DTLB_MISS_2M	Marked data TLB reload (after a miss) page size 2 MB, which implies radix page table translation was used. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	30130	PM_MRK_INST_FIN	Marked instruction finished. Excludes instructions that finish at dispatch. Note that stores always finish twice because the address is issued to the LSU and the data is issued to the VSU.
PMC3	30132	PM_MRK_VSU_FIN	VSU marked instruction finished. Excludes simple FX instructions issued to the store unit.
PMC3	30134	PM_MRK_ST_CMPL_INT	Marked store finished with intervention.
PMC3	3013C	PM_MRK_DTLB_MISS_4K	Marked data TLB reload (after a miss) page size 4 KB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	34146	PM_MRK_LD_CMPL	Marked load instruction completed.
PMC3	34149	PM_MRK_STORE_DATA	A marked instruction was waiting for data. This event uses edge detection.
PMC3	3C142	PM_MRK_XFER_FROM_SRC_PMC3	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[30:42]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.
PMC3	3C144	PM_MRK_XFER_FROM_SRC_CYC_PMC3	Cycles taken for a marked demand miss to reload a line from the source specified in MMCR3[30:42].



Table E-29. Marked Events (Sheet 4 of 5)

PMC	Event Code	Event Name	Description
PMC3	3C146	PM_MRK_DATA_FLUSHED_FROM_SRC	Marked demand loads that attempted a reload, but were flushed.
PMC3	3C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC	Cycles spent attempting to reload a line from any source for a marked demand miss that was later flushed.
PMC3	3C14C	PM_MRK_DATA_SYS_PUMP_MPRED_RTY	Final pump scope (system) ended up larger than initial pump scope (chip/group) for a marked demand load.
PMC3	30154	PM_MRK_FAB_RSP_DCLAIM	Marked store had to do a dclaim.
PMC3	3015C	PM_MRK_PTESYNC_CYC	Ptesync latency from the cycle the marked instruction becomes next-to-complete (NTC) until it completes in the nest.
PMC3	3015E	PM_MRK_FAB_RSP_CLAIM_RTY	Sampled store did a <b>rwitm</b> and got a rty.
PMC3	3D154	PM_MRK_DERAT_MISS_16G	Data ERAT miss (data TLB access) page size 16 GB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	3E158	PM_MRK_STCX_FAIL	Marked conditional store instruction ( <b>stcx</b> ) failed. <b>Larx</b> and <b>stcx</b> are instructions used to acquire a lock .
PMC3	3E15A	PM_MRK_ST_FIN	Marked store instruction finished.
PMC3	3F150	PM_MRK_ST_DRAIN_CYC	Cycles in which the marked store drained from the core to the L2 cache.
PMC3	30162	PM_MRK_ISSUE_DEPENDENT_LOAD	The marked instruction was dependent on a load. It is eligible for issue kill.
PMC3	301E2	PM_MRK_ST_CMPL	Marked store completed and sent to nest. Note that this count excludes cache-inhibited stores.
PMC3	301E4	PM_MRK_BR_MPRED_CMPL	Marked branch mispredicted. Includes direction and target.
PMC3	301E6	PM_MRK_DERAT_MISS	Marked ERAT miss (data TLB access); all page sizes. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	4010E	PM_MRK_TLBIE_FIN	Marked <b>tlbie</b> instruction finished. Includes <b>tlbie</b> and <b>tlbiel</b> instructions
PMC4	40116	PM_MRK_LARX_FIN	Marked load and reserve instruction ( <b>larx</b> ) finished. <b>Larx</b> and <b>stcx</b> are instructions used to acquire a lock.
PMC4	40118	PM_MRK_DCACHE_RELOAD_INTV	The marked instruction reloaded data from any of the caches on the system excluding the local caches. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	40132	PM_MRK_LSU_FIN	LSU marked instruction finish.
PMC4	40136	PM_MRK_STORE_DATA_CYC	Cycles in which the marked instruction was waiting for data. Divide by PM_MRK_STORE_DATA to obtain the average Store data latency.
PMC4	44146	PM_MRK_STCX_CORE_CYC	Cycles spent in the core portion of a marked <b>stcx</b> instruction. It starts counting when the instruction is decoded and stops counting when it drains into the L2.
PMC4	4C142	PM_MRK_XFER_FROM_SRC_PMC4	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[45:57]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.

Table E-29. Marked Events (Sheet 5 of 5)

PMC	Event Code	Event Name	Description
PMC4	4C144	PM_MRK_XFER_FROM_SRC_CYC_PMC4	Cycles taken for a marked demand miss to reload a line from the source specified in MMCR3[45:57].
PMC4	4C146	PM_MRK_DATA_FLUSHED_FROM_SRC	Marked demand loads that attempted a reload, but were flushed.
PMC4	4C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC	Cycles spent attempting to reload a line from any source for a marked demand miss that was later flushed.
PMC4	40152	PM_MRK_PUMP_MPRED	Pump misprediction. Counts across all types of pumps with a marked instruction for all data types excluding data prefetch (demand load, instruction prefetch, instruction fetch, xlate).
PMC4	40154	PM_MRK_FAB_RSP_BKILL	Marked store had to do a bkill.
PMC4	40156	PM_MRK_GRP_PUMP_MPRED_RTY	Final pump scope (group) ended up larger than initial pump scope (chip) for a marked instruction. Includes instruction and data lines.
PMC4	4015E	PM_MRK_FAB_RSP_RD_RTY	Sampled L2 reads retry count.
PMC4	4C15C	PM_MRK_DERAT_MISS_1G	Data ERAT miss (data TLB access) page size 1 GB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	4C15E	PM_MRK_DTLB_MISS_64K	Marked Data TLB reload (after a miss) page size 64K. When MMCR1[16]=0 this event counts only for demand misses. When MMCR1[16]=1 this event includes demand misses and prefetches
PMC4	4E15E	PM_MRK_INST_FLUSHED	The marked instruction was flushed.
PMC4	4F150	PM_MRK_FAB_RSP_RWITM_CYC	Cycles L2 RC took for a <b>rwitm</b> .
PMC4	40164	PM_MRK_DERAT_MISS_2M	Data ERAT miss (data TLB access) page size 2 MB for a marked instruction. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	401E0	PM_MRK_INST_CMPL	Marked instruction completed.
PMC4	401E4	PM_MRK_DTLB_MISS	The <b>DPTEG</b> required for the marked load/store instruction in execution was missing from the TLB. It includes pages of all sizes for demand and prefetch activity
PMC4	401E6	PM_MRK_INST_FROM_L3MISS	The processor's instruction cache was reloaded from a source other than the local core's L1, L2, or L3 cache due to a demand miss for a marked instruction.
PMC4	401E8	PM_MRK_DATA_FROM_L2MISS	The processor's data cache was reloaded from a source other than the local core's L1 or L2 cache due to a demand miss for a marked load.



## E.5.10 MMU Events

Table E-30 lists the Power10 events related to memory management (MMU).

Table E-30. MMU Events (Sheet 1 of 6)

PMC	Event Code	Event Name	Description
PMC1	1C056	PM_DERAT_MISS_4K	Data ERAT miss (data TLB access) page size 4 KB. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1C058	PM_DTLB_MISS_16G	Data TLB reload (after a miss) page size 16 GB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1C05A	PM_DERAT_MISS_2M	Data ERAT miss (data TLB access) page size 2 MB. Implies radix translation. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1C05C	PM_DTLB_MISS_2M	Data TLB reload (after a miss) page size 2 MB. Implies radix translation was used. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1C05E	PM_ITLB_MISS_64K	Instruction TLB reload (after a miss) page size 64 KB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC1	1D050	PM_DTLB_HIT_16G	Data TLB hit (DERAT reload) page size 16 GB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1D052	PM_IERAT_MISS_1G	IERAT miss (instruction TLB access) page size 1 GB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC1	1D054	PM_DTLB_HIT_2M	Data TLB hit (DERAT reload) page size 2 MB. Implies radix translation. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC1	1D058	PM_ITLB_HIT_64K	Instruction TLB hit (IERAT reload) page size 64 KB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC1	1F054	PM_DTLB_HIT	The PTE required by the instruction was resident in the TLB (data TLB access). When MMCR1[16] = 0, this event counts only demand hits. When MMCR1[16] = 1, this event includes demand and prefetch. Applies to both HPT and RPT.
PMC1	100F6	PM_IERAT_MISS	IERAT reloaded to satisfy an IERAT miss. All page sizes are counted by this event.
PMC2	2001A	PM_ITLB_HIT	The PTE required to translate the instruction address was resident in the TLB (instruction TLB access/IERAT reload). Applies to both HPT and RPT. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC2	20050	PM_HPT_RELOAD	A page was reloaded for an HPT translation.
PMC2	2C054	PM_DERAT_MISS_64K	Data ERAT miss (data TLB access) page size 64 KB. When MMCR1[16] = 0 this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1 this event includes demand misses and prefetches.
PMC2	2C056	PM_DTLB_MISS_4K	Data TLB reload (after a miss) page size 4 KB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.

Table E-30. MMU Events (Sheet 2 of 6)

PMC	Event Code	Event Name	Description
PMC2	2C05A	PM_DERAT_MISS_1G	Data ERAT miss (data TLB access) page size 1 GB. Implies radix translation. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2D058	PM_ITLB_MISS_2M	Instruction TLB reload (after a miss) page size 2 MB, which implies radix page table translation is in use. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC2	2D05A	PM_ITLB_MISS_16M	Instruction TLB reload (after a miss) page size 16 MB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC2	2D05C	PM_IERAT_MISS_16G	IERAT miss (instruction TLB access) page size 16 GB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC2	2E050	PM_DTLB_HIT_4K	Data TLB hit (DERAT reload) page size 64 KB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC2	2E054	PM_ITLB_HIT_2M	Instruction TLB hit (IERAT reload) page size 2 MB, which implies radix page table translation is in use. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC2	2E056	PM_ITLB_HIT_16M	Instruction TLB hit (IERAT reload) page size 16 MB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC2	20064	PM_IERAT_MISS_4K	IERAT miss (instruction TLB access) page size 4 KB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC2	200F6	PM_DERAT_MISS	DERAT reloaded to satisfy a DERAT miss. All page sizes are counted by this event. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	3003E	PM_ITLB_MISS_1G	Instruction TLB reload (after a miss) page size 1 GB, which implies radix page table translation is in use. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC3	34040	PM_ITLB_MISS_16G	Instruction TLB reload (after a miss) page size 16 GB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC3	34042	PM_HPT_TLB_PARENT_HIT_CHILD_MISS	Hierarchical TLB found parent entry, but child entry was not found in the TLB. Two-level TLB, second level miss.
PMC3	34044	PM_DERAT_MISS_PREF	DERAT miss (TLB access) while servicing a data prefetch.
PMC3	3F046	PM_ITLB_HIT_1G	Instruction TLB hit (IERAT reload) page size 1 GB, which implies radix page table translation is in use. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC3	3F048	PM_ITLB_HIT_16G	Instruction TLB hit (IERAT reload) page size 16 GB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC3	3C054	PM_DERAT_MISS_16M	Data ERAT miss (data TLB access) page size 16 MB. When MMCR1[16] = 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.



Table E-30. MMU Events (Sheet 3 of 6)

PMC	Event Code	Event Name	Description
PMC3	3C056	PM_DTLB_MISS_64K	Data TLB reload (after a miss) page size 64 KB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	3C05A	PM_DTLB_HIT_64K	Data TLB hit (DERAT reload) page size 64 KB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC3	3006A	PM_IERAT_MISS_64K	IERAT Miss (Instruction TLB Access) page size 64 KB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC3	300FC	PM_DTLB_MISS	The DPTEG required for the load/store instruction in execution was missing from the TLB. It includes pages of all sizes for demand and prefetch activity.
PMC4	44040	PM_L2_PWC_HIT	A translation reload hits in the L2 page walk cache.
PMC4	44042	PM_L3_PWC_HIT	A translation reload hits in the L3 page walk cache.
PMC4	4C054	PM_DERAT_MISS_16G	Data ERAT miss (data TLB access) page size 16 GB. When MMCR1[16] 0, this event counts only DERAT reloads for demand misses. When MMCR1[16] = 1 this event includes demand misses and prefetches.
PMC4	4C056	PM_DTLB_MISS_16M	Data TLB reload (after a miss) page size 16 MB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	4C05A	PM_DTLB_MISS_1G	Data TLB reload (after a miss) page size 1 GB. Implies radix translation was used. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	4E052	PM_DTLB_HIT_16M	Data TLB hit (DERAT reload) page size 16 MB. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	4E054	PM_DTLB_HIT_1G	Data TLB hit (DERAT reload) page size 1 GB. Implies radix translation. When MMCR1[16] = 0, this event counts only for demand misses. When MMCR1[16] = 1, this event includes demand misses and prefetches.
PMC4	40066	PM_ITLB_HIT_4K	Instruction TLB hit (IERAT reload) page size 4 KB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC4	40068	PM_IERAT_MISS_2M	IERAT miss (instruction TLB access) page size 2 MB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC4	4006A	PM_IERAT_MISS_16M	IERAT miss (Instruction TLB Access) page size 16 MB. When MMCR1[16] = 0, this event counts only IERAT reloads for demand misses. When MMCR1[16] = 1, this event will count incorrectly.
PMC4	4006C	PM_ITLB_MISS_4K	Instruction TLB reload (after a miss) page size 4 KB. When MMCR1[17] = 0, this event counts only for demand misses. When MMCR1[17] = 1, this event includes demand misses and prefetches.
PMC4	400FC	PM_ITLB_MISS	Instruction TLB reload (after a miss), all page sizes. Includes only demand misses.
Any PMC	0000008080	PM_TLB_ACCESS_L3PREF	Incoming L3 prefetches that access the TLB (there is no L3 instruction prefetching in the Power10 processor). L3 prefetches access the TLB directly, without accessing the ERAT.
Any PMC	0000008880	PM_TLB_CHILD_PURGE_CYC	Cycles spent in a TLB scrub. Counted for radix or HPT translations. Implies a parent eviction.
Any PMC	0000008884	PM_REJ_MMU	The MMU rejects a translation for any reason. This includes loads/stores/ifetches.



Table E-30. MMU Events (Sheet 4 of 6)

PMC	Event Code	Event Name	Description
Any PMC	0000008088	PM_REJ_EXEC_NTC_SLEEP_SELECT	The operation is rejected because it must be NTC to translate. This includes loads/stores/ifetches.
Any PMC	0000008090	PM_REJ_TABLEWALK_L2_PDE_MERGE	A translation is rejected on a level-2 PDE boundary. This is only valid for radix. This is when the incoming translation matches on the 1 GB address boundary, but not a 2 MB address boundary.
Any PMC	0000008890	PM_REJ_TABLEWALK_L3_PDE_MERGE	A translation is rejected on a level-3 PDE boundary. This is when the incoming translation matches on the 2 MB address boundary, but not the 64 KB address boundary. This is only valid in radix translations.
Any PMC	0000008094	PM_REJ_TABLEWALK_PTE_MERGE	A translation is rejected on a PTE boundary. This indicates that the table walk merged due to a full 4k or 64k address match. This is valid for both radix and HPT. In HPT, this is both the parent hit, child miss case, and the parent miss case.
Any PMC	0000008894	PM_MRK_DTABLEWALK_CYC	Cycles a data (MMU type LD, ST, L1 PF) tablewalk is in progress for a marked instruction.
Any PMC	0000008098	PM_XLATE_HPT_MODE_CYC	MMU reports every cycle the thread is in HPT translation mode (as opposed to radix mode).
Any PMC	0000008898	PM_XLATE_RADIX_MODE_CYC	MMU reports every cycle the thread is in radix translation mode (as opposed to HPT mode).
Any PMC	000000809C	PM_XLATE_L2_REQ	The MMU requested a line from the L2 for translation. It may be satisfied from L2 and beyond. Includes speculative instructions. Includes instruction, prefetch and demand.
Any PMC	000000908C	PM_DTLB_ACCESS_ERAT_MISS	The TLB was read to satisfy a load or store ERAT miss. This includes demands and L1 prefetching data (MMU type LD, ST, L1PF).
Any PMC	000000988C	PM_ITLB_ACCESS_ERAT_MISS	The TLB was read to satisfy an instruction ERAT miss. This includes demands and L1 prefetching for instruction (MMU type fetch).
Any PMC	0000009090	PM_HPTWALK_INSTR_CYC	Cycles when an instruction tablewalk is active. This is qualified for HPT translations.
Any PMC	0000009890	PM_HPTWALK_DATA_CYC	Data tablewalk cycles. Cycles in which at least one tablewalk is in progress. This event includes data and instruction, demand and prefetch. There can be up to 4 tablewalks in one cycle (MMU Type LD, ST, L1PF).
Any PMC	0000009094	PM_REJ_PROBE_MATCH_PRS0	A load/store, fetch, or prefetch was rejected because it matched on a partition-scoped probe. This is set for both radix and HPT.
Any PMC	0000009894	PM_REJ_PROBE_MATCH_PRS1	A load/store, fetch, or prefetch was rejected because it matched on a process-scoped probe. This is radix only.
Any PMC	0000009098	PM_REJ_PROBE_MATCH_MML	A load/store, fetch, or prefetch was rejected because it matched on the MML lock. This is set for both radix and HPT.
Any PMC	0000009898	PM_REJ_PIPE_COLLISION_2ND_RELOAD	A load/store or fetch was rejected because there was a collision in the MMU pipeline with a 2nd pass and a reload.
Any PMC	000000909C	PM_REJ_XMQ_FULL	A translation was rejected because the MMU was full.
Any PMC	000000989C	PM_BLOCK_ERAT_WRITE	A valid translation reloaded the ERAT with block write enabled.
Any PMC	00000090A0	PM_2ND_PASS_RADIX	The second pass for radix was initiated.
Any PMC	00000098A0	PM_2ND_PASS_HPT	The second pass for HPT was initiated. This does not include VA hash mispredicts.
Any PMC	00000090A4	PM_VA_HASH_MPRED	The first pass for HPT was rejected due to VA hash mispredicts.
Any PMC	000000A080	PM_CHILD_PURGE_HIT	Total amount of congruence classes that found at least one invalidation match for a child purge.
Any PMC	000000A090	PM_CASE_A_HIT_MTPID	A <b>mtpid</b> instruction found a case A context table hit.



Table E-30. MMU Events (Sheet 5 of 6)

PMC	Event Code	Event Name	Description
Any PMC	000000A890	PM_CASE_A_HIT_MTLPID	A <b>mtlid</b> instruction found a case A context table hit.
Any PMC	000000A094	PM_CASE_B_HIT_MTLPID	A <b>mtlid</b> instruction found a case B context table hit.
Any PMC	000000A894	PM_CASE_C_HIT_MTPID	A <b>mtpid</b> instruction found a case C context table hit.
Any PMC	000000A098	PM_CASE_A_MISS_MTPID	A <b>mtpid</b> instruction found a case A context table miss
Any PMC	000000A898	PM_CASE_A_MISS_MTLPID	A <b>mtlid</b> instruction found a case A context table miss
Any PMC	000000A09C	PM_CASE_B_MISS_MTLPID	A <b>mtlid</b> instruction found a case B context table miss
Any PMC	000000A89C	PM_CASE_C_MISS_MTPID	A <b>mtpid</b> instruction found a case C context table miss.
Any PMC	000000A0A0	PM_MTPID	The thread executed a <b>mtpid</b> instruction.
Any PMC	000000A8A0	PM_MTLPID	The thread executed a <b>mtlid</b> instruction.
Any PMC	000000808C	PM_RDXWALK_INSTR_CYC	Cycles when an instruction (MMU type fetch) tablewalk is active.
Any PMC	000000888C	PM_RDXWALK_DATA_CYC	Data tablewalk cycles. Cycles in which at least one tablewalk is in progress. This event includes data, demand, and prefetch (MMU type LD, ST, L1 Prefetch). There can be up to 4 tablewalks in one cycle.
Any PMC	0000009080	PM_SNOOP_TLBIE_ARB_CYC	Cycles the MMU is in the process of arbitrating an incoming TLBIE snoop, but the snoop has not yet acquired the MML lock.
Any PMC	0000009880	PM_SNOOP_TLBIE_TLB_INV_CYC	Cycles the MMU is in the process of walking the TLB for a TLBIE. From the time the probe is installed to the time the probe is uninstalled.
Any PMC	0000009084	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_HIT_CYC	Total duration of the snoop TLBIE when there is a context table hit, but a not-my-lpar (LSU is counting my-lpar case).
Any PMC	0000009884	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_MISS_CYC	Total duration of the snoop TLBIE when there is a context table miss.
Any PMC	0000009088	PM_SNOOP_TLBIE	Total number of TLBIE snoops
Any PMC	000000A084	PM_TLBIE_INV_CC_PRS0_COMPOSITE_CHILD_HASH	Congruence classes that found at least 1 invalidation match for a partition-scoped hash match.
Any PMC	000000A884	PM_TLBIE_INV_CC_PRS0_COMPOSITE_PARENT_HASH	Congruence classes that found at least 1 invalidation match for a partition-scoped parent hash match.
Any PMC	000000A088	PM_TLBIE_INV_ATTEMPT_CC_PRS0_COMPOSITE_CHILD	A partition-scoped child hash invalidation was sent to the TLB.
Any PMC	000000A888	PM_TLBIE_INV_ATTEMPT_CC_PRS0_COMPOSITE_PARENT	A partition-scoped parent hash invalidation was sent to the TLB.
Any PMC	000000A08C	PM_TLBIE_INV_ATTEMPT_HPT_PRECISE	Precise TLBIEs sent to the TLB in HPT.
Any PMC	000000A88C	PM_TLBIE_INV_CC_HPT_PRECISE	Precise TLBIEs sent to the TLB for HPT invalidations and had at least 1 matching entry in the congruence class.
Any PMC	000000A0A4	PM_CASE_A_SNOOP_TLBIE_HIT	A snoop hit on a case A context tag.
Any PMC	000000A8A4	PM_CASE_B_SNOOP_TLBIE_HIT	A snoop hit on a case B context tag.



Table E-30. MMU Events (Sheet 6 of 6)

PMC	Event Code	Event Name	Description
Any PMC	000000A0A8	PM_CASE_C_SNOOP_TLBIE_HIT	A snoop hit on a case C context tag.
Any PMC	000000A8A8	PM_CASE_D_SNOOP_TLBIE_HIT	A snoop hit on a case D context tag.
Any PMC	000000A0AC	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_HIT	A snoop found a not-my-lpar but found a context table hit.
Any PMC	000000A8AC	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_MISS	A snoop found a not-my-lpar but found a context table miss.
Any PMC	000000A0B0	PM_LPAR_SNOOP_HIT	A snoop found a "my-lpar" match.



### E.5.11 PMC Events

*Table E-31* lists the Power10 PMC events.

*Table E-31. PMC Events* (Sheet 1 of 3)

PMC	Event Code	Event Name	Description
PMC1	10000	PM_SUSPENDED	Counter off.
PMC1	10002	PM_INST_CMPL	PowerPC instruction completed.
PMC1	10008	PM_RUN_SPURR	SPURR bit 59 toggled.
PMC1	10010	PM_PMC4_OVERFLOW	The event selected for PMC4 caused the event counter to overflow.
PMC1	10020	PM_PMC4_REWIND	The speculative event selected for PMC4 rewinds and the counter for PMC4 is not charged.
PMC1	10022	PM_PMC2_SAVED	The conditions for the speculative event selected for PMC2 are met and PMC2 is charged.
PMC1	10024	PM_PMC5_OVERFLOW	The event selected for PMC5 caused the event counter to overflow.
PMC1	10026	PM_EXT_INT_EBB	Cycles an external interrupt was asserted due to an event-based branch. This means that the PMU interrupt is being handled by user code.
PMC1	1000A	PM_PMC3_REWIND	The speculative event selected for PMC3 rewinds and the counter for PMC3 is not charged.
PMC1	1001C	PM_ULTRAVISOR_INST_CMPL	PowerPC instruction completed while the thread was in ultrvisor state.
PMC1	1001E	PM_CYC	Processor cycles.
PMC1	1002A	PM_PMC3_HELD_CYC	Cycles when the speculative counter for PMC3 is frozen.
PMC1	1006C	PM_RUN_CYC_ST_MODE	Cycles when the run latch is set and the core is in ST mode.
PMC1	100F0	PM_CYC	Processor cycles.
PMC1	100FA	PM_RUN_LATCH_ANY_THREAD_CYC	Cycles when at least one thread has the run latch set.
PMC1	100FE	PM_INST_CMPL	PowerPC instruction completed.
PMC1	101E6	PM_THRESH_EXC_4096	Threshold counter exceeded a count of 4096.
PMC1	101E8	PM_THRESH_EXC_256	Threshold counter exceeded a count of 256.
PMC1	101EC	PM_THRESH_MET	Threshold exceeded.
PMC1	1C04E	PM_SIAR_LOADED	SIAR written per thread.
PMC1	1F15E	PM_MRK_START_PROBE_NOP_CMPL	Marked start probe nop (AND R0,R0,R0) completed.
PMC2	20000	PM_SUSPENDED	Counter off.
PMC2	20002	PM_INST_CMPL	PowerPC instruction completed.
PMC2	20010	PM_PMC1_OVERFLOW	The event selected for PMC1 caused the event counter to overflow.
PMC2	2000A	PM_HYPERVISOR_CYC	Cycles when the thread is in Hypervisor state. MSR[S, HV, PR] = 010.
PMC2	2000C	PM_RUN_LATCH_ALL_THREADS_CYC	Cycles when the run latch is set for all threads.
PMC2	2001E	PM_CYC	Processor cycles.
PMC2	2006C	PM_RUN_CYC_SMT4_MODE	Cycles when this thread's run latch is set and the core is in SMT4 mode.
PMC2	200F4	PM_RUN_CYC	Processor cycles gated by the run latch.



Table E-31. PMC Events (Sheet 2 of 3)

PMC	Event Code	Event Name	Description
PMC2	200F8	PM_EXT_INT	Cycles an external interrupt was active.
PMC2	200FE	PM_DATA_FROM_L2MISS	The processor's data cache was reloaded from a source other than the local core's L1 or L2 cache due to a demand miss.
PMC2	201E0	PM_MRK_DATA_FROM_MEMORY	The processor's data cache was reloaded from local, remote, or distant memory due to a demand miss for a marked load.
PMC2	201E4	PM_MRK_DATA_FROM_L3MISS	The processor's data cache was reloaded from a source other than the local core's L1, L2, or L3 cache due to a demand miss for a marked load
PMC2	201E6	PM_THRESH_EXC_32	Threshold counter exceeded a value of 32.
PMC2	201E8	PM_THRESH_EXC_512	Threshold counter exceeded a value of 512.
PMC2	2504C	PM_PMC4_HELD_CYC	Cycles when the speculative counter for PMC4 is frozen.
PMC2	2C04E	PM_SDAR_LOADED	SDAR writes per thread.
PMC2	2E016	PM_EXT_INT_HYP	Cycles an external interrupt was active that was initiated by the hypervisor.
PMC3	30000	PM_SUSPENDED	Counter off.
PMC3	30010	PM_PMC2_OVERFLOW	The event selected for PMC2 caused the event counter to overflow.
PMC3	30020	PM_PMC2_REWIND	The speculative event selected for PMC2 rewinds and the counter for PMC2 is not charged.
PMC3	30022	PM_PMC4_SAVED	The conditions for the speculative event selected for PMC4 are met and PMC4 is charged.
PMC3	30024	PM_PMC6_OVERFLOW	The event selected for PMC6 caused the event counter to overflow.
PMC3	3000C	PM_FREQ_DOWN	Power management: below threshold B.
PMC3	3001E	PM_CYC	Processor cycles.
PMC3	3006C	PM_RUN_CYC_SMT2_MODE	Cycles when this thread's run latch is set and the core is in SMT2 mode.
PMC3	3006E	PM_CONSTANT_CLK	This event increments at the 32 MHz rate of TOD_step.
PMC3	300F8	PM_TB_BIT_TRANS	Timebase event.
PMC3	300FE	PM_DATA_FROM_L3MISS	The processor's data cache was reloaded from a source other than the local core's L1, L2, or L3 cache due to a demand miss.
PMC3	301E8	PM_THRESH_EXC_64	Threshold counter exceeded a value of 64.
PMC3	301EA	PM_THRESH_EXC_1024	Threshold counter exceeded a value of 1024.
PMC3	3C04E	PM_PMC1_HELD_CYC	Cycles when the speculative counter for PMC1 is frozen.
PMC3	3D15E	PM_MULT_MRK	Mult marked instruction.
PMC3	3E050	PM_EXT_INT_OS	Cycles an external interrupt was active that was initiated by the operating system.
PMC4	40000	PM_SUSPENDED	Counter off.
PMC4	40002	PM_INST_CMPL	PowerPC instruction completed.
PMC4	40010	PM_PMC3_OVERFLOW	The event selected for PMC3 caused the event counter to overflow.
PMC4	40030	PM_INST_FIN	Instruction finished
PMC4	40114	PM_MRK_START_PROBE_NOP_DISP	Marked start probe NOP dispatched. Instruction AND R0,R0,R0.
PMC4	40134	PM_MRK_INST_TIMEOUT	Marked instruction finish timeout (instruction was lost).



Table E-31. PMC Events (Sheet 3 of 3)

PMC	Event Code	Event Name	Description
PMC4	4000A	PM_DEBUG_TRIGGER	Counts a trigger generated from the debug logic for lab use.
PMC4	4000C	PM_FREQ_UP	Power management: above threshold A.
PMC4	4001A	PM_SMT_MODE_SWITCH	Counts every time the SMT mode changes, SMT up or SMT down.
PMC4	4001E	PM_CYC	Processor cycles.
PMC4	400F4	PM_RUN_PURR	PURR bit 59 toggled.
PMC4	400FA	PM_RUN_INST_CMPL	PowerPC instruction completed while the run latch is set.
PMC4	400FE	PM_DATA_FROM_MEMORY	The processor's data cache was reloaded from local, remote, or distant memory due to a demand miss.
PMC4	4016E	PM_THRESH_NOT_MET	Threshold counter did not meet threshold.
PMC4	401E8	PM_MRK_DATA_FROM_L2MISS	The processor's data cache was reloaded from a source other than the local core's L1 or L2 cache due to a demand miss for a marked load.
PMC4	401EA	PM_THRESH_EXC_128	Threshold counter exceeded a value of 128.
PMC4	401EC	PM_THRESH_EXC_2048	Threshold counter exceeded a value of 2048.
PMC4	4C01C	PM_INT_DOORBELL	Cycles an internal doorbell interrupt was active.
PMC4	4C04A	PM_PMC2_HELD_CYC	Cycles when the speculative counter for PMC2 is frozen.
PMC4	4D010	PM_PMC1_SAVED	The conditions for the speculative event selected for PMC1 are met and PMC1 is charged.
PMC4	4D012	PM_PMC3_SAVED	The conditions for the speculative event selected for PMC3 are met and PMC3 is charged.
PMC4	4D022	PM_HYPERVISOR_INST_CMPL	PowerPC instruction completed while the thread was in hypervisor state.
PMC4	4D024	PM_PROBLEM_INST_CMPL	PowerPC instruction completed while the thread was in problem state.
PMC4	4D026	PM_ULTRAVISOR_CYC	Cycles when the thread is in ultrvisor state. MSR[S, HV, PR] = 110.
PMC4	4D028	PM_PRIVILEGED_CYC	Cycles when the thread is in privileged state. MSR[S HV, PR] = x00.
PMC4	4D02C	PM_PMC1_REWIND	The speculative event selected for PMC1 rewinds and the counter for PMC1 is not charged.
Any PMC	00000020B0	PM_START_PROBE_NOP_DISP	A start probe NOP was dispatched. Instruction AND 0,0,0.
Any PMC	00000021B4	PM_MRK_STOP_PROBE_NOP_DISP	A stop probe NOP was marked at dispatch. Instruction AND 1,1,1.
Any PMC	00000028B0	PM_STOP_PROBE_NOP_DISP	A stop probe NOP was dispatched. Instruction AND 1,1,1.

### E.5.12 Reload Source Events

The PMU provides a set of reload events that can be programmed to count for data reloads, instruction reloads, data page table reloads, and instruction page table reloads from the source specified by the user. The source and reload type can be programmed by setting the MMCR3 bits in the select event code formation using the data cache encoding table (see *Table E-32*).

*Table E-32. Source Encodings* (Sheet 1 of 2)

Performance Monitor Reload Bus Source Encoding (Binary)	Implementation-Dependent Bits Description
L2 Hit	
000000	Private L2 cache for this core sourced data (or NCU loads) without dispatch conflicts.
000001	Private L2 cache for this core sourced data in the Mepf state without dispatch conflicts. (L3 prefetch brought line in Me.)
000010	Private L2 cache for this core sourced data that had a dispatch conflict on load-hit-store.
000011	Private L2 cache for this core sourced data that had a dispatch conflict other than load-hit-store.
L3 Hit	
000100	Private L3 cache for this core sourced data without dispatch conflicts.
000101	Private L3 cache for this core sourced data in the Mepf state without dispatch conflicts. (L3 prefetch brought line in Me.)
000111	Private L3 cache for this core sourced data that had a dispatch conflict.
Memory	
100101	Local (on-chip) memory controller.
100110	Local OpenCAPP cache.
100111	Local OpenCAPP memory.
110101	Remote (same group, off-chip) memory controller.
110110	Remote OpenCAPP cache.
110111	Remote OpenCAPP memory.
111101	Distant (outside group, off-chip) memory controller.
111110	Distant OpenCAPP cache.
111111	Distant OpenCAPP memory.
On-Chip Cache	
100000	Data sourced from another core's L2 on the same chip in the same regent in valid, but non-exclusive state.
100001	Data sourced from another core's L2 on the same chip in the same regent in exclusive state.
100010	Data sourced from another core's L3 on the same chip in the same regent in valid, but non-exclusive state.
100011	Data sourced from another core's L3 on the same chip in the same regent in exclusive state.
101000	Data sourced from another core's L2 on the same chip outside the regent domain in valid, but non-exclusive state.
101001	Data sourced from another core's L2 on the same chip outside the regent domain in exclusive state.



Table E-32. Source Encodings (Sheet 2 of 2)

Performance Monitor Reload Bus Source Encoding (Binary)	Implementation-Dependent Bits Description
101010	Data sourced from another core's L3 on the same chip outside the regent domain in valid, but non-exclusive state.
101011	Data sourced from another core's L3 on the same chip outside the regent domain in exclusive state.
Off-Chip Cache	
110000	Data sourced from a remote (same group, off-chip) core's L2 in valid, but non-exclusive state.
110010	Data sourced from a remote (same group, off-chip) core's L3 in valid, but non-exclusive state.
110001	Data sourced from a remote (same group, off-chip) core's L2 in exclusive state.
110011	Data sourced from a remote (same group, off-chip) core's L3 in exclusive state.
111000	Data sourced from a distant (outside group, off-chip) core's L2 in valid, but non-exclusive state.
111010	Data sourced from a distant (outside group, off-chip) core's L3 in valid, but non-exclusive state.
111001	Data sourced from a distant (outside group, off-chip) core's L2 in exclusive state.
111011	Data sourced from a distant (outside group, off-chip) core's L3 in exclusive state.

Table E-33 lists the Power10 events related to reloads.

Table E-33. Reload Events (Sheet 1 of 2)

PMC	Event Code	Event Name	Description
PMC1	1C040	PM_XFER_FROM_SRC_PMC1	The processor's L1 data cache was reloaded from the source specified in MMCR3[0:14]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC1	1C142	PM_MRK_XFER_FROM_SRC_PMC1	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[0:14]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC1	1C144	PM_MRK_XFER_FROM_SRC_CYC_PMC1	Cycles between the first time a marked instruction misses in the L1 cache until a line is reloaded from the source specified in MMCR3[0:14]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads. Divide this count by PM_MRK_XFER_FROM_SRC_PMC1 when PM_MRK_XFER_FROM_SRC_PMC1 is set with the same reload source, to obtain the average latency for the specified source.
PMC2	2C040	PM_XFER_FROM_SRC_PMC2	The processor's L1 data cache was reloaded from the source specified in MMCR3[15:29]. If MMCR1[16 17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.



Table E-33. Reload Events (Sheet 2 of 2)

PMC	Event Code	Event Name	Description
PMC2	2C142	PM_MRK_XFER_FROM_SRC_PMC2	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[15:29]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC2	2C144	PM_MRK_XFER_FROM_SRC_CYC_PMC2	Cycles between the first time a marked instruction misses in the L1 cache until a line is reloaded from the source specified in MMCR3[15:29]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads. Divide this count by PM_MRK_XFER_FROM_SRC_PMC2 when PM_MRK_XFER_FROM_SRC_PMC2 is set with the same reload source, to obtain the average latency for the specified source.
PMC3	3C040	PM_XFER_FROM_SRC_PMC3	The processor's L1 data cache was reloaded from the source specified in MMCR3[30:44]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC3	3C142	PM_MRK_XFER_FROM_SRC_PMC3	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[30:44]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is 1, this count includes both demand misses and prefetch reloads.
PMC3	3C144	PM_MRK_XFER_FROM_SRC_CYC_PMC3	Cycles between the first time a marked instruction misses in the L1 cache until a line is reloaded from the source specified in MMCR3[30:44]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads. Divide this count by PM_MRK_XFER_FROM_SRC_PMC3 when PM_MRK_XFER_FROM_SRC_PMC3 is set with the same reload source, to obtain the average latency for the specified source.
PMC4	4C040	PM_XFER_FROM_SRC_PMC4	The processor's L1 data cache was reloaded from the source specified in MMCR3[45:59]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC4	4C142	PM_MRK_XFER_FROM_SRC_PMC4	For a marked data transfer instruction, the processor's L1 data cache was reloaded from the source specified in MMCR3[45:59]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads.
PMC4	4C144	PM_MRK_XFER_FROM_SRC_CYC_PMC4	Cycles between the first time a marked instruction misses in the L1 cache until a line is reloaded from the source specified in MMCR3[45:59]. If MMCR1[16 17] is '0' (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16 17] is '1', this count includes both demand misses and prefetch reloads. Divide this count by PM_MRK_XFER_FROM_SRC_PMC4 when PM_MRK_XFER_FROM_SRC_PMC4 is set with the same reload source, to obtain the average latency for the specified source.



### E.5.13 Radix Events

Table E-34 lists the Power10 radix events.

**Table E-34. Radix Events (Sheet 1 of 10)**

PMC	Event Code	Event Name	Description
PMC1	14042	PM_DATA_RADIX_L2_PTE_FROM_L2	A data page table entry was reloaded to a level-2 page walk cache from the core's L2 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	14044	PM_DATA_RADIX_L2_PDE_FROM_L2	A data page directory entry was reloaded to a level-2 page walk cache from the core's L2 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	14046	PM_DATA_RADIX_L3_PTE_FROM_L2	A data page table entry was reloaded to a level-3 page walk cache from the core's L2 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	14048	PM_DATA_RADIX_L3_PDE_FROM_L2	A data page directory entry was reloaded to a level-3 page walk cache from the core's L2 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	1404A	PM_DATA_RADIX_L4_PTE_FROM_L2	A data page table entry was reloaded to a level-4 page walk cache from the core's L2 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	1404E	PM_INST_RADIX_L2_PTE_FROM_L2	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L2 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.



Table E-34. Radix Events (Sheet 2 of 10)

PMC	Event Code	Event Name	Description
PMC1	15042	PM_INST_RADIX_L3_PTE_FROM_L2	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L2 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	15044	PM_INST_RADIX_L3_PDE_FROM_L2	An instruction page directory entry was reloaded to a level-3 page walk cache from the core's L2 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	15046	PM_INST_RADIX_L4_PTE_FROM_L2	An instruction page table entry was reloaded to a level-4 page walk cache from the core's L2 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	24042	PM_DATA_RADIX_L2_PTE_FROM_L3	A data page table entry was reloaded to a level-2 page walk cache from the core's L3 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	24044	PM_DATA_RADIX_L2_PDE_FROM_L3	A data page directory entry was reloaded to a level-2 page walk cache from the core's L3 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	24046	PM_DATA_RADIX_L3_PTE_FROM_L3	A data page table entry was reloaded to a level-3 page walk cache from the core's L3 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	24048	PM_DATA_RADIX_L3_PDE_FROM_L3	A data page directory entry was reloaded to a level-3 page walk cache from the core's L3 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.



Table E-34. Radix Events (Sheet 3 of 10)

PMC	Event Code	Event Name	Description
PMC2	2404A	PM_DATA_RADIX_L4_PTE_FROM_L3	A data page table entry was reloaded to a level-4 page walk cache from the core's L3 cache. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	2404E	PM_INST_RADIX_L2_PTE_FROM_L3	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L3 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	25042	PM_INST_RADIX_L3_PTE_FROM_L3	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L3 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	25044	PM_INST_RADIX_L3_PDE_FROM_L3	An instruction page directory entry was reloaded to a level-3 page walk cache from the core's L3 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC2	25046	PM_INST_RADIX_L4_PTE_FROM_L3	An instruction page table entry was reloaded to a level-4 page walk cache from the core's L3 cache. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	3404A	PM_DATA_RADIX_L2_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	3404C	PM_DATA_RADIX_L2_PDE_FROM_L3MISS	A data page directory entry was reloaded to a level-2 page walk cache from a source beyond the core's caches. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.

Table E-34. Radix Events (Sheet 4 of 10)

PMC	Event Code	Event Name	Description
PMC3	3404E	PM_DATA_RADIX_L3_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	35040	PM_DATA_RADIX_L3_PDE_FROM_L3MISS	A data page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	35042	PM_DATA_RADIX_L4_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	35046	PM_INST_RADIX_L2_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	3504A	PM_INST_RADIX_L3_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	3504C	PM_INST_RADIX_L3_PDE_FROM_L3MISS	An instruction page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.



Table E-34. Radix Events (Sheet 5 of 10)

PMC	Event Code	Event Name	Description
PMC3	3504E	PM_INST_RADIX_L4_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC3	3F04C	PM_RADIX_RELOAD	A page was reloaded for a radix translation in the ERAT or TLB.
PMC4	4404A	PM_DATA_RADIX_L2_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-2 page walk cache from distant memory. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	4404C	PM_DATA_RADIX_L2_PDE_FROM_DISTANT	A data page directory entry was reloaded to a level-2 page walk cache from distant memory. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	4404E	PM_DATA_RADIX_L3_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-3 page walk cache from distant memory. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	45040	PM_DATA_RADIX_L3_PDE_FROM_DISTANT	A data page directory entry was reloaded to a level-3 page walk cache from distant memory. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	45042	PM_DATA_RADIX_L4_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-4 page walk cache from distant memory. If MMCR1[16] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[16] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	45046	PM_INST_RADIX_L2_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-2 page walk cache from distant memory. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.



Table E-34. Radix Events (Sheet 6 of 10)

PMC	Event Code	Event Name	Description
PMC4	4504A	PM_INST_RADIX_L3_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-3 page walk cache from distant memory. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	4504C	PM_INST_RADIX_L3_PDE_FROM_DISTANT	An instruction page directory entry was reloaded to a level-3 page walk cache from distant memory. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC4	4504E	PM_INST_RADIX_L4_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-4 page walk cache from distant memory. If MMCR1[17] is 0 (default), this count includes only lines that were reloaded to satisfy a demand miss. If MMCR1[17] is 1, this count includes both demand miss and prefetch reloads. Further, if MMCR1[18] is 0 (default), only partition-scoped accesses are included. If MMCR1[18] is 1, only process-scoped accesses are included in this count. Only valid for radix tests. Not valid for HPT tests or radix and HPT tests.
PMC1	000000000014242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L2	A data page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000014244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L2	A data page directory entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000014246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L2	A data page table entry was reloaded to a level-3 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000014248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L2	A data page directory entry was reloaded to a level-3 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	00000000001424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L2	A data page table entry was reloaded to a level-4 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	00000000001424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L2	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000015242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L2	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000015244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L2	An instruction page directory entry was reloaded to a level-3 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC1	000000000015246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L2	An instruction page table entry was reloaded to a level-4 page walk cache from the core's L2 cache due to a process-scoped demand miss.
PMC2	000000000024242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3	A data page table entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	000000000024244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3	A data page directory entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand miss.



*Table E-34. Radix Events (Sheet 7 of 10)*

PMC	Event Code	Event Name	Description
PMC2	000000000024246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3	A data page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	000000000024248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3	A data page directory entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	00000000002424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3	A data page table entry was reloaded to a level-4 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	00000000002424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	000000000025242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	000000000025244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3	An instruction page directory entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC2	000000000025246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3	An instruction page table entry was reloaded to a level-4 page walk cache from the core's L3 data cache due to a process-scoped demand miss.
PMC3	00000000003434A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	00000000003424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3MISS	A data page directory entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	00000000003424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	000000000035240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3MISS	A data page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	000000000035242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3MISS	A data page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	000000000035246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	00000000003524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	00000000003524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3MISS	An instruction page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC3	00000000003524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3MISS	An instruction page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches due to a process-scoped demand miss.
PMC4	00000000004424A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand miss.



Table E-34. Radix Events (Sheet 8 of 10)

PMC	Event Code	Event Name	Description
PMC4	00000000004424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_DISTANT	A data page directory entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	00000000004424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	000000000045240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_DISTANT	A data page directory entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	000000000045242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_DISTANT	A data page table entry was reloaded to a level-4 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	000000000045246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	00000000004524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	00000000004524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_DISTANT	An instruction page directory entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand miss.
PMC4	00000000004524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_DISTANT	An instruction page table entry was reloaded to a level-4 page walk cache from distant memory due to a process-scoped demand miss.
PMC1	000000000214242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L2_ALL	A data page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000214244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L2_ALL	A data page directory entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000214246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L2_ALL	A data page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000214248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L2_ALL	A data page directory entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	00000000021424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L2_ALL	A data page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	00000000011424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L2_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000115242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L2_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000115244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L2_ALL	An instruction page directory entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC1	000000000115246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L2_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L2 cache due to a process-scoped demand or prefetch miss.
PMC2	000000000224242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3_ALL	A data page table entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	000000000224244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3_ALL	A data page directory entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.



Table E-34. Radix Events (Sheet 9 of 10)

PMC	Event Code	Event Name	Description
PMC2	000000000224246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3_ALL	A data page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	000000000224248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3_ALL	A data page directory entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	00000000022424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3_ALL	A data page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	00000000012424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	000000000125242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3_ALL	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	000000000125244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3_ALL	An instruction page directory entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC2	000000000125246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3_ALL	An instruction page table entry was reloaded to a level-3 page walk cache from the core's L3 data cache due to a process-scoped demand or prefetch miss.
PMC3	00000000023434A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3MISS_ALL	A data page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	00000000023424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3MISS_ALL	A data page directory entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	00000000023424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3MISS_ALL	A data page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	000000000235240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3MISS_ALL	A data page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	000000000235242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3MISS_ALL	A data page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	000000000135246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3MISS_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	00000000013524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3MISS_ALL	An instruction page table entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	00000000013524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3MISS_ALL	An instruction page directory entry was reloaded to a level-3 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC3	00000000013524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3MISS_ALL	An instruction page table entry was reloaded to a level-4 page walk cache from a source beyond the core's caches due to a process-scoped demand or prefetch miss.
PMC4	00000000024424A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_DISTANT_ALL	A data page table entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand or prefetch miss.



Table E-34. Radix Events (Sheet 10 of 10)

PMC	Event Code	Event Name	Description
PMC4	0000000024424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_DISTANT_ALL	A data page directory entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	0000000024424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_DISTANT_ALL	A data page table entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	00000000245240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_DISTANT_ALL	A data page directory entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	00000000245242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_DISTANT_ALL	A data page table entry was reloaded to a level-4 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	00000000145246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_DISTANT_ALL	An instruction page table entry was reloaded to a level-2 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	0000000014524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_DISTANT_ALL	An instruction page table entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	0000000014524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_DISTANT_ALL	An instruction page directory entry was reloaded to a level-3 page walk cache from distant memory due to a process-scoped demand or prefetch miss.
PMC4	0000000014524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_DISTANT_ALL	An instruction page table entry was reloaded to a level-4 page walk cache from distant memory due to a process-scoped demand or prefetch miss.



## E.5.14 Metrics

*Appendix E.5.14.1* list the metrics by category (group) and *Appendix E.5.14.2 Power10 Metric Events and Formulas* on page 636 lists the detailed Power10 metric events and formulas.

### E.5.14.1 Power10 Events by Group

Table E-35. Power10 Events by Group (Sheet 1 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
0	PM_STAT	10008	PM_RUN_SPURR
		20002	PM_INST_CMPL
		3001E	PM_CYC
		400F4	PM_RUN_PURR
1	PM_CPI_1	1003A	PM_DISP_STALL_BR_MPRED_IC_L2
		2C01E	PM_DISP_STALL_BR_MPRED_IC_L3
		34058	PM_DISP_STALL_BR_MPRED_ICMISS
		4C010	PM_DISP_STALL_BR_MPRED_IC_L3MISS
2	PM_CPI_2	10064	PM_DISP_STALL_IC_L2
		2D01A	PM_DISP_STALL_IC_MISS
		3E052	PM_DISP_STALL_IC_L3
		4E010	PM_DISP_STALL_IC_L3MISS
3	PM_CPI_3	10038	PM_DISP_STALL_TRANSLATION
		2C016	PM_DISP_STALL_IERAT_ONLY_MISS
		3000A	PM_DISP_STALL_ITLB_MISS
		4D01E	PM_DISP_STALL_BR_MPRED
4	PM_CPI_4	1E050	PM_DISP_STALL_HELD_STF_MAPPER_CYC
		2E01A	PM_DISP_STALL_HELD_XVFC_MAPPER_CYC
		3D05C	PM_DISP_STALL_HELD_RENAME_CYC
		4003C	PM_DISP_STALL_HELD_SYNC_CYC
5	PM_CPI_5	10006	PM_DISP_STALL_HELD_OTHER_CYC
		20000	PM_SUSPENDED
		30018	PM_DISP_STALL_HELD_SCOREBOARD_CYC
		4E01A	PM_DISP_STALL_HELD_CYC
6	PM_CPI_6	1D05E	PM_DISP_STALL_HELD_HALT_CYC
		20000	PM_SUSPENDED
		30000	PM_SUSPENDED
		40000	PM_SUSPENDED



Table E-35. Power10 Events by Group (Sheet 2 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
7	PM_CPI_7	100F8	PM_DISP_STALL_CYC
		2E018	PM_DISP_STALL_FETCH
		30004	PM_DISP_STALL_FLUSH
		40000	PM_SUSPENDED
8	PM_CPI_8	1E056	PM_EXEC_STALL_STORE_PIPE
		2E01C	PM_EXEC_STALL_TLBIE
		30026	PM_EXEC_STALL_STORE_MISS
		4D016	PM_EXEC_STALL_PTESYNC
9	PM_CPI_9	1E054	PM_EXEC_STALL_DMISS_L21_L31
		2C01C	PM_EXEC_STALL_DMISS_OFF_CHIP
		30038	PM_EXEC_STALL_DMISS_LMEM
		4C01A	PM_EXEC_STALL_DMISS_OFF_NODE
10	PM_CPI_10	1003C	PM_EXEC_STALL_DMISS_L2L3
		2C018	PM_EXEC_STALL_DMISS_L3MISS
		34054	PM_EXEC_STALL_DMISS_L2L3_NOCONFLICT
		4C016	PM_EXEC_STALL_DMISS_L2L3_CONFLICT
11	PM_CPI_11	10000	PM_SUSPENDED
		2C010	PM_EXEC_STALL_LSU
		30014	PM_EXEC_STALL_STORE
		4D014	PM_EXEC_STALL_LOAD
12	PM_CPI_12	10004	PM_EXEC_STALL_TRANSLATION
		2D018	PM_EXEC_STALL_VSU
		30016	PM_EXEC_STALL_DERAT_DTLB_MISS
		4C012	PM_EXEC_STALL_DERAT_ONLY_MISS
13	PM_CPI_13	10058	PM_EXEC_STALL_FIN_AT_DISP
		2E01E	PM_EXEC_STALL_NTC_FLUSH
		30036	PM_EXEC_STALL_SIMPLE_FX
		4D018	PM_EXEC_STALL_BRU
14	PM_CPI_14	100F2	PM_1PLUS_PPC_CMPL
		20006	PM_DISP_STALL_HELD_ISSQ_FULL_CYC
		30008	PM_EXEC_STALL
		4001E	PM_CYC
15	PM_CPI_15	10000	PM_SUSPENDED
		2405A	PM_NTC_FIN
		30014	PM_EXEC_STALL_STORE
		4D01C	PM_EXEC_STALL_TLBIEL



Table E-35. Power10 Events by Group (Sheet 3 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
16	PM_CPI_16	10000	PM_SUSPENDED
		20004	PM_ISSUE_STALL
		34056	PM_EXEC_STALL_LOAD_FINISH
		40002	PM_INST_CMPL
17	PM_CPI_17	1E05A	PM_CMPL_STALL_LWSYNC
		2D01C	PM_CMPL_STALL_STCX
		3003A	PM_CMPL_STALL_EXCEPTION
		4C018	PM_CMPL_STALL
18	PM_CPI_18	10000	PM_SUSPENDED
		2C014	PM_CMPL_STALL_SPECIAL
		30028	PM_CMPL_STALL_MEM_ECC
		4D01A	PM_CMPL_STALL_HWSYNC
19	PM_L3_1	100000016080	PM_L3_PF_MISS_L3
		100000026080	PM_L3_CO_MEM
		100000036080	PM_L3_PF_ON_CHIP_CACHE
		100000046080	PM_L3_PF_ON_CHIP_MEM
20	PM_L3_2	100000016880	PM_L3_CO_MEFP
		100000026880	PM_L3_CO_L31
		100000036880	PM_L3_PF_OFF_CHIP_CACHE
		100000046880	PM_L3_PF_OFF_CHIP_MEM
21	PM_L3_3	10000	PM_SUSPENDED
		110000026080	PM_L3_CI_HIT
		110000036080	PM_L3_L2_CO_HIT
		110000046080	PM_L3_LAT_CI_HIT
22	PM_L3_4	10000	PM_SUSPENDED
		110000026880	PM_L3_CI_MISS
		110000036880	PM_L3_L2_CO_MISS
		110000046880	PM_L3_LAT_CI_MISS
23	PM_L3_5	120000016080	PM_L3_HIT
		120000026080	PM_L3_LD_HIT
		120000036080	PM_L3_CO_LCO
		40000	PM_SUSPENDED
24	PM_L3_6	120000016880	PM_L3_MISS
		120000026880	PM_L3_LD_MISS
		120000036880	PM_L3_CINJ
		120000046880	PM_L3_TRANS_PF



Table E-35. Power10 Events by Group (Sheet 4 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
25	PM_L3_7	130000016080	PM_TM_SC_CO
		130000026080	PM_NON_TM_RST_SC
		130000036080	PM_SNP_TM_HIT_M
		130000046080	PM_RD_FORMING_SC
26	PM_L3_8	130000016880	PM_TM_CAM_OVERFLOW
		130000026880	PM_TM_RST_SC
		130000036880	PM_SNP_TM_HIT_T
		130000046880	PM_RD_CLEARING_SC
27	PM_L3_9	10000	PM_SUSPENDED
		140000026080	PM_L3_PF_HIT_L3
		140000036080	PM_L3_CO
		140000046080	PM_SN_HIT
28	PM_L3_10	140000016880	PM_L3_WI_USAGE
		140000026880	PM_RD_HIT_PF
		140000036880	PM_SN_INVL
		140000046880	PM_SN_MISS
29	PM_L3_11	150000016080	PM_L3_P0_LCO_NO_DATA
		150000026080	PM_L3_P0_LCO_DATA
		150000036080	PM_L3_P0_CO_MEM
		150000046080	PM_L3_P0_CO_L31
30	PM_L3_12	150000016880	PM_L3_P1_LCO_NO_DATA
		150000026880	PM_L3_P1_LCO_DATA
		150000036880	PM_L3_P1_CO_MEM
		150000046880	PM_L3_P1_CO_L31
31	PM_L3_13	160000016080	PM_L3_SN_USAGE
		160000026080	PM_L3_PF_USAGE
		160000036080	PM_L3_SN0_BUSY
		160000046080	PM_L3_SN0_BUSY
32	PM_L3_14	160000016880	PM_L3_CI_USAGE
		160000026880	PM_L3_RD_USAGE
		160000036880	PM_L3_CO0_BUSY
		160000046880	PM_L3_CO0_BUSY
33	PM_L3_15	170000016080	PM_L3_P0_PF_RTY
		170000026080	PM_L3_P2_PF_RTY
		170000036080	PM_L3_P0_CO_RTY
		170000046080	PM_L3_P2_CO_RTY



Table E-35. Power10 Events by Group (Sheet 5 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
34	PM_L3_16	170000016880	PM_L3_P1_PF_RTY
		170000026880	PM_L3_P3_PF_RTY
		170000036880	PM_L3_P1_CO_RTY
		170000046880	PM_L3_P3_CO_RTY
35	PM_L3_17	180000016080	PM_L3_P0_NODE_PUMP
		180000026080	PM_L3_P0_GRP_PUMP
		180000036080	PM_L3_P0_SYS_PUMP
		40000	PM_SUSPENDED
36	PM_L3_18	180000016880	PM_L3_P1_NODE_PUMP
		180000026880	PM_L3_P1_GRP_PUMP
		180000036880	PM_L3_P1_SYS_PUMP
		40000	PM_SUSPENDED
37	PM_L3_19	190000016080	PM_L3_LOC_GUESS_CORRECT
		190000026080	PM_L3_SYS_GUESS_CORRECT
		190000036080	PM_L3_GRP_GUESS_WRONG_LOW
		190000046080	PM_L3_SYS_GUESS_WRONG
38	PM_L3_20	190000016880	PM_L3_GRP_GUESS_CORRECT
		190000026880	PM_L3_LOC_GUESS_WRONG
		190000036880	PM_L3_GRP_GUESS_WRONG_HIGH
		40000	PM_SUSPENDED
39	PM_L3_21	1A0000016080	PM_L3_P0_LCO_RTY
		1A0000026080	PM_L3_P2_LCO_RTY
		1A0000036080	PM_L3_PF0_BUSY
		1A0000046080	PM_L3_PF0_BUSY
40	PM_L3_22	1A0000016880	PM_L3_P1_LCO_RTY
		1A0000026880	PM_L3_P3_LCO_RTY
		1A0000036880	PM_L3_RD0_BUSY
		1A0000046880	PM_L3_RD0_BUSY
41	PM_L3_23	1B0000016080	PM_L3_WI0_BUSY
		1B0000026080	PM_L3_WI0_BUSY
		1B0000036080	PM_L3_PF_MPRED_DUALMCMDATA
		1B0000046080	PM_L3_PF_CPRED_DUALMCMDATA
42	PM_L3_24	10000	PM_SUSPENDED
		20000	PM_SUSPENDED
		1B0000036880	PM_L3_CO_L31_INST_XLAT
		40000	PM_SUSPENDED



Table E-35. Power10 Events by Group (Sheet 6 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
43	PM_L2_1	000000016080	PM_L2_LD
		000000026080	PM_L2_LD_MISS
		000000036080	PM_L2_INST
		000000046080	PM_L2_ALL_MISS
44	PM_L2_2	000000016880	PM_L2_ST
		000000026880	PM_L2_ST_MISS
		000000036880	PM_L2_INST_MISS
		000000046880	PM_L2_ISIDE_MRU_TOUCH
45	PM_L2_3	010000016080	PM_L2_CASTOUT_MOD
		010000026080	PM_L2_IC_INV
		010000036080	PM_L2_ISIDE_DSIDE
		010000046080	PM_L2_ST
46	PM_L2_4	010000016880	PM_L2_CASTOUT_SHR
		010000026880	PM_L2_DC_INV
		010000036880	PM_L2_ISIDE_DSIDE_HIT
		010000046880	PM_L2_ST_HIT
47	PM_L2_5	020000016080	PM_L2_ISIDE_DSIDE_ATTEMPT
		020000026080	PM_L2_ISIDE_DSIDE_FAIL_OTHER
		020000036080	PM_L2_ST_ATTEMPT
		020000046080	PM_L2_ST_DISP_FAIL_OTHER
48	PM_L2_6	020000016880	PM_L2_ISIDE_DSIDE_FAIL_ADDR
		20000	PM_SUSPENDED
		020000036880	PM_L2_ST_DISP_FAIL_ADDR
		40000	PM_SUSPENDED
49	PM_L2_7	030000016080	PM_L2_SN_M_WR_DONE
		030000026080	PM_L2_CO_TM_SC_FOOTPRINT
		030000036080	PM_L2_RC_ST_DONE
		030000046080	PM_L2_SN_M_RD_DONE
50	PM_L2_8	030000016880	PM_CO_DISP_FAIL
		20000	PM_SUSPENDED
		030000036880	PM_L2_SN_SX_I_DONE
		030000046880	PM_L2_SN_M_WR_DONE
51	PM_L2_9	040000016080	PM_L2_LOC_GUESS_CORRECT
		040000026080	PM_L2_GRP_GUESS_CORRECT
		040000036080	PM_L2_SYS_GUESS_CORRECT
		040000046080	PM_L2_CHIP_PUMP



Table E-35. Power10 Events by Group (Sheet 7 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
52	PM_L2_10	040000016880	PM_L2_LOC_GUESS_WRONG
		040000026880	PM_L2_GRP_GUESS_WRONG
		040000036880	PM_L2_SYS_GUESS_WRONG
		040000046880	PM_L2_GROUP_PUMP
53	PM_L2_11	050000016080	PM_L2_ST_ALL
		050000026080	PM_ISIDE_FAIL_ADDR
		050000036080	PM_L2_RTY_ST
		050000046080	PM_L2_ST_GATHER_ALL
54	PM_L2_12	050000016880	PM_ISIDE_ATTEMPT
		050000026880	PM_ISIDE_FAIL_OTHER
		050000036880	PM_L2_RTY_LD
		050000046880	PM_L2_SYS_PUMP
55	PM_L2_13	060000016080	PM_L2_RC0_BUSY
		060000026080	PM_L2_CO0_BUSY
		060000036080	PM_L2_CO0_BUSY
		060000046080	PM_L2_CO0_BUSY
56	PM_L2_14	060000016880	PM_L2_RC_USAGE
		060000026880	PM_L2_CO_USAGE
		060000036880	PM_L2_SN_USAGE
		40000	PM_SUSPENDED
57	PM_L2_15	070000016080	PM_L2_ST CAUSED_TM_FAIL
		070000026080	PM_TM_LD_FAIL
		070000036080	PM_TM_ST_FAIL
		070000046080	PM_TM_FOOTPR_OVERFLOW
58	PM_L2_16	070000016880	PM_L2_LD CAUSED_TM_FAIL
		070000026880	PM_TM_FAV CAUSED_FAIL
		070000036880	PM_TM_ST CAUSED_FAIL
		40000	PM_SUSPENDED
59	PM_L2_17	080000016080	PM_L2_SN0_BUSY
		080000026080	PM_L2_SN0_BUSY
		080000036080	PM_L2_DSIDE_READ
		080000046080	PM_DUALMCDATA_REQ_USE_INTV_DATA
60	PM_L2_18	080000016880	PM_L2_L1PF_READ
		080000026880	PM_L2_ISIDE_READ
		080000036880	PM_DUALMCDATA_REQ_USE_LPC_DATA
		40000	PM_SUSPENDED



Table E-35. Power10 Events by Group (Sheet 8 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
61	PM_L2_19	090000016080	PM_L2_ISIDE_DSIDE_ST_ATTEMPT
		090000026080	PM_L2_ISIDE_DSIDE_ST_SUCCESS
		090000036080	PM_L2_RC_CACHE_READ
		090000046080	PM_L2_CO_CACHE_READ
62	PM_L2_21	0A0000016080	PM_L2_TM_FOOTPR_OVERFLOW_LD
		0A0000026080	PM_L2_TM_FOOTPR_OVERFLOW_ST
		0A0000036080	PM_L2_TM_FOOTPR_OVERFLOW_LD_MULTI
		0A0000046080	PM_L2_TM_FOOTPR_OVERFLOW_ST_MULTI
63	PM_L2_22	0A0000016880	PM_L2_TM_END
		0A0000026880	PM_L2_TM_FOOTPR_LD_LINES
		0A0000036880	PM_L2_TM_FOOTPR_ST_LINES
		40000	PM_SUSPENDED
64	PM_L2_23	0B0000016080	PM_L2_TLBIE_SLBIE_START
		0B0000026080	PM_L2_SNP_TLBIE_SLBIE_START
		0B0000036080	PM_NCU_SNP_TLBIE_CYC
		40000	PM_SUSPENDED
65	PM_L2_24	0B0000016880	PM_L2_TLBIE_SLBIE_DELAY
		0B0000026880	PM_L2_SNP_TLBIE_SLBIE_DELAY
		30000	PM_SUSPENDED
		40000	PM_SUSPENDED
66	PM_L2_25	0C0000016080	PM_ST_DATA_FROM_L2
		0C0000026080	PM_ST_DATA_FROM_L21_L31
		0C0000036080	PM_ST_DATA_FROM_RL2L3
		0C0000046080	PM_ST_DATA_FROM_DL2L3
67	PM_L2_26	0C0000016880	PM_ST_DATA_FROM_L3
		0C0000026880	PM_ST_DATA_FROM_LMEM
		0C0000036880	PM_ST_DATA_FROM_RMEM
		0C0000046880	PM_ST_DATA_FROM_DMEM
68	PM_L2_31	0F0000016080	PM_L2_LD_DISP
		0F0000026080	PM_L2_LD_HIT
		30000	PM_SUSPENDED
		0F0000046080	PM_L2_INST_MISS
69	PM_L2_32	0F0000016880	PM_L2_ST_DISP
		0F0000026880	PM_L2_ST_HIT
		30000	PM_SUSPENDED
		0F0000046880	PM_L2_RTY_ST



Table E-35. Power10 Events by Group (Sheet 9 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
70	PM_RADIX_DS_1	000000000014242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L2
		000000000024242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3
		00000000003434A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3MISS
		00000000004424A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_DISTANT
71	PM_RADIX_DS_2	000000000014244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L2
		000000000024244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3
		00000000003424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3MISS
		00000000004424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_DISTANT
72	PM_RADIX_DS_3	000000000014246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L2
		000000000024246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3
		00000000003424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3MISS
		00000000004424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_DISTANT
73	PM_RADIX_DS_4	000000000014248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L2
		000000000024248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3
		000000000035240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3MISS
		000000000045240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_DISTANT
74	PM_RADIX_DS_5	00000000001424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L2
		00000000002424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3
		000000000035242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3MISS
		000000000045242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_DISTANT
75	PM_RADIX_DS_6	00000000001424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L2
		00000000002424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3
		000000000035246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3MISS
		000000000045246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_DISTANT
76	PM_RADIX_DS_7	000000000015242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L2
		000000000025242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3
		00000000003524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3MISS
		00000000004524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_DISTANT
77	PM_RADIX_DS_8	000000000015244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L2
		000000000025244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3
		00000000003524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3MISS
		00000000004524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_DISTANT
78	PM_RADIX_DS_9	000000000015246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L2
		000000000025246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3
		00000000003524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3MISS
		00000000004524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_DISTANT



Table E-35. Power10 Events by Group (Sheet 10 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
79	PM_RADIX_DS_10	000000000214242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L2_ALL
		000000000224242	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3_ALL
		00000000023434A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_L3MISS_ALL
		00000000024424A	PM_DATA_RADIX_PROCESS_L2_PTE_FROM_DISTANT_ALL
80	PM_RADIX_DS_11	000000000214244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L2_ALL
		000000000224244	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3_ALL
		00000000023424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_L3MISS_ALL
		00000000024424C	PM_DATA_RADIX_PROCESS_L2_PDE_FROM_DISTANT_ALL
81	PM_RADIX_DS_12	000000000214246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L2_ALL
		000000000224246	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3_ALL
		00000000023424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_L3MISS_ALL
		00000000024424E	PM_DATA_RADIX_PROCESS_L3_PTE_FROM_DISTANT_ALL
82	PM_RADIX_DS_13	000000000214248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L2_ALL
		000000000224248	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3_ALL
		000000000235240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_L3MISS_ALL
		000000000245240	PM_DATA_RADIX_PROCESS_L3_PDE_FROM_DISTANT_ALL
83	PM_RADIX_DS_14	00000000021424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L2_ALL
		00000000022424A	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3_ALL
		000000000235242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_L3MISS_ALL
		000000000245242	PM_DATA_RADIX_PROCESS_L4_PTE_FROM_DISTANT_ALL
84	PM_RADIX_DS_15	00000000011424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L2_ALL
		00000000012424E	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3_ALL
		000000000135246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_L3MISS_ALL
		000000000145246	PM_INST_RADIX_PROCESS_L2_PTE_FROM_DISTANT_ALL
85	PM_RADIX_DS_16	000000000115242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L2_ALL
		000000000125242	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3_ALL
		00000000013524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_L3MISS_ALL
		00000000014524A	PM_INST_RADIX_PROCESS_L3_PTE_FROM_DISTANT_ALL
86	PM_RADIX_DS_17	000000000115244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L2_ALL
		000000000125244	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3_ALL
		00000000013524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_L3MISS_ALL
		00000000014524C	PM_INST_RADIX_PROCESS_L3_PDE_FROM_DISTANT_ALL
87	PM_RADIX_DS_18	000000000115246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L2_ALL
		000000000125246	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3_ALL
		00000000013524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_L3MISS_ALL
		00000000014524E	PM_INST_RADIX_PROCESS_L4_PTE_FROM_DISTANT_ALL



Table E-35. Power10 Events by Group (Sheet 11 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
88	PM_DS_1	100FE	PM_INST_CMPL
		000300000002C040	PM_INST_FROM_L2
		000300000003C142	PM_MRK_INST_FROM_L2
		000300000004C144	PM_MRK_INST_FROM_L2_CYC
89	PM_DS_2	100FE	PM_INST_CMPL
		000320000002C040	PM_IPTEG_FROM_L2
		000320000003C142	PM_MRK_IPTEG_FROM_L2
		000320000004C144	PM_MRK_IPTEG_FROM_L2_CYC
90	PM_DS_3	100FE	PM_INST_CMPL
		000340000002C040	PM_DATA_FROM_L2
		000340000003C142	PM_MRK_DATA_FROM_L2
		000340000004C144	PM_MRK_DATA_FROM_L2_CYC
91	PM_DS_4	100FE	PM_INST_CMPL
		000360000002C040	PM_DPTEG_FROM_L2
		000360000003C142	PM_MRK_DPTEG_FROM_L2
		000360000004C144	PM_MRK_DPTEG_FROM_L2_CYC
92	PM_DS_5	100FE	PM_INST_CMPL
		000300000012C040	PM_INST_FROM_L2_ALL
		000300000013C142	PM_MRK_INST_FROM_L2_ALL
		000300000014C144	PM_MRK_INST_FROM_L2_ALL_CYC
93	PM_DS_6	100FE	PM_INST_CMPL
		000320000012C040	PM_IPTEG_FROM_L2_ALL
		000320000013C142	PM_MRK_IPTEG_FROM_L2_ALL
		000320000014C144	PM_MRK_IPTEG_FROM_L2_ALL_CYC
94	PM_DS_7	100FE	PM_INST_CMPL
		000340000022C040	PM_DATA_FROM_L2_ALL
		000340000023C142	PM_MRK_DATA_FROM_L2_ALL
		000340000024C144	PM_MRK_DATA_FROM_L2_ALL_CYC
95	PM_DS_8	100FE	PM_INST_CMPL
		000360000022C040	PM_DPTEG_FROM_L2_ALL
		000360000023C142	PM_MRK_DPTEG_FROM_L2_ALL
		000360000024C144	PM_MRK_DPTEG_FROM_L2_ALL_CYC
96	PM_DS_9	100FE	PM_INST_CMPL
		003F00000002C040	PM_INST_FROM_L1MISS
		003F00000003C142	PM_MRK_INST_FROM_L1MISS
		003F00000004C144	PM_MRK_INST_FROM_L1MISS_CYC



Table E-35. Power10 Events by Group (Sheet 12 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
97	PM_DS_10	100FE	PM_INST_CMPL
		003F20000002C040	PM_IPTEG_FROM_L1MISS
		003F20000003C142	PM_MRK_IPTEG_FROM_L1MISS
		003F20000004C144	PM_MRK_IPTEG_FROM_L1MISS_CYC
98	PM_DS_11	100FE	PM_INST_CMPL
		003F40000002C040	PM_DATA_FROM_L1MISS
		003F40000003C142	PM_MRK_DATA_FROM_L1MISS
		003F40000004C144	PM_MRK_DATA_FROM_L1MISS_CYC
99	PM_DS_12	100FE	PM_INST_CMPL
		003F60000002C040	PM_DPTEG_FROM_L1MISS
		003F60000003C142	PM_MRK_DPTEG_FROM_L1MISS
		003F60000004C144	PM_MRK_DPTEG_FROM_L1MISS_CYC
100	PM_DS_13	100FE	PM_INST_CMPL
		003F00000012C040	PM_INST_FROM_L1MISS_ALL
		003F00000013C142	PM_MRK_INST_FROM_L1MISS_ALL
		003F00000014C144	PM_MRK_INST_FROM_L1MISS_ALL_CYC
101	PM_DS_14	100FE	PM_INST_CMPL
		003F20000012C040	PM_IPTEG_FROM_L1MISS_ALL
		003F20000013C142	PM_MRK_IPTEG_FROM_L1MISS_ALL
		003F20000014C144	PM_MRK_IPTEG_FROM_L1MISS_ALL_CYC
102	PM_DS_15	100FE	PM_INST_CMPL
		003F40000022C040	PM_DATA_FROM_L1MISS_ALL
		003F40000023C142	PM_MRK_DATA_FROM_L1MISS_ALL
		003F40000024C144	PM_MRK_DATA_FROM_L1MISS_ALL_CYC
103	PM_DS_16	100FE	PM_INST_CMPL
		003F60000022C040	PM_DPTEG_FROM_L1MISS_ALL
		003F60000023C142	PM_MRK_DPTEG_FROM_L1MISS_ALL
		003F60000024C144	PM_MRK_DPTEG_FROM_L1MISS_ALL_CYC
104	PM_DS_17	100FE	PM_INST_CMPL
		000020000002C040	PM_IPTEG_FROM_L2_NO_CONFLICT
		000020000003C142	PM_MRK_IPTEG_FROM_L2_NO_CONFLICT
		000020000004C144	PM_MRK_IPTEG_FROM_L2_NO_CONFLICT_CYC
105	PM_DS_18	100FE	PM_INST_CMPL
		000040000002C040	PM_DATA_FROM_L2_NO_CONFLICT
		000040000003C142	PM_MRK_DATA_FROM_L2_NO_CONFLICT
		000040000004C144	PM_MRK_DATA_FROM_L2_NO_CONFLICT_CYC



Table E-35. Power10 Events by Group (Sheet 13 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
106	PM_DS_19	100FE	PM_INST_CMPL
		000060000002C040	PM_DPTEG_FROM_L2_NO_CONFLICT
		000060000003C142	PM_MRK_DPTEG_FROM_L2_NO_CONFLICT
		000060000004C144	PM_MRK_DPTEG_FROM_L2_NO_CONFLICT_CYC
107	PM_DS_20	100FE	PM_INST_CMPL
		000020000012C040	PM_IPTEG_FROM_L2_NO_CONFLICT_ALL
		000020000013C142	PM_MRK_IPTEG_FROM_L2_NO_CONFLICT_ALL
		000020000014C144	PM_MRK_IPTEG_FROM_L2_NO_CONFLICT_ALL_CYC
108	PM_DS_21	100FE	PM_INST_CMPL
		000040000022C040	PM_DATA_FROM_L2_NO_CONFLICT_ALL
		000040000023C142	PM_MRK_DATA_FROM_L2_NO_CONFLICT_ALL
		000040000024C144	PM_MRK_DATA_FROM_L2_NO_CONFLICT_ALL_CYC
109	PM_DS_22	100FE	PM_INST_CMPL
		000060000022C040	PM_DPTEG_FROM_L2_NO_CONFLICT_ALL
		000060000023C142	PM_MRK_DPTEG_FROM_L2_NO_CONFLICT_ALL
		000060000024C144	PM_MRK_DPTEG_FROM_L2_NO_CONFLICT_ALL_CYC
110	PM_DS_23	100FE	PM_INST_CMPL
		004020000002C040	PM_IPTEG_FROM_L2_MEPF
		004020000003C142	PM_MRK_IPTEG_FROM_L2_MEPF
		004020000004C144	PM_MRK_IPTEG_FROM_L2_MEPF_CYC
111	PM_DS_24	100FE	PM_INST_CMPL
		004040000002C040	PM_DATA_FROM_L2_MEPF
		004040000003C142	PM_MRK_DATA_FROM_L2_MEPF
		004040000004C144	PM_MRK_DATA_FROM_L2_MEPF_CYC
112	PM_DS_25	100FE	PM_INST_CMPL
		004060000002C040	PM_DPTEG_FROM_L2_MEPF
		004060000003C142	PM_MRK_DPTEG_FROM_L2_MEPF
		004060000004C144	PM_MRK_DPTEG_FROM_L2_MEPF_CYC
113	PM_DS_26	100FE	PM_INST_CMPL
		004020000012C040	PM_IPTEG_FROM_L2_MEPF_ALL
		004020000013C142	PM_MRK_IPTEG_FROM_L2_MEPF_ALL
		004020000014C144	PM_MRK_IPTEG_FROM_L2_MEPF_ALL_CYC
114	PM_DS_27	100FE	PM_INST_CMPL
		004040000022C040	PM_DATA_FROM_L2_MEPF_ALL
		004040000023C142	PM_MRK_DATA_FROM_L2_MEPF_ALL
		004040000024C144	PM_MRK_DATA_FROM_L2_MEPF_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 14 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
115	PM_DS_28	100FE	PM_INST_CMPL
		004060000022C040	PM_DPTEG_FROM_L2_MEPE_ALL
		004060000023C142	PM_MRK_DPTEG_FROM_L2_MEPE_ALL
		004060000024C144	PM_MRK_DPTEG_FROM_L2_MEPE_ALL_CYC
116	PM_DS_29	100FE	PM_INST_CMPL
		008020000002C040	PM_IPTEG_FROM_L2_LDHTST_CONFLICT
		008020000003C142	PM_MRK_IPTEG_FROM_L2_LDHTST_CONFLICT
		008020000004C144	PM_MRK_IPTEG_FROM_L2_LDHTST_CONFLICT_CYC
117	PM_DS_30	100FE	PM_INST_CMPL
		008040000002C040	PM_DATA_FROM_L2_LDHTST_CONFLICT
		008040000003C142	PM_MRK_DATA_FROM_L2_LDHTST_CONFLICT
		008040000004C144	PM_MRK_DATA_FROM_L2_LDHTST_CONFLICT_CYC
118	PM_DS_31	100FE	PM_INST_CMPL
		008060000002C040	PM_DPTEG_FROM_L2_LDHTST_CONFLICT
		008060000003C142	PM_MRK_DPTEG_FROM_L2_LDHTST_CONFLICT
		008060000004C144	PM_MRK_DPTEG_FROM_L2_LDHTST_CONFLICT_CYC
119	PM_DS_32	100FE	PM_INST_CMPL
		008020000012C040	PM_IPTEG_FROM_L2_LDHTST_CONFLICT_ALL
		008020000013C142	PM_MRK_IPTEG_FROM_L2_LDHTST_CONFLICT_ALL
		008020000014C144	PM_MRK_IPTEG_FROM_L2_LDHTST_CONFLICT_ALL_CYC
120	PM_DS_33	100FE	PM_INST_CMPL
		008040000022C040	PM_DATA_FROM_L2_LDHTST_CONFLICT_ALL
		008040000023C142	PM_MRK_DATA_FROM_L2_LDHTST_CONFLICT_ALL
		008040000024C144	PM_MRK_DATA_FROM_L2_LDHTST_CONFLICT_ALL_CYC
121	PM_DS_34	100FE	PM_INST_CMPL
		008060000022C040	PM_DPTEG_FROM_L2_LDHTST_CONFLICT_ALL
		008060000023C142	PM_MRK_DPTEG_FROM_L2_LDHTST_CONFLICT_ALL
		008060000024C144	PM_MRK_DPTEG_FROM_L2_LDHTST_CONFLICT_ALL_CYC
122	PM_DS_35	100FE	PM_INST_CMPL
		00C020000002C040	PM_IPTEG_FROM_L2_OTHER_CONFLICT
		00C020000003C142	PM_MRK_IPTEG_FROM_L2_OTHER_CONFLICT
		00C020000004C144	PM_MRK_IPTEG_FROM_L2_OTHER_CONFLICT_CYC
123	PM_DS_36	100FE	PM_INST_CMPL
		00C040000002C040	PM_DATA_FROM_L2_OTHER_CONFLICT
		00C040000003C142	PM_MRK_DATA_FROM_L2_OTHER_CONFLICT
		00C040000004C144	PM_MRK_DATA_FROM_L2_OTHER_CONFLICT_CYC



Table E-35. Power10 Events by Group (Sheet 15 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
124	PM_DS_37	100FE	PM_INST_CMPL
		00C060000002C040	PM_DPTEG_FROM_L2_OTHER_CONFLICT
		00C060000003C142	PM_MRK_DPTEG_FROM_L2_OTHER_CONFLICT
		00C060000004C144	PM_MRK_DPTEG_FROM_L2_OTHER_CONFLICT_CYC
125	PM_DS_38	100FE	PM_INST_CMPL
		00C020000012C040	PM_IPTEG_FROM_L2_OTHER_CONFLICT_ALL
		00C020000013C142	PM_MRK_IPTEG_FROM_L2_OTHER_CONFLICT_ALL
		00C020000014C144	PM_MRK_IPTEG_FROM_L2_OTHER_CONFLICT_ALL_CYC
126	PM_DS_39	100FE	PM_INST_CMPL
		00C040000022C040	PM_DATA_FROM_L2_OTHER_CONFLICT_ALL
		00C040000023C142	PM_MRK_DATA_FROM_L2_OTHER_CONFLICT_ALL
		00C040000024C144	PM_MRK_DATA_FROM_L2_OTHER_CONFLICT_ALL_CYC
127	PM_DS_40	100FE	PM_INST_CMPL
		00C060000022C040	PM_DPTEG_FROM_L2_OTHER_CONFLICT_ALL
		00C060000023C142	PM_MRK_DPTEG_FROM_L2_OTHER_CONFLICT_ALL
		00C060000024C144	PM_MRK_DPTEG_FROM_L2_OTHER_CONFLICT_ALL_CYC
128	PM_DS_41	100FE	PM_INST_CMPL
		000380000002C040	PM_INST_FROM_L2MISS
		000380000003C142	PM_MRK_INST_FROM_L2MISS
		000380000004C144	PM_MRK_INST_FROM_L2MISS_CYC
129	PM_DS_42	100FE	PM_INST_CMPL
		0003A0000002C040	PM_IPTEG_FROM_L2MISS
		0003A0000003C142	PM_MRK_IPTEG_FROM_L2MISS
		0003A0000004C144	PM_MRK_IPTEG_FROM_L2MISS_CYC
130	PM_DS_43	100FE	PM_INST_CMPL
		200FE	PM_DATA_FROM_L2MISS
		0003C0000003C142	PM_MRK_DATA_FROM_L2MISS
		0003C0000004C144	PM_MRK_DATA_FROM_L2MISS_CYC
131	PM_DS_44	100FE	PM_INST_CMPL
		0003E0000002C040	PM_DPTEG_FROM_L2MISS
		0003E0000003C142	PM_MRK_DPTEG_FROM_L2MISS
		0003E0000004C144	PM_MRK_DPTEG_FROM_L2MISS_CYC
132	PM_DS_45	100FE	PM_INST_CMPL
		000380000012C040	PM_INST_FROM_L2MISS_ALL
		000380000013C142	PM_MRK_INST_FROM_L2MISS_ALL
		000380000014C144	PM_MRK_INST_FROM_L2MISS_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 16 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
133	PM_DS_46	100FE	PM_INST_CMPL
		0003A0000012C040	PM_IPTEG_FROM_L2MISS_ALL
		0003A0000013C142	PM_MRK_IPTEG_FROM_L2MISS_ALL
		0003A0000014C144	PM_MRK_IPTEG_FROM_L2MISS_ALL_CYC
134	PM_DS_47	100FE	PM_INST_CMPL
		0003C0000022C040	PM_DATA_FROM_L2MISS_ALL
		0003C0000023C142	PM_MRK_DATA_FROM_L2MISS_ALL
		0003C0000024C144	PM_MRK_DATA_FROM_L2MISS_ALL_CYC
135	PM_DS_48	100FE	PM_INST_CMPL
		0003E0000022C040	PM_DPTEG_FROM_L2MISS_ALL
		0003E0000023C142	PM_MRK_DPTEG_FROM_L2MISS_ALL
		0003E0000024C144	PM_MRK_DPTEG_FROM_L2MISS_ALL_CYC
136	PM_DS_49	100FE	PM_INST_CMPL
		010300000002C040	PM_INST_FROM_L3
		010300000003C142	PM_MRK_INST_FROM_L3
		010300000004C144	PM_MRK_INST_FROM_L3_CYC
137	PM_DS_50	100FE	PM_INST_CMPL
		010320000002C040	PM_IPTEG_FROM_L3
		010320000003C142	PM_MRK_IPTEG_FROM_L3
		010320000004C144	PM_MRK_IPTEG_FROM_L3_CYC
138	PM_DS_51	100FE	PM_INST_CMPL
		010340000002C040	PM_DATA_FROM_L3
		010340000003C142	PM_MRK_DATA_FROM_L3
		010340000004C144	PM_MRK_DATA_FROM_L3_CYC
139	PM_DS_52	100FE	PM_INST_CMPL
		010360000002C040	PM_DPTEG_FROM_L3
		010360000003C142	PM_MRK_DPTEG_FROM_L3
		010360000004C144	PM_MRK_DPTEG_FROM_L3_CYC
140	PM_DS_53	100FE	PM_INST_CMPL
		010300000012C040	PM_INST_FROM_L3_ALL
		010300000013C142	PM_MRK_INST_FROM_L3_ALL
		010300000014C144	PM_MRK_INST_FROM_L3_ALL_CYC
141	PM_DS_54	100FE	PM_INST_CMPL
		010320000012C040	PM_IPTEG_FROM_L3_ALL
		010320000013C142	PM_MRK_IPTEG_FROM_L3_ALL
		010320000014C144	PM_MRK_IPTEG_FROM_L3_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 17 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
142	PM_DS_55	100FE	PM_INST_CMPL
		010340000022C040	PM_DATA_FROM_L3_ALL
		010340000023C142	PM_MRK_DATA_FROM_L3_ALL
		010340000024C144	PM_MRK_DATA_FROM_L3_ALL_CYC
143	PM_DS_56	100FE	PM_INST_CMPL
		010360000022C040	PM_DPTEG_FROM_L3_ALL
		010360000023C142	PM_MRK_DPTEG_FROM_L3_ALL
		010360000024C144	PM_MRK_DPTEG_FROM_L3_ALL_CYC
144	PM_DS_57	100FE	PM_INST_CMPL
		010020000002C040	PM_IPTEG_FROM_L3_NO_CONFLICT
		010020000003C142	PM_MRK_IPTEG_FROM_L3_NO_CONFLICT
		010020000004C144	PM_MRK_IPTEG_FROM_L3_NO_CONFLICT_CYC
145	PM_DS_58	100FE	PM_INST_CMPL
		010040000002C040	PM_DATA_FROM_L3_NO_CONFLICT
		010040000003C142	PM_MRK_DATA_FROM_L3_NO_CONFLICT
		010040000004C144	PM_MRK_DATA_FROM_L3_NO_CONFLICT_CYC
146	PM_DS_59	100FE	PM_INST_CMPL
		010060000002C040	PM_DPTEG_FROM_L3_NO_CONFLICT
		010060000003C142	PM_MRK_DPTEG_FROM_L3_NO_CONFLICT
		010060000004C144	PM_MRK_DPTEG_FROM_L3_NO_CONFLICT_CYC
147	PM_DS_60	100FE	PM_INST_CMPL
		010020000012C040	PM_IPTEG_FROM_L3_NO_CONFLICT_ALL
		010020000013C142	PM_MRK_IPTEG_FROM_L3_NO_CONFLICT_ALL
		010020000014C144	PM_MRK_IPTEG_FROM_L3_NO_CONFLICT_ALL_CYC
148	PM_DS_61	100FE	PM_INST_CMPL
		010040000022C040	PM_DATA_FROM_L3_NO_CONFLICT_ALL
		010040000023C142	PM_MRK_DATA_FROM_L3_NO_CONFLICT_ALL
		010040000024C144	PM_MRK_DATA_FROM_L3_NO_CONFLICT_ALL_CYC
149	PM_DS_62	100FE	PM_INST_CMPL
		010060000022C040	PM_DPTEG_FROM_L3_NO_CONFLICT_ALL
		010060000023C142	PM_MRK_DPTEG_FROM_L3_NO_CONFLICT_ALL
		010060000024C144	PM_MRK_DPTEG_FROM_L3_NO_CONFLICT_ALL_CYC
150	PM_DS_63	100FE	PM_INST_CMPL
		014020000002C040	PM_IPTEG_FROM_L3_MEPF
		014020000003C142	PM_MRK_IPTEG_FROM_L3_MEPF
		014020000004C144	PM_MRK_IPTEG_FROM_L3_MEPF_CYC



Table E-35. Power10 Events by Group (Sheet 18 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
151	PM_DS_64	100FE	PM_INST_CMPL
		014040000002C040	PM_DATA_FROM_L3_MEPP
		014040000003C142	PM_MRK_DATA_FROM_L3_MEPP
		014040000004C144	PM_MRK_DATA_FROM_L3_MEPP_CYC
152	PM_DS_65	100FE	PM_INST_CMPL
		014060000002C040	PM_DPTEG_FROM_L3_MEPP
		014060000003C142	PM_MRK_DPTEG_FROM_L3_MEPP
		014060000004C144	PM_MRK_DPTEG_FROM_L3_MEPP_CYC
153	PM_DS_66	100FE	PM_INST_CMPL
		014020000012C040	PM_IPTEG_FROM_L3_MEPP_ALL
		014020000013C142	PM_MRK_IPTEG_FROM_L3_MEPP_ALL
		014020000014C144	PM_MRK_IPTEG_FROM_L3_MEPP_ALL_CYC
154	PM_DS_67	100FE	PM_INST_CMPL
		014040000022C040	PM_DATA_FROM_L3_MEPP_ALL
		014040000023C142	PM_MRK_DATA_FROM_L3_MEPP_ALL
		014040000024C144	PM_MRK_DATA_FROM_L3_MEPP_ALL_CYC
155	PM_DS_68	100FE	PM_INST_CMPL
		014060000022C040	PM_DPTEG_FROM_L3_MEPP_ALL
		014060000023C142	PM_MRK_DPTEG_FROM_L3_MEPP_ALL
		014060000024C144	PM_MRK_DPTEG_FROM_L3_MEPP_ALL_CYC
156	PM_DS_69	100FE	PM_INST_CMPL
		01C020000002C040	PM_IPTEG_FROM_L3_CONFLICT
		01C020000003C142	PM_MRK_IPTEG_FROM_L3_CONFLICT
		01C020000004C144	PM_MRK_IPTEG_FROM_L3_CONFLICT_CYC
157	PM_DS_70	100FE	PM_INST_CMPL
		01C040000002C040	PM_DATA_FROM_L3_CONFLICT
		01C040000003C142	PM_MRK_DATA_FROM_L3_CONFLICT
		01C040000004C144	PM_MRK_DATA_FROM_L3_CONFLICT_CYC
158	PM_DS_71	100FE	PM_INST_CMPL
		01C060000002C040	PM_DPTEG_FROM_L3_CONFLICT
		01C060000003C142	PM_MRK_DPTEG_FROM_L3_CONFLICT
		01C060000004C144	PM_MRK_DPTEG_FROM_L3_CONFLICT_CYC
159	PM_DS_72	100FE	PM_INST_CMPL
		01C0200000012C040	PM_IPTEG_FROM_L3_CONFLICT_ALL
		01C0200000013C142	PM_MRK_IPTEG_FROM_L3_CONFLICT_ALL
		01C0200000014C144	PM_MRK_IPTEG_FROM_L3_CONFLICT_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 19 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
160	PM_DS_73	100FE	PM_INST_CMPL
		01C040000022C040	PM_DATA_FROM_L3_CONFLICT_ALL
		01C040000023C142	PM_MRK_DATA_FROM_L3_CONFLICT_ALL
		01C040000024C144	PM_MRK_DATA_FROM_L3_CONFLICT_ALL_CYC
161	PM_DS_74	100FE	PM_INST_CMPL
		01C060000022C040	PM_DPTEG_FROM_L3_CONFLICT_ALL
		01C060000023C142	PM_MRK_DPTEG_FROM_L3_CONFLICT_ALL
		01C060000024C144	PM_MRK_DPTEG_FROM_L3_CONFLICT_ALL_CYC
162	PM_DS_75	100FE	PM_INST_CMPL
		000780000002C040	PM_INST_FROM_L3MISS
		000780000003C142	PM_MRK_INST_FROM_L3MISS
		000780000004C144	PM_MRK_INST_FROM_L3MISS_CYC
163	PM_DS_76	100FE	PM_INST_CMPL
		0007A0000002C040	PM_IPTEG_FROM_L3MISS
		0007A0000003C142	PM_MRK_IPTEG_FROM_L3MISS
		0007A0000004C144	PM_MRK_IPTEG_FROM_L3MISS_CYC
164	PM_DS_77	100FE	PM_INST_CMPL
		0007C0000002C040	PM_DATA_FROM_L3MISS
		0007C0000003C142	PM_MRK_DATA_FROM_L3MISS
		0007C0000004C144	PM_MRK_DATA_FROM_L3MISS_CYC
165	PM_DS_78	100FE	PM_INST_CMPL
		0007E0000002C040	PM_DPTEG_FROM_L3MISS
		0007E0000003C142	PM_MRK_DPTEG_FROM_L3MISS
		0007E0000004C144	PM_MRK_DPTEG_FROM_L3MISS_CYC
166	PM_DS_79	100FE	PM_INST_CMPL
		000780000012C040	PM_INST_FROM_L3MISS_ALL
		000780000013C142	PM_MRK_INST_FROM_L3MISS_ALL
		000780000014C144	PM_MRK_INST_FROM_L3MISS_ALL_CYC
167	PM_DS_80	100FE	PM_INST_CMPL
		0007A0000012C040	PM_IPTEG_FROM_L3MISS_ALL
		0007A0000013C142	PM_MRK_IPTEG_FROM_L3MISS_ALL
		0007A0000014C144	PM_MRK_IPTEG_FROM_L3MISS_ALL_CYC
168	PM_DS_81	100FE	PM_INST_CMPL
		0007C0000022C040	PM_DATA_FROM_L3MISS_ALL
		0007C0000023C142	PM_MRK_DATA_FROM_L3MISS_ALL
		0007C0000024C144	PM_MRK_DATA_FROM_L3MISS_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 20 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
169	PM_DS_82	100FE	PM_INST_CMPL
		0007E0000022C040	PM_DPTEG_FROM_L3MISS_ALL
		0007E0000023C142	PM_MRK_DPTEG_FROM_L3MISS_ALL
		0007E0000024C144	PM_MRK_DPTEG_FROM_L3MISS_ALL_CYC
170	PM_DS_83	100FE	PM_INST_CMPL
		080020000002C040	PM_IPTEG_FROM_L21_REGENT_SHR
		080020000003C142	PM_MRK_IPTEG_FROM_L21_REGENT_SHR
		080020000004C144	PM_MRK_IPTEG_FROM_L21_REGENT_SHR_CYC
171	PM_DS_84	100FE	PM_INST_CMPL
		080040000002C040	PM_DATA_FROM_L21_REGENT_SHR
		080040000003C142	PM_MRK_DATA_FROM_L21_REGENT_SHR
		080040000004C144	PM_MRK_DATA_FROM_L21_REGENT_SHR_CYC
172	PM_DS_85	100FE	PM_INST_CMPL
		080060000002C040	PM_DPTEG_FROM_L21_REGENT_SHR
		080060000003C142	PM_MRK_DPTEG_FROM_L21_REGENT_SHR
		080060000004C144	PM_MRK_DPTEG_FROM_L21_REGENT_SHR_CYC
173	PM_DS_86	100FE	PM_INST_CMPL
		080020000012C040	PM_IPTEG_FROM_L21_REGENT_SHR_ALL
		080020000013C142	PM_MRK_IPTEG_FROM_L21_REGENT_SHR_ALL
		080020000014C144	PM_MRK_IPTEG_FROM_L21_REGENT_SHR_ALL_CYC
174	PM_DS_87	100FE	PM_INST_CMPL
		080040000022C040	PM_DATA_FROM_L21_REGENT_SHR_ALL
		080040000023C142	PM_MRK_DATA_FROM_L21_REGENT_SHR_ALL
		080040000024C144	PM_MRK_DATA_FROM_L21_REGENT_SHR_ALL_CYC
175	PM_DS_88	100FE	PM_INST_CMPL
		080060000022C040	PM_DPTEG_FROM_L21_REGENT_SHR_ALL
		080060000023C142	PM_MRK_DPTEG_FROM_L21_REGENT_SHR_ALL
		080060000024C144	PM_MRK_DPTEG_FROM_L21_REGENT_SHR_ALL_CYC
176	PM_DS_89	100FE	PM_INST_CMPL
		084020000002C040	PM_IPTEG_FROM_L21_REGENT_MOD
		084020000003C142	PM_MRK_IPTEG_FROM_L21_REGENT_MOD
		084020000004C144	PM_MRK_IPTEG_FROM_L21_REGENT_MOD_CYC
177	PM_DS_90	100FE	PM_INST_CMPL
		084040000002C040	PM_DATA_FROM_L21_REGENT_MOD
		084040000003C142	PM_MRK_DATA_FROM_L21_REGENT_MOD
		084040000004C144	PM_MRK_DATA_FROM_L21_REGENT_MOD_CYC



Table E-35. Power10 Events by Group (Sheet 21 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
178	PM_DS_91	100FE	PM_INST_CMPL
		084060000002C040	PM_DPTEG_FROM_L21_REGENT_MOD
		084060000003C142	PM_MRK_DPTEG_FROM_L21_REGENT_MOD
		084060000004C144	PM_MRK_DPTEG_FROM_L21_REGENT_MOD_CYC
179	PM_DS_92	100FE	PM_INST_CMPL
		084020000012C040	PM_IPTEG_FROM_L21_REGENT_MOD_ALL
		084020000013C142	PM_MRK_IPTEG_FROM_L21_REGENT_MOD_ALL
		084020000014C144	PM_MRK_IPTEG_FROM_L21_REGENT_MOD_ALL_CYC
180	PM_DS_93	100FE	PM_INST_CMPL
		084040000022C040	PM_DATA_FROM_L21_REGENT_MOD_ALL
		084040000023C142	PM_MRK_DATA_FROM_L21_REGENT_MOD_ALL
		084040000024C144	PM_MRK_DATA_FROM_L21_REGENT_MOD_ALL_CYC
181	PM_DS_94	100FE	PM_INST_CMPL
		084060000022C040	PM_DPTEG_FROM_L21_REGENT_MOD_ALL
		084060000023C142	PM_MRK_DPTEG_FROM_L21_REGENT_MOD_ALL
		084060000024C144	PM_MRK_DPTEG_FROM_L21_REGENT_MOD_ALL_CYC
182	PM_DS_95	100FE	PM_INST_CMPL
		080100000002C040	PM_INST_FROM_L21_REGENT
		080100000003C142	PM_MRK_INST_FROM_L21_REGENT
		080100000004C144	PM_MRK_INST_FROM_L21_REGENT_CYC
183	PM_DS_96	100FE	PM_INST_CMPL
		080120000002C040	PM_IPTEG_FROM_L21_REGENT
		080120000003C142	PM_MRK_IPTEG_FROM_L21_REGENT
		080120000004C144	PM_MRK_IPTEG_FROM_L21_REGENT_CYC
184	PM_DS_97	100FE	PM_INST_CMPL
		080140000002C040	PM_DATA_FROM_L21_REGENT
		080140000003C142	PM_MRK_DATA_FROM_L21_REGENT
		080140000004C144	PM_MRK_DATA_FROM_L21_REGENT_CYC
185	PM_DS_98	100FE	PM_INST_CMPL
		080160000002C040	PM_DPTEG_FROM_L21_REGENT
		080160000003C142	PM_MRK_DPTEG_FROM_L21_REGENT
		080160000004C144	PM_MRK_DPTEG_FROM_L21_REGENT_CYC
186	PM_DS_99	100FE	PM_INST_CMPL
		080100000012C040	PM_INST_FROM_L21_REGENT_ALL
		080100000013C142	PM_MRK_INST_FROM_L21_REGENT_ALL
		080100000014C144	PM_MRK_INST_FROM_L21_REGENT_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 22 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
187	PM_DS_100	100FE	PM_INST_CMPL
		080120000012C040	PM_IPTEG_FROM_L21_REGENT_ALL
		080120000013C142	PM_MRK_IPTEG_FROM_L21_REGENT_ALL
		080120000014C144	PM_MRK_IPTEG_FROM_L21_REGENT_ALL_CYC
188	PM_DS_101	100FE	PM_INST_CMPL
		080140000022C040	PM_DATA_FROM_L21_REGENT_ALL
		080140000023C142	PM_MRK_DATA_FROM_L21_REGENT_ALL
		080140000024C144	PM_MRK_DATA_FROM_L21_REGENT_ALL_CYC
189	PM_DS_102	100FE	PM_INST_CMPL
		080160000022C040	PM_DPTEG_FROM_L21_REGENT_ALL
		080160000023C142	PM_MRK_DPTEG_FROM_L21_REGENT_ALL
		080160000024C144	PM_MRK_DPTEG_FROM_L21_REGENT_ALL_CYC
190	PM_DS_103	100FE	PM_INST_CMPL
		088020000002C040	PM_IPTEG_FROM_L31_REGENT_SHR
		088020000003C142	PM_MRK_IPTEG_FROM_L31_REGENT_SHR
		088020000004C144	PM_MRK_IPTEG_FROM_L31_REGENT_SHR_CYC
191	PM_DS_104	100FE	PM_INST_CMPL
		088040000002C040	PM_DATA_FROM_L31_REGENT_SHR
		088040000003C142	PM_MRK_DATA_FROM_L31_REGENT_SHR
		088040000004C144	PM_MRK_DATA_FROM_L31_REGENT_SHR_CYC
192	PM_DS_105	100FE	PM_INST_CMPL
		088060000002C040	PM_DPTEG_FROM_L31_REGENT_SHR
		088060000003C142	PM_MRK_DPTEG_FROM_L31_REGENT_SHR
		088060000004C144	PM_MRK_DPTEG_FROM_L31_REGENT_SHR_CYC
193	PM_DS_106	100FE	PM_INST_CMPL
		088020000012C040	PM_IPTEG_FROM_L31_REGENT_SHR_ALL
		088020000013C142	PM_MRK_IPTEG_FROM_L31_REGENT_SHR_ALL
		088020000014C144	PM_MRK_IPTEG_FROM_L31_REGENT_SHR_ALL_CYC
194	PM_DS_107	100FE	PM_INST_CMPL
		088040000022C040	PM_DATA_FROM_L31_REGENT_SHR_ALL
		088040000023C142	PM_MRK_DATA_FROM_L31_REGENT_SHR_ALL
		088040000024C144	PM_MRK_DATA_FROM_L31_REGENT_SHR_ALL_CYC
195	PM_DS_108	100FE	PM_INST_CMPL
		088060000022C040	PM_DPTEG_FROM_L31_REGENT_SHR_ALL
		088060000023C142	PM_MRK_DPTEG_FROM_L31_REGENT_SHR_ALL
		088060000024C144	PM_MRK_DPTEG_FROM_L31_REGENT_SHR_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 23 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
196	PM_DS_109	100FE	PM_INST_CMPL
		08C020000002C040	PM_IPTEG_FROM_L31_REGENT_MOD
		08C020000003C142	PM_MRK_IPTEG_FROM_L31_REGENT_MOD
		08C020000004C144	PM_MRK_IPTEG_FROM_L31_REGENT_MOD_CYC
197	PM_DS_110	100FE	PM_INST_CMPL
		08C040000002C040	PM_DATA_FROM_L31_REGENT_MOD
		08C040000003C142	PM_MRK_DATA_FROM_L31_REGENT_MOD
		08C040000004C144	PM_MRK_DATA_FROM_L31_REGENT_MOD_CYC
198	PM_DS_111	100FE	PM_INST_CMPL
		08C060000002C040	PM_DPTEG_FROM_L31_REGENT_MOD
		08C060000003C142	PM_MRK_DPTEG_FROM_L31_REGENT_MOD
		08C060000004C144	PM_MRK_DPTEG_FROM_L31_REGENT_MOD_CYC
199	PM_DS_112	100FE	PM_INST_CMPL
		08C020000012C040	PM_IPTEG_FROM_L31_REGENT_MOD_ALL
		08C020000013C142	PM_MRK_IPTEG_FROM_L31_REGENT_MOD_ALL
		08C020000014C144	PM_MRK_IPTEG_FROM_L31_REGENT_MOD_ALL_CYC
200	PM_DS_113	100FE	PM_INST_CMPL
		08C040000022C040	PM_DATA_FROM_L31_REGENT_MOD_ALL
		08C040000023C142	PM_MRK_DATA_FROM_L31_REGENT_MOD_ALL
		08C040000024C144	PM_MRK_DATA_FROM_L31_REGENT_MOD_ALL_CYC
201	PM_DS_114	100FE	PM_INST_CMPL
		08C060000022C040	PM_DPTEG_FROM_L31_REGENT_MOD_ALL
		08C060000023C142	PM_MRK_DPTEG_FROM_L31_REGENT_MOD_ALL
		08C060000024C144	PM_MRK_DPTEG_FROM_L31_REGENT_MOD_ALL_CYC
202	PM_DS_115	100FE	PM_INST_CMPL
		088100000002C040	PM_INST_FROM_L31_REGENT
		088100000003C142	PM_MRK_INST_FROM_L31_REGENT
		088100000004C144	PM_MRK_INST_FROM_L31_REGENT_CYC
203	PM_DS_116	100FE	PM_INST_CMPL
		088120000002C040	PM_IPTEG_FROM_L31_REGENT
		088120000003C142	PM_MRK_IPTEG_FROM_L31_REGENT
		088120000004C144	PM_MRK_IPTEG_FROM_L31_REGENT_CYC
204	PM_DS_117	100FE	PM_INST_CMPL
		088140000002C040	PM_DATA_FROM_L31_REGENT
		088140000003C142	PM_MRK_DATA_FROM_L31_REGENT
		088140000004C144	PM_MRK_DATA_FROM_L31_REGENT_CYC



Table E-35. Power10 Events by Group (Sheet 24 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
205	PM_DS_118	100FE	PM_INST_CMPL
		088160000002C040	PM_DPTEG_FROM_L31_REGENT
		088160000003C142	PM_MRK_DPTEG_FROM_L31_REGENT
		088160000004C144	PM_MRK_DPTEG_FROM_L31_REGENT_CYC
206	PM_DS_119	100FE	PM_INST_CMPL
		088100000012C040	PM_INST_FROM_L31_REGENT_ALL
		088100000013C142	PM_MRK_INST_FROM_L31_REGENT_ALL
		088100000014C144	PM_MRK_INST_FROM_L31_REGENT_ALL_CYC
207	PM_DS_120	100FE	PM_INST_CMPL
		088120000012C040	PM_IPTEG_FROM_L31_REGENT_ALL
		088120000013C142	PM_MRK_IPTEG_FROM_L31_REGENT_ALL
		088120000014C144	PM_MRK_IPTEG_FROM_L31_REGENT_ALL_CYC
208	PM_DS_121	100FE	PM_INST_CMPL
		088140000022C040	PM_DATA_FROM_L31_REGENT_ALL
		088140000023C142	PM_MRK_DATA_FROM_L31_REGENT_ALL
		088140000024C144	PM_MRK_DATA_FROM_L31_REGENT_ALL_CYC
209	PM_DS_122	100FE	PM_INST_CMPL
		088160000022C040	PM_DPTEG_FROM_L31_REGENT_ALL
		088160000023C142	PM_MRK_DPTEG_FROM_L31_REGENT_ALL
		088160000024C144	PM_MRK_DPTEG_FROM_L31_REGENT_ALL_CYC
210	PM_DS_123	100FE	PM_INST_CMPL
		080220000002C040	PM_IPTEG_FROM_REGENT_L2L3_SHR
		080220000003C142	PM_MRK_IPTEG_FROM_REGENT_L2L3_SHR
		080220000004C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_SHR_CYC
211	PM_DS_124	100FE	PM_INST_CMPL
		080240000002C040	PM_DATA_FROM_REGENT_L2L3_SHR
		080240000003C142	PM_MRK_DATA_FROM_REGENT_L2L3_SHR
		080240000004C144	PM_MRK_DATA_FROM_REGENT_L2L3_SHR_CYC
212	PM_DS_125	100FE	PM_INST_CMPL
		080260000002C040	PM_DPTEG_FROM_REGENT_L2L3_SHR
		080260000003C142	PM_MRK_DPTEG_FROM_REGENT_L2L3_SHR
		080260000004C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_SHR_CYC
213	PM_DS_126	100FE	PM_INST_CMPL
		080220000012C040	PM_IPTEG_FROM_REGENT_L2L3_SHR_ALL
		080220000013C142	PM_MRK_IPTEG_FROM_REGENT_L2L3_SHR_ALL
		080220000014C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_SHR_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 25 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
214	PM_DS_127	100FE	PM_INST_CMPL
		080240000022C040	PM_DATA_FROM_REGENT_L2L3_SHR_ALL
		080240000023C142	PM_MRK_DATA_FROM_REGENT_L2L3_SHR_ALL
		080240000024C144	PM_MRK_DATA_FROM_REGENT_L2L3_SHR_ALL_CYC
215	PM_DS_128	100FE	PM_INST_CMPL
		080260000022C040	PM_DPTEG_FROM_REGENT_L2L3_SHR_ALL
		080260000023C142	PM_MRK_DPTEG_FROM_REGENT_L2L3_SHR_ALL
		080260000024C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_SHR_ALL_CYC
216	PM_DS_129	100FE	PM_INST_CMPL
		084220000002C040	PM_IPTEG_FROM_REGENT_L2L3_MOD
		084220000003C142	PM_MRK_IPTEG_FROM_REGENT_L2L3_MOD
		084220000004C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_MOD_CYC
217	PM_DS_130	100FE	PM_INST_CMPL
		084240000002C040	PM_DATA_FROM_REGENT_L2L3_MOD
		084240000003C142	PM_MRK_DATA_FROM_REGENT_L2L3_MOD
		084240000004C144	PM_MRK_DATA_FROM_REGENT_L2L3_MOD_CYC
218	PM_DS_131	100FE	PM_INST_CMPL
		084260000002C040	PM_DPTEG_FROM_REGENT_L2L3_MOD
		084260000003C142	PM_MRK_DPTEG_FROM_REGENT_L2L3_MOD
		084260000004C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_MOD_CYC
219	PM_DS_132	100FE	PM_INST_CMPL
		084220000012C040	PM_IPTEG_FROM_REGENT_L2L3_MOD_ALL
		084220000013C142	PM_MRK_IPTEG_FROM_REGENT_L2L3_MOD_ALL
		084220000014C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_MOD_ALL_CYC
220	PM_DS_133	100FE	PM_INST_CMPL
		084240000022C040	PM_DATA_FROM_REGENT_L2L3_MOD_ALL
		084240000023C142	PM_MRK_DATA_FROM_REGENT_L2L3_MOD_ALL
		084240000024C144	PM_MRK_DATA_FROM_REGENT_L2L3_MOD_ALL_CYC
221	PM_DS_134	100FE	PM_INST_CMPL
		084260000022C040	PM_DPTEG_FROM_REGENT_L2L3_MOD_ALL
		084260000023C142	PM_MRK_DPTEG_FROM_REGENT_L2L3_MOD_ALL
		084260000024C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_MOD_ALL_CYC
222	PM_DS_135	100FE	PM_INST_CMPL
		080300000002C040	PM_INST_FROM_REGENT_L2L3
		080300000003C142	PM_MRK_INST_FROM_REGENT_L2L3
		080300000004C144	PM_MRK_INST_FROM_REGENT_L2L3_CYC



Table E-35. Power10 Events by Group (Sheet 26 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
223	PM_DS_136	100FE	PM_INST_CMPL
		080320000002C040	PM_IPTEG_FROM_REGENT_L2L3
		080320000003C142	PM_MRK_IPTEG_FROM_REGENT_L2L3
		080320000004C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_CYC
224	PM_DS_137	100FE	PM_INST_CMPL
		080340000002C040	PM_DATA_FROM_REGENT_L2L3
		080340000003C142	PM_MRK_DATA_FROM_REGENT_L2L3
		080340000004C144	PM_MRK_DATA_FROM_REGENT_L2L3_CYC
225	PM_DS_138	100FE	PM_INST_CMPL
		080360000002C040	PM_DPTEG_FROM_REGENT_L2L3
		080360000003C142	PM_MRK_DPTEG_FROM_REGENT_L2L3
		080360000004C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_CYC
226	PM_DS_139	100FE	PM_INST_CMPL
		080300000012C040	PM_INST_FROM_REGENT_L2L3_ALL
		080300000013C142	PM_MRK_INST_FROM_REGENT_L2L3_ALL
		080300000014C144	PM_MRK_INST_FROM_REGENT_L2L3_ALL_CYC
227	PM_DS_140	100FE	PM_INST_CMPL
		080320000012C040	PM_IPTEG_FROM_REGENT_L2L3_ALL
		080320000013C142	PM_MRK_IPTEG_FROM_REGENT_L2L3_ALL
		080320000014C144	PM_MRK_IPTEG_FROM_REGENT_L2L3_ALL_CYC
228	PM_DS_141	100FE	PM_INST_CMPL
		080340000022C040	PM_DATA_FROM_REGENT_L2L3_ALL
		080340000023C142	PM_MRK_DATA_FROM_REGENT_L2L3_ALL
		080340000024C144	PM_MRK_DATA_FROM_REGENT_L2L3_ALL_CYC
229	PM_DS_142	100FE	PM_INST_CMPL
		080360000022C040	PM_DPTEG_FROM_REGENT_L2L3_ALL
		080360000023C142	PM_MRK_DPTEG_FROM_REGENT_L2L3_ALL
		080360000024C144	PM_MRK_DPTEG_FROM_REGENT_L2L3_ALL_CYC
230	PM_DS_143	100FE	PM_INST_CMPL
		0A0020000002C040	PM_IPTEG_FROM_L21_NON_REGENT_SHR
		0A0020000003C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT_SHR
		0A0020000004C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_SHR_CYC
231	PM_DS_144	100FE	PM_INST_CMPL
		0A0040000002C040	PM_DATA_FROM_L21_NON_REGENT_SHR
		0A0040000003C142	PM_MRK_DATA_FROM_L21_NON_REGENT_SHR
		0A0040000004C144	PM_MRK_DATA_FROM_L21_NON_REGENT_SHR_CYC



Table E-35. Power10 Events by Group (Sheet 27 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
232	PM_DS_145	100FE	PM_INST_CMPL
		0A0060000002C040	PM_DPTEG_FROM_L21_NON_REGENT_SHR
		0A0060000003C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT_SHR
		0A0060000004C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_SHR_CYC
233	PM_DS_146	100FE	PM_INST_CMPL
		0A0020000012C040	PM_IPTEG_FROM_L21_NON_REGENT_SHR_ALL
		0A0020000013C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT_SHR_ALL
		0A0020000014C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_SHR_ALL_CYC
234	PM_DS_147	100FE	PM_INST_CMPL
		0A0040000022C040	PM_DATA_FROM_L21_NON_REGENT_SHR_ALL
		0A0040000023C142	PM_MRK_DATA_FROM_L21_NON_REGENT_SHR_ALL
		0A0040000024C144	PM_MRK_DATA_FROM_L21_NON_REGENT_SHR_ALL_CYC
235	PM_DS_148	100FE	PM_INST_CMPL
		0A0060000022C040	PM_DPTEG_FROM_L21_NON_REGENT_SHR_ALL
		0A0060000023C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT_SHR_ALL
		0A0060000024C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_SHR_ALL_CYC
236	PM_DS_149	100FE	PM_INST_CMPL
		0A4020000002C040	PM_IPTEG_FROM_L21_NON_REGENT_MOD
		0A4020000003C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT_MOD
		0A4020000004C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_MOD_CYC
237	PM_DS_150	100FE	PM_INST_CMPL
		0A4040000002C040	PM_DATA_FROM_L21_NON_REGENT_MOD
		0A4040000003C142	PM_MRK_DATA_FROM_L21_NON_REGENT_MOD
		0A4040000004C144	PM_MRK_DATA_FROM_L21_NON_REGENT_MOD_CYC
238	PM_DS_151	100FE	PM_INST_CMPL
		0A4060000002C040	PM_DPTEG_FROM_L21_NON_REGENT_MOD
		0A4060000003C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT_MOD
		0A4060000004C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_MOD_CYC
239	PM_DS_152	100FE	PM_INST_CMPL
		0A4020000012C040	PM_IPTEG_FROM_L21_NON_REGENT_MOD_ALL
		0A4020000013C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT_MOD_ALL
		0A4020000014C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_MOD_ALL_CYC
240	PM_DS_153	100FE	PM_INST_CMPL
		0A4040000022C040	PM_DATA_FROM_L21_NON_REGENT_MOD_ALL
		0A4040000023C142	PM_MRK_DATA_FROM_L21_NON_REGENT_MOD_ALL
		0A4040000024C144	PM_MRK_DATA_FROM_L21_NON_REGENT_MOD_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 28 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
241	PM_DS_154	100FE	PM_INST_CMPL
		0A4060000022C040	PM_DPTEG_FROM_L21_NON_REGENT_MOD_ALL
		0A4060000023C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT_MOD_ALL
		0A4060000024C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_MOD_ALL_CYC
242	PM_DS_155	100FE	PM_INST_CMPL
		0A0100000002C040	PM_INST_FROM_L21_NON_REGENT
		0A0100000003C142	PM_MRK_INST_FROM_L21_NON_REGENT
		0A0100000004C144	PM_MRK_INST_FROM_L21_NON_REGENT_CYC
243	PM_DS_156	100FE	PM_INST_CMPL
		0A0120000002C040	PM_IPTEG_FROM_L21_NON_REGENT
		0A0120000003C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT
		0A0120000004C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_CYC
244	PM_DS_157	100FE	PM_INST_CMPL
		0A0140000002C040	PM_DATA_FROM_L21_NON_REGENT
		0A0140000003C142	PM_MRK_DATA_FROM_L21_NON_REGENT
		0A0140000004C144	PM_MRK_DATA_FROM_L21_NON_REGENT_CYC
245	PM_DS_158	100FE	PM_INST_CMPL
		0A0160000002C040	PM_DPTEG_FROM_L21_NON_REGENT
		0A0160000003C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT
		0A0160000004C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_CYC
246	PM_DS_159	100FE	PM_INST_CMPL
		0A0100000012C040	PM_INST_FROM_L21_NON_REGENT_ALL
		0A0100000013C142	PM_MRK_INST_FROM_L21_NON_REGENT_ALL
		0A0100000014C144	PM_MRK_INST_FROM_L21_NON_REGENT_ALL_CYC
247	PM_DS_160	100FE	PM_INST_CMPL
		0A0120000012C040	PM_IPTEG_FROM_L21_NON_REGENT_ALL
		0A0120000013C142	PM_MRK_IPTEG_FROM_L21_NON_REGENT_ALL
		0A0120000014C144	PM_MRK_IPTEG_FROM_L21_NON_REGENT_ALL_CYC
248	PM_DS_161	100FE	PM_INST_CMPL
		0A0140000022C040	PM_DATA_FROM_L21_NON_REGENT_ALL
		0A0140000023C142	PM_MRK_DATA_FROM_L21_NON_REGENT_ALL
		0A0140000024C144	PM_MRK_DATA_FROM_L21_NON_REGENT_ALL_CYC
249	PM_DS_162	100FE	PM_INST_CMPL
		0A0160000022C040	PM_DPTEG_FROM_L21_NON_REGENT_ALL
		0A0160000023C142	PM_MRK_DPTEG_FROM_L21_NON_REGENT_ALL
		0A0160000024C144	PM_MRK_DPTEG_FROM_L21_NON_REGENT_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 29 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
250	PM_DS_163	100FE	PM_INST_CMPL
		0A8020000002C040	PM_IPTEG_FROM_L31_NON_REGENT_SHR
		0A8020000003C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT_SHR
		0A8020000004C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_SHR_CYC
251	PM_DS_164	100FE	PM_INST_CMPL
		0A8040000002C040	PM_DATA_FROM_L31_NON_REGENT_SHR
		0A8040000003C142	PM_MRK_DATA_FROM_L31_NON_REGENT_SHR
		0A8040000004C144	PM_MRK_DATA_FROM_L31_NON_REGENT_SHR_CYC
252	PM_DS_165	100FE	PM_INST_CMPL
		0A8060000002C040	PM_DPTEG_FROM_L31_NON_REGENT_SHR
		0A8060000003C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT_SHR
		0A8060000004C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_SHR_CYC
253	PM_DS_166	100FE	PM_INST_CMPL
		0A8020000012C040	PM_IPTEG_FROM_L31_NON_REGENT_SHR_ALL
		0A8020000013C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT_SHR_ALL
		0A8020000014C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_SHR_ALL_CYC
254	PM_DS_167	100FE	PM_INST_CMPL
		0A8040000022C040	PM_DATA_FROM_L31_NON_REGENT_SHR_ALL
		0A8040000023C142	PM_MRK_DATA_FROM_L31_NON_REGENT_SHR_ALL
		0A8040000024C144	PM_MRK_DATA_FROM_L31_NON_REGENT_SHR_ALL_CYC
255	PM_DS_168	100FE	PM_INST_CMPL
		0A8060000022C040	PM_DPTEG_FROM_L31_NON_REGENT_SHR_ALL
		0A8060000023C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT_SHR_ALL
		0A8060000024C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_SHR_ALL_CYC
256	PM_DS_169	100FE	PM_INST_CMPL
		0AC020000002C040	PM_IPTEG_FROM_L31_NON_REGENT_MOD
		0AC020000003C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT_MOD
		0AC020000004C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_MOD_CYC
257	PM_DS_170	100FE	PM_INST_CMPL
		0AC040000002C040	PM_DATA_FROM_L31_NON_REGENT_MOD
		0AC040000003C142	PM_MRK_DATA_FROM_L31_NON_REGENT_MOD
		0AC040000004C144	PM_MRK_DATA_FROM_L31_NON_REGENT_MOD_CYC
258	PM_DS_171	100FE	PM_INST_CMPL
		0AC060000002C040	PM_DPTEG_FROM_L31_NON_REGENT_MOD
		0AC060000003C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT_MOD
		0AC060000004C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_MOD_CYC



Table E-35. Power10 Events by Group (Sheet 30 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
259	PM_DS_172	100FE	PM_INST_CMPL
		0AC020000012C040	PM_IPTEG_FROM_L31_NON_REGENT_MOD_ALL
		0AC020000013C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT_MOD_ALL
		0AC020000014C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_MOD_ALL_CYC
260	PM_DS_173	100FE	PM_INST_CMPL
		0AC040000022C040	PM_DATA_FROM_L31_NON_REGENT_MOD_ALL
		0AC040000023C142	PM_MRK_DATA_FROM_L31_NON_REGENT_MOD_ALL
		0AC040000024C144	PM_MRK_DATA_FROM_L31_NON_REGENT_MOD_ALL_CYC
261	PM_DS_174	100FE	PM_INST_CMPL
		0AC060000022C040	PM_DPTEG_FROM_L31_NON_REGENT_MOD_ALL
		0AC060000023C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT_MOD_ALL
		0AC060000024C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_MOD_ALL_CYC
262	PM_DS_175	100FE	PM_INST_CMPL
		0A8100000002C040	PM_INST_FROM_L31_NON_REGENT
		0A8100000003C142	PM_MRK_INST_FROM_L31_NON_REGENT
		0A8100000004C144	PM_MRK_INST_FROM_L31_NON_REGENT_CYC
263	PM_DS_176	100FE	PM_INST_CMPL
		0A8120000002C040	PM_IPTEG_FROM_L31_NON_REGENT
		0A8120000003C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT
		0A8120000004C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_CYC
264	PM_DS_177	100FE	PM_INST_CMPL
		0A8140000002C040	PM_DATA_FROM_L31_NON_REGENT
		0A8140000003C142	PM_MRK_DATA_FROM_L31_NON_REGENT
		0A8140000004C144	PM_MRK_DATA_FROM_L31_NON_REGENT_CYC
265	PM_DS_178	100FE	PM_INST_CMPL
		0A8160000002C040	PM_DPTEG_FROM_L31_NON_REGENT
		0A8160000003C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT
		0A8160000004C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_CYC
266	PM_DS_179	100FE	PM_INST_CMPL
		0A8100000012C040	PM_INST_FROM_L31_NON_REGENT_ALL
		0A8100000013C142	PM_MRK_INST_FROM_L31_NON_REGENT_ALL
		0A8100000014C144	PM_MRK_INST_FROM_L31_NON_REGENT_ALL_CYC
267	PM_DS_180	100FE	PM_INST_CMPL
		0A8120000012C040	PM_IPTEG_FROM_L31_NON_REGENT_ALL
		0A8120000013C142	PM_MRK_IPTEG_FROM_L31_NON_REGENT_ALL
		0A8120000014C144	PM_MRK_IPTEG_FROM_L31_NON_REGENT_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 31 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
268	PM_DS_181	100FE	PM_INST_CMPL
		0A8140000022C040	PM_DATA_FROM_L31_NON_REGENT_ALL
		0A8140000023C142	PM_MRK_DATA_FROM_L31_NON_REGENT_ALL
		0A8140000024C144	PM_MRK_DATA_FROM_L31_NON_REGENT_ALL_CYC
269	PM_DS_182	100FE	PM_INST_CMPL
		0A8160000022C040	PM_DPTEG_FROM_L31_NON_REGENT_ALL
		0A8160000023C142	PM_MRK_DPTEG_FROM_L31_NON_REGENT_ALL
		0A8160000024C144	PM_MRK_DPTEG_FROM_L31_NON_REGENT_ALL_CYC
270	PM_DS_183	100FE	PM_INST_CMPL
		0A0220000002C040	PM_IPTEG_FROM_NON_REGENT_L2L3_SHR
		0A0220000003C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_SHR
		0A0220000004C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_SHR_CYC
271	PM_DS_184	100FE	PM_INST_CMPL
		0A0240000002C040	PM_DATA_FROM_NON_REGENT_L2L3_SHR
		0A0240000003C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3_SHR
		0A0240000004C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_SHR_CYC
272	PM_DS_185	100FE	PM_INST_CMPL
		0A0260000002C040	PM_DPTEG_FROM_NON_REGENT_L2L3_SHR
		0A0260000003C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_SHR
		0A0260000004C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_SHR_CYC
273	PM_DS_186	100FE	PM_INST_CMPL
		0A0220000012C040	PM_IPTEG_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0220000013C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0220000014C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_SHR_ALL_CYC
274	PM_DS_187	100FE	PM_INST_CMPL
		0A0240000022C040	PM_DATA_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0240000023C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0240000024C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_SHR_ALL_CYC
275	PM_DS_188	100FE	PM_INST_CMPL
		0A0260000022C040	PM_DPTEG_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0260000023C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_SHR_ALL
		0A0260000024C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_SHR_ALL_CYC
276	PM_DS_189	100FE	PM_INST_CMPL
		0A4220000002C040	PM_IPTEG_FROM_NON_REGENT_L2L3_MOD
		0A4220000003C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_MOD
		0A4220000004C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_MOD_CYC



Table E-35. Power10 Events by Group (Sheet 32 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
277	PM_DS_190	100FE	PM_INST_CMPL
		0A4240000002C040	PM_DATA_FROM_NON_REGENT_L2L3_MOD
		0A4240000003C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3_MOD
		0A4240000004C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_MOD_CYC
278	PM_DS_191	100FE	PM_INST_CMPL
		0A4260000002C040	PM_DPTEG_FROM_NON_REGENT_L2L3_MOD
		0A4260000003C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_MOD
		0A4260000004C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_MOD_CYC
279	PM_DS_192	100FE	PM_INST_CMPL
		0A4220000012C040	PM_IPTEG_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4220000013C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4220000014C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_MOD_ALL_CYC
280	PM_DS_193	100FE	PM_INST_CMPL
		0A4240000022C040	PM_DATA_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4240000023C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4240000024C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_MOD_ALL_CYC
281	PM_DS_194	100FE	PM_INST_CMPL
		0A4260000022C040	PM_DPTEG_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4260000023C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_MOD_ALL
		0A4260000024C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_MOD_ALL_CYC
282	PM_DS_195	100FE	PM_INST_CMPL
		0A0300000002C040	PM_INST_FROM_NON_REGENT_L2L3
		0A0300000003C142	PM_MRK_INST_FROM_NON_REGENT_L2L3
		0A0300000004C144	PM_MRK_INST_FROM_NON_REGENT_L2L3_CYC
283	PM_DS_196	100FE	PM_INST_CMPL
		0A0320000002C040	PM_IPTEG_FROM_NON_REGENT_L2L3
		0A0320000003C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3
		0A0320000004C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_CYC
284	PM_DS_197	100FE	PM_INST_CMPL
		0A0340000002C040	PM_DATA_FROM_NON_REGENT_L2L3
		0A0340000003C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3
		0A0340000004C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_CYC
285	PM_DS_198	100FE	PM_INST_CMPL
		0A0360000002C040	PM_DPTEG_FROM_NON_REGENT_L2L3
		0A0360000003C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3
		0A0360000004C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_CYC



Table E-35. Power10 Events by Group (Sheet 33 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
286	PM_DS_199	100FE	PM_INST_CMPL
		0A0300000012C040	PM_INST_FROM_NON_REGENT_L2L3_ALL
		0A0300000013C142	PM_MRK_INST_FROM_NON_REGENT_L2L3_ALL
		0A0300000014C144	PM_MRK_INST_FROM_NON_REGENT_L2L3_ALL_CYC
287	PM_DS_200	100FE	PM_INST_CMPL
		0A0320000012C040	PM_IPTEG_FROM_NON_REGENT_L2L3_ALL
		0A0320000013C142	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_ALL
		0A0320000014C144	PM_MRK_IPTEG_FROM_NON_REGENT_L2L3_ALL_CYC
288	PM_DS_201	100FE	PM_INST_CMPL
		0A0340000022C040	PM_DATA_FROM_NON_REGENT_L2L3_ALL
		0A0340000023C142	PM_MRK_DATA_FROM_NON_REGENT_L2L3_ALL
		0A0340000024C144	PM_MRK_DATA_FROM_NON_REGENT_L2L3_ALL_CYC
289	PM_DS_202	100FE	PM_INST_CMPL
		0A0360000022C040	PM_DPTEG_FROM_NON_REGENT_L2L3_ALL
		0A0360000023C142	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_ALL
		0A0360000024C144	PM_MRK_DPTEG_FROM_NON_REGENT_L2L3_ALL_CYC
290	PM_DS_203	100FE	PM_INST_CMPL
		094100000002C040	PM_INST_FROM_LMEM
		094100000003C142	PM_MRK_INST_FROM_LMEM
		094100000004C144	PM_MRK_INST_FROM_LMEM_CYC
291	PM_DS_204	100FE	PM_INST_CMPL
		094020000002C040	PM_IPTEG_FROM_LMEM
		094020000003C142	PM_MRK_IPTEG_FROM_LMEM
		094020000004C144	PM_MRK_IPTEG_FROM_LMEM_CYC
292	PM_DS_205	100FE	PM_INST_CMPL
		094040000002C040	PM_DATA_FROM_LMEM
		094040000003C142	PM_MRK_DATA_FROM_LMEM
		094040000004C144	PM_MRK_DATA_FROM_LMEM_CYC
293	PM_DS_206	100FE	PM_INST_CMPL
		094060000002C040	PM_DPTEG_FROM_LMEM
		094060000003C142	PM_MRK_DPTEG_FROM_LMEM
		094060000004C144	PM_MRK_DPTEG_FROM_LMEM_CYC
294	PM_DS_207	100FE	PM_INST_CMPL
		094100000012C040	PM_INST_FROM_LMEM_ALL
		094100000013C142	PM_MRK_INST_FROM_LMEM_ALL
		094100000014C144	PM_MRK_INST_FROM_LMEM_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 34 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
295	PM_DS_208	100FE	PM_INST_CMPL
		094020000012C040	PM_IPTEG_FROM_LMEM_ALL
		094020000013C142	PM_MRK_IPTEG_FROM_LMEM_ALL
		094020000014C144	PM_MRK_IPTEG_FROM_LMEM_ALL_CYC
296	PM_DS_209	100FE	PM_INST_CMPL
		094040000022C040	PM_DATA_FROM_LMEM_ALL
		094040000023C142	PM_MRK_DATA_FROM_LMEM_ALL
		094040000024C144	PM_MRK_DATA_FROM_LMEM_ALL_CYC
297	PM_DS_210	100FE	PM_INST_CMPL
		094060000022C040	PM_DPTEG_FROM_LMEM_ALL
		094060000023C142	PM_MRK_DPTEG_FROM_LMEM_ALL
		094060000024C144	PM_MRK_DPTEG_FROM_LMEM_ALL_CYC
298	PM_DS_211	100FE	PM_INST_CMPL
		098020000002C040	PM_IPTEG_FROM_L_OC_CACHE
		098020000003C142	PM_MRK_IPTEG_FROM_L_OC_CACHE
		098020000004C144	PM_MRK_IPTEG_FROM_L_OC_CACHE_CYC
299	PM_DS_212	100FE	PM_INST_CMPL
		098040000002C040	PM_DATA_FROM_L_OC_CACHE
		098040000003C142	PM_MRK_DATA_FROM_L_OC_CACHE
		098040000004C144	PM_MRK_DATA_FROM_L_OC_CACHE_CYC
300	PM_DS_213	100FE	PM_INST_CMPL
		098060000002C040	PM_DPTEG_FROM_L_OC_CACHE
		098060000003C142	PM_MRK_DPTEG_FROM_L_OC_CACHE
		098060000004C144	PM_MRK_DPTEG_FROM_L_OC_CACHE_CYC
301	PM_DS_214	100FE	PM_INST_CMPL
		098020000012C040	PM_IPTEG_FROM_L_OC_CACHE_ALL
		098020000013C142	PM_MRK_IPTEG_FROM_L_OC_CACHE_ALL
		098020000014C144	PM_MRK_IPTEG_FROM_L_OC_CACHE_ALL_CYC
302	PM_DS_215	100FE	PM_INST_CMPL
		098040000022C040	PM_DATA_FROM_L_OC_CACHE_ALL
		098040000023C142	PM_MRK_DATA_FROM_L_OC_CACHE_ALL
		098040000024C144	PM_MRK_DATA_FROM_L_OC_CACHE_ALL_CYC
303	PM_DS_216	100FE	PM_INST_CMPL
		098060000022C040	PM_DPTEG_FROM_L_OC_CACHE_ALL
		098060000023C142	PM_MRK_DPTEG_FROM_L_OC_CACHE_ALL
		098060000024C144	PM_MRK_DPTEG_FROM_L_OC_CACHE_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 35 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
304	PM_DS_217	100FE	PM_INST_CMPL
		09C020000002C040	PM_IPTEG_FROM_L_OC_MEM
		09C020000003C142	PM_MRK_IPTEG_FROM_L_OC_MEM
		09C020000004C144	PM_MRK_IPTEG_FROM_L_OC_MEM_CYC
305	PM_DS_218	100FE	PM_INST_CMPL
		09C040000002C040	PM_DATA_FROM_L_OC_MEM
		09C040000003C142	PM_MRK_DATA_FROM_L_OC_MEM
		09C040000004C144	PM_MRK_DATA_FROM_L_OC_MEM_CYC
306	PM_DS_219	100FE	PM_INST_CMPL
		09C060000002C040	PM_DPTEG_FROM_L_OC_MEM
		09C060000003C142	PM_MRK_DPTEG_FROM_L_OC_MEM
		09C060000004C144	PM_MRK_DPTEG_FROM_L_OC_MEM_CYC
307	PM_DS_220	100FE	PM_INST_CMPL
		09C020000012C040	PM_IPTEG_FROM_L_OC_MEM_ALL
		09C020000013C142	PM_MRK_IPTEG_FROM_L_OC_MEM_ALL
		09C020000014C144	PM_MRK_IPTEG_FROM_L_OC_MEM_ALL_CYC
308	PM_DS_221	100FE	PM_INST_CMPL
		09C040000022C040	PM_DATA_FROM_L_OC_MEM_ALL
		09C040000023C142	PM_MRK_DATA_FROM_L_OC_MEM_ALL
		09C040000024C144	PM_MRK_DATA_FROM_L_OC_MEM_ALL_CYC
309	PM_DS_222	100FE	PM_INST_CMPL
		09C060000022C040	PM_DPTEG_FROM_L_OC_MEM_ALL
		09C060000023C142	PM_MRK_DPTEG_FROM_L_OC_MEM_ALL
		09C060000024C144	PM_MRK_DPTEG_FROM_L_OC_MEM_ALL_CYC
310	PM_DS_223	100FE	PM_INST_CMPL
		098100000002C040	PM_INST_FROM_L_OC_ANY
		098100000003C142	PM_MRK_INST_FROM_L_OC_ANY
		098100000004C144	PM_MRK_INST_FROM_L_OC_ANY_CYC
311	PM_DS_224	100FE	PM_INST_CMPL
		098120000002C040	PM_IPTEG_FROM_L_OC_ANY
		098120000003C142	PM_MRK_IPTEG_FROM_L_OC_ANY
		098120000004C144	PM_MRK_IPTEG_FROM_L_OC_ANY_CYC
312	PM_DS_225	100FE	PM_INST_CMPL
		098140000002C040	PM_DATA_FROM_L_OC_ANY
		098140000003C142	PM_MRK_DATA_FROM_L_OC_ANY
		098140000004C144	PM_MRK_DATA_FROM_L_OC_ANY_CYC



Table E-35. Power10 Events by Group (Sheet 36 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
313	PM_DS_226	100FE	PM_INST_CMPL
		098160000002C040	PM_DPTEG_FROM_L_OC_ANY
		098160000003C142	PM_MRK_DPTEG_FROM_L_OC_ANY
		098160000004C144	PM_MRK_DPTEG_FROM_L_OC_ANY_CYC
314	PM_DS_227	100FE	PM_INST_CMPL
		098100000012C040	PM_INST_FROM_L_OC_ANY_ALL
		098100000013C142	PM_MRK_INST_FROM_L_OC_ANY_ALL
		098100000014C144	PM_MRK_INST_FROM_L_OC_ANY_ALL_CYC
315	PM_DS_228	100FE	PM_INST_CMPL
		098120000012C040	PM_IPTEG_FROM_L_OC_ANY_ALL
		098120000013C142	PM_MRK_IPTEG_FROM_L_OC_ANY_ALL
		098120000014C144	PM_MRK_IPTEG_FROM_L_OC_ANY_ALL_CYC
316	PM_DS_229	100FE	PM_INST_CMPL
		098140000022C040	PM_DATA_FROM_L_OC_ANY_ALL
		098140000023C142	PM_MRK_DATA_FROM_L_OC_ANY_ALL
		098140000024C144	PM_MRK_DATA_FROM_L_OC_ANY_ALL_CYC
317	PM_DS_230	100FE	PM_INST_CMPL
		098160000022C040	PM_DPTEG_FROM_L_OC_ANY_ALL
		098160000023C142	PM_MRK_DPTEG_FROM_L_OC_ANY_ALL
		098160000024C144	PM_MRK_DPTEG_FROM_L_OC_ANY_ALL_CYC
318	PM_DS_231	100FE	PM_INST_CMPL
		0C0020000002C040	PM_IPTEG_FROM_RL2_SHR
		0C0020000003C142	PM_MRK_IPTEG_FROM_RL2_SHR
		0C0020000004C144	PM_MRK_IPTEG_FROM_RL2_SHR_CYC
319	PM_DS_232	100FE	PM_INST_CMPL
		0C0040000002C040	PM_DATA_FROM_RL2_SHR
		0C0040000003C142	PM_MRK_DATA_FROM_RL2_SHR
		0C0040000004C144	PM_MRK_DATA_FROM_RL2_SHR_CYC
320	PM_DS_233	100FE	PM_INST_CMPL
		0C0060000002C040	PM_DPTEG_FROM_RL2_SHR
		0C0060000003C142	PM_MRK_DPTEG_FROM_RL2_SHR
		0C0060000004C144	PM_MRK_DPTEG_FROM_RL2_SHR_CYC
321	PM_DS_234	100FE	PM_INST_CMPL
		0C0020000012C040	PM_IPTEG_FROM_RL2_SHR_ALL
		0C0020000013C142	PM_MRK_IPTEG_FROM_RL2_SHR_ALL
		0C0020000014C144	PM_MRK_IPTEG_FROM_RL2_SHR_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 37 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
322	PM_DS_235	100FE	PM_INST_CMPL
		0C0040000022C040	PM_DATA_FROM_RL2_SHR_ALL
		0C0040000023C142	PM_MRK_DATA_FROM_RL2_SHR_ALL
		0C0040000024C144	PM_MRK_DATA_FROM_RL2_SHR_ALL_CYC
323	PM_DS_236	100FE	PM_INST_CMPL
		0C0060000022C040	PM_DPTEG_FROM_RL2_SHR_ALL
		0C0060000023C142	PM_MRK_DPTEG_FROM_RL2_SHR_ALL
		0C0060000024C144	PM_MRK_DPTEG_FROM_RL2_SHR_ALL_CYC
324	PM_DS_237	100FE	PM_INST_CMPL
		0C4020000002C040	PM_IPTEG_FROM_RL2_MOD
		0C4020000003C142	PM_MRK_IPTEG_FROM_RL2_MOD
		0C4020000004C144	PM_MRK_IPTEG_FROM_RL2_MOD_CYC
325	PM_DS_238	100FE	PM_INST_CMPL
		0C4040000002C040	PM_DATA_FROM_RL2_MOD
		0C4040000003C142	PM_MRK_DATA_FROM_RL2_MOD
		0C4040000004C144	PM_MRK_DATA_FROM_RL2_MOD_CYC
326	PM_DS_239	100FE	PM_INST_CMPL
		0C4060000002C040	PM_DPTEG_FROM_RL2_MOD
		0C4060000003C142	PM_MRK_DPTEG_FROM_RL2_MOD
		0C4060000004C144	PM_MRK_DPTEG_FROM_RL2_MOD_CYC
327	PM_DS_240	100FE	PM_INST_CMPL
		0C4020000012C040	PM_IPTEG_FROM_RL2_MOD_ALL
		0C4020000013C142	PM_MRK_IPTEG_FROM_RL2_MOD_ALL
		0C4020000014C144	PM_MRK_IPTEG_FROM_RL2_MOD_ALL_CYC
328	PM_DS_241	100FE	PM_INST_CMPL
		0C4040000022C040	PM_DATA_FROM_RL2_MOD_ALL
		0C4040000023C142	PM_MRK_DATA_FROM_RL2_MOD_ALL
		0C4040000024C144	PM_MRK_DATA_FROM_RL2_MOD_ALL_CYC
329	PM_DS_242	100FE	PM_INST_CMPL
		0C4060000022C040	PM_DPTEG_FROM_RL2_MOD_ALL
		0C4060000023C142	PM_MRK_DPTEG_FROM_RL2_MOD_ALL
		0C4060000024C144	PM_MRK_DPTEG_FROM_RL2_MOD_ALL_CYC
330	PM_DS_243	100FE	PM_INST_CMPL
		0C0100000002C040	PM_INST_FROM_RL2
		0C0100000003C142	PM_MRK_INST_FROM_RL2
		0C0100000004C144	PM_MRK_INST_FROM_RL2_CYC



Table E-35. Power10 Events by Group (Sheet 38 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
331	PM_DS_244	100FE	PM_INST_CMPL
		0C0120000002C040	PM_IPTEG_FROM_RL2
		0C0120000003C142	PM_MRK_IPTEG_FROM_RL2
		0C0120000004C144	PM_MRK_IPTEG_FROM_RL2_CYC
332	PM_DS_245	100FE	PM_INST_CMPL
		0C0140000002C040	PM_DATA_FROM_RL2
		0C0140000003C142	PM_MRK_DATA_FROM_RL2
		0C0140000004C144	PM_MRK_DATA_FROM_RL2_CYC
333	PM_DS_246	100FE	PM_INST_CMPL
		0C0160000002C040	PM_DPTEG_FROM_RL2
		0C0160000003C142	PM_MRK_DPTEG_FROM_RL2
		0C0160000004C144	PM_MRK_DPTEG_FROM_RL2_CYC
334	PM_DS_247	100FE	PM_INST_CMPL
		0C0100000012C040	PM_INST_FROM_RL2_ALL
		0C0100000013C142	PM_MRK_INST_FROM_RL2_ALL
		0C0100000014C144	PM_MRK_INST_FROM_RL2_ALL_CYC
335	PM_DS_248	100FE	PM_INST_CMPL
		0C0120000012C040	PM_IPTEG_FROM_RL2_ALL
		0C0120000013C142	PM_MRK_IPTEG_FROM_RL2_ALL
		0C0120000014C144	PM_MRK_IPTEG_FROM_RL2_ALL_CYC
336	PM_DS_249	100FE	PM_INST_CMPL
		0C0140000022C040	PM_DATA_FROM_RL2_ALL
		0C0140000023C142	PM_MRK_DATA_FROM_RL2_ALL
		0C0140000024C144	PM_MRK_DATA_FROM_RL2_ALL_CYC
337	PM_DS_250	100FE	PM_INST_CMPL
		0C0160000022C040	PM_DPTEG_FROM_RL2_ALL
		0C0160000023C142	PM_MRK_DPTEG_FROM_RL2_ALL
		0C0160000024C144	PM_MRK_DPTEG_FROM_RL2_ALL_CYC
338	PM_DS_251	100FE	PM_INST_CMPL
		0C8020000002C040	PM_IPTEG_FROM_RL3_SHR
		0C8020000003C142	PM_MRK_IPTEG_FROM_RL3_SHR
		0C8020000004C144	PM_MRK_IPTEG_FROM_RL3_SHR_CYC
339	PM_DS_252	100FE	PM_INST_CMPL
		0C8040000002C040	PM_DATA_FROM_RL3_SHR
		0C8040000003C142	PM_MRK_DATA_FROM_RL3_SHR
		0C8040000004C144	PM_MRK_DATA_FROM_RL3_SHR_CYC



Table E-35. Power10 Events by Group (Sheet 39 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
340	PM_DS_253	100FE	PM_INST_CMPL
		0C8060000002C040	PM_DPTEG_FROM_RL3_SHR
		0C8060000003C142	PM_MRK_DPTEG_FROM_RL3_SHR
		0C8060000004C144	PM_MRK_DPTEG_FROM_RL3_SHR_CYC
341	PM_DS_254	100FE	PM_INST_CMPL
		0C8020000012C040	PM_IPTEG_FROM_RL3_SHR_ALL
		0C8020000013C142	PM_MRK_IPTEG_FROM_RL3_SHR_ALL
		0C8020000014C144	PM_MRK_IPTEG_FROM_RL3_SHR_ALL_CYC
342	PM_DS_255	100FE	PM_INST_CMPL
		0C8040000022C040	PM_DATA_FROM_RL3_SHR_ALL
		0C8040000023C142	PM_MRK_DATA_FROM_RL3_SHR_ALL
		0C8040000024C144	PM_MRK_DATA_FROM_RL3_SHR_ALL_CYC
343	PM_DS_256	100FE	PM_INST_CMPL
		0C8060000022C040	PM_DPTEG_FROM_RL3_SHR_ALL
		0C8060000023C142	PM_MRK_DPTEG_FROM_RL3_SHR_ALL
		0C8060000024C144	PM_MRK_DPTEG_FROM_RL3_SHR_ALL_CYC
344	PM_DS_257	100FE	PM_INST_CMPL
		0CC020000002C040	PM_IPTEG_FROM_RL3_MOD
		0CC020000003C142	PM_MRK_IPTEG_FROM_RL3_MOD
		0CC020000004C144	PM_MRK_IPTEG_FROM_RL3_MOD_CYC
345	PM_DS_258	100FE	PM_INST_CMPL
		0CC040000002C040	PM_DATA_FROM_RL3_MOD
		0CC040000003C142	PM_MRK_DATA_FROM_RL3_MOD
		0CC040000004C144	PM_MRK_DATA_FROM_RL3_MOD_CYC
346	PM_DS_259	100FE	PM_INST_CMPL
		0CC060000002C040	PM_DPTEG_FROM_RL3_MOD
		0CC060000003C142	PM_MRK_DPTEG_FROM_RL3_MOD
		0CC060000004C144	PM_MRK_DPTEG_FROM_RL3_MOD_CYC
347	PM_DS_260	100FE	PM_INST_CMPL
		0CC020000012C040	PM_IPTEG_FROM_RL3_MOD_ALL
		0CC020000013C142	PM_MRK_IPTEG_FROM_RL3_MOD_ALL
		0CC020000014C144	PM_MRK_IPTEG_FROM_RL3_MOD_ALL_CYC
348	PM_DS_261	100FE	PM_INST_CMPL
		0CC040000022C040	PM_DATA_FROM_RL3_MOD_ALL
		0CC040000023C142	PM_MRK_DATA_FROM_RL3_MOD_ALL
		0CC040000024C144	PM_MRK_DATA_FROM_RL3_MOD_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 40 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
349	PM_DS_262	100FE	PM_INST_CMPL
		0CC060000022C040	PM_DPTEG_FROM_RL3_MOD_ALL
		0CC060000023C142	PM_MRK_DPTEG_FROM_RL3_MOD_ALL
		0CC060000024C144	PM_MRK_DPTEG_FROM_RL3_MOD_ALL_CYC
350	PM_DS_263	100FE	PM_INST_CMPL
		0C8100000002C040	PM_INST_FROM_RL3
		0C8100000003C142	PM_MRK_INST_FROM_RL3
		0C8100000004C144	PM_MRK_INST_FROM_RL3_CYC
351	PM_DS_264	100FE	PM_INST_CMPL
		0C8120000002C040	PM_IPTEG_FROM_RL3
		0C8120000003C142	PM_MRK_IPTEG_FROM_RL3
		0C8120000004C144	PM_MRK_IPTEG_FROM_RL3_CYC
352	PM_DS_265	100FE	PM_INST_CMPL
		0C8140000002C040	PM_DATA_FROM_RL3
		0C8140000003C142	PM_MRK_DATA_FROM_RL3
		0C8140000004C144	PM_MRK_DATA_FROM_RL3_CYC
353	PM_DS_266	100FE	PM_INST_CMPL
		0C8160000002C040	PM_DPTEG_FROM_RL3
		0C8160000003C142	PM_MRK_DPTEG_FROM_RL3
		0C8160000004C144	PM_MRK_DPTEG_FROM_RL3_CYC
354	PM_DS_267	100FE	PM_INST_CMPL
		0C8100000012C040	PM_INST_FROM_RL3_ALL
		0C8100000013C142	PM_MRK_INST_FROM_RL3_ALL
		0C8100000014C144	PM_MRK_INST_FROM_RL3_ALL_CYC
355	PM_DS_268	100FE	PM_INST_CMPL
		0C8120000012C040	PM_IPTEG_FROM_RL3_ALL
		0C8120000013C142	PM_MRK_IPTEG_FROM_RL3_ALL
		0C8120000014C144	PM_MRK_IPTEG_FROM_RL3_ALL_CYC
356	PM_DS_269	100FE	PM_INST_CMPL
		0C8140000022C040	PM_DATA_FROM_RL3_ALL
		0C8140000023C142	PM_MRK_DATA_FROM_RL3_ALL
		0C8140000024C144	PM_MRK_DATA_FROM_RL3_ALL_CYC
357	PM_DS_270	100FE	PM_INST_CMPL
		0C8160000022C040	PM_DPTEG_FROM_RL3_ALL
		0C8160000023C142	PM_MRK_DPTEG_FROM_RL3_ALL
		0C8160000024C144	PM_MRK_DPTEG_FROM_RL3_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 41 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
358	PM_DS_271	100FE	PM_INST_CMPL
		0C0220000002C040	PM_IPTEG_FROM_RL2L3_SHR
		0C0220000003C142	PM_MRK_IPTEG_FROM_RL2L3_SHR
		0C0220000004C144	PM_MRK_IPTEG_FROM_RL2L3_SHR_CYC
359	PM_DS_272	100FE	PM_INST_CMPL
		0C0240000002C040	PM_DATA_FROM_RL2L3_SHR
		0C0240000003C142	PM_MRK_DATA_FROM_RL2L3_SHR
		0C0240000004C144	PM_MRK_DATA_FROM_RL2L3_SHR_CYC
360	PM_DS_273	100FE	PM_INST_CMPL
		0C0260000002C040	PM_DPTEG_FROM_RL2L3_SHR
		0C0260000003C142	PM_MRK_DPTEG_FROM_RL2L3_SHR
		0C0260000004C144	PM_MRK_DPTEG_FROM_RL2L3_SHR_CYC
361	PM_DS_274	100FE	PM_INST_CMPL
		0C0220000012C040	PM_IPTEG_FROM_RL2L3_SHR_ALL
		0C0220000013C142	PM_MRK_IPTEG_FROM_RL2L3_SHR_ALL
		0C0220000014C144	PM_MRK_IPTEG_FROM_RL2L3_SHR_ALL_CYC
362	PM_DS_275	100FE	PM_INST_CMPL
		0C0240000022C040	PM_DATA_FROM_RL2L3_SHR_ALL
		0C0240000023C142	PM_MRK_DATA_FROM_RL2L3_SHR_ALL
		0C0240000024C144	PM_MRK_DATA_FROM_RL2L3_SHR_ALL_CYC
363	PM_DS_276	100FE	PM_INST_CMPL
		0C0260000022C040	PM_DPTEG_FROM_RL2L3_SHR_ALL
		0C0260000023C142	PM_MRK_DPTEG_FROM_RL2L3_SHR_ALL
		0C0260000024C144	PM_MRK_DPTEG_FROM_RL2L3_SHR_ALL_CYC
364	PM_DS_277	100FE	PM_INST_CMPL
		0C4220000002C040	PM_IPTEG_FROM_RL2L3_MOD
		0C4220000003C142	PM_MRK_IPTEG_FROM_RL2L3_MOD
		0C4220000004C144	PM_MRK_IPTEG_FROM_RL2L3_MOD_CYC
365	PM_DS_278	100FE	PM_INST_CMPL
		0C4240000002C040	PM_DATA_FROM_RL2L3_MOD
		0C4240000003C142	PM_MRK_DATA_FROM_RL2L3_MOD
		0C4240000004C144	PM_MRK_DATA_FROM_RL2L3_MOD_CYC
366	PM_DS_279	100FE	PM_INST_CMPL
		0C4260000002C040	PM_DPTEG_FROM_RL2L3_MOD
		0C4260000003C142	PM_MRK_DPTEG_FROM_RL2L3_MOD
		0C4260000004C144	PM_MRK_DPTEG_FROM_RL2L3_MOD_CYC



Table E-35. Power10 Events by Group (Sheet 42 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
367	PM_DS_280	100FE	PM_INST_CMPL
		0C4220000012C040	PM_IPTEG_FROM_RL2L3_MOD_ALL
		0C4220000013C142	PM_MRK_IPTEG_FROM_RL2L3_MOD_ALL
		0C4220000014C144	PM_MRK_IPTEG_FROM_RL2L3_MOD_ALL_CYC
368	PM_DS_281	100FE	PM_INST_CMPL
		0C4240000022C040	PM_DATA_FROM_RL2L3_MOD_ALL
		0C4240000023C142	PM_MRK_DATA_FROM_RL2L3_MOD_ALL
		0C4240000024C144	PM_MRK_DATA_FROM_RL2L3_MOD_ALL_CYC
369	PM_DS_282	100FE	PM_INST_CMPL
		0C4260000022C040	PM_DPTEG_FROM_RL2L3_MOD_ALL
		0C4260000023C142	PM_MRK_DPTEG_FROM_RL2L3_MOD_ALL
		0C4260000024C144	PM_MRK_DPTEG_FROM_RL2L3_MOD_ALL_CYC
370	PM_DS_283	100FE	PM_INST_CMPL
		0C0300000002C040	PM_INST_FROM_RL2L3
		0C0300000003C142	PM_MRK_INST_FROM_RL2L3
		0C0300000004C144	PM_MRK_INST_FROM_RL2L3_CYC
371	PM_DS_284	100FE	PM_INST_CMPL
		0C0320000002C040	PM_IPTEG_FROM_RL2L3
		0C0320000003C142	PM_MRK_IPTEG_FROM_RL2L3
		0C0320000004C144	PM_MRK_IPTEG_FROM_RL2L3_CYC
372	PM_DS_285	100FE	PM_INST_CMPL
		0C0340000002C040	PM_DATA_FROM_RL2L3
		0C0340000003C142	PM_MRK_DATA_FROM_RL2L3
		0C0340000004C144	PM_MRK_DATA_FROM_RL2L3_CYC
373	PM_DS_286	100FE	PM_INST_CMPL
		0C0360000002C040	PM_DPTEG_FROM_RL2L3
		0C0360000003C142	PM_MRK_DPTEG_FROM_RL2L3
		0C0360000004C144	PM_MRK_DPTEG_FROM_RL2L3_CYC
374	PM_DS_287	100FE	PM_INST_CMPL
		0C0300000012C040	PM_INST_FROM_RL2L3_ALL
		0C0300000013C142	PM_MRK_INST_FROM_RL2L3_ALL
		0C0300000014C144	PM_MRK_INST_FROM_RL2L3_ALL_CYC
375	PM_DS_288	100FE	PM_INST_CMPL
		0C0320000012C040	PM_IPTEG_FROM_RL2L3_ALL
		0C0320000013C142	PM_MRK_IPTEG_FROM_RL2L3_ALL
		0C0320000014C144	PM_MRK_IPTEG_FROM_RL2L3_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 43 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
376	PM_DS_289	100FE	PM_INST_CMPL
		0C0340000022C040	PM_DATA_FROM_RL2L3_ALL
		0C0340000023C142	PM_MRK_DATA_FROM_RL2L3_ALL
		0C0340000024C144	PM_MRK_DATA_FROM_RL2L3_ALL_CYC
377	PM_DS_290	100FE	PM_INST_CMPL
		0C0360000022C040	PM_DPTEG_FROM_RL2L3_ALL
		0C0360000023C142	PM_MRK_DPTEG_FROM_RL2L3_ALL
		0C0360000024C144	PM_MRK_DPTEG_FROM_RL2L3_ALL_CYC
378	PM_DS_291	100FE	PM_INST_CMPL
		0D4100000002C040	PM_INST_FROM_RMEM
		0D4100000003C142	PM_MRK_INST_FROM_RMEM
		0D4100000004C144	PM_MRK_INST_FROM_RMEM_CYC
379	PM_DS_292	100FE	PM_INST_CMPL
		0D4020000002C040	PM_IPTEG_FROM_RMEM
		0D4020000003C142	PM_MRK_IPTEG_FROM_RMEM
		0D4020000004C144	PM_MRK_IPTEG_FROM_RMEM_CYC
380	PM_DS_293	100FE	PM_INST_CMPL
		0D4040000002C040	PM_DATA_FROM_RMEM
		0D4040000003C142	PM_MRK_DATA_FROM_RMEM
		0D4040000004C144	PM_MRK_DATA_FROM_RMEM_CYC
381	PM_DS_294	100FE	PM_INST_CMPL
		0D4060000002C040	PM_DPTEG_FROM_RMEM
		0D4060000003C142	PM_MRK_DPTEG_FROM_RMEM
		0D4060000004C144	PM_MRK_DPTEG_FROM_RMEM_CYC
382	PM_DS_295	100FE	PM_INST_CMPL
		0D4100000012C040	PM_INST_FROM_RMEM_ALL
		0D4100000013C142	PM_MRK_INST_FROM_RMEM_ALL
		0D4100000014C144	PM_MRK_INST_FROM_RMEM_ALL_CYC
383	PM_DS_296	100FE	PM_INST_CMPL
		0D4020000012C040	PM_IPTEG_FROM_RMEM_ALL
		0D4020000013C142	PM_MRK_IPTEG_FROM_RMEM_ALL
		0D4020000014C144	PM_MRK_IPTEG_FROM_RMEM_ALL_CYC
384	PM_DS_297	100FE	PM_INST_CMPL
		0D4040000022C040	PM_DATA_FROM_RMEM_ALL
		0D4040000023C142	PM_MRK_DATA_FROM_RMEM_ALL
		0D4040000024C144	PM_MRK_DATA_FROM_RMEM_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 44 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
385	PM_DS_298	100FE	PM_INST_CMPL
		0D4060000022C040	PM_DPTEG_FROM_RMEM_ALL
		0D4060000023C142	PM_MRK_DPTEG_FROM_RMEM_ALL
		0D4060000024C144	PM_MRK_DPTEG_FROM_RMEM_ALL_CYC
386	PM_DS_299	100FE	PM_INST_CMPL
		0D8020000002C040	PM_IPTEG_FROM_R_OC_CACHE
		0D8020000003C142	PM_MRK_IPTEG_FROM_R_OC_CACHE
		0D8020000004C144	PM_MRK_IPTEG_FROM_R_OC_CACHE_CYC
387	PM_DS_300	100FE	PM_INST_CMPL
		0D8040000002C040	PM_DATA_FROM_R_OC_CACHE
		0D8040000003C142	PM_MRK_DATA_FROM_R_OC_CACHE
		0D8040000004C144	PM_MRK_DATA_FROM_R_OC_CACHE_CYC
388	PM_DS_301	100FE	PM_INST_CMPL
		0D8060000002C040	PM_DPTEG_FROM_R_OC_CACHE
		0D8060000003C142	PM_MRK_DPTEG_FROM_R_OC_CACHE
		0D8060000004C144	PM_MRK_DPTEG_FROM_R_OC_CACHE_CYC
389	PM_DS_302	100FE	PM_INST_CMPL
		0D8020000012C040	PM_IPTEG_FROM_R_OC_CACHE_ALL
		0D8020000013C142	PM_MRK_IPTEG_FROM_R_OC_CACHE_ALL
		0D8020000014C144	PM_MRK_IPTEG_FROM_R_OC_CACHE_ALL_CYC
390	PM_DS_303	100FE	PM_INST_CMPL
		0D8040000022C040	PM_DATA_FROM_R_OC_CACHE_ALL
		0D8040000023C142	PM_MRK_DATA_FROM_R_OC_CACHE_ALL
		0D8040000024C144	PM_MRK_DATA_FROM_R_OC_CACHE_ALL_CYC
391	PM_DS_304	100FE	PM_INST_CMPL
		0D8060000022C040	PM_DPTEG_FROM_R_OC_CACHE_ALL
		0D8060000023C142	PM_MRK_DPTEG_FROM_R_OC_CACHE_ALL
		0D8060000024C144	PM_MRK_DPTEG_FROM_R_OC_CACHE_ALL_CYC
392	PM_DS_305	100FE	PM_INST_CMPL
		0DC020000002C040	PM_IPTEG_FROM_R_OC_MEM
		0DC020000003C142	PM_MRK_IPTEG_FROM_R_OC_MEM
		0DC020000004C144	PM_MRK_IPTEG_FROM_R_OC_MEM_CYC
393	PM_DS_306	100FE	PM_INST_CMPL
		0DC040000002C040	PM_DATA_FROM_R_OC_MEM
		0DC040000003C142	PM_MRK_DATA_FROM_R_OC_MEM
		0DC040000004C144	PM_MRK_DATA_FROM_R_OC_MEM_CYC



Table E-35. Power10 Events by Group (Sheet 45 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
394	PM_DS_307	100FE	PM_INST_CMPL
		0DC060000002C040	PM_DPTEG_FROM_R_OC_MEM
		0DC060000003C142	PM_MRK_DPTEG_FROM_R_OC_MEM
		0DC060000004C144	PM_MRK_DPTEG_FROM_R_OC_MEM_CYC
395	PM_DS_308	100FE	PM_INST_CMPL
		0DC020000012C040	PM_IPTEG_FROM_R_OC_MEM_ALL
		0DC020000013C142	PM_MRK_IPTEG_FROM_R_OC_MEM_ALL
		0DC020000014C144	PM_MRK_IPTEG_FROM_R_OC_MEM_ALL_CYC
396	PM_DS_309	100FE	PM_INST_CMPL
		0DC040000022C040	PM_DATA_FROM_R_OC_MEM_ALL
		0DC040000023C142	PM_MRK_DATA_FROM_R_OC_MEM_ALL
		0DC040000024C144	PM_MRK_DATA_FROM_R_OC_MEM_ALL_CYC
397	PM_DS_310	100FE	PM_INST_CMPL
		0DC060000022C040	PM_DPTEG_FROM_R_OC_MEM_ALL
		0DC060000023C142	PM_MRK_DPTEG_FROM_R_OC_MEM_ALL
		0DC060000024C144	PM_MRK_DPTEG_FROM_R_OC_MEM_ALL_CYC
398	PM_DS_311	100FE	PM_INST_CMPL
		0D8100000002C040	PM_INST_FROM_R_OC_ANY
		0D8100000003C142	PM_MRK_INST_FROM_R_OC_ANY
		0D8100000004C144	PM_MRK_INST_FROM_R_OC_ANY_CYC
399	PM_DS_312	100FE	PM_INST_CMPL
		0D8120000002C040	PM_IPTEG_FROM_R_OC_ANY
		0D8120000003C142	PM_MRK_IPTEG_FROM_R_OC_ANY
		0D8120000004C144	PM_MRK_IPTEG_FROM_R_OC_ANY_CYC
400	PM_DS_313	100FE	PM_INST_CMPL
		0D8140000002C040	PM_DATA_FROM_R_OC_ANY
		0D8140000003C142	PM_MRK_DATA_FROM_R_OC_ANY
		0D8140000004C144	PM_MRK_DATA_FROM_R_OC_ANY_CYC
401	PM_DS_314	100FE	PM_INST_CMPL
		0D8160000002C040	PM_DPTEG_FROM_R_OC_ANY
		0D8160000003C142	PM_MRK_DPTEG_FROM_R_OC_ANY
		0D8160000004C144	PM_MRK_DPTEG_FROM_R_OC_ANY_CYC
402	PM_DS_315	100FE	PM_INST_CMPL
		0D8100000012C040	PM_INST_FROM_R_OC_ANY_ALL
		0D8100000013C142	PM_MRK_INST_FROM_R_OC_ANY_ALL
		0D8100000014C144	PM_MRK_INST_FROM_R_OC_ANY_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 46 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
403	PM_DS_316	100FE	PM_INST_CMPL
		0D8120000012C040	PM_IPTEG_FROM_R_OC_ANY_ALL
		0D8120000013C142	PM_MRK_IPTEG_FROM_R_OC_ANY_ALL
		0D8120000014C144	PM_MRK_IPTEG_FROM_R_OC_ANY_ALL_CYC
404	PM_DS_317	100FE	PM_INST_CMPL
		0D8140000022C040	PM_DATA_FROM_R_OC_ANY_ALL
		0D8140000023C142	PM_MRK_DATA_FROM_R_OC_ANY_ALL
		0D8140000024C144	PM_MRK_DATA_FROM_R_OC_ANY_ALL_CYC
405	PM_DS_318	100FE	PM_INST_CMPL
		0D8160000022C040	PM_DPTEG_FROM_R_OC_ANY_ALL
		0D8160000023C142	PM_MRK_DPTEG_FROM_R_OC_ANY_ALL
		0D8160000024C144	PM_MRK_DPTEG_FROM_R_OC_ANY_ALL_CYC
406	PM_DS_319	100FE	PM_INST_CMPL
		0E0020000002C040	PM_IPTEG_FROM_DL2_SHR
		0E0020000003C142	PM_MRK_IPTEG_FROM_DL2_SHR
		0E0020000004C144	PM_MRK_IPTEG_FROM_DL2_SHR_CYC
407	PM_DS_320	100FE	PM_INST_CMPL
		0E0040000002C040	PM_DATA_FROM_DL2_SHR
		0E0040000003C142	PM_MRK_DATA_FROM_DL2_SHR
		0E0040000004C144	PM_MRK_DATA_FROM_DL2_SHR_CYC
408	PM_DS_321	100FE	PM_INST_CMPL
		0E0060000002C040	PM_DPTEG_FROM_DL2_SHR
		0E0060000003C142	PM_MRK_DPTEG_FROM_DL2_SHR
		0E0060000004C144	PM_MRK_DPTEG_FROM_DL2_SHR_CYC
409	PM_DS_322	100FE	PM_INST_CMPL
		0E0020000012C040	PM_IPTEG_FROM_DL2_SHR_ALL
		0E0020000013C142	PM_MRK_IPTEG_FROM_DL2_SHR_ALL
		0E0020000014C144	PM_MRK_IPTEG_FROM_DL2_SHR_ALL_CYC
410	PM_DS_323	100FE	PM_INST_CMPL
		0E0040000022C040	PM_DATA_FROM_DL2_SHR_ALL
		0E0040000023C142	PM_MRK_DATA_FROM_DL2_SHR_ALL
		0E0040000024C144	PM_MRK_DATA_FROM_DL2_SHR_ALL_CYC
411	PM_DS_324	100FE	PM_INST_CMPL
		0E0060000022C040	PM_DPTEG_FROM_DL2_SHR_ALL
		0E0060000023C142	PM_MRK_DPTEG_FROM_DL2_SHR_ALL
		0E0060000024C144	PM_MRK_DPTEG_FROM_DL2_SHR_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 47 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
412	PM_DS_325	100FE	PM_INST_CMPL
		0E4020000002C040	PM_IPTEG_FROM_DL2_MOD
		0E4020000003C142	PM_MRK_IPTEG_FROM_DL2_MOD
		0E4020000004C144	PM_MRK_IPTEG_FROM_DL2_MOD_CYC
413	PM_DS_326	100FE	PM_INST_CMPL
		0E4040000002C040	PM_DATA_FROM_DL2_MOD
		0E4040000003C142	PM_MRK_DATA_FROM_DL2_MOD
		0E4040000004C144	PM_MRK_DATA_FROM_DL2_MOD_CYC
414	PM_DS_327	100FE	PM_INST_CMPL
		0E4060000002C040	PM_DPTEG_FROM_DL2_MOD
		0E4060000003C142	PM_MRK_DPTEG_FROM_DL2_MOD
		0E4060000004C144	PM_MRK_DPTEG_FROM_DL2_MOD_CYC
415	PM_DS_328	100FE	PM_INST_CMPL
		0E4020000012C040	PM_IPTEG_FROM_DL2_MOD_ALL
		0E4020000013C142	PM_MRK_IPTEG_FROM_DL2_MOD_ALL
		0E4020000014C144	PM_MRK_IPTEG_FROM_DL2_MOD_ALL_CYC
416	PM_DS_329	100FE	PM_INST_CMPL
		0E4040000022C040	PM_DATA_FROM_DL2_MOD_ALL
		0E4040000023C142	PM_MRK_DATA_FROM_DL2_MOD_ALL
		0E4040000024C144	PM_MRK_DATA_FROM_DL2_MOD_ALL_CYC
417	PM_DS_330	100FE	PM_INST_CMPL
		0E4060000022C040	PM_DPTEG_FROM_DL2_MOD_ALL
		0E4060000023C142	PM_MRK_DPTEG_FROM_DL2_MOD_ALL
		0E4060000024C144	PM_MRK_DPTEG_FROM_DL2_MOD_ALL_CYC
418	PM_DS_331	100FE	PM_INST_CMPL
		0E0100000002C040	PM_INST_FROM_DL2
		0E0100000003C142	PM_MRK_INST_FROM_DL2
		0E0100000004C144	PM_MRK_INST_FROM_DL2_CYC
419	PM_DS_332	100FE	PM_INST_CMPL
		0E0120000002C040	PM_IPTEG_FROM_DL2
		0E0120000003C142	PM_MRK_IPTEG_FROM_DL2
		0E0120000004C144	PM_MRK_IPTEG_FROM_DL2_CYC
420	PM_DS_333	100FE	PM_INST_CMPL
		0E0140000002C040	PM_DATA_FROM_DL2
		0E0140000003C142	PM_MRK_DATA_FROM_DL2
		0E0140000004C144	PM_MRK_DATA_FROM_DL2_CYC



Table E-35. Power10 Events by Group (Sheet 48 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
421	PM_DS_334	100FE	PM_INST_CMPL
		0E0160000002C040	PM_DPTEG_FROM_DL2
		0E0160000003C142	PM_MRK_DPTEG_FROM_DL2
		0E0160000004C144	PM_MRK_DPTEG_FROM_DL2_CYC
422	PM_DS_335	100FE	PM_INST_CMPL
		0E0100000012C040	PM_INST_FROM_DL2_ALL
		0E0100000013C142	PM_MRK_INST_FROM_DL2_ALL
		0E0100000014C144	PM_MRK_INST_FROM_DL2_ALL_CYC
423	PM_DS_336	100FE	PM_INST_CMPL
		0E0120000012C040	PM_IPTEG_FROM_DL2_ALL
		0E0120000013C142	PM_MRK_IPTEG_FROM_DL2_ALL
		0E0120000014C144	PM_MRK_IPTEG_FROM_DL2_ALL_CYC
424	PM_DS_337	100FE	PM_INST_CMPL
		0E0140000022C040	PM_DATA_FROM_DL2_ALL
		0E0140000023C142	PM_MRK_DATA_FROM_DL2_ALL
		0E0140000024C144	PM_MRK_DATA_FROM_DL2_ALL_CYC
425	PM_DS_338	100FE	PM_INST_CMPL
		0E0160000022C040	PM_DPTEG_FROM_DL2_ALL
		0E0160000023C142	PM_MRK_DPTEG_FROM_DL2_ALL
		0E0160000024C144	PM_MRK_DPTEG_FROM_DL2_ALL_CYC
426	PM_DS_339	100FE	PM_INST_CMPL
		0E8020000002C040	PM_IPTEG_FROM_DL3_SHR
		0E8020000003C142	PM_MRK_IPTEG_FROM_DL3_SHR
		0E8020000004C144	PM_MRK_IPTEG_FROM_DL3_SHR_CYC
427	PM_DS_340	100FE	PM_INST_CMPL
		0E8040000002C040	PM_DATA_FROM_DL3_SHR
		0E8040000003C142	PM_MRK_DATA_FROM_DL3_SHR
		0E8040000004C144	PM_MRK_DATA_FROM_DL3_SHR_CYC
428	PM_DS_341	100FE	PM_INST_CMPL
		0E8060000002C040	PM_DPTEG_FROM_DL3_SHR
		0E8060000003C142	PM_MRK_DPTEG_FROM_DL3_SHR
		0E8060000004C144	PM_MRK_DPTEG_FROM_DL3_SHR_CYC
429	PM_DS_342	100FE	PM_INST_CMPL
		0E8020000012C040	PM_IPTEG_FROM_DL3_SHR_ALL
		0E8020000013C142	PM_MRK_IPTEG_FROM_DL3_SHR_ALL
		0E8020000014C144	PM_MRK_IPTEG_FROM_DL3_SHR_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 49 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
430	PM_DS_343	100FE	PM_INST_CMPL
		0E8040000022C040	PM_DATA_FROM_DL3_SHR_ALL
		0E8040000023C142	PM_MRK_DATA_FROM_DL3_SHR_ALL
		0E8040000024C144	PM_MRK_DATA_FROM_DL3_SHR_ALL_CYC
431	PM_DS_344	100FE	PM_INST_CMPL
		0E8060000022C040	PM_DPTEG_FROM_DL3_SHR_ALL
		0E8060000023C142	PM_MRK_DPTEG_FROM_DL3_SHR_ALL
		0E8060000024C144	PM_MRK_DPTEG_FROM_DL3_SHR_ALL_CYC
432	PM_DS_345	100FE	PM_INST_CMPL
		0EC020000002C040	PM_IPTEG_FROM_DL3_MOD
		0EC020000003C142	PM_MRK_IPTEG_FROM_DL3_MOD
		0EC020000004C144	PM_MRK_IPTEG_FROM_DL3_MOD_CYC
433	PM_DS_346	100FE	PM_INST_CMPL
		0EC040000002C040	PM_DATA_FROM_DL3_MOD
		0EC040000003C142	PM_MRK_DATA_FROM_DL3_MOD
		0EC040000004C144	PM_MRK_DATA_FROM_DL3_MOD_CYC
434	PM_DS_347	100FE	PM_INST_CMPL
		0EC060000002C040	PM_DPTEG_FROM_DL3_MOD
		0EC060000003C142	PM_MRK_DPTEG_FROM_DL3_MOD
		0EC060000004C144	PM_MRK_DPTEG_FROM_DL3_MOD_CYC
435	PM_DS_348	100FE	PM_INST_CMPL
		0EC020000012C040	PM_IPTEG_FROM_DL3_MOD_ALL
		0EC020000013C142	PM_MRK_IPTEG_FROM_DL3_MOD_ALL
		0EC020000014C144	PM_MRK_IPTEG_FROM_DL3_MOD_ALL_CYC
436	PM_DS_349	100FE	PM_INST_CMPL
		0EC040000022C040	PM_DATA_FROM_DL3_MOD_ALL
		0EC040000023C142	PM_MRK_DATA_FROM_DL3_MOD_ALL
		0EC040000024C144	PM_MRK_DATA_FROM_DL3_MOD_ALL_CYC
437	PM_DS_350	100FE	PM_INST_CMPL
		0EC060000022C040	PM_DPTEG_FROM_DL3_MOD_ALL
		0EC060000023C142	PM_MRK_DPTEG_FROM_DL3_MOD_ALL
		0EC060000024C144	PM_MRK_DPTEG_FROM_DL3_MOD_ALL_CYC
438	PM_DS_351	100FE	PM_INST_CMPL
		0E8100000002C040	PM_INST_FROM_DL3
		0E8100000003C142	PM_MRK_INST_FROM_DL3
		0E8100000004C144	PM_MRK_INST_FROM_DL3_CYC



Table E-35. Power10 Events by Group (Sheet 50 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
439	PM_DS_352	100FE	PM_INST_CMPL
		0E8120000002C040	PM_IPTEG_FROM_DL3
		0E8120000003C142	PM_MRK_IPTEG_FROM_DL3
		0E8120000004C144	PM_MRK_IPTEG_FROM_DL3_CYC
440	PM_DS_353	100FE	PM_INST_CMPL
		0E8140000002C040	PM_DATA_FROM_DL3
		0E8140000003C142	PM_MRK_DATA_FROM_DL3
		0E8140000004C144	PM_MRK_DATA_FROM_DL3_CYC
441	PM_DS_354	100FE	PM_INST_CMPL
		0E8160000002C040	PM_DPTEG_FROM_DL3
		0E8160000003C142	PM_MRK_DPTEG_FROM_DL3
		0E8160000004C144	PM_MRK_DPTEG_FROM_DL3_CYC
442	PM_DS_355	100FE	PM_INST_CMPL
		0E8100000012C040	PM_INST_FROM_DL3_ALL
		0E8100000013C142	PM_MRK_INST_FROM_DL3_ALL
		0E8100000014C144	PM_MRK_INST_FROM_DL3_ALL_CYC
443	PM_DS_356	100FE	PM_INST_CMPL
		0E8120000012C040	PM_IPTEG_FROM_DL3_ALL
		0E8120000013C142	PM_MRK_IPTEG_FROM_DL3_ALL
		0E8120000014C144	PM_MRK_IPTEG_FROM_DL3_ALL_CYC
444	PM_DS_357	100FE	PM_INST_CMPL
		0E8140000022C040	PM_DATA_FROM_DL3_ALL
		0E8140000023C142	PM_MRK_DATA_FROM_DL3_ALL
		0E8140000024C144	PM_MRK_DATA_FROM_DL3_ALL_CYC
445	PM_DS_358	100FE	PM_INST_CMPL
		0E8160000022C040	PM_DPTEG_FROM_DL3_ALL
		0E8160000023C142	PM_MRK_DPTEG_FROM_DL3_ALL
		0E8160000024C144	PM_MRK_DPTEG_FROM_DL3_ALL_CYC
446	PM_DS_359	100FE	PM_INST_CMPL
		0E0220000002C040	PM_IPTEG_FROM_DL2L3_SHR
		0E0220000003C142	PM_MRK_IPTEG_FROM_DL2L3_SHR
		0E0220000004C144	PM_MRK_IPTEG_FROM_DL2L3_SHR_CYC
447	PM_DS_360	100FE	PM_INST_CMPL
		0E0240000002C040	PM_DATA_FROM_DL2L3_SHR
		0E0240000003C142	PM_MRK_DATA_FROM_DL2L3_SHR
		0E0240000004C144	PM_MRK_DATA_FROM_DL2L3_SHR_CYC



Table E-35. Power10 Events by Group (Sheet 51 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
448	PM_DS_361	100FE	PM_INST_CMPL
		0E0260000002C040	PM_DPTEG_FROM_DL2L3_SHR
		0E0260000003C142	PM_MRK_DPTEG_FROM_DL2L3_SHR
		0E0260000004C144	PM_MRK_DPTEG_FROM_DL2L3_SHR_CYC
449	PM_DS_362	100FE	PM_INST_CMPL
		0E0220000012C040	PM_IPTEG_FROM_DL2L3_SHR_ALL
		0E0220000013C142	PM_MRK_IPTEG_FROM_DL2L3_SHR_ALL
		0E0220000014C144	PM_MRK_IPTEG_FROM_DL2L3_SHR_ALL_CYC
450	PM_DS_363	100FE	PM_INST_CMPL
		0E0240000022C040	PM_DATA_FROM_DL2L3_SHR_ALL
		0E0240000023C142	PM_MRK_DATA_FROM_DL2L3_SHR_ALL
		0E0240000024C144	PM_MRK_DATA_FROM_DL2L3_SHR_ALL_CYC
451	PM_DS_364	100FE	PM_INST_CMPL
		0E0260000022C040	PM_DPTEG_FROM_DL2L3_SHR_ALL
		0E0260000023C142	PM_MRK_DPTEG_FROM_DL2L3_SHR_ALL
		0E0260000024C144	PM_MRK_DPTEG_FROM_DL2L3_SHR_ALL_CYC
452	PM_DS_365	100FE	PM_INST_CMPL
		0E4220000002C040	PM_IPTEG_FROM_DL2L3_MOD
		0E4220000003C142	PM_MRK_IPTEG_FROM_DL2L3_MOD
		0E4220000004C144	PM_MRK_IPTEG_FROM_DL2L3_MOD_CYC
453	PM_DS_366	100FE	PM_INST_CMPL
		0E4240000002C040	PM_DATA_FROM_DL2L3_MOD
		0E4240000003C142	PM_MRK_DATA_FROM_DL2L3_MOD
		0E4240000004C144	PM_MRK_DATA_FROM_DL2L3_MOD_CYC
454	PM_DS_367	100FE	PM_INST_CMPL
		0E4260000002C040	PM_DPTEG_FROM_DL2L3_MOD
		0E4260000003C142	PM_MRK_DPTEG_FROM_DL2L3_MOD
		0E4260000004C144	PM_MRK_DPTEG_FROM_DL2L3_MOD_CYC
455	PM_DS_368	100FE	PM_INST_CMPL
		0E4220000012C040	PM_IPTEG_FROM_DL2L3_MOD_ALL
		0E4220000013C142	PM_MRK_IPTEG_FROM_DL2L3_MOD_ALL
		0E4220000014C144	PM_MRK_IPTEG_FROM_DL2L3_MOD_ALL_CYC
456	PM_DS_369	100FE	PM_INST_CMPL
		0E4240000022C040	PM_DATA_FROM_DL2L3_MOD_ALL
		0E4240000023C142	PM_MRK_DATA_FROM_DL2L3_MOD_ALL
		0E4240000024C144	PM_MRK_DATA_FROM_DL2L3_MOD_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 52 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
457	PM_DS_370	100FE	PM_INST_CMPL
		0E4260000022C040	PM_DPTEG_FROM_DL2L3_MOD_ALL
		0E4260000023C142	PM_MRK_DPTEG_FROM_DL2L3_MOD_ALL
		0E4260000024C144	PM_MRK_DPTEG_FROM_DL2L3_MOD_ALL_CYC
458	PM_DS_371	100FE	PM_INST_CMPL
		0E0300000002C040	PM_INST_FROM_DL2L3
		0E0300000003C142	PM_MRK_INST_FROM_DL2L3
		0E0300000004C144	PM_MRK_INST_FROM_DL2L3_CYC
459	PM_DS_372	100FE	PM_INST_CMPL
		0E0320000002C040	PM_IPTEG_FROM_DL2L3
		0E0320000003C142	PM_MRK_IPTEG_FROM_DL2L3
		0E0320000004C144	PM_MRK_IPTEG_FROM_DL2L3_CYC
460	PM_DS_373	100FE	PM_INST_CMPL
		0E0340000002C040	PM_DATA_FROM_DL2L3
		0E0340000003C142	PM_MRK_DATA_FROM_DL2L3
		0E0340000004C144	PM_MRK_DATA_FROM_DL2L3_CYC
461	PM_DS_374	100FE	PM_INST_CMPL
		0E0360000002C040	PM_DPTEG_FROM_DL2L3
		0E0360000003C142	PM_MRK_DPTEG_FROM_DL2L3
		0E0360000004C144	PM_MRK_DPTEG_FROM_DL2L3_CYC
462	PM_DS_375	100FE	PM_INST_CMPL
		0E0300000012C040	PM_INST_FROM_DL2L3_ALL
		0E0300000013C142	PM_MRK_INST_FROM_DL2L3_ALL
		0E0300000014C144	PM_MRK_INST_FROM_DL2L3_ALL_CYC
463	PM_DS_376	100FE	PM_INST_CMPL
		0E0320000012C040	PM_IPTEG_FROM_DL2L3_ALL
		0E0320000013C142	PM_MRK_IPTEG_FROM_DL2L3_ALL
		0E0320000014C144	PM_MRK_IPTEG_FROM_DL2L3_ALL_CYC
464	PM_DS_377	100FE	PM_INST_CMPL
		0E0340000022C040	PM_DATA_FROM_DL2L3_ALL
		0E0340000023C142	PM_MRK_DATA_FROM_DL2L3_ALL
		0E0340000024C144	PM_MRK_DATA_FROM_DL2L3_ALL_CYC
465	PM_DS_378	100FE	PM_INST_CMPL
		0E0360000022C040	PM_DPTEG_FROM_DL2L3_ALL
		0E0360000023C142	PM_MRK_DPTEG_FROM_DL2L3_ALL
		0E0360000024C144	PM_MRK_DPTEG_FROM_DL2L3_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 53 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
466	PM_DS_379	100FE	PM_INST_CMPL
		0F4100000002C040	PM_INST_FROM_DMEM
		0F4100000003C142	PM_MRK_INST_FROM_DMEM
		0F4100000004C144	PM_MRK_INST_FROM_DMEM_CYC
467	PM_DS_380	100FE	PM_INST_CMPL
		0F4020000002C040	PM_IPTEG_FROM_DMEM
		0F4020000003C142	PM_MRK_IPTEG_FROM_DMEM
		0F4020000004C144	PM_MRK_IPTEG_FROM_DMEM_CYC
468	PM_DS_381	100FE	PM_INST_CMPL
		0F4040000002C040	PM_DATA_FROM_DMEM
		0F4040000003C142	PM_MRK_DATA_FROM_DMEM
		0F4040000004C144	PM_MRK_DATA_FROM_DMEM_CYC
469	PM_DS_382	100FE	PM_INST_CMPL
		0F4060000002C040	PM_DPTEG_FROM_DMEM
		0F4060000003C142	PM_MRK_DPTEG_FROM_DMEM
		0F4060000004C144	PM_MRK_DPTEG_FROM_DMEM_CYC
470	PM_DS_383	100FE	PM_INST_CMPL
		0F4100000012C040	PM_INST_FROM_DMEM_ALL
		0F4100000013C142	PM_MRK_INST_FROM_DMEM_ALL
		0F4100000014C144	PM_MRK_INST_FROM_DMEM_ALL_CYC
471	PM_DS_384	100FE	PM_INST_CMPL
		0F4020000012C040	PM_IPTEG_FROM_DMEM_ALL
		0F4020000013C142	PM_MRK_IPTEG_FROM_DMEM_ALL
		0F4020000014C144	PM_MRK_IPTEG_FROM_DMEM_ALL_CYC
472	PM_DS_385	100FE	PM_INST_CMPL
		0F4040000022C040	PM_DATA_FROM_DMEM_ALL
		0F4040000023C142	PM_MRK_DATA_FROM_DMEM_ALL
		0F4040000024C144	PM_MRK_DATA_FROM_DMEM_ALL_CYC
473	PM_DS_386	100FE	PM_INST_CMPL
		0F4060000022C040	PM_DPTEG_FROM_DMEM_ALL
		0F4060000023C142	PM_MRK_DPTEG_FROM_DMEM_ALL
		0F4060000024C144	PM_MRK_DPTEG_FROM_DMEM_ALL_CYC
474	PM_DS_387	100FE	PM_INST_CMPL
		0F8020000002C040	PM_IPTEG_FROM_D_OC_CACHE
		0F8020000003C142	PM_MRK_IPTEG_FROM_D_OC_CACHE
		0F8020000004C144	PM_MRK_IPTEG_FROM_D_OC_CACHE_CYC



Table E-35. Power10 Events by Group (Sheet 54 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
475	PM_DS_388	100FE	PM_INST_CMPL
		0F8040000002C040	PM_DATA_FROM_D_OC_CACHE
		0F8040000003C142	PM_MRK_DATA_FROM_D_OC_CACHE
		0F8040000004C144	PM_MRK_DATA_FROM_D_OC_CACHE_CYC
476	PM_DS_389	100FE	PM_INST_CMPL
		0F8060000002C040	PM_DPTEG_FROM_D_OC_CACHE
		0F8060000003C142	PM_MRK_DPTEG_FROM_D_OC_CACHE
		0F8060000004C144	PM_MRK_DPTEG_FROM_D_OC_CACHE_CYC
477	PM_DS_390	100FE	PM_INST_CMPL
		0F8020000012C040	PM_IPTEG_FROM_D_OC_CACHE_ALL
		0F8020000013C142	PM_MRK_IPTEG_FROM_D_OC_CACHE_ALL
		0F8020000014C144	PM_MRK_IPTEG_FROM_D_OC_CACHE_ALL_CYC
478	PM_DS_391	100FE	PM_INST_CMPL
		0F8040000022C040	PM_DATA_FROM_D_OC_CACHE_ALL
		0F8040000023C142	PM_MRK_DATA_FROM_D_OC_CACHE_ALL
		0F8040000024C144	PM_MRK_DATA_FROM_D_OC_CACHE_ALL_CYC
479	PM_DS_392	100FE	PM_INST_CMPL
		0F8060000022C040	PM_DPTEG_FROM_D_OC_CACHE_ALL
		0F8060000023C142	PM_MRK_DPTEG_FROM_D_OC_CACHE_ALL
		0F8060000024C144	PM_MRK_DPTEG_FROM_D_OC_CACHE_ALL_CYC
480	PM_DS_393	100FE	PM_INST_CMPL
		0FC020000002C040	PM_IPTEG_FROM_D_OC_MEM
		0FC020000003C142	PM_MRK_IPTEG_FROM_D_OC_MEM
		0FC020000004C144	PM_MRK_IPTEG_FROM_D_OC_MEM_CYC
481	PM_DS_394	100FE	PM_INST_CMPL
		0FC040000002C040	PM_DATA_FROM_D_OC_MEM
		0FC040000003C142	PM_MRK_DATA_FROM_D_OC_MEM
		0FC040000004C144	PM_MRK_DATA_FROM_D_OC_MEM_CYC
482	PM_DS_395	100FE	PM_INST_CMPL
		0FC060000002C040	PM_DPTEG_FROM_D_OC_MEM
		0FC060000003C142	PM_MRK_DPTEG_FROM_D_OC_MEM
		0FC060000004C144	PM_MRK_DPTEG_FROM_D_OC_MEM_CYC
483	PM_DS_396	100FE	PM_INST_CMPL
		0FC0200000012C040	PM_IPTEG_FROM_D_OC_MEM_ALL
		0FC0200000013C142	PM_MRK_IPTEG_FROM_D_OC_MEM_ALL
		0FC0200000014C144	PM_MRK_IPTEG_FROM_D_OC_MEM_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 55 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
484	PM_DS_397	100FE	PM_INST_CMPL
		0FC040000022C040	PM_DATA_FROM_D_OC_MEM_ALL
		0FC040000023C142	PM_MRK_DATA_FROM_D_OC_MEM_ALL
		0FC040000024C144	PM_MRK_DATA_FROM_D_OC_MEM_ALL_CYC
485	PM_DS_398	100FE	PM_INST_CMPL
		0FC060000022C040	PM_DPTEG_FROM_D_OC_MEM_ALL
		0FC060000023C142	PM_MRK_DPTEG_FROM_D_OC_MEM_ALL
		0FC060000024C144	PM_MRK_DPTEG_FROM_D_OC_MEM_ALL_CYC
486	PM_DS_399	100FE	PM_INST_CMPL
		0F8100000002C040	PM_INST_FROM_D_OC_ANY
		0F8100000003C142	PM_MRK_INST_FROM_D_OC_ANY
		0F8100000004C144	PM_MRK_INST_FROM_D_OC_ANY_CYC
487	PM_DS_400	100FE	PM_INST_CMPL
		0F8120000002C040	PM_IPTEG_FROM_D_OC_ANY
		0F8120000003C142	PM_MRK_IPTEG_FROM_D_OC_ANY
		0F8120000004C144	PM_MRK_IPTEG_FROM_D_OC_ANY_CYC
488	PM_DS_401	100FE	PM_INST_CMPL
		0F8140000002C040	PM_DATA_FROM_D_OC_ANY
		0F8140000003C142	PM_MRK_DATA_FROM_D_OC_ANY
		0F8140000004C144	PM_MRK_DATA_FROM_D_OC_ANY_CYC
489	PM_DS_402	100FE	PM_INST_CMPL
		0F8160000002C040	PM_DPTEG_FROM_D_OC_ANY
		0F8160000003C142	PM_MRK_DPTEG_FROM_D_OC_ANY
		0F8160000004C144	PM_MRK_DPTEG_FROM_D_OC_ANY_CYC
490	PM_DS_403	100FE	PM_INST_CMPL
		0F8100000012C040	PM_INST_FROM_D_OC_ANY_ALL
		0F8100000013C142	PM_MRK_INST_FROM_D_OC_ANY_ALL
		0F8100000014C144	PM_MRK_INST_FROM_D_OC_ANY_ALL_CYC
491	PM_DS_404	100FE	PM_INST_CMPL
		0F8120000012C040	PM_IPTEG_FROM_D_OC_ANY_ALL
		0F8120000013C142	PM_MRK_IPTEG_FROM_D_OC_ANY_ALL
		0F8120000014C144	PM_MRK_IPTEG_FROM_D_OC_ANY_ALL_CYC
492	PM_DS_405	100FE	PM_INST_CMPL
		0F8140000022C040	PM_DATA_FROM_D_OC_ANY_ALL
		0F8140000023C142	PM_MRK_DATA_FROM_D_OC_ANY_ALL
		0F8140000024C144	PM_MRK_DATA_FROM_D_OC_ANY_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 56 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
493	PM_DS_406	100FE	PM_INST_CMPL
		0F8160000022C040	PM_DPTEG_FROM_D_OC_ANY_ALL
		0F8160000023C142	PM_MRK_DPTEG_FROM_D_OC_ANY_ALL
		0F8160000024C144	PM_MRK_DPTEG_FROM_D_OC_ANY_ALL_CYC
494	PM_DS_407	100FE	PM_INST_CMPL
		080B00000002C040	PM_INST_FROM_ONCHIP_CACHE
		080B00000003C142	PM_MRK_INST_FROM_ONCHIP_CACHE
		080B00000004C144	PM_MRK_INST_FROM_ONCHIP_CACHE_CYC
495	PM_DS_408	100FE	PM_INST_CMPL
		080B20000002C040	PM_IPTEG_FROM_ONCHIP_CACHE
		080B20000003C142	PM_MRK_IPTEG_FROM_ONCHIP_CACHE
		080B20000004C144	PM_MRK_IPTEG_FROM_ONCHIP_CACHE_CYC
496	PM_DS_409	100FE	PM_INST_CMPL
		080B40000002C040	PM_DATA_FROM_ONCHIP_CACHE
		080B40000003C142	PM_MRK_DATA_FROM_ONCHIP_CACHE
		080B40000004C144	PM_MRK_DATA_FROM_ONCHIP_CACHE_CYC
497	PM_DS_410	100FE	PM_INST_CMPL
		080B60000002C040	PM_DPTEG_FROM_ONCHIP_CACHE
		080B60000003C142	PM_MRK_DPTEG_FROM_ONCHIP_CACHE
		080B60000004C144	PM_MRK_DPTEG_FROM_ONCHIP_CACHE_CYC
498	PM_DS_411	100FE	PM_INST_CMPL
		080B00000012C040	PM_INST_FROM_ONCHIP_CACHE_ALL
		080B00000013C142	PM_MRK_INST_FROM_ONCHIP_CACHE_ALL
		080B00000014C144	PM_MRK_INST_FROM_ONCHIP_CACHE_ALL_CYC
499	PM_DS_412	100FE	PM_INST_CMPL
		080B20000012C040	PM_IPTEG_FROM_ONCHIP_CACHE_ALL
		080B20000013C142	PM_MRK_IPTEG_FROM_ONCHIP_CACHE_ALL
		080B20000014C144	PM_MRK_IPTEG_FROM_ONCHIP_CACHE_ALL_CYC
500	PM_DS_413	100FE	PM_INST_CMPL
		080B40000022C040	PM_DATA_FROM_ONCHIP_CACHE_ALL
		080B40000023C142	PM_MRK_DATA_FROM_ONCHIP_CACHE_ALL
		080B40000024C144	PM_MRK_DATA_FROM_ONCHIP_CACHE_ALL_CYC
501	PM_DS_414	100FE	PM_INST_CMPL
		080B60000022C040	PM_DPTEG_FROM_ONCHIP_CACHE_ALL
		080B60000023C142	PM_MRK_DPTEG_FROM_ONCHIP_CACHE_ALL
		080B60000024C144	PM_MRK_DPTEG_FROM_ONCHIP_CACHE_ALL_CYC



Table E-35. Power10 Events by Group (Sheet 57 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
502	PM_DS_415	100FE	PM_INST_CMPL
		0C0B00000002C040	PM_INST_FROM_OFFCHIP_CACHE
		0C0B00000003C142	PM_MRK_INST_FROM_OFFCHIP_CACHE
		0C0B00000004C144	PM_MRK_INST_FROM_OFFCHIP_CACHE_CYC
503	PM_DS_416	100FE	PM_INST_CMPL
		0C0B20000002C040	PM_IPTEG_FROM_OFFCHIP_CACHE
		0C0B20000003C142	PM_MRK_IPTEG_FROM_OFFCHIP_CACHE
		0C0B20000004C144	PM_MRK_IPTEG_FROM_OFFCHIP_CACHE_CYC
504	PM_DS_417	100FE	PM_INST_CMPL
		0C0B40000002C040	PM_DATA_FROM_OFFCHIP_CACHE
		0C0B40000003C142	PM_MRK_DATA_FROM_OFFCHIP_CACHE
		0C0B40000004C144	PM_MRK_DATA_FROM_OFFCHIP_CACHE_CYC
505	PM_DS_418	100FE	PM_INST_CMPL
		0C0B60000002C040	PM_DPTEG_FROM_OFFCHIP_CACHE
		0C0B60000003C142	PM_MRK_DPTEG_FROM_OFFCHIP_CACHE
		0C0B60000004C144	PM_MRK_DPTEG_FROM_OFFCHIP_CACHE_CYC
506	PM_DS_419	100FE	PM_INST_CMPL
		0C0B00000012C040	PM_INST_FROM_OFFCHIP_CACHE_ALL
		0C0B00000013C142	PM_MRK_INST_FROM_OFFCHIP_CACHE_ALL
		0C0B00000014C144	PM_MRK_INST_FROM_OFFCHIP_CACHE_ALL_CYC
507	PM_DS_420	100FE	PM_INST_CMPL
		0C0B20000012C040	PM_IPTEG_FROM_OFFCHIP_CACHE_ALL
		0C0B20000013C142	PM_MRK_IPTEG_FROM_OFFCHIP_CACHE_ALL
		0C0B20000014C144	PM_MRK_IPTEG_FROM_OFFCHIP_CACHE_ALL_CYC
508	PM_DS_421	100FE	PM_INST_CMPL
		0C0B40000022C040	PM_DATA_FROM_OFFCHIP_CACHE_ALL
		0C0B40000023C142	PM_MRK_DATA_FROM_OFFCHIP_CACHE_ALL
		0C0B40000024C144	PM_MRK_DATA_FROM_OFFCHIP_CACHE_ALL_CYC
509	PM_DS_422	100FE	PM_INST_CMPL
		0C0B60000022C040	PM_DPTEG_FROM_OFFCHIP_CACHE_ALL
		0C0B60000023C142	PM_MRK_DPTEG_FROM_OFFCHIP_CACHE_ALL
		0C0B60000024C144	PM_MRK_DPTEG_FROM_OFFCHIP_CACHE_ALL_CYC
510	PM_DS_423	100FE	PM_INST_CMPL
		095900000002C040	PM_INST_FROM_ANY_MEMORY
		095900000003C142	PM_MRK_INST_FROM_ANY_MEMORY
		095900000004C144	PM_MRK_INST_FROM_ANY_MEMORY_CYC



Table E-35. Power10 Events by Group (Sheet 58 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
511	PM_DS_424	100FE	PM_INST_CMPL
		095820000002C040	PM_IPTEG_FROM_ANY_MEMORY
		095820000003C142	PM_MRK_IPTEG_FROM_ANY_MEMORY
		095820000004C144	PM_MRK_IPTEG_FROM_ANY_MEMORY_CYC
512	PM_DS_425	100FE	PM_INST_CMPL
		095840000002C040	PM_DATA_FROM_ANY_MEMORY
		095840000003C142	PM_MRK_DATA_FROM_ANY_MEMORY
		095840000004C144	PM_MRK_DATA_FROM_ANY_MEMORY_CYC
513	PM_DS_426	100FE	PM_INST_CMPL
		095860000002C040	PM_DPTEG_FROM_ANY_MEMORY
		095860000003C142	PM_MRK_DPTEG_FROM_ANY_MEMORY
		095860000004C144	PM_MRK_DPTEG_FROM_ANY_MEMORY_CYC
514	PM_DS_427	100FE	PM_INST_CMPL
		095900000012C040	PM_INST_FROM_ANY_MEMORY_ALL
		095900000013C142	PM_MRK_INST_FROM_ANY_MEMORY_ALL
		095900000014C144	PM_MRK_INST_FROM_ANY_MEMORY_ALL_CYC
515	PM_DS_428	100FE	PM_INST_CMPL
		095820000012C040	PM_IPTEG_FROM_ANY_MEMORY_ALL
		095820000013C142	PM_MRK_IPTEG_FROM_ANY_MEMORY_ALL
		095820000014C144	PM_MRK_IPTEG_FROM_ANY_MEMORY_ALL_CYC
516	PM_DS_429	100FE	PM_INST_CMPL
		095840000022C040	PM_DATA_FROM_ANY_MEMORY_ALL
		095840000023C142	PM_MRK_DATA_FROM_ANY_MEMORY_ALL
		095840000024C144	PM_MRK_DATA_FROM_ANY_MEMORY_ALL_CYC
517	PM_DS_430	100FE	PM_INST_CMPL
		095860000022C040	PM_DPTEG_FROM_ANY_MEMORY_ALL
		095860000023C142	PM_MRK_DPTEG_FROM_ANY_MEMORY_ALL
		095860000024C144	PM_MRK_DPTEG_FROM_ANY_MEMORY_ALL_CYC
518	PM_MRK_OTHER_1	100FE	PM_INST_CMPL
		20108	PM_MRK_TLBIE_CYC
		3012A	PM_MRK_L2_RC_DONE
		4010E	PM_MRK_TLBIE_FIN
519	PM_MRK_OTHER_2	100FE	PM_INST_CMPL
		20112	PM_MRK_NTF_FIN
		3012C	PM_MRK_ST_FWD
		40114	PM_MRK_START_PROBE_NOP_DISP



Table E-35. Power10 Events by Group (Sheet 59 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
520	PM_MRK_OTHER_3	100FE	PM_INST_CMPL
		20114	PM_MRK_L2_RC_DISP
		3012E	PM_MRK_DTLB_MISS_2M
		40116	PM_MRK_LARX_FIN
521	PM_MRK_OTHER_4	100FE	PM_INST_CMPL
		2011C	PM_MRK_NTF_CYC
		30130	PM_MRK_INST_FIN
		40118	PM_MRK_DCACHE_RELOAD_INTV
522	PM_MRK_OTHER_5	10132	PM_MRK_INST_ISSUED
		2D12C	PM_MRK_DATA_GRP_PUMP_MPRED_RTY
		30132	PM_MRK_VSU_FIN
		40002	PM_INST_CMPL
523	PM_MRK_OTHER_6	10134	PM_MRK_ST_DONE_L2
		2D12E	PM_MRK_DTLB_HIT
		30134	PM_MRK_ST_CMPL_INT
		40002	PM_INST_CMPL
524	PM_MRK_OTHER_7	1013E	PM_MRK_LD_MISS_EXPOSED_CYC
		20130	PM_MRK_INST_DECODED
		3013C	PM_MRK_DTLB_MISS_4K
		40002	PM_INST_CMPL
525	PM_MRK_OTHER_8	100FE	PM_INST_CMPL
		20132	PM_MRK_DFU_ISSUE
		34146	PM_MRK_LD_CMPL
		40132	PM_MRK_LSU_FIN
526	PM_MRK_OTHER_9	100FE	PM_INST_CMPL
		20134	PM_MRK_FXU_ISSUE
		34149	PM_MRK_STORE_DATA
		40134	PM_MRK_INST_TIMEO
527	PM_MRK_OTHER_10	1C142	PM_MRK_XFER_FROM_SRC_PMC1
		20138	PM_MRK_ST_NEST
		30002	PM_INST_CMPL
		40136	PM_MRK_STORE_DATA_CYC
528	PM_MRK_OTHER_11	1C144	PM_MRK_XFER_FROM_SRC_CYC_PMC1
		2013A	PM_MRK_BRU_FIN
		3C142	PM_MRK_XFER_FROM_SRC_PMC3
		40002	PM_INST_CMPL



Table E-35. Power10 Events by Group (Sheet 60 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
529	PM_MRK_OTHER_12	1C146	PM_MRK_DATA_FLUSHED_FROM_SRC_PMC1
		2013C	PM_MRK_FX_LSU_FIN
		30002	PM_INST_CMPL
		44146	PM_MRK_STCX_CORE_CYC
530	PM_MRK_OTHER_13	1C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC_PMC1
		2C142	PM_MRK_XFER_FROM_SRC_PMC2
		3C144	PM_MRK_XFER_FROM_SRC_CYC_PMC3
		40002	PM_INST_CMPL
531	PM_MRK_OTHER_14	1C14A	PM_MRK_SYS_PUMP_MPRED_RTY
		2C144	PM_MRK_XFER_FROM_SRC_CYC_PMC2
		3C146	PM_MRK_DATA_FLUSHED_FROM_SRC_PMC3
		40002	PM_INST_CMPL
532	PM_MRK_OTHER_15	1015E	PM_MRK_FAB_RSP_RD_T_INTV
		2C146	PM_MRK_DATA_FLUSHED_FROM_SRC_PMC2
		3C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC_PMC3
		40002	PM_INST_CMPL
533	PM_MRK_OTHER_16	100FE	PM_INST_CMPL
		2C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC_PMC2
		3C14C	PM_MRK_DATA_SYS_PUMP_MPRED_RTY
		4C142	PM_MRK_XFER_FROM_SRC_PMC4
534	PM_MRK_OTHER_17	100FE	PM_INST_CMPL
		2015E	PM_MRK_FAB_RSP_RWITM_RTY
		30154	PM_MRK_FAB_RSP_DCLAIM
		4C144	PM_MRK_XFER_FROM_SRC_CYC_PMC4
535	PM_MRK_OTHER_18	100FE	PM_INST_CMPL
		24156	PM_MRK_STCX_FIN
		3015C	PM_MRK_PTESYNC_CYC
		4C146	PM_MRK_DATA_FLUSHED_FROM_SRC_PMC4
536	PM_MRK_OTHER_19	100FE	PM_INST_CMPL
		24158	PM_MRK_INST
		3015E	PM_MRK_FAB_RSP CLAIM_RTY
		4C148	PM_MRK_DATA_FLUSHED_FROM_SRC_CYC_PMC4
537	PM_MRK_OTHER_20	1D156	PM_MRK_LD_MISS_L1_CYC
		2415C	PM_MRK_BR_CMPL
		30002	PM_INST_CMPL
		40152	PM_MRK_PUMP_MPRED



Table E-35. Power10 Events by Group (Sheet 61 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
538	PM_MRK_OTHER_21	1D15C	PM_MRK_DTLB_MISS_1G
		2D150	PM_MRK_DERAT_MISS_4K
		30002	PM_INST_CMPL
		40154	PM_MRK_FAB_RSP_BKILL
539	PM_MRK_OTHER_22	1E152	PM_MRK_DERAT_MISS_16M
		20002	PM_INST_CMPL
		3D154	PM_MRK_DERAT_MISS_16G
		40156	PM_MRK_GRP_PUMP_MPRED_RTY
540	PM_MRK_OTHER_23	1E15E	PM_MRK_L2_TM_REQ_ABORT
		2D154	PM_MRK_DERAT_MISS_64K
		30002	PM_INST_CMPL
		4015E	PM_MRK_FAB_RSP_RD_RTY
541	PM_MRK_OTHER_24	1F150	PM_MRK_ST_L2_CYC
		2D156	PM_MRK_DTLB_MISS_16M
		3D15E	PM_MULT_MRK
		40002	PM_INST_CMPL
542	PM_MRK_OTHER_25	1F152	PM_MRK_FAB_RSP_BKILL_CYC
		2D15E	PM_MRK_DTLB_MISS_16G
		3E158	PM_MRK_STCX_FAIL
		40002	PM_INST_CMPL
543	PM_MRK_OTHER_26	1F15C	PM_MRK_STCX_L2_CYC
		2F152	PM_MRK_FAB_RSP_DCLAIM_CYC
		3E15A	PM_MRK_ST_FIN
		40002	PM_INST_CMPL
544	PM_MRK_OTHER_27	1F15E	PM_MRK_START_PROBE_NOP_CMPL
		201E0	PM_MRK_DATA_FROM_MEMORY
		3F150	PM_MRK_ST_DRAIN_CYC
		40002	PM_INST_CMPL
545	PM_MRK_OTHER_28	101E0	PM_MRK_INST_DISP
		20002	PM_INST_CMPL
		30162	PM_MRK_ISSUE_DEPENDENT_LOAD
		4C15C	PM_MRK_DERAT_MISS_1G
546	PM_MRK_OTHER_29	101E2	PM_MRK_BR_TAKEN_CMPL
		00000021B4	PM_MRK_STOP_PROBE_NOP_DISP
		30002	PM_INST_CMPL
		4C15E	PM_MRK_DTLB_MISS_64K



Table E-35. Power10 Events by Group (Sheet 62 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
547	PM_MRK_OTHER_30	101E4	PM_MRK_L1_ICACHE_MISS
		000000399C	PM_MRK_MMA_ACC_VSR_CONFLICT
		30002	PM_INST_CMPL
		4E15E	PM_MRK_INST_FLUSHED
548	PM_MRK_OTHER_31	101EA	PM_MRK_L1_RELOAD_VALID
		20002	PM_INST_CMPL
		30000	PM_SUSPENDED
		4F150	PM_MRK_FAB_RSP_RWITM_CYC
549	PM_MRK_OTHER_32	100FE	PM_INST_CMPL
		20000	PM_SUSPENDED
		30000	PM_SUSPENDED
		40164	PM_MRK_DERAT_MISS_2M
550	PM_MRK_OTHER_33	1F150	PM_MRK_ST_L2_CYC
		20114	PM_MRK_L2_RC_DISP
		30002	PM_INST_CMPL
		000300000004C040	PM_INST_FROM_L2
551	PM_OTHER_1	1000A	PM_PMC3_REWIND
		2F04A	PM_BR_FIN
		30006	PM_MMA_ACTIVE_NAVAIL
		40004	PM_FXU_ISSUE
552	PM_OTHER_2	1000C	PM_LSU_LD0_FIN
		2000A	PM_HYPERVISOR_CYC
		3000C	PM_FREQ_DOWN
		40006	PM_ISSUE_KILL
553	PM_OTHER_3	1000E	PM_MMA_ISSUED
		2000C	PM_RUN_LATCH_ALL_THREADS_CYC
		3000E	PM_ST_REJECT_TIQ
		40008	PM_NTC_ALL_FIN
554	PM_OTHER_4	10010	PM_PMC4_OVERFLOW
		2000E	PM_LSU_LD1_FIN
		30010	PM_PMC2_OVERFLOW
		4000A	PM_DEBUG_TRIGGER
555	PM_OTHER_5	10012	PM_LSU_ST0_FIN
		20010	PM_PMC1_OVERFLOW
		30012	PM_FLUSH_COMPLETION
		4000C	PM_FREQ_UP



Table E-35. Power10 Events by Group (Sheet 63 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
556	PM_OTHER_6	10014	PM_LSU_ST4_FIN
		20016	PM_ST_FIN
		3001A	PM_LSU_ST2_FIN
		40010	PM_PMC3_OVERFLOW
557	PM_OTHER_7	10016	PM_VSU0_ISSUE
		20018	PM_ST_FWD
		30020	PM_PMC2_REWIND
		40012	PM_L1_ICACHE_RELOADED_ALL
558	PM_OTHER_8	10018	PM_IC_DEMAND_CYC
		2001A	PM_ITLB_HIT
		30022	PM_PMC4_SAVED
		4001A	PM_SMT_MODE_SWITCH
559	PM_OTHER_9	1001C	PM_ULTRAVISOR_INST_CMPL
		2C012	PM_MMA_AVAIL_NACTIVE
		30024	PM_PMC6_OVERFLOW
		4001C	PM_VSU_FIN
560	PM_OTHER_10	10020	PM_PMC4_REWIND
		2D010	PM_LSU_ST1_FIN
		3003E	PM_ITLB_MISS_1G
		4C01C	PM_INT_DOORBELL
561	PM_OTHER_11	10022	PM_PMC2_SAVED
		2D012	PM_VSU1_ISSUE
		34040	PM_ITLB_MISS_16G
		4C01E	PM_LSU_ST3_FIN
562	PM_OTHER_12	10024	PM_PMC5_OVERFLOW
		2E010	PM_ADJUNCT_INST_CMPL
		34042	PM_HPT_TLB_PARENT_HIT_CHILD_MISS
		4D010	PM_PMC1_SAVED
563	PM_OTHER_13	10026	PM_EXT_INT_EBB
		2E014	PM_STCX_FIN
		34044	PM_DERAT_MISS_PREF
		4D012	PM_PMC3_SAVED
564	PM_OTHER_14	10028	PM_NTC_FLUSH
		2E016	PM_EXT_INT_HYP
		3404A	PM_DATA_RADIX_L2_PTE_FROM_L3MISS
		4E012	PM_EXEC_STALL_UNKNOWN



Table E-35. Power10 Events by Group (Sheet 64 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
565	PM_OTHER_15	1002A	PM_PMC3_HELD_CYC
		2003E	PM_PTESYNC_FIN
		3404C	PM_DATA_RADIX_L2_PDE_FROM_L3MISS
		4D020	PM_VSU3_ISSUE
566	PM_OTHER_16	1002C	PM_LD_PREFETCH_CACHE_LINE_MISS
		24042	PM_DATA_RADIX_L2_PTE_FROM_L3
		3404E	PM_DATA_RADIX_L3_PTE_FROM_L3MISS
		4D022	PM_HYPERVISOR_INST_CMPL
567	PM_OTHER_17	14042	PM_DATA_RADIX_L2_PTE_FROM_L2
		24044	PM_DATA_RADIX_L2_PDE_FROM_L3
		35040	PM_DATA_RADIX_L3_PDE_FROM_L3MISS
		4D024	PM_PROBLEM_INST_CMPL
568	PM_OTHER_18	14044	PM_DATA_RADIX_L2_PDE_FROM_L2
		24046	PM_DATA_RADIX_L3_PTE_FROM_L3
		35042	PM_DATA_RADIX_L4_PTE_FROM_L3MISS
		4D026	PM_ULTRAVISOR_CYC
569	PM_OTHER_19	14046	PM_DATA_RADIX_L3_PTE_FROM_L2
		24048	PM_DATA_RADIX_L3_PDE_FROM_L3
		35046	PM_INST_RADIX_L2_PTE_FROM_L3MISS
		4D028	PM_PRIVILEGED_CYC
570	PM_OTHER_20	14048	PM_DATA_RADIX_L3_PDE_FROM_L2
		2404A	PM_DATA_RADIX_L4_PTE_FROM_L3
		3504A	PM_INST_RADIX_L3_PTE_FROM_L3MISS
		4D02C	PM_PMC1_REWIND
571	PM_OTHER_21	1404A	PM_DATA_RADIX_L4_PTE_FROM_L2
		2404E	PM_INST_RADIX_L2_PTE_FROM_L3
		3504C	PM_INST_RADIX_L3_PDE_FROM_L3MISS
		4D02E	PM_NO_FETCH_CYC
572	PM_OTHER_22	1404E	PM_INST_RADIX_L2_PTE_FROM_L2
		25042	PM_INST_RADIX_L3_PTE_FROM_L3
		3504E	PM_INST_RADIX_L4_PTE_FROM_L3MISS
		40030	PM_INST_FIN
573	PM_OTHER_23	15042	PM_INST_RADIX_L3_PTE_FROM_L2
		25044	PM_INST_RADIX_L3_PDE_FROM_L3
		3C040	PM_XFER_FROM_SRC_PMC3
		4003E	PM_LD_CMPL

Table E-35. Power10 Events by Group (Sheet 65 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
574	PM_OTHER_24	15044	PM_INST_RADIX_L3_PDE_FROM_L2
		25046	PM_INST_RADIX_L4_PTE_FROM_L3
		3C04E	PM_PMC1_HELD_CYC
		44040	PM_L2_PWC_HIT
575	PM_OTHER_25	15046	PM_INST_RADIX_L4_PTE_FROM_L2
		2504C	PM_PMC4_HELD_CYC
		3F042	PM_HOT_LOCK_COLLISION
		44042	PM_L3_PWC_HIT
576	PM_OTHER_26	1C040	PM_XFER_FROM_SRC_PMC1
		2C040	PM_XFER_FROM_SRC_PMC2
		3F044	PM_VSU2_ISSUE
		4404A	PM_DATA_RADIX_L2_PTE_FROM_DISTANT
577	PM_OTHER_27	1C04E	PM_SIAR_LOADED
		2C04E	PM_SDAR_LOADED
		3F046	PM_ITLB_HIT_1G
		4404C	PM_DATA_RADIX_L2_PDE_FROM_DISTANT
578	PM_OTHER_28	10056	PM_MEM_READ
		2F04C	PM_ICBI_FIN
		3F048	PM_ITLB_HIT_16G
		4404E	PM_DATA_RADIX_L3_PTE_FROM_DISTANT
579	PM_OTHER_29	1005A	PM_FLUSH_MPRED
		20050	PM_HPT_RELOAD
		3F04A	PM_LSU_ST5_FIN
		45040	PM_DATA_RADIX_L3_PDE_FROM_DISTANT
580	PM_OTHER_30	14050	PM_INST_CHIP_PUMP_CPRED
		20052	PM_GRP_PUMP_MPRED_TOO_BIG
		3F04C	PM_RADIX_RELOAD
		45042	PM_DATA_RADIX_L4_PTE_FROM_DISTANT
581	PM_OTHER_31	14052	PM_INST_GRP_PUMP_MPRED_RTY
		20058	PM_LD_CACHE_INHIBITED
		3F04E	PM_PROBLEM_CYC
		45046	PM_INST_RADIX_L2_PTE_FROM_DISTANT
582	PM_OTHER_32	14054	PM_INST_PUMP_CPRED
		24050	PM_IOPS_DISP
		30050	PM_GRP_PUMP_MPRED_RTY
		4504A	PM_INST_RADIX_L3_PTE_FROM_DISTANT



Table E-35. Power10 Events by Group (Sheet 66 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
583	PM_OTHER_33	1505E	PM_LD_HIT_L1
		2405E	PM_ISSUE_CANCEL
		30052	PM_SYS_PUMP_MPRED_TOO_BIG
		4504C	PM_INST_RADIX_L3_PDE_FROM_DISTANT
584	PM_OTHER_34	1C050	PM_DATA_CHIP_PUMP_CPRED
		2505C	PM_VSU_ISSUE
		30058	PM_TLBIE_FIN
		4504E	PM_INST_RADIX_L4_PTE_FROM_DISTANT
585	PM_OTHER_35	1C052	PM_DATA_GRP_PUMP_MPRED_RTY
		2C050	PM_DATA_GRP_PUMP_CPRED
		34050	PM_INST_SYS_PUMP_CPRED
		4C040	PM_XFER_FROM_SRC_PMC4
586	PM_OTHER_36	1C054	PM_DATA_PUMP_CPRED
		2C052	PM_DATA_GRP_PUMP_MPRED_TOO_BIG
		34052	PM_INST_SYS_PUMP_MPRED_TOO_BIG
		4C04A	PM_PMC2_HELD_CYC
587	PM_OTHER_37	1C056	PM_DERAT_MISS_4K
		2C054	PM_DERAT_MISS_64K
		3405A	PM_PRIVILEGED_INST_CMPL
		4D04E	PM_VECTOR_FSQRT_FDIV_ISSUE
588	PM_OTHER_38	1C058	PM_DTLB_MISS_16G
		2C056	PM_DTLB_MISS_4K
		3C050	PM_DATA_SYS_PUMP_CPRED
		4E040	PM_SP_MMA_CMPL
589	PM_OTHER_39	1C05A	PM_DERAT_MISS_2M
		2C058	PM_MEM_PREF
		3C052	PM_DATA_SYS_PUMP_MPRED_TOO_BIG
		4E042	PM_DP_MMA_CMPL
590	PM_OTHER_40	1C05C	PM_DTLB_MISS_2M
		2C05A	PM_DERAT_MISS_1G
		3C054	PM_DERAT_MISS_16M
		4E044	PM_HP_MMA_CMPL
591	PM_OTHER_41	1C05E	PM_ITLB_MISS_64K
		2C05C	PM_INST_GRP_PUMP_CPRED
		3C056	PM_DTLB_MISS_64K
		4E046	PM_4INT_MMA_CMPL



Table E-35. Power10 Events by Group (Sheet 67 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
592	PM_OTHER_42	1D050	PM_DTLB_HIT_16G
		2C05E	PM_INST_GRP_PUMP_MPRED_TOO_BIG
		3C058	PM_LARX_FIN
		4E048	PM_8INT_MMA_CMPL
593	PM_OTHER_43	1D054	PM_DTLB_HIT_2M
		2D058	PM_ITLB_MISS_2M
		3C05A	PM_DTLB_HIT_64K
		4E04A	PM_16INT_MMA_CMPL
594	PM_OTHER_44	1D058	PM_ITLB_HIT_64K
		2D05A	PM_ITLB_MISS_16M
		3C05E	PM_MEM_RWITM
		4E04C	PM_BF16_MMA_CMPL
595	PM_OTHER_45	1E058	PM_STCX_FAIL_FIN
		2E050	PM_DTLB_HIT_4K
		3D058	PM_SCALAR_FSQRT_FDIV_ISSUE
		4E04E	PM_MOVE_TO_MMA_CMPL
596	PM_OTHER_46	1F054	PM_DTLB_HIT
		2E054	PM_ITLB_HIT_2M
		3E050	PM_EXT_INT_OS
		4F040	PM_MOVE_FROM_MMA_CMPL
597	PM_OTHER_47	1F056	PM_DISP_SS0_2_INSTR_CYC
		2E056	PM_ITLB_HIT_16M
		3E054	PM_LD_MISS_L1
		4F042	PM_MUL_MMA_CMPL
598	PM_OTHER_48	1F058	PM_DISP_HELD_CYC
		2E05A	PM_LD_REJECT_TIQ
		3F054	PM_DISP_SS0_4_INSTR_CYC
		40050	PM_SYS_PUMP_MPRED_RTY
599	PM_OTHER_49	1F05A	PM_DISP_HELD_SYNC_CYC
		2F054	PM_DISP_SS1_2_INSTR_CYC
		3F056	PM_DISP_SS0_8_INSTR_CYC
		44050	PM_INST_SYS_PUMP_MPRED_RTY
600	PM_OTHER_50	10060	PM_TM_TRANS_RUN_CYC
		2F056	PM_DISP_SS1_4_INSTR_CYC
		3F05E	PM_DISP_HELD_ISSQ_FULL_CYC
		44054	PM_VECTOR_LD_CMPL



Table E-35. Power10 Events by Group (Sheet 68 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
601	PM_OTHER_51	10062	PM_LD_L3MISS_PEND_CYC
		20066	PM_DISP_HELD_OTHER_CYC
		30060	PM_DISP_HELD_XVFC_MAPPER_CYC
		44056	PM_VECTOR_ST_CMPL
602	PM_OTHER_52	10066	PM_ADJUNCT_CYC
		20068	PM_DISP_HELD_HALT_CYC
		30064	PM_NTF_ISSUE_HOLD_CYC
		44058	PM_LWSYNC_CMPL
603	PM_OTHER_53	1006A	PM_FX_LSU_FIN
		2006A	PM_DISP_HELD_STF_MAPPER_CYC
		30066	PM_LSU_FIN
		4405A	PM_HWSYNC_CMPL
604	PM_OTHER_54	1006C	PM_RUN_CYC_ST_MODE
		2006C	PM_RUN_CYC_SMT4_MODE
		30068	PM_L1_ICACHE_RELOADED_PREF
		4405E	PM_ANY_FLOP_CMPL
605	PM_OTHER_55	101E6	PM_THRESH_EXC_4096
		0000002080	PM_EE_OFF_EXT_INT_CYC
		3006C	PM_RUN_CYC_SMT2_MODE
		45050	PM_1FLOP_CMPL
606	PM_OTHER_56	101E8	PM_THRESH_EXC_256
		0000002880	PM_ISU_FLUSH
		3006E	PM_CONSTANT_CLK
		45052	PM_4FLOP_CMPL
607	PM_OTHER_57	101EC	PM_THRESH_MET
		0000002084	PM_ISU_FLUSH_DISP
		300F0	PM_ST_MISS_L1
		45054	PM_FMA_CMPL
608	PM_OTHER_58	0000002884	PM_ISU_FLUSH_BALANCE
		0000002088	PM_ISU_FLUSH_PARTIAL
		0000002888	PM_ISU_FLUSH_DISP_SRQ_EMPTY
		45056	PM_SCALAR_FLOP_CMPL
609	PM_OTHER_59	000000208C	PM_ISU_FLUSH_LD_ECC_ERROR
		000000288C	PM_ISU_FLUSH_LWSYNC
		0000002090	PM_ISU_FLUSH_ISYNC
		45058	PM_IC_MISS_CMPL



Table E-35. Power10 Events by Group (Sheet 69 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
610	PM_OTHER_60	0000002094	PM_ISU_FLUSH_HWSYNC
		0000002894	PM_ISU_FLUSH_MMA_OFF_CYC
		00000028AC	PM_ISU_FLUSH_DISP_STF_REBAL
		4505A	PM_SP_FLOP_CMPL
611	PM_OTHER_61	00000020B0	PM_START_PROBE_NOP_DISP
		00000028B0	PM_STOP_PROBE_NOP_DISP
		00000020B8	PM_FUNCTION_CALL_DISP
		4505C	PM_MATH_FLOP_CMPL
612	PM_OTHER_62	00000028B8	PM_FUNCTION_RETURN_DISP
		00000020BC	PM_0CYC_CONST_DISP
		0000003080	PM_ISSUE_HOLD_STAGS_CYC
		4C050	PM_DATA_SYS_PUMP_MPRED_RTY
613	PM_OTHER_63	0000003880	PM_ISSUE_HOLD_LTAGS_CYC
		0000003084	PM_ISSUE_KILL_DL_MISS
		0000003884	PM_ISSUE_KILL_RESOURCE
		4C054	PM_DERAT_MISS_16G
614	PM_OTHER_64	0000003088	PM_ISSUE_KILL_THROTTLE
		000000388C	PM_SHL_HIT
		0000003890	PM_LHS_HIT
		4C056	PM_DTLB_MISS_16M
615	PM_OTHER_65	0000003094	PM_LHS_CREATED
		0000003894	PM_LARX_HIT_LARX_HIT
		0000003098	PM_LARX_HIT_LARX_CREATED
		4C05A	PM_DTLB_MISS_1G
616	PM_OTHER_66	0000003898	PM_MMA_VSR_CONFLICT_FLUSH
		000000309C	PM_MMA_ACC_CONFLICT_FLUSH
		00000030A0	PM_MMA_IN_USE_CYC
		4D050	PM_VSU_NON_FLOP_CMPL
617	PM_OTHER_67	00000038A0	PM_MMA_ON_CYC
		00000030B4	PM_DISP_HELD_OUT_OF_LTAGS_CYC
		00000038B4	PM_DISP_HELD_OUT_OF_STAGS_CYC
		4D052	PM_2FLOP_CMPL
618	PM_OTHER_68	00000030B8	PM_DISP_CLB_HELD_BALANCE_CYC
		00000038B8	PM_DISP_CLB_HELD_SRQ_EMPTY_CYC
		00000030BC	PM_DISP_PARTIAL
		4D054	PM_8FLOP_CMPL



Table E-35. Power10 Events by Group (Sheet 70 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
619	PM_OTHER_69	00000038BC	PM_ISYNC_CMPL
		000000C080	PM_LD0_8B_FIN
		000000C880	PM_LD1_8B_FIN
		4D056	PM_NON_FMA_FLOP_CMPL
620	PM_OTHER_70	000000C084	PM_LD0_16B_FIN
		000000C884	PM_LD1_16B_FIN
		000000C088	PM_LD0_32B_FIN
		4D058	PM_VECTOR_FLOP_CMPL
621	PM_OTHER_71	000000C888	PM_LD1_32B_FIN
		000000C08C	PM_LD0_VECTOR_FIN
		000000C88C	PM_LD1_VECTOR_FIN
		4D05A	PM_NON_MATH_FLOP_CMPL
622	PM_OTHER_72	000000C090	PM_LD0_UNALIGNED_FIN
		000000C890	PM_LD1_UNALIGNED_FIN
		000000C094	PM_ST0_8B_FIN
		4D05C	PM_DPP_FLOP_CMPL
623	PM_OTHER_73	000000C894	PM_ST1_8B_FIN
		000000C098	PM_ST0_16B_FIN
		000000C898	PM_ST1_16B_FIN
		4D05E	PM_BR_CMPL
624	PM_OTHER_74	000000C09C	PM_ST0_32B_FIN
		000000C89C	PM_ST1_32B_FIN
		000000C0A0	PM_ST0_VECTOR_FIN
		4E050	PM_STCX_PASS_FIN
625	PM_OTHER_75	000000C8A0	PM_ST1_VECTOR_FIN
		000000C0A4	PM_ST0_UNALIGNED_FIN
		000000C8A4	PM_ST1_UNALIGNED_FIN
		4E052	PM_DTLB_HIT_16M
626	PM_OTHER_76	000000C0A8	PM_FALSE_LHS
		000000C8A8	PM_LD0_LHS_REJECT
		000000C0AC	PM_LD1_LHS_REJECT
		4E054	PM_DTLB_HIT_1G
627	PM_OTHER_77	000000C8AC	PM_ST0_STORE_REJECT
		000000C0B0	PM_ST1_STORE_REJECT
		000000C8B0	PM_ST_DRAIN_MERGE
		4E058	PM_BR_COND_CMPL



Table E-35. Power10 Events by Group (Sheet 71 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
628	PM_OTHER_78	000000C0B4	PM_LMQ_MERGE
		000000C8B4	PM_STCX_CMPL
		000000C0B8	PM_NCST_CMPL
		4E05A	PM_PREFIXED_CMPL
629	PM_OTHER_79	000000C8B8	PM_STCX_SUCCESS_CMPL
		000000C0BC	PM_DC_RELOAD_COLLISIONS
		000000C8BC	PM_DC_STORE_WRITE_COLLISIONS
		4F056	PM_DISP_SS1_8_INSTR_CYC
630	PM_OTHER_80	000000D080	PM_LSU_SET_MPRED
		000000D880	PM_DERAT_HIT
		000000D084	PM_IERAT_HIT
		40060	PM_DISP_HELD_SCOREBOARD_CYC
631	PM_OTHER_81	000000D884	PM_TIQ_BYPASS
		000000D088	PM_TIQ_ALLOC_CYC
		000000D888	PM_TIQ_HALF_FULL_CYC
		40062	PM_DISP_HELD_RENAME_CYC
632	PM_OTHER_82	000000D08C	PM_TIQ_ERAT_MISS_EMB_FULL_RESPIN
		000000D88C	PM_EMB_FULL_CYC
		000000D090	PM_LSU_FLUSH_CYC
		40066	PM_ITLB_HIT_4K
633	PM_OTHER_83	000000D890	PM_LSU_FLUSH_CI
		000000D094	PM_LSU_FLUSH_ALL_WAYS_LOCKED
		000000D894	PM_LSU_FLUSH_LHL
		4006C	PM_ITLB_MISS_4K
634	PM_OTHER_84	000000D098	PM_LSU_FLUSH_SAME_ICT_GRP
		000000D898	PM_LSU_REJECT_LHS
		000000D09C	PM_LSU_FLUSH_SPECIAL
		4016E	PM_THRESH_NOT_MET
635	PM_OTHER_85	000000D89C	PM_LSU_FLUSH_SHL
		000000D0A0	PM_LSU_FLUSH_SAO
		000000D8A0	PM_LSU_FLUSH_LARX_STCX
		400F0	PM_LD_DEMAND_MISS_L1_FIN
636	PM_OTHER_86	000000D0A4	PM_LSU_FLUSH_OTHER
		000000D8A4	PM_DC_PREF_HW_ALLOC
		000000D0A8	PM_DC_PREF_SW_ALLOC
		000000D8A8	PM_DC_PREF_STRIDED_ALLOC



Table E-35. Power10 Events by Group (Sheet 72 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
637	PM_OTHER_87	000000D0AC	PM_DC_PREF_CONS_ALLOC
		000000D8AC	PM_DC_PREF_XCONS_ALLOC
		000000D0B0	PM_DC_PREF_CONF
		000000D8B0	PM_DC_PREF_FUZZY_CONF
638	PM_OTHER_88	000000D0B4	PM_DC_PREF_STRIDED_CONF
		000000D8B4	PM_DC_PREF DEALLOC_NO_CONF
		000000D0B8	PM_L1_SW_PREF
		000000D0BC	PM_L3_SW_PREF
639	PM_OTHER_89	000000F080	PM_SNOOP_TLBIE_MY_LPAR_CYC
		000000F880	PM_SNOOP_TLBIE_CYC
		000000F084	PM_SNOOP_TLBIE_CACHE_WALK_CYC
		000000F884	PM_SNOOP_TLBIE_WAIT_ST_CYC
640	PM_OTHER_90	000000F088	PM_SNOOP_TLBIE_WAIT_LD_CYC
		000000F888	PM_SNOOP_TLBIE_WAIT_IFU_CYC
		000000F08C	PM_SNOOP_TLBIE_WAIT_MMU_CYC
		000000F094	PM_LD0_SETP_HIT_EADIR_MISS
641	PM_OTHER_91	000000F898	PM_SAME_EA_DIFF_CTXTAG_RADIR_HIT
		000000F09C	PM_CTXT_MP4_ALLOC
		000000F89C	PM_CTXT_ALIAS_HIT CONTRIB
		000000F0A0	PM_START_NEW_RENAME
642	PM_OTHER_92	000000F8A0	PM_STORE_ALLOCATE
		000000F0A4	PM_LOAD_ALLOC_DEPRA_FOR_ALL_WAYS_LOCKED
		000000F8A4	PM_STORE_REJECT_FOR_ALL_WAYS_LOCKED
		000000F0B0	PM_DERAT_HIT_4K
643	PM_OTHER_93	000000F8B0	PM_IERAT_HIT_4K
		000000F0B4	PM_DERAT_HIT_64K
		000000F8B4	PM_IERAT_HIT_64K
		000000F0B8	PM_DERAT_HIT_2M
644	PM_OTHER_94	000000F8B8	PM_IERAT_HIT_2M
		000000F0BC	PM_DERAT_HIT_1G
		000000F8BC	PM_IERAT_HIT_1G
		0000004080	PM_INST_FROM_L1
645	PM_OTHER_95	0000004880	PM_NO_FETCH_BANK_CONFLICT_CYC
		0000004084	PM_NO_FETCH_EAT_FULL_CYC
		0000004884	PM_NO_FETCH_IBUF_FULL_CYC
		0000004088	PM_NO_FETCH_THROTTLE_CYC

Table E-35. Power10 Events by Group (Sheet 73 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
646	PM_OTHER_96	0000004888	PM_FETCH_CYC
		000000408C	PM_NO_FETCH_THROTTLE_POWMAN_CYC
		000000488C	PM_NO_FETCH_THROTTLE_OTHER_CYC
		0000004090	PM_NO_FETCH_THROTTLE_REL_PRIO_CYC
647	PM_OTHER_97	0000004890	PM_NO_FETCH_THROTTLE_DYN_PRIO_CYC
		0000004894	PM_DECODE_THROTTLE_IIF_CYC
		0000004098	PM_DECODE_HOLD_NO_ITAGS
		0000004898	PM_DECODE_THROTTLE_IPC_CYC
648	PM_OTHER_98	000000409C	PM_IC_INVALIDATE
		000000489C	PM_IC_RELOAD_PRIVATE
		00000040A0	PM_IC_PREF_REQ
		00000048A0	PM_FUSED_BACK_TO_BACK
649	PM_OTHER_99	00000040A4	PM_FUSED_DESTRUCTIVE
		00000048A4	PM_FUSED_RESULT
		00000040A8	PM_FUSED_TOGETHER
		00000048A8	PM_FUSED_LOADCOMPARE
650	PM_OTHER_100	00000040AC	PM_FUSED_LOAD_LOAD
		00000040B0	PM_BCQ_FULL_CYC
		00000048B0	PM_BR_FIN_FROM_BCQ
		00000040B4	PM_BR_TKN_FIN
651	PM_OTHER_101	00000048B4	PM_BR_TKN_UNCOND_FIN
		00000040B8	PM_PRED_BR_TKN_COND_DIR
		00000048B8	PM_PRED_BR_NTKN_COND_DIR
		00000040BC	PM_MPRED_BR_TKN_COND_DIR
652	PM_OTHER_102	00000048BC	PM_MPRED_BR_NTKN_COND_DIR
		0000005080	PM_PRED_BR_TKN_COND_TGT_DIR
		0000005880	PM_PRED_BR_NTKN_COND_TGT_DIR
		0000005884	PM_MPRED_BR_NTKN_COND_TGT_DIR
653	PM_OTHER_103	0000005088	PM_BR_PRED_TKN_COND_DIR_LBHT_LSEL
		0000005888	PM_BR_PRED_TKN_COND_DIR_LBHT_GSEL
		000000508C	PM_BR_PRED_TKN_COND_DIR_GBHT
		000000588C	PM_BR_PRED_TKN_COND_DIR_TAGE
654	PM_OTHER_104	0000005090	PM_BR_PRED_TKN_COND_DIR_TOP
		0000005890	PM_BR_PRED_TKN_COND_TGT_COUNT_LCC
		0000005094	PM_BR_PRED_TKN_COND_TGT_COUNT_GCC
		0000005894	PM_BR_PRED_TKN_COND_TGT_COUNT_TIP



Table E-35. Power10 Events by Group (Sheet 74 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
655	PM_OTHER_105	0000005098	PM_BR_PRED_TKN_TGT_LINK
		0000005898	PM_BR_PRED_NTKN_COND_DIR_LBHT_LSEL
		000000509C	PM_BR_PRED_NTKN_COND_DIR_LBHT_GSEL
		000000589C	PM_BR_PRED_NTKN_COND_DIR_GBHT
656	PM_OTHER_106	00000050A0	PM_BR_PRED_NTKN_COND_DIR_TAGE
		00000058A0	PM_BR_PRED_NTKN_COND_DIR_TOP
		00000050A4	PM_SHL_CREATED
		00000058A4	PM_SHL_ST_DEPENDENCY
657	PM_OTHER_107	00000050A8	PM_Ieadir_HIT_IDIR_MISS
		00000058A8	PM_ICACHE_MISS_DUE_TO_CTXTTAG
		00000050AC	PM_BR_MPRED_TKN_COND_DIR_LBHT_LSEL
		00000058AC	PM_BR_MPRED_TKN_COND_DIR_LBHT_GSEL
658	PM_OTHER_108	00000050B0	PM_BR_MPRED_TKN_COND_DIR_GBHT
		00000058B0	PM_BR_MPRED_TKN_COND_DIR_TAGE
		00000050B4	PM_BR_MPRED_TKN_COND_DIR_TOP
		00000058B4	PM_BR_MPRED_TKN_COND_TGT_COUNT_LCC
659	PM_OTHER_109	00000050B8	PM_BR_MPRED_TKN_COND_TGT_COUNT_GCC
		00000058B8	PM_BR_MPRED_TKN_COND_TGT_COUNT_TIP
		00000050BC	PM_BR_MPRED_TKN_TGT_LINK
		00000058BC	PM_BR_MPRED_NTKN_COND_DIR_LBHT_LSEL
660	PM_OTHER_110	000000E080	PM_BR_MPRED_NTKN_COND_DIR_LBHT_GSEL
		000000E880	PM_BR_MPRED_NTKN_COND_DIR_GBHT
		000000E084	PM_BR_MPRED_NTKN_COND_DIR_TAGE
		000000E884	PM_BR_MPRED_NTKN_COND_DIR_TOP
661	PM_OTHER_111	000000E088	PM_Iea_ALIAS_TABLE_WRITE
		000000E888	PM_Iea_ALIAS_TABLE_HIT
		000000E08C	PM_Iea_TRACKING_TABLE_WRITE
		000000E88C	PM_Iea_ICACHE_SHARED_HIT
662	PM_OTHER_112	000000E890	PM_BR_BTAC_INV_TGT
		000000E094	PM_BR_BTAC_INV_DIR
		000000E894	PM_BR_PRED_COND_BTAC
		000000E098	PM_MPRED_BR_FIN
663	PM_OTHER_113	000000E898	PM_BR_PRED_TKN_SWHINT
		000000E09C	PM_BR_PRED_NTKN_SWHINT
		000000E89C	PM_BR_MPRED_TKN_SWHINT
		000000E0A0	PM_BR_MPRED_NTKN_SWHINT



Table E-35. Power10 Events by Group (Sheet 75 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
664	PM_OTHER_114	000000E0A4	PM_BACK_BRANCH
		20000	PM_SUSPENDED
		30000	PM_SUSPENDED
		40000	PM_SUSPENDED
665	PM_MMU_1	0000008080	PM_TLB_ACCESS_L3PREF
		0000008880	PM_TLB_CHILD_PURGE_CYC
		0000008884	PM_REJ_MMU
		0000008088	PM_REJ_EXEC_NTC_SLEEP_SELECT
666	PM_MMU_2	000000808C	PM_RDXWALK_INSTR_CYC
		000000888C	PM_RDXWALK_DATA_CYC
		0000008090	PM_REJ_TABLEWALK_L2_PDE_MERGE
		0000008890	PM_REJ_TABLEWALK_L3_PDE_MERGE
667	PM_MMU_3	0000008094	PM_REJ_TABLEWALK_PTE_MERGE
		0000008894	PM_MRK_DTABLEWALK_CYC
		0000008098	PM_XLATE_HPT_MODE_CYC
		0000008898	PM_XLATE_RADIX_MODE_CYC
668	PM_MMU_4	000000809C	PM_XLATE_L2_REQ
		00000080A0	PM_RDXTLB_ANY_NSTD_ANY_HST_HIT
		00000088A0	PM_RDXTLB_ANY_NSTD_4K_HST_HIT
		00000080A4	PM_RDXTLB_ANY_NSTD_64K_HST_HIT
669	PM_MMU_5	00000088A4	PM_RDXTLB_ANY_NSTD_2M_HST_HIT
		00000080A8	PM_RDXTLB_ANY_NSTD_1G_HST_HIT
		00000088A8	PM_RDXTLB_ANY_NSTD_ACC
		00000080AC	PM_RDXTLB_ANY_NSTD_MISS
670	PM_MMU_6	0000009080	PM_SNOOP_TLBIE_ARB_CYC
		0000009880	PM_SNOOP_TLBIE_TLB_INV_CYC
		0000009084	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_HIT_CYC
		0000009884	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_MISS_CYC
671	PM_MMU_7	0000009088	PM_SNOOP_TLBIE
		000000908C	PM_DTLB_ACCESS_ERAT_MISS
		000000988C	PM_ITLB_ACCESS_ERAT_MISS
		0000009090	PM_HPTWALK_INSTR_CYC
672	PM_MMU_8	0000009890	PM_HPTWALK_DATA_CYC
		0000009094	PM_REJ_PROBE_MATCH_PRS0
		0000009894	PM_REJ_PROBE_MATCH_PRS1
		0000009098	PM_REJ_PROBE_MATCH_MML



Table E-35. Power10 Events by Group (Sheet 76 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
673	PM_MMU_9	0000009898	PM_REJ_PIPE_COLLISION_2ND_RELOAD
		000000909C	PM_REJ_XMQ_FULL
		000000989C	PM_BLOCK_ERAT_WRITE
		00000090A0	PM_2ND_PASS_RADIX
674	PM_MMU_10	00000098A0	PM_2ND_PASS_HPT
		00000090A4	PM_VA_HASH_MPRED
		000000A080	PM_CHILD_PURGE_HIT
		000000A084	PM_TLBIE_INV_CC_PRS0_COMPOSITE_CHILD_HASH
675	PM_MMU_11	000000A884	PM_TLBIE_INV_CC_PRS0_COMPOSITE_PARENT_HASH
		000000A088	PM_TLBIE_INV_ATTEMPT_CC_PRS0_COMPOSITE_CHILD
		000000A888	PM_TLBIE_INV_ATTEMPT_CC_PRS0_COMPOSITE_PARENT
		000000A08C	PM_TLBIE_INV_ATTEMPT_HPT_PRECISE
676	PM_MMU_12	000000A88C	PM_TLBIE_INV_CC_HPT_PRECISE
		000000A090	PM_CASE_A_HIT_MTPID
		000000A890	PM_CASE_A_HIT_MTLPID
		000000A094	PM_CASE_B_HIT_MTLPID
677	PM_MMU_13	000000A894	PM_CASE_C_HIT_MTPID
		000000A098	PM_CASE_A_MISS_MTPID
		000000A898	PM_CASE_A_MISS_MTLPID
		000000A09C	PM_CASE_B_MISS_MTLPID
678	PM_MMU_14	000000A89C	PM_CASE_C_MISS_MTPID
		000000A0A0	PM_MTPID
		000000A8A0	PM_MTLPID
		000000A0A4	PM_CASE_A_SNOOP_TLBIE_HIT
679	PM_MMU_15	000000A8A4	PM_CASE_B_SNOOP_TLBIE_HIT
		000000A0A8	PM_CASE_C_SNOOP_TLBIE_HIT
		000000A8A8	PM_CASE_D_SNOOP_TLBIE_HIT
		000000A0AC	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_HIT
680	PM_MMU_16	000000A8AC	PM_SNOOP_TLBIE_NOT_MY_LPAR_CONTEXT_TABLE_MISS
		000000A0B0	PM_LPAR_SNOOP_HIT
		30000	PM_SUSPENDED
		40000	PM_SUSPENDED
681	PM_COMPAT_GROUP_1	100FE	PM_INST_CMPL
		201E2	PM_MRK_LD_MISS_L1
		300F2	PM_INST_DISP
		401E0	PM_MRK_INST_CMPL



Table E-35. Power10 Events by Group (Sheet 77 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
682	PM_COMPAT_GROUP_2	100F4	PM_FLOP_CMPL
		20002	PM_INST_CMPL
		301E2	PM_MRK_ST_CMPL
		401E4	PM_MRK_DTLB_MISS
683	PM_COMPAT_GROUP_3	100F6	PM_IERAT_MISS
		201E6	PM_THRESH_EXC_32
		30002	PM_INST_CMPL
		401E6	PM_MRK_INST_FROM_L3MISS
684	PM_COMPAT_GROUP_4	100FE	PM_INST_CMPL
		201E8	PM_THRESH_EXC_512
		301E6	PM_MRK_DERAT_MISS
		401E8	PM_MRK_DATA_FROM_L2MISS
685	PM_COMPAT_GROUP_5	100FA	PM_RUN_LATCH_ANY_THREAD_CYC
		200F0	PM_ST_CMPL
		301E8	PM_THRESH_EXC_64
		401EA	PM_THRESH_EXC_128
686	PM_COMPAT_GROUP_6	100FC	PM_LD_REF_L1
		200F4	PM_RUN_CYC
		301EA	PM_THRESH_EXC_1024
		401EC	PM_THRESH_EXC_2048
687	PM_COMPAT_GROUP_7	100FE	PM_INST_CMPL
		200F6	PM_DERAT_MISS
		30000	PM_SUSPENDED
		40000	PM_SUSPENDED
688	PM_COMPAT_GROUP_8	100FE	PM_INST_CMPL
		200F8	PM_EXT_INT
		300F4	PM_RUN_INST_CMPL_CONC
		400F2	PM_1PLUS_PPC_DISP
689	PM_COMPAT_GROUP_9	100FE	PM_INST_CMPL
		200FA	PM_BR_TAKEN_CMPL
		300F6	PM_LD_DEMAND_MISS_L1
		400F4	PM_RUN_PURR
690	PM_COMPAT_GROUP_10	100FE	PM_INST_CMPL
		200FD	PM_L1_ICACHE_MISS
		300F8	PM_TB_BIT_TRANS
		400F6	PM_BR_MPRED_CMPL



Table E-35. Power10 Events by Group (Sheet 78 of 78)

Group ID	Group Name	Group Event Codes	Group Event Names
691	PM_COMPAT_GROUP_11	100FE	PM_INST_CMPL
		200FE	PM_DATA_FROM_L2MISS
		300FA	PM_INST_FROM_L3MISS
		400F8	PM_FLUSH
692	PM_COMPAT_GROUP_12	100F2	PM_1PLUS_PPC_CMPL
		20002	PM_INST_CMPL
		300FC	PM_DTLB_MISS
		0000000	PM_RUN_INST_CMPL
693	PM_COMPAT_GROUP_13	100F8	PM_DISP_STALL_CYC
		20002	PM_INST_CMPL
		300FE	PM_DATA_FROM_L3MISS
		400FC	PM_ITLB_MISS
694	PM_COMPAT_GROUP_14	100FE	PM_INST_CMPL
		201E4	PM_MRK_DATA_FROM_L3MISS
		301E4	PM_MRK_BR_MPRED_CMPL
		400FE	PM_DATA_FROM_MEMORY



### E.5.14.2 Power10 Metric Events and Formulas

Table E-36. Metric Events and Formulas (Sheet 1 of 20)

Metric Name	Metric Description	Formula
elapsed_cycles	Time base elapsed cycles	proc_freq * total_time
Run_Cycles(%)	Percentage of cycles that are run cycles	PM_RUN_CYC/PM_CYC*100
CPI	Average cycles per completed instruction	PM_CYC/PM_INST_CMPL
RUN_CPI	Average number of run cycles per completed run instruction	PM_RUN_CYC/PM_RUN_INST_CMPL
Custom_secs	Total number of run cycles	PM_RUN_CYC
IPC	Average number of completed instructions per cycle	PM_INST_CMPL/PM_CYC
RUN_IPC	Average number of completed run instructions per run cycle	PM_RUN_INST_CMPL/PM_RUN_CYC
Cycles/Completed_Instructions_Set	Average number of cycles per completed instruction group	PM_CYC/PM_1PLUS_PPC_CMPL
Run_Latch_Cyc(%)	Percentage of cycles when the run latch was set, assuming fixed frequency.	(PM_RUN_CYC / elapsed_cycles) * 100
Cycles_Atleast_One_Inst_Dispatched(%)	Percentage of cycles when at least 1 instruction dispatched	PM_1PLUS_PPC_DISP/PM_CYC * 100
Speculation	Average number of instruction dispatched per instruction completed	PM_INST_DISP/PM_RUN_INST_CMPL
Average Completed Instruction Set Size	Average number of instruction completed per instruction group	PM_RUN_INST_CMPL/PM_1PLUS_PPC_CMPL
inst_fin_per_cmpl	Rate of finished instructions per completed instructions	PM_INST_FIN/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_CPI	Average cycles per instruction when dispatch was stalled for any reason	PM_DISP_STALL_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_FLUSH_CPI	Average cycles per instruction when dispatch was stalled because there was a flush	PM_DISP_STALL_FLUSH/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_FETCH_CPI	Average cycles per instruction when dispatch was stalled because Fetch was being held, so there was nothing in the pipeline for this thread	PM_DISP_STALL_FETCH/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_TRANSLATION_CPI	Average cycles per instruction when dispatch was stalled because the MMU was handling a translation miss	PM_DISP_STALL_TRANSLATION/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_IERAT_ONLY_MISS_CPI	Average cycles per instruction when dispatch was stalled waiting to resolve an instruction ERAT miss	PM_DISP_STALL_IERAT_ONLY_MISS/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_ITLB_MISS_CPI	Average cycles per instruction when dispatch was stalled waiting to resolve an instruction TLB miss	PM_DISP_STALL_ITLB_MISS/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_IC_MISS_CPI	Average cycles per instruction when dispatch was stalled due to an icache miss	PM_DISP_STALL_IC_MISS/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_IC_L2_CPI	Average cycles per instruction when dispatch was stalled while the instruction was fetched from the local L2	PM_DISP_STALL_IC_L2/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_IC_L3_CPI	Average cycles per instruction when dispatch was stalled while the instruction was fetched from the local L3	PM_DISP_STALL_IC_L3/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_IC_L3MISS_CPI	Average cycles per instruction when dispatch was stalled while the instruction was fetched from any source beyond the local L3	PM_DISP_STALL_IC_L3MISS/PM_RUN_INST_CMPL

Table E-36. Metric Events and Formulas (Sheet 2 of 20)

Metric Name	Metric Description	Formula
NOTHING_DISPATCHED_BR_MPRED_ICMISS_CPI	Average cycles per instruction when dispatch was stalled due to an icache miss after a branch mispredict	PM_DISP_STALL_BR_MPRED_ICMISS/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_BR_MPRED_IC_L2_CPI	Average cycles per instruction when dispatch was stalled while instruction was fetched from the local L2 after suffering a branch mispredict	PM_DISP_STALL_BR_MPRED_IC_L2/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_BR_MPRED_IC_L3_CPI	Average cycles per instruction when dispatch was stalled while instruction was fetched from the local L3 after suffering a branch mispredict	PM_DISP_STALL_BR_MPRED_IC_L3/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_BR_MPRED_IC_L3MISS_CPI	Average cycles per instruction when dispatch was stalled while instruction was fetched from any source beyond the local L3 after suffering a branch mispredict	PM_DISP_STALL_BR_MPRED_IC_L3MISS/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_BR_MPRED_CPI	Average cycles per instruction when dispatch was stalled due to a branch mispredict	PM_DISP_STALL_BR_MPRED/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_CPI	Average cycles per instruction when the ntc instruction was held at dispatch for any reason	PM_DISP_STALL_HELD_CYC/PM_RUN_INST_CMPL
DISP_HELD_STALL_SYNC_CPI	Average cycles per instruction when the ntc instruction was held at dispatch because of a synchronizing instruction that requires the ICT to be empty before dispatch	PM_DISP_STALL_HELD_SYNC_CYC/PM_RUN_INST_CMPL
DISP_HELD_STALL_SCOREBOARD_CPI	Average cycles per instruction when the ntc instruction was held at dispatch while waiting on the scoreboard	PM_DISP_STALL_HELD_SCOREBOARD_CYC/PM_RUN_INST_CMPL
DISP_HELD_STALL_ISSQ_FULL_CPI	Average cycles per instruction when the ntc instruction was held at dispatch due to issue q full	PM_DISP_STALL_HELD_ISSQ_FULL_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_RENAME_CPI	Average cycles per instruction when the ntc instruction was held at dispatch because the mapper/SRB was full	PM_DISP_STALL_HELD_RENAME_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_STF_MAPPER_CPI	Average cycles per instruction when the ntc instruction was held at dispatch because the STF mapper/SRB was full	PM_DISP_STALL_HELD_STF_MAPPER_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_XVFC_MAPPER_CPI	Average cycles per instruction when the ntc instruction was held at dispatch because the XVFC mapper/SRB was full	PM_DISP_STALL_HELD_XVFC_MAPPER_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_HALT_CPI	Average cycles per instruction when the ntc instruction was held at dispatch because of power management	PM_DISP_STALL_HELD_HALT_CYC/PM_RUN_INST_CMPL
NOTHING_DISPATCHED_HELD_OTHER_CPI	Average cycles per instruction when the ntc instruction was held at dispatch for any other reason	PM_DISP_STALL_HELD_OTHER_CYC/PM_RUN_INST_CMPL
ISSUE_STALL_CPI	Average cycles per instruction when the ntc instruction has been dispatched but not issued for any reason	PM_ISSUE_STALL/PM_RUN_INST_CMPL
EXECUTION_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting to be finished in one of the execution units	PM_EXEC_STALL/PM_RUN_INST_CMPL
NTC_FLUSH_STALL_CPI	Average cycles per instruction spent executing an NTC instruction that gets flushed some time after dispatch	PM_EXEC_STALL_NTC_FLUSH/PM_RUN_INST_CMPL
FIN_AT_DISP_STALL_CPI	Average cycles per instruction when the instruction finishes at dispatch	PM_EXEC_STALL_FIN_AT_DISP/PM_RUN_INST_CMPL
BRU_STALL_CPI	Average cycles per instruction when the ntc instruction is executing in the branch unit	PM_EXEC_STALL_BRU/PM_RUN_INST_CMPL
SIMPLE_FX_STALL_CPI	Average cycles per instruction when the ntc instruction is a simple fixed point instr that is executing in the lsu unit	PM_EXEC_STALL_SIMPLE_FX/PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 3 of 20)

Metric Name	Metric Description	Formula
VSU_STALL_CPI	Average cycles per instruction when the ntc instruction is executing in the vsu unit	PM_EXEC_STALL_VSU/PM_RUN_INST_CMPL
TRANSLATION_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting to be finished in one of the execution units	PM_EXEC_STALL_TRANSLATION/PM_RUN_INST_CMPL
DERAT_ONLY_MISS_STALL_CPI	Average cycles per instruction when the ntc instruction is a load or store that suffered a translation miss	PM_EXEC_STALL_DERAT_ONLY_MISS/PM_RUN_INST_CMPL
DERAT_DTLB_MISS_STALL_CPI	Average cycles per instruction when the ntc instruction is recovering from a TLB miss	PM_EXEC_STALL_DERAT_DTLB_MISS/PM_RUN_INST_CMPL
LSU_STALL_CPI	Average cycles per instruction when the ntc instruction is executing in the lsu unit	PM_EXEC_STALL_LSU/PM_RUN_INST_CMPL
LOAD_STALL_CPI	Average cycles per instruction when the ntc instruction is a load that is executing in the lsu unit	PM_EXEC_STALL_LOAD/PM_RUN_INST_CMPL
DMISS_L2L3_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from either the local L2 or local L3	PM_EXEC_STALL_DMISS_L2L3/PM_RUN_INST_CMPL
DMISS_L2L3_CONFLICT_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from either the local L2 or local L3, with an RC dispatch conflict	PM_EXEC_STALL_DMISS_L2L3_CONFLICT/PM_RUN_INST_CMPL
DMISS_L2L3_NOCONFLICT_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from either the local L2 or local L3, without an RC dispatch conflict	PM_EXEC_STALL_DMISS_L2L3_NOCONFLICT/PM_RUN_INST_CMPL
DMISS_L3MISS_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from a source beyond the local L2 and local L3	PM_EXEC_STALL_DMISS_L3MISS/PM_RUN_INST_CMPL
DMISS_L21_L31_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from a neighbor chiplet's L2 or L3 in the same chip.	PM_EXEC_STALL_DMISS_L21_L31/PM_RUN_INST_CMPL
DMISS_LMEM_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from local memory, L4 or OpenCAPP chip	PM_EXEC_STALL_DMISS_LMEM/PM_RUN_INST_CMPL
DMISS_OFF_CHIP_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from a remote chip (cache, L4, memory or CAPP) in the same group	PM_EXEC_STALL_DMISS_OFF_CHIP/PM_RUN_INST_CMPL
DMISS_OFF_NODE_STALL_CPI	Average cycles per instruction when the ntc instruction is waiting for a load miss to resolve from a distant chip (cache, L4, memory or CAPP chip)	PM_EXEC_STALL_DMISS_OFF_NODE/PM_RUN_INST_CMPL
TLBIEL_STALL_CPI	Average cycles per instruction when the ntc instruction is executing a TLBIEL instruction	PM_EXEC_STALL_TLBIEL/PM_RUN_INST_CMPL
LOAD_FINISH_STALL_CPI	Average cycles per instruction when the ntc instruction is finishing a load after its data has been reloaded from a data source beyond the local L1, OR when the LSU is processing an L1-hit, OR when the NTF instruction merged with another load in the LMQ	PM_EXEC_STALL_LOAD_FINISH/PM_RUN_INST_CMPL
STORE_STALL_CPI	Average cycles per instruction when the ntc instruction is a store that is executing in the lsu unit	PM_EXEC_STALL_STORE/PM_RUN_INST_CMPL
STORE_PIPE_STALL_CPI	Average cycles per instruction when the ntc instruction is in the store unit outside of handling store misses or other special store operations	PM_EXEC_STALL_STORE_PIPE/PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 4 of 20)

Metric Name	Metric Description	Formula
STORE_MISS_STALL_CPI	Average cycles per instruction when the ntc instruction is a store whose cache line was not resident in the L1 and had to wait for allocation of the missing line into the L1	PM_EXEC_STALL_STORE_MISS/ PM_RUN_INST_CMPL
TLBIE_STALL_CPI	Average cycles per instruction when the ntc instruction is a TLBIE instruction waiting for a response from the L2	PM_EXEC_STALL_TLBIE/ PM_RUN_INST_CMPL
PTESYNC_STALL_CPI	Average cycles per instruction when the ntc instruction is executing a PTESYNC instruction	PM_EXEC_STALL_PTESYNC/ PM_RUN_INST_CMPL
COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction cannot complete because the thread was blocked	PM_CMPL_STALL/ PM_RUN_INST_CMPL
EXCEPTION_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction cannot complete because it was interrupted by ANY exception	PM_CMPL_STALL_EXCEPTION/ PM_RUN_INST_CMPL
MEM_ECC_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction is stuck at finish waiting for the non-speculative finish of either a stcx waiting for its result or a load waiting for non-critical sectors of data and ECC	PM_CMPL_STALL_MEM_ECC/ PM_RUN_INST_CMPL
STCX_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction cannot complete the instruction is a stcx waiting for resolution from the nest.	PM_CMPL_STALL_STCX/ PM_RUN_INST_CMPL
LWSYNC_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction is a LWSYNC instruction waiting to complete	PM_CMPL_STALL_LWSYNC/ PM_RUN_INST_CMPL
HWSYNC_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction is a HWSYNC instruction stuck at finish waiting for a response from the L2	PM_CMPL_STALL_HWSYNC/ PM_RUN_INST_CMPL
SPECIAL_COMPLETION_STALL_CPI	Average cycles per instruction when the ntc instruction required special handling before completion	PM_CMPL_STALL_SPECIAL/ PM_RUN_INST_CMPL
UNKNOWN_CPI	Average cycles per instruction when the ntf instruction is completing and the finish was overlooked	PM_EXEC_STALL_UNKNOWN/ PM_RUN_INST_CMPL
Flush_Rate(%)	Percentage of flushes per completed instruction	PM_FLUSH * 100 / PM_RUN_INST_CMPL
LSU_Flush_Rate(%)	Percentage of cycles with at least 1 LSU flush per completed instruction	PM_LSU_FLUSH *100 / PM_RUN_INST_CMPL
ISU_Flush_Rate(%)	Percentage of flushes initiated by the ISU per completed instruction	PM_ISU_FLUSH *100 / PM_RUN_INST_CMPL
Disp_Flush_Rate(%)	Percentage of dispatch flushes per completed instruction	PM_ISU_FLUSH_DISP / PM_RUN_INST_CMPL * 100
Br_Mpred_Flush_Rate(%)	Percentage of flushes due to a branch mispredict per instruction	PM_FLUSH_MPRED / PM_RUN_INST_CMPL * 100
DTLB_Miss_Rate(%)	Percentage of instructions when the DPTEG required for the load/store instruction in execution was missing from the TLB	PM_DTLB_MISS * 100/ PM_RUN_INST_CMPL
DC_Reload_Collision_Rate(%)	Percentage of bank collisions caused by a load reading the L1 cache per completed instruction	PM_DC_RELOAD_COLLISIONS * 100 / PM_RUN_INST_CMPL
Issue_Collision_Rate(%)	Percentage of resource collisions that occurred at issue per completed instruction	PM_ISSUE_COLLISION * 100 / PM_RUN_INST_CMPL
DC_Store_Write_Collision_Rate(%)	Percentage of bank collisions caused by a store writing to the L1 cache per completed instruction	PM_DC_STORE_WRITE_COLLISIONS * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 5 of 20)

Metric Name	Metric Description	Formula
Branch_misprediction_rate	Percentage of branch mispredictions per completed instruction	$PM\_BR\_MPRED\_CMPL * 100 / PM\_RUN\_INST\_CMPL$
Branch_miss_direction	Average number of conditional branches that finished with mispredicted direction per conditional branch that finished	$(PM\_MPRED\_BR\_TKN\_COND\_DIR + PM\_MPRED\_BR\_NTKN\_COND\_DIR) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_target_address	Average number of conditional branches that finished with mispredicted target per conditional branch that finished	$(PM\_MPRED\_BR\_TKN\_COND\_TGT + PM\_MPRED\_BR\_NTKN\_COND\_TGT) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_direction_lbht_lsel	Average number of conditional branches that finished with mispredicted direction using the Local Branch History Table selected by the local selector per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_COND\_DIR\_LBHT\_LSEL + PM\_BR\_MPRED\_NTKN\_COND\_DIR\_LBHT\_LSEL) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_direction_lbht_gsel	Average number of conditional branches that finished with mispredicted direction using the Local Branch History Table selected by the global selector per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_COND\_DIR\_LBHT\_GSEL + PM\_BR\_MPRED\_NTKN\_COND\_DIR\_LBHT\_GSEL) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_direction_gbht	Average number of conditional branches that finished with mispredicted direction using the Global Branch History Table per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_COND\_DIR\_GBHT + PM\_BR\_MPRED\_NTKN\_COND\_DIR\_GBHT) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_direction_tage	Average number of conditional branches that finished with mispredicted direction using a TAGE override per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_COND\_DIR\_TAGE + PM\_BR\_MPRED\_NTKN\_COND\_DIR\_TAGE) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_direction_top	Average number of conditional branches that finished with mispredicted direction using a TOP override per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_COND\_DIR\_TOP + PM\_BR\_MPRED\_NTKN\_COND\_DIR\_TOP) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Branch_miss_hint	Average number of conditional branches that finished with misprediction using a software hint per conditional branch that finished	$(PM\_BR\_MPRED\_TKN\_SWHINT + PM\_BR\_MPRED\_NTKN\_SWHINT) / (PM\_BR\_FIN - PM\_BR\_TKN\_UNCOND\_FIN)$
Taken_Branches(%)	Percentage of finished branches that were taken	$PM\_BR\_TAKEN\_CMPL * 100 / PM\_BR\_FIN$
L1_LD_Miss_Ratio(%)	Percentage of finished loads that missed in the L1	$PM\_LD\_MISS\_L1 / PM\_LD\_REF\_L1 * 100$
L1_ST_Miss_Ratio(%)	Percentage of finished stores that missed in the L1	$PM\_ST\_MISS\_L1 / PM\_ST\_FIN * 100$
L1_LD_Miss_Rate(%)	Percentage of completed instructions that were loads that missed the L1	$PM\_LD\_MISS\_L1 * 100 / PM\_RUN\_INST\_CMPL$
L1_ST_Miss_Rate(%)	Percentage of completed instructions that were stores that missed the L1	$PM\_ST\_MISS\_L1 * 100 / PM\_RUN\_INST\_CMPL$
L2_LD_Miss_Rate(%)	Percentage of completed instructions that were a demand load that did not hit in the L1 or L2	$PM\_DATA\_FROM\_L2MISS * 100 / PM\_RUN\_INST\_CMPL$
L3_LD_Miss_Rate(%)	Percentage of completed instructions that were a demand load that did not hit in the L1, L2, or the L3	$PM\_DATA\_FROM\_L3MISS * 100 / PM\_RUN\_INST\_CMPL$
L2_PTEG_Miss_Rate(%)	Percentage of completed instructions that were a data side request that did not hit in the L1 or L2	$PM\_DPTEG\_FROM\_L2MISS * 100 / PM\_RUN\_INST\_CMPL$



Table E-36. Metric Events and Formulas (Sheet 6 of 20)

Metric Name	Metric Description	Formula
L3_PTEG_Miss_Rate(%)	Percentage of completed instructions that were a data side request that did not hit in the L1 or L2, or L3	PM_DPTEG_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
L1_Inst_Miss_Rate(%)	Percentage of completed instructions that were demand fetches that missed the L1 instruction cache	PM_L1_ICACHE_MISS * 100 / PM_RUN_INST_CMPL
L2_Inst_Miss_Rate(%)	Percentage of completed instructions that were demand fetches that missed the L1 and L2 instruction cache	PM_INST_FROM_L2MISS * 100 / PM_RUN_INST_CMPL
L3_Inst_Miss_Rate(%)	Percentage of completed instructions that were demand fetches that reloaded from beyond the L3 instruction cache	PM_INST_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
Loads_per_inst	Rate of finished loads per completed instruction	PM_LD_REF_L1 / PM_RUN_INST_CMPL
Stores_per_inst	Rate of finished stores per completed instruction	PM_ST_FIN / PM_RUN_INST_CMPL
Branches_per_inst	Rate of finished branches per completed instruction	PM_BR_FIN / PM_RUN_INST_CMPL
LSU_per_inst	Rate of instructions finished in the LSU per completed instruction	PM_LSU_FIN / PM_RUN_INST_CMPL
VSU_per_inst	Rate of instructions finished in the VSU per completed instruction	PM_VSU_FIN / PM_RUN_INST_CMPL
TLBIE_per_inst	Rate of TLBIE instructions finished in the LSU per completed instruction	PM_TLBIE_FIN / PM_RUN_INST_CMPL
STXC_per_inst	Rate of STCX instructions finished per completed instruction	PM_STCX_FIN / PM_RUN_INST_CMPL
LARX_per_inst	Rate of LARX instructions finished per completed instruction	PM_LARX_FIN / PM_RUN_INST_CMPL
PTESYNC_per_inst	Rate of ptesync instructions finished per completed instruction	PM_PTESYNC_FIN / PM_RUN_INST_CMPL
ICBI_per_inst	Rate of ICBI instructions finished per completed instruction	PM_ICBI_FIN / PM_RUN_INST_CMPL
FX_per_inst	Rate of simple fixed-point instructions finished in the store unit per completed instruction	PM_FX_LSU_FIN / PM_RUN_INST_CMPL
dL1_Miss_Reloads(%)	Percentage of demand load misses that reloaded the L1 cache	PM_LD_DEMAND_MISS_L1 * 100 / PM_LD_MISS_L1
dL1_Reload_FROM_L2_Rate(%)	Percentage of demand loads that reloaded from the L2 per completed instruction	PM_DATA_FROM_L2 * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_L2_Miss_Rate(%)	Percentage of demand loads that reloaded from beyond the L2 per completed instruction	PM_DATA_FROM_L2MISS * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_RL2L3_MOD_Rate(%)	Percentage of demand loads that reloaded using modified data from another core's L2 or L3 on a remote chip, per completed instruction	PM_DATA_FROM_RL2L3_MOD * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_RL2L3_SHR_Rate(%)	Percentage of demand loads that reloaded using shared data from another core's L2 or L3 on a remote chip, per completed instruction	PM_DATA_FROM_RL2L3_SHR * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_L3_Rate(%)	Percentage of demand loads that reloaded from the L3 per completed instruction	PM_DATA_FROM_L3 * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_L3_MEPPF_Rate(%)	Percentage of demand loads that reloaded with data brought into the L3 by prefetch per completed instruction	PM_DATA_FROM_L3_MEPPF * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 7 of 20)

Metric Name	Metric Description	Formula
dL1_Reload_FROM_L3_Miss_Rate(%)	Percentage of demand loads that reloaded from beyond the L3 per completed instruction	PM_DATA_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_DL2L3_MOD_Rate(%)	Percentage of demand loads that reloaded using modified data from another core's L2 or L3 on a distant chip, per completed instruction	PM_DATA_FROM_DL2L3_MOD * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_DL2L3_SHR_Rate(%)	Percentage of demand loads that reloaded using shared data from another core's L2 or L3 on a distant chip, per completed instruction	PM_DATA_FROM_DL2L3_SHR * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_LMEM_Rate(%)	Percentage of demand loads that reloaded from local memory per completed instruction	PM_DATA_FROM_LMEM * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_RMEM_Rate(%)	Percentage of demand loads that reloaded from remote memory per completed instruction	PM_DATA_FROM_RMEM * 100 / PM_RUN_INST_CMPL
dL1_Reload_FROM_DMEM_Rate(%)	Percentage of demand loads that reloaded from distant memory per completed instruction	PM_DATA_FROM_DMEM * 100 / PM_RUN_INST_CMPL
L2_Latency	Average marked L2 load latency, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L2_CYC / PM_MRK_DATA_FROM_L2
L2_Mepf_state_Latency	Average marked L2 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L2_MEFP_CYC / PM_MRK_DATA_FROM_L2_MEFP
L2_Disp_Conflict_Other_Latency	Average marked L2 load latency for data with dispatch conflict other, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L2_DISP_CONFLICT_OTHER_CYC / PM_MRK_DATA_FROM_L2_DISP_CONFLICT_OTHER
L2_Disp_Conflict_LDHITST_Latency	Average marked L2 load latency for data with dispatch conflict load-hit-store, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L2_DISP_CONFLICT_LDHITST_CYC / PM_MRK_DATA_FROM_L2_DISP_CONFLICT_LDHITST
L3_Latency	Average marked L3 load latency, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L3_CYC / PM_MRK_DATA_FROM_L3
L3_Mepf_state_Latency	Average marked L3 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L3_MEFP_CYC / PM_MRK_DATA_FROM_L3_MEFP
L3_Disp_Conflict_Latency	Average marked L3 load latency for data with dispatch conflict, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L3_DISP_CONFLICT_CYC / PM_MRK_DATA_FROM_L3_DISP_CONFLICT
L21_Regent_MOD_Latency	Average marked regent L21 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L21_REGENT_MOD_CYC / PM_MRK_DATA_FROM_L21_REGENT_MOD
L21_Regent_SHR_Latency	Average marked regent L21 load latency for shared data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L21_REGENT_SHR_CYC / PM_MRK_DATA_FROM_L21_REGENT_SHR
L21_Non_Regent_MOD_Latency	Average marked non-regent L21 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L21_NON_REGENT_MOD_CYC / PM_MRK_DATA_FROM_L21_NON_REGENT_MOD
L21_Non_Regent_SHR_Latency	Average marked non-regent L21 load latency for shared data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L21_NON_REGENT_SHR_CYC / PM_MRK_DATA_FROM_L21_NON_REGENT_SHR

Table E-36. Metric Events and Formulas (Sheet 8 of 20)

Metric Name	Metric Description	Formula
L31_Regent_MOD_Latency	Average marked regent L31 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L31_REGENT_MOD_CYC / PM_MRK_DATA_FROM_L31_REGENT_MOD
L31_Regent_SHR_Latency	Average marked regent L31 load latency for shared data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L31_REGENT_SHR_CYC / PM_MRK_DATA_FROM_L31_REGENT_SHR
L31_Non_Regent_MOD_Latency	Average marked non-regent L31 load latency for modified data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L31_NON_REGENT_MOD_CYC / PM_MRK_DATA_FROM_L31_NON_REGENT_MOD
L31_Non_Regent_SHR_Latency	Average marked non-regent L31 load latency for shared data, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L31_NON_REGENT_SHR_CYC / PM_MRK_DATA_FROM_L31_NON_REGENT_SHR
RL2_MOD_Latency	Average marked load latency for modified data from another core's L2 cache on a remote chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_RL2_MOD_CYC / PM_MRK_DATA_FROM_RL2_MOD
RL2_SHR_Latency	Average marked load latency for shared data from another core's L2 on a remote chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_RL2_SHR_CYC / PM_MRK_DATA_FROM_RL2_SHR
RL3_MOD_Latency	Average marked load latency for modified data from another core's L3 cache on a remote chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_RL3_MOD_CYC / PM_MRK_DATA_FROM_RL3_MOD
RL3_SHR_Latency	Average marked load latency for shared data from another core's L3 cache on a remote chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_RL3_SHR_CYC / PM_MRK_DATA_FROM_RL3_SHR
DL2_MOD_Latency	Average marked load latency for modified data from another core's L2 cache on a distant chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_DL2_MOD_CYC / PM_MRK_DATA_FROM_DL2_MOD
DL2_SHR_Latency	Average marked load latency for shared data from another core's L2 cache on a distant chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_DL2_SHR_CYC / PM_MRK_DATA_FROM_DL2_SHR
DL3_MOD_Latency	Average marked load latency for modified data from another core's L3 cache on a distant chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_DL3_MOD_CYC / PM_MRK_DATA_FROM_DL3_MOD
DL3_SHR_Latency	Average marked load latency for shared data from another core's L3 cache on a distant chip, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_DL3_SHR_CYC / PM_MRK_DATA_FROM_DL3_SHR
LMEM_Latency	Average marked load latency for local memory, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_LMEM_CYC / PM_MRK_DATA_FROM_LMEM
L_OC_CACHE_Latency	Average marked load latency for the local chip's OpenCAPP cache, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L_OC_CACHE_CYC / PM_MRK_DATA_FROM_L_OC_CACHE
L_OC_MEM_Latency	Average marked load latency for the local chip's OpenCAPP memory, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_L_OC_MEM_CYC / PM_MRK_DATA_FROM_L_OC_MEM

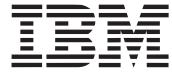


Table E-36. Metric Events and Formulas (Sheet 9 of 20)

Metric Name	Metric Description	Formula
RMEM_Latency	Average marked remote memory load latency, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_RMEM_CYC / PM_MRK_DATA_FROM_RMEM
R_OC_CACHE_Latency	Average marked load latency for a remote chip's OpenCAPP cache, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_R_OC_CACHE_CYC / PM_MRK_DATA_FROM_R_OC_CACHE
R_OC_MEM_Latency	Average marked load latency for a remote chip's OpenCAPP memory, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_R_OC_MEM_CYC / PM_MRK_DATA_FROM_R_OC_MEM
DMEM_Latency	Average marked distant memory load latency, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_DMEM_CYC / PM_MRK_DATA_FROM_DMEM
D_OC_CACHE_Latency	Average marked load latency for a distant chip's OpenCAPP cache, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_D_OC_CACHE_CYC / PM_MRK_DATA_FROM_D_OC_CACHE
D_OC_MEM_Latency	Average marked load latency for a distant chip's OpenCAPP memory, where latency is the average cycles between the first miss and the reload	PM_MRK_DATA_FROM_D_OC_MEM_CYC / PM_MRK_DATA_FROM_D_OC_MEM
Estimated_dL1Miss_Latency	Average marked L1 miss latency, where latency is the average cycles between the first miss and the reload	PM_MRK_LD_MISS_L1_CYC / PM_MRK_LD_MISS_L1
dL1_Reload_FROM_L2(%)	Percentage of demand load misses that reloaded from the local L2 cache	PM_DATA_FROM_L2 * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L2_Miss(%)	Percentage of demand load misses that reloaded from beyond the local L2 cache	PM_DATA_FROM_L2MISS * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L3(%)	Percentage of demand load misses that reloaded from the local L3 cache	PM_DATA_FROM_L3 * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L3_Miss(%)	Percentage of demand load misses that reloaded from beyond the local L3 cache	PM_DATA_FROM_L3MISS * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L3_MEPF(%)	Percentage of demand load misses that reloaded from the local L3 with modified data	PM_DATA_FROM_L3_MEPF * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L21_Regent_MOD(%)	Percentage of demand load misses that reloaded from another core's L2 on the same regent with modified data	PM_DATA_FROM_L21_REGENT_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L21_Regent_SHR(%)	Percentage of demand load misses that reloaded from another core's L2 on the same regent with shared data	PM_DATA_FROM_L21_REGENT_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L21_Non_Regent_MOD(%)	Percentage of demand load misses that reloaded from another core's L2 on the same chip in a different regent with modified data	PM_DATA_FROM_L21_NON_REGENT_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L21_Non_Regent_SHR(%)	Percentage of demand load misses that reloaded from another core's L2 on the same chip in a different regent with shared data	PM_DATA_FROM_L21_NON_REGENT_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L31_Regent_MOD(%)	Percentage of demand load misses that reloaded from another core's L3 on the same regent with modified data	PM_DATA_FROM_L31_REGENT_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L31_Regent_SHR(%)	Percentage of demand load misses that reloaded from another core's L3 on the same regent with shared data	PM_DATA_FROM_L31_REGENT_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L31_Non_Regent_MOD(%)	Percentage of demand load misses that reloaded from another core's L3 on the same chip in a different regent with modified data	PM_DATA_FROM_L31_NON_REGENT_MOD * 100 / PM_LD_DEMAND_MISS_L1



Table E-36. Metric Events and Formulas (Sheet 10 of 20)

Metric Name	Metric Description	Formula
dL1_Reload_FROM_L31_Non_Regent_SHR(%)	Percentage of demand load misses that reloaded from another core's L3 on the same chip in a different regent with shared data	PM_DATA_FROM_L31_NON_REGENT_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_RL2_MOD(%)	Percentage of demand load misses that reloaded from another core's L2 on a remote chip with modified data	PM_DATA_FROM_RL2_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_RL2_SHR(%)	Percentage of demand load misses that reloaded from another core's L2 on a remote chip with shared data	PM_DATA_FROM_RL2_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_RL3_MOD(%)	Percentage of demand load misses that reloaded from another core's L3 on a remote chip with modified data	PM_DATA_FROM_RL3_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_RL3_SHR(%)	Percentage of demand load misses that reloaded from another core's L3 on a remote chip with shared data	PM_DATA_FROM_RL3_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_DL2_MOD(%)	Percentage of demand load misses that reloaded from another core's L2 on a distant chip with modified data	PM_DATA_FROM_DL2_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_DL2_SHR(%)	Percentage of demand load misses that reloaded from another core's L2 on a distant chip with shared data	PM_DATA_FROM_DL2_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_DL3_MOD(%)	Percentage of demand load misses that reloaded from another core's L3 on a distant chip with modified data	PM_DATA_FROM_DL3_MOD * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_DL3_SHR(%)	Percentage of demand load misses that reloaded from another core's L3 on a distant chip with shared data	PM_DATA_FROM_DL3_SHR * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_LMEM(%)	Percentage of demand load misses that reloaded from the local chip's memory	PM_DATA_FROM_LMEM * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L_OC_CACHE(%)	Percentage of demand load misses that reloaded from the local chip's OpenCAPP Cache	PM_DATA_FROM_L_OC_CACHE * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_L_OC_MEM(%)	Percentage of demand load misses that reloaded from the local chip's OpenCAPP memory	PM_DATA_FROM_L_OC_MEM * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_RMEM(%)	Percentage of demand load misses that reloaded from a remote chip's memory	PM_DATA_FROM_RMEM * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_R_OC_CACHE(%)	Percentage of demand load misses that reloaded from a remote chip's OpenCAPP Cache	PM_DATA_FROM_R_OC_CACHE * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_R_OC_MEM(%)	Percentage of demand load misses that reloaded from a remote chip's OpenCAPP memory	PM_DATA_FROM_R_OC_MEM * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_DMEM(%)	Percentage of demand load misses that reloaded from a distant chip's memory	PM_DATA_FROM_DMEM * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_D_OC_CACHE(%)	Percentage of demand load misses that reloaded from a distant chip's OpenCAPP Cache	PM_DATA_FROM_D_OC_CACHE * 100 / PM_LD_DEMAND_MISS_L1
dL1_Reload_FROM_D_OC_MEM(%)	Percentage of demand load misses that reloaded from a distant chip's OpenCAPP memory	PM_DATA_FROM_D_OC_MEM * 100 / PM_LD_DEMAND_MISS_L1
dcache_miss_cpi(%)	Percentage of cycles stalled due to the ntc instruction waiting for a load miss to resolve from a source beyond the local L2 and local L3	DMISS_L3MISS_STALL_CPI / RUN_CPI * 100
MEM_LOCALITY(%)	Percentage of data reloads from local memory per data reloads from any memory	PM_DATA_FROM_LMEM * 100 / (PM_DATA_FROM_ANY_MEMORY)
LD_LMEM_PER_LD_RMEM	Number of data reloads from local memory per data reloads from remote memory	PM_DATA_FROM_LMEM / PM_DATA_FROM_RMEM
LD_LMEM_PER_LD_DMEM	Number of data reloads from local memory per data reloads from distant memory	PM_DATA_FROM_LMEM / PM_DATA_FROM_DMEM



Table E-36. Metric Events and Formulas (Sheet 11 of 20)

Metric Name	Metric Description	Formula
LD_LMEM_PER_LD_MEM	Number of data reloads from local memory per data reloads from distant and remote memory	PM_DATA_FROM_LMEM / (PM_DATA_FROM_DMEM + PM_DATA_FROM_RMEM)
LD_RMEM_PER_LD_DMEM	Number of data reloads from remote memory per data reloads from distant memory	PM_DATA_FROM_RMEM / PM_DATA_FROM_DMEM
ITLB_Miss_Rate(%)	Percentage of ITLB misses per completed run instruction	PM_ITLB_MISS / PM_RUN_INST_CMPL *100
DERAT_Miss_Rate(%)	Percentage of DERAT misses per completed run instruction	PM_DERAT_MISS * 100 / PM_RUN_INST_CMPL
DERAT_4K_Miss_Rate(%)	Percentage of DERAT misses with 4k page size per completed run instruction	PM_DERAT_MISS_4K * 100 / PM_RUN_INST_CMPL
DERAT_2M_Miss_Rate(%)	Percentage of DERAT misses with 2M page size per completed run instruction	PM_DERAT_MISS_2M * 100 / PM_RUN_INST_CMPL
DERAT_16M_Miss_Rate(%)	Percentage of DERAT misses with 16M page size per completed run instruction	PM_DERAT_MISS_16M * 100 / PM_RUN_INST_CMPL
DERAT_16G_Miss_Rate(%)	Percentage of DERAT misses with 16G page size per completed run instruction	PM_DERAT_MISS_16G * 100 / PM_RUN_INST_CMPL
DERAT_1G_Miss_Rate(%)	Percentage of DERAT misses with 1G page size per completed run instruction	PM_DERAT_MISS_1G* 100 / PM_RUN_INST_CMPL
DERAT_64K_Miss_Rate(%)	Percentage of DERAT misses with 64k page size per completed run instruction	PM_DERAT_MISS_64K * 100 / PM_RUN_INST_CMPL
DERAT_4K_Miss_Ratio	DERAT miss ratio for 4K page size	PM_DERAT_MISS_4K / PM_DERAT_MISS
DERAT_2M_Miss_Ratio	DERAT miss ratio for 2M page size	PM_DERAT_MISS_2M / PM_DERAT_MISS
DERAT_16M_Miss_Ratio	DERAT miss ratio for 16M page size	PM_DERAT_MISS_16M / PM_DERAT_MISS
DERAT_16G_Miss_Ratio	DERAT miss ratio for 16G page size	PM_DERAT_MISS_16G / PM_DERAT_MISS
DERAT_1G_Miss_Ratio	DERAT miss ratio for 1G page size	PM_DERAT_MISS_1G / PM_DERAT_MISS
DERAT_64K_Miss_Ratio	DERAT miss ratio for 64K page size	PM_DERAT_MISS_64K / PM_DERAT_MISS
DERAT_MISS_RELOAD(%)	Percentage of DERAT misses that resulted in TLB reloads	PM_DTLB_MISS * 100 / PM_DERAT_MISS
DPTEG_FROM_L2(%)	Percentage of DTLB misses that were reloaded from the L2	PM_DPTEG_FROM_L2 * 100 / PM_DTLB_MISS
DPTEG_FROM_L2_Miss(%)	Percentage of DTLB misses that were reloaded from beyond the L2	PM_DPTEG_FROM_L2MISS * 100 / PM_DTLB_MISS
DPTEG_FROM_L3(%)	Percentage of DTLB misses that were reloaded from the L3	PM_DPTEG_FROM_L3 * 100 / PM_DTLB_MISS
DPTEG_FROM_L3_Miss(%)	Percentage of DTLB misses that were reloaded from beyond the L3	PM_DPTEG_FROM_L3MISS * 100 / PM_DTLB_MISS
DPTEG_FROM_L21_Regent_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L2 on the same regent with modified data	PM_DPTEG_FROM_L21_REGENT_MOD * 100 / PM_DTLB_MISS

Table E-36. Metric Events and Formulas (Sheet 12 of 20)

Metric Name	Metric Description	Formula
DPTEG_FROM_L21_Regent_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L2 on the same regent with shared data	PM_DPTEG_FROM_L21_REGENT_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_L21_Non_Regent_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a different regent with modified data	PM_DPTEG_FROM_L21_NON_REGENT_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_L21_Non_Regent_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a different regent with shared data	PM_DPTEG_FROM_L21_NON_REGENT_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_L31_Regent_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L3 on the same regent with modified data	PM_DPTEG_FROM_L31_REGENT_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_L31_Regent_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L3 on the same regent with shared data	PM_DPTEG_FROM_L31_REGENT_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_L31_Non_Regent_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a different regent with modified data	PM_DPTEG_FROM_L31_NON_REGENT_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_L31_Non_Regent_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a different regent with shared data	PM_DPTEG_FROM_L31_NON_REGENT_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_RL2_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a remote chip with modified data	PM_DPTEG_FROM_RL2_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_RL2_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a remote chip with shared data	PM_DPTEG_FROM_RL2_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_RL3_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a remote chip with modified data	PM_DPTEG_FROM_RL3_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_RL3_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a remote chip with shared data	PM_DPTEG_FROM_RL3_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_DL2_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a distant chip with modified data	PM_DPTEG_FROM_DL2_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_DL2_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L2 on a distant chip with shared data	PM_DPTEG_FROM_DL2_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_DL3_MOD(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a distant chip with modified data	PM_DPTEG_FROM_DL3_MOD * 100 / PM_DTLB_MISS
DPTEG_FROM_DL3_SHR(%)	Percentage of DTLB misses that were reloaded from another core's L3 on a distant chip with shared data	PM_DPTEG_FROM_DL3_SHR * 100 / PM_DTLB_MISS
DPTEG_FROM_LMEM(%)	Percentage of DTLB misses that were reloaded from local memory	PM_DPTEG_FROM_LMEM * 100 / PM_DTLB_MISS
DPTEG_FROM_L_OC_CACHE(%)	Percentage of DTLB misses that were reloaded from the local chip's OpenCAPP cache	PM_DPTEG_FROM_L_OC_CACHE * 100 / PM_DTLB_MISS
DPTEG_FROM_L_OC_MEM(%)	Percentage of DTLB misses that were reloaded from the local chip's OpenCAPP memory	PM_DPTEG_FROM_L_OC_MEM * 100 / PM_DTLB_MISS
DPTEG_FROM_RMEM(%)	Percentage of DTLB misses that were reloaded from remote memory	PM_DPTEG_FROM_RMEM * 100 / PM_DTLB_MISS
DPTEG_FROM_R_OC_CACHE(%)	Percentage of DTLB misses that were reloaded from a remote chip's OpenCAPP cache	PM_DPTEG_FROM_R_OC_CACHE * 100 / PM_DTLB_MISS
DPTEG_FROM_R_OC_MEM(%)	Percentage of DTLB misses that were reloaded from a remote chip's OpenCAPP memory	PM_DPTEG_FROM_R_OC_MEM * 100 / PM_DTLB_MISS
DPTEG_FROM_DMEM(%)	Percentage of DTLB misses that were reloaded from distant memory	PM_DPTEG_FROM_DMEM * 100 / PM_DTLB_MISS



Table E-36. Metric Events and Formulas (Sheet 13 of 20)

Metric Name	Metric Description	Formula
DPTEG_FROM_D_OC_CACHE(%)	Percentage of DTLB misses that were reloaded from a distant chip's OpenCAPP cache	PM_DPTEG_FROM_D_OC_CACHE * 100 / PM_DTLB_MISS
DPTEG_FROM_D_OC_MEM(%)	Percentage of DTLB misses that were reloaded from a distant chip's OpenCAPP memory	PM_DPTEG_FROM_D_OC_MEM * 100 / PM_DTLB_MISS
DPTEG_FROM_L2_Rate(%)	Percentage of DERAT misses that reloaded from the L2 per completed run instruction	PM_DPTEG_FROM_L2 * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L2_Miss_Rate(%)	Percentage of DERAT misses that reloaded from beyond the L2 per completed run instruction	PM_DPTEG_FROM_L2MISS * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L3_Rate(%)	Percentage of DERAT misses that reloaded from the L3 per completed run instruction	PM_DPTEG_FROM_L3 * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L3_Miss_Rate(%)	Percentage of DERAT misses that reloaded from beyond the L3 per completed run instruction	PM_DPTEG_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L21_Regent_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on the same regent with modified data per completed run instruction	PM_DPTEG_FROM_L21_REGENT_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L21_Regent_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on the same regent with shared data per completed run instruction	PM_DPTEG_FROM_L21_REGENT_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L21_Non_Regent_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a different regent with modified data per completed run instruction	PM_DPTEG_FROM_L21_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L21_Non_Regent_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a different regent with shared data per completed run instruction	PM_DPTEG_FROM_L21_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L31_Regent_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on the same regent with modified data per completed run instruction	PM_DPTEG_FROM_L31_REGENT_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L31_Regent_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on the same regent with shared data per completed run instruction	PM_DPTEG_FROM_L31_REGENT_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L31_Non_Regent_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a different regent with modified data per completed run instruction	PM_DPTEG_FROM_L31_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L31_Non_Regent_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a different regent with shared data per completed run instruction	PM_DPTEG_FROM_L31_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_RL2_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a remote chip with modified data per completed run instruction	PM_DPTEG_FROM_RL2_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_RL2_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a remote chip with shared data per completed run instruction	PM_DPTEG_FROM_RL2_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_RL3_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a remote chip with modified data per completed run instruction	PM_DPTEG_FROM_RL3_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_RL3_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a remote chip with shared data per completed run instruction	PM_DPTEG_FROM_RL3_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_DL2_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a distant chip with modified data per completed run instruction	PM_DPTEG_FROM_DL2_MOD * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 14 of 20)

Metric Name	Metric Description	Formula
DPTEG_FROM_DL2_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L2 on a distant chip with shared data per completed run instruction	PM_DPTEG_FROM_DL2_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_DL3_MOD_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a distant chip with modified data per completed run instruction	PM_DPTEG_FROM_DL3_MOD * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_DL3_SHR_Rate(%)	Percentage of DERAT misses that reloaded from another core's L3 on a distant chip with shared data per completed run instruction	PM_DPTEG_FROM_DL3_SHR * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_LMEM_Rate(%)	Percentage of DERAT misses that reloaded from local memory per completed run instruction	PM_DPTEG_FROM_LMEM * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L_OC_CACHE_Rate(%)	Percentage of DERAT misses that reloaded from the local chip's OpenCAPP cache per completed run instruction	PM_DPTEG_FROM_L_OC_CACHE * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_L_OC_MEM_Rate(%)	Percentage of DERAT misses that reloaded from the local chip's OpenCAPP memory per completed run instruction	PM_DPTEG_FROM_L_OC_MEM * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_RMEM_Rate(%)	Percentage of DERAT misses that reloaded from remote memory per completed run instruction	PM_DPTEG_FROM_RMEM * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_R_OC_CACHE_Rate(%)	Percentage of DERAT misses that reloaded from a remote chip's OpenCAPP cache per completed run instruction	PM_DPTEG_FROM_R_OC_CACHE * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_R_OC_MEM_Rate(%)	Percentage of DERAT misses that reloaded from a remote chip's OpenCAPP memory per completed run instruction	PM_DPTEG_FROM_R_OC_MEM * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_DMEM_Rate(%)	Percentage of DERAT misses that reloaded from distant memory per completed run instruction	PM_DPTEG_FROM_DMEM * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_D_OC_CACHE_Rate(%)	Percentage of DERAT misses that reloaded from a distant chip's OpenCAPP cache per completed run instruction	PM_DPTEG_FROM_D_OC_CACHE * 100 / PM_RUN_INST_CMPL
DPTEG_FROM_D_OC_MEM_Rate(%)	Percentage of DERAT misses that reloaded from a distant chip's OpenCAPP memory per completed run instruction	PM_DPTEG_FROM_D_OC_MEM * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L2_Rate(%)	Percentage of IERAT misses that reloaded from the L2 per completed run instruction	PM_IPTEG_FROM_L2 * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L2_Miss_Rate(%)	Percentage of IERAT misses that reloaded from beyond the L2 per completed run instruction	PM_IPTEG_FROM_L2MISS * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L3_Rate(%)	Percentage of IERAT misses that reloaded from the L3 per completed run instruction	PM_IPTEG_FROM_L3 * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L3_Miss_Rate(%)	Percentage of IERAT misses that reloaded from beyond the L3 per completed run instruction	PM_IPTEG_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L21_Regent_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on the same regent with modified data per completed run instruction	PM_IPTEG_FROM_L21_REGENT_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L21_Regent_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on the same regent with shared data per completed run instruction	PM_IPTEG_FROM_L21_REGENT_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L21_Non_Regent_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a different regent with modified data per completed run instruction	PM_IPTEG_FROM_L21_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 15 of 20)

Metric Name	Metric Description	Formula
IPTEG_FROM_L21_Non_Regent_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a different regent with shared data per completed run instruction	PM_IPTEG_FROM_L21_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L31_Regent_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on the same regent with modified data per completed run instruction	PM_IPTEG_FROM_L31_REGENT_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L31_Regent_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on the same regent with shared data per completed run instruction	PM_IPTEG_FROM_L31_REGENT_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L31_Non_Regent_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a different regent with modified data per completed run instruction	PM_IPTEG_FROM_L31_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L31_Non_Regent_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a different regent with shared data per completed run instruction	PM_IPTEG_FROM_L31_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_RL2_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a remote chip with modified data per completed run instruction	PM_IPTEG_FROM_RL2_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_RL2_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a remote chip with shared data per completed run instruction	PM_IPTEG_FROM_RL2_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_RL3_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a remote chip with modified data per completed run instruction	PM_IPTEG_FROM_RL3_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_RL3_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a remote chip with shared data per completed run instruction	PM_IPTEG_FROM_RL3_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_DL2_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a distant chip with modified data per completed run instruction	PM_IPTEG_FROM_DL2_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_DL2_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L2 on a distant chip with shared data per completed run instruction	PM_IPTEG_FROM_DL2_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_DL3_MOD_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a distant chip with modified data per completed run instruction	PM_IPTEG_FROM_DL3_MOD * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_DL3_SHR_Rate(%)	Percentage of IERAT misses that reloaded from another core's L3 on a distant chip with shared data per completed run instruction	PM_IPTEG_FROM_DL3_SHR * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_LMEM_Rate(%)	Percentage of IERAT misses that reloaded from local memory per completed run instruction	PM_IPTEG_FROM_LMEM * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L_OC_CACHE_Rate(%)	Percentage of IERAT misses that reloaded from the local chip's OpenCAPP cache per completed run instruction	PM_IPTEG_FROM_L_OC_CACHE * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_L_OC_MEM_Rate(%)	Percentage of IERAT misses that reloaded from the local chip's OpenCAPP memory per completed run instruction	PM_IPTEG_FROM_L_OC_MEM * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_RMEM_Rate(%)	Percentage of IERAT misses that reloaded from remote memory per completed run instruction	PM_IPTEG_FROM_RMEM * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 16 of 20)

Metric Name	Metric Description	Formula
IPTEG_FROM_R_OC_CACHE_Rate(%)	Percentage of IERAT misses that reloaded from a remote chip's OpenCAPP cache per completed run instruction	PM_IPTEG_FROM_R_OC_CACHE * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_R_OC_MEM_Rate(%)	Percentage of IERAT misses that reloaded from a remote chip's OpenCAPP memory per completed run instruction	PM_IPTEG_FROM_R_OC_MEM * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_DMEM_Rate(%)	Percentage of IERAT misses that reloaded from distant memory per completed run instruction	PM_IPTEG_FROM_DMEM * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_D_OC_CACHE_Rate(%)	Percentage of IERAT misses that reloaded from a distant chip's OpenCAPP cache per completed run instruction	PM_IPTEG_FROM_D_OC_CACHE * 100 / PM_RUN_INST_CMPL
IPTEG_FROM_D_OC_MEM_Rate(%)	Percentage of IERAT misses that reloaded from a distant chip's OpenCAPP memory per completed run instruction	PM_IPTEG_FROM_D_OC_MEM * 100 / PM_RUN_INST_CMPL
INST_FROM_L2(%)	Percentage of I-cache misses that were reloaded from the L2 cache	PM_INST_FROM_L2 * 100 / PM_L1_ICACHE_MISS
INST_FROM_L2_Miss(%)	Percentage of I-cache misses that were reloaded from beyond the local L2 cache	PM_INST_FROM_L2MISS * 100 / PM_L1_ICACHE_MISS
INST_FROM_L3(%)	Percentage of I-cache misses that were reloaded from the L3 cache	PM_INST_FROM_L3 * 100 / PM_L1_ICACHE_MISS
INST_FROM_L3_Miss(%)	Percentage of I-cache misses that were reloaded from beyond the local L3 cache	PM_INST_FROM_L3MISS * 100 / PM_L1_ICACHE_MISS
INST_FROM_L21_Regent_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L2 on the same regent	PM_INST_FROM_L21_REGENT_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_L21_Regent_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L2 on the same regent	PM_INST_FROM_L21_REGENT_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_L21_Non_Regent_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L2 on the same chip in a different regent	PM_INST_FROM_L21_NON_REGENT_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_L21_Non_Regent_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L2 on the same chip in a different regent	PM_INST_FROM_L21_NON_REGENT_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_L31_Regent_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L3 on the same regent	PM_INST_FROM_L31_REGENT_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_L31_Regent_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L3 on the same regent	PM_INST_FROM_L31_REGENT_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_L31_Non_Regent_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L3 on the same chip in a different regent	PM_INST_FROM_L31_NON_REGENT_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_L31_Non_Regent_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L3 on the same chip in a different regent	PM_INST_FROM_L31_NON_REGENT_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_RL2_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L2 on a remote chip	PM_INST_FROM_RL2_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_RL2_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L2 on a remote chip	PM_INST_FROM_RL2_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_RL3_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L3 on a remote chip	PM_INST_FROM_RL3_MOD * 100 / PM_L1_ICACHE_MISS



Table E-36. Metric Events and Formulas (Sheet 17 of 20)

Metric Name	Metric Description	Formula
INST_FROM_RL3_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L3 on a remote chip	PM_INST_FROM_RL3_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_DL2_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L2 on a distant chip	PM_INST_FROM_DL2_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_DL2_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L2 on a distant chip	PM_INST_FROM_DL2_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_DL3_MOD(%)	Percentage of I-cache misses that were reloaded with modified data from another core's L3 on a distant chip	PM_INST_FROM_DL3_MOD * 100 / PM_L1_ICACHE_MISS
INST_FROM_DL3_SHR(%)	Percentage of I-cache misses that were reloaded with shared data from another core's L3 on a distant chip	PM_INST_FROM_DL3_SHR * 100 / PM_L1_ICACHE_MISS
INST_FROM_LMEM(%)	Percentage of I-cache misses that were reloaded from local memory	PM_INST_FROM_LMEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_L_OC_CACHE(%)	Percentage of I-cache misses that were reloaded from the local chip's OpenCAPP Cache	PM_INST_FROM_L_OC_CACHE * 100 / PM_L1_ICACHE_MISS
INST_FROM_L_OC_MEM(%)	Percentage of I-cache misses that were reloaded from the local chip's OpenCAPP memory	PM_INST_FROM_L_OC_MEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_RMEM(%)	Percentage of I-cache misses that were reloaded from remote memory	PM_INST_FROM_RMEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_R_OC_CACHE(%)	Percentage of I-cache misses that were reloaded from a remote chip's OpenCAPP Cache	PM_INST_FROM_R_OC_CACHE * 100 / PM_L1_ICACHE_MISS
INST_FROM_R_OC_MEM(%)	Percentage of I-cache misses that were reloaded from a remote chip's OpenCAPP memory	PM_INST_FROM_R_OC_MEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_DMEM(%)	Percentage of I-cache misses that were reloaded from distant memory	PM_INST_FROM_DMEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_D_OC_CACHE(%)	Percentage of I-cache misses that were reloaded from a distant chip's OpenCAPP Cache	PM_INST_FROM_D_OC_CACHE * 100 / PM_L1_ICACHE_MISS
INST_FROM_D_OC_MEM(%)	Percentage of I-cache misses that were reloaded from a distant chip's OpenCAPP memory	PM_INST_FROM_D_OC_MEM * 100 / PM_L1_ICACHE_MISS
INST_FROM_L2_Rate(%)	Percentage of I-cache reloads from the L2 per completed run instruction	PM_INST_FROM_L2 * 100 / PM_RUN_INST_CMPL
INST_FROM_L2_Miss_Rate(%)	Percentage of I-cache reloads from beyond the L2 per completed run instruction	PM_INST_FROM_L2MISS * 100 / PM_RUN_INST_CMPL
INST_FROM_L3_Rate(%)	Percentage of I-cache reloads from the L3 per completed run instruction	PM_INST_FROM_L3 * 100 / PM_RUN_INST_CMPL
INST_FROM_L3_Miss_Rate(%)	Percentage of I-cache reloads from beyond the L3 per completed run instruction	PM_INST_FROM_L3MISS * 100 / PM_RUN_INST_CMPL
INST_FROM_L21_Regent_MOD_Rate(%)	Percentage of I-cache reloads from another core's L2 on the same regent with modified data per completed run instruction	PM_INST_FROM_L21_REGENT_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_L21_Regent_SHR_Rate(%)	Percentage of I-cache reloads from another core's L2 on the same regent with shared data per completed run instruction	PM_INST_FROM_L21_REGENT_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_L21_Non_Regent_MOD_Rate(%)	Percentage of I-cache reloads from another core's L2 on the same chip in a different regent with modified data per completed run instruction	PM_INST_FROM_L21_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_L21_Non_Regent_SHR_Rate(%)	Percentage of I-cache reloads from another core's L2 on the same chip in a different regent with shared data per completed run instruction	PM_INST_FROM_L21_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 18 of 20)

Metric Name	Metric Description	Formula
INST_FROM_L31_Regent_MOD_Rate(%)	Percentage of I-cache reloads from another core's L3 on the same regent with modified data per completed run instruction	PM_INST_FROM_L31_REGENT_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_L31_Regent_SHR_Rate(%)	Percentage of I-cache reloads from another core's L3 on the same regent with shared data per completed run instruction	PM_INST_FROM_L31_REGENT_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_L31_Non_Regent_MOD_Rate(%)	Percentage of I-cache reloads from another core's L3 on the same chip in a different regent with modified data per completed run instruction	PM_INST_FROM_L31_NON_REGENT_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_L31_Non_Regent_SHR_Rate(%)	Percentage of I-cache reloads from another core's L3 on the same chip in a different regent with shared data per completed run instruction	PM_INST_FROM_L31_NON_REGENT_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_RL2_MOD_Rate(%)	Percentage of I-cache reloads from another core's L2 on a remote chip with modified data per completed run instruction	PM_INST_FROM_RL2_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_RL2_SHR_Rate(%)	Percentage of I-cache reloads from another core's L2 on a remote chip with shared data per completed run instruction	PM_INST_FROM_RL2_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_RL3_MOD_Rate(%)	Percentage of I-cache reloads from another core's L3 on a remote chip with modified data per completed run instruction	PM_INST_FROM_RL3_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_RL3_SHR_Rate(%)	Percentage of I-cache reloads from another core's L3 on a remote chip with shared data per completed run instruction	PM_INST_FROM_RL3_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_DL2_MOD_Rate(%)	Percentage of I-cache reloads from another core's L2 on a distant chip with modified data per completed run instruction	PM_INST_FROM_DL2_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_DL2_SHR_Rate(%)	Percentage of I-cache reloads from another core's L2 on a distant chip with shared data per completed run instruction	PM_INST_FROM_DL2_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_DL3_MOD_Rate(%)	Percentage of I-cache reloads from another core's L3 on a distant chip with modified data per completed run instruction	PM_INST_FROM_DL3_MOD * 100 / PM_RUN_INST_CMPL
INST_FROM_DL3_SHR_Rate(%)	Percentage of I-cache reloads from another core's L3 on a distant chip with shared data per completed run instruction	PM_INST_FROM_DL3_SHR * 100 / PM_RUN_INST_CMPL
INST_FROM_LMEM_Rate(%)	Percentage of I-cache reloads from local memory per completed run instruction	PM_INST_FROM_LMEM * 100 / PM_RUN_INST_CMPL
INST_FROM_L_OC_CACHE_Rate(%)	Percentage of I-cache reloads from the local chip's OpenCAPP Cache per completed run instruction	PM_INST_FROM_L_OC_CACHE * 100 / PM_RUN_INST_CMPL
INST_FROM_L_OC_MEM_Rate(%)	Percentage of I-cache reloads from the local chip's OpenCAPP memory per completed run instruction	PM_INST_FROM_L_OC_MEM * 100 / PM_RUN_INST_CMPL
INST_FROM_RMEM_Rate(%)	Percentage of I-cache reloads from remote memory per completed run instruction	PM_INST_FROM_RMEM * 100 / PM_RUN_INST_CMPL
INST_FROM_R_OC_CACHE_Rate(%)	Percentage of I-cache reloads from a remote chip's OpenCAPP Cache per completed run instruction	PM_INST_FROM_R_OC_CACHE * 100 / PM_RUN_INST_CMPL
INST_FROM_R_OC_MEM_Rate(%)	Percentage of I-cache reloads from a remote chip's OpenCAPP memory per completed run instruction	PM_INST_FROM_R_OC_MEM * 100 / PM_RUN_INST_CMPL
INST_FROM_DMEM_Rate(%)	Percentage of I-cache reloads from distant memory per completed run instruction	PM_INST_FROM_DMEM * 100 / PM_RUN_INST_CMPL



Table E-36. Metric Events and Formulas (Sheet 19 of 20)

Metric Name	Metric Description	Formula
INST_FROM_D_OC_CACHE_Rate(%)	Percentage of I-cache reloads from a distant chip's OpenCAPP Cache per completed run instruction	PM_INST_FROM_D_OC_CACHE * 100 / PM_RUN_INST_CMPL
INST_FROM_D_OC_MEM_Rate(%)	Percentage of I-cache reloads from a distant chip's OpenCAPP memory per completed run instruction	PM_INST_FROM_D_OC_MEM * 100 / PM_RUN_INST_CMPL
L2_LD_Miss_Ratio(%)	Percentage of L2 load dispatches that were L2 load misses	PM_L2_LD_MISS / PM_L2_LD * 100
L2_Id_hit_frequency	Average number of cycles between L2 load hits	PM_RUN_CYC / (PM_L2_LD_HIT * 2)
L2_Id_miss_frequency	Average number of cycles between L2 load misses	PM_RUN_CYC / (PM_L2_LD_MISS * 2)
L2_ST_Miss_Ratio(%)	Percentage of L2 store dispatches that were L2 store misses	PM_L2_ST_MISS / PM_L2_ST * 100
L2_INST_Miss_Ratio(%)	Percentage of L2 instruction dispatches that were L2 instruction misses	PM_L2_INST_MISS / PM_L2_INST * 100
L2_IC_Inv_Rate(%)	Percentage of L2 I-cache invalidates per completed run instruction	(PM_L2_IC_INV * 2) / PM_RUN_INST_CMPL * 100
L2_DC_Inv_Rate(%)	Percentage of L2 D-cache invalidates per completed run instruction	(PM_L2_DC_INV * 2) / PM_RUN_INST_CMPL * 100
L2_Dem_LD_Disp(%)	Percentage of L2 load dispatches that were demand load misses	PM_LD_DEMAND_MISS_L1 / (PM_L2_LD * 2) * 100
L2_Mod_CO(%)	Percentage of L2 castouts that were in modified state	PM_L2_CASTOUT_MOD / (PM_L2_CASTOUT_MOD + PM_L2_CASTOUT_SHR) * 100
L2_Sh_Co(%)	Percentage of L2 castouts that were in shared state	PM_L2_CASTOUT_SHR / (PM_L2_CASTOUT_MOD + PM_L2_CASTOUT_SHR) * 100
L2_Local_Pred_Correct(%)	Percentage of L2 local pump predictions that were correct	PM_L2_LOC_GUESS_CORRECT / (PM_L2_LOC_GUESS_CORRECT + PM_L2_LOC_GUESS_WRONG) * 100
L2_CO_M_Rd_Util	Percentage of run cycles when the L2 castout a modified line	(PM_L2_CASTOUT_MOD * 2) / PM_RUN_CYC * 100
L2_LD_Rd_Util	Percentage of run cycles when the L2 made a load dispatch attempt	(PM_L2_ISIDE_DSIDE_ATTEMPT * 2) / PM_RUN_CYC * 100
L2_ST_Rd_Util	Percentage of run cycles when the L2 made a store dispatch attempt	(PM_L2_ST_ATTEMPT_DISP * 2) / PM_RUN_CYC * 100
L2_Rd_Util(%)	Percentage of cycles when L2 Cache read is utilized	L2_LD_Rd_Util + L2_ST_Rd_Util + L2_CO_M_Rd_Util
L2_LDMISS_Wr_Util	Percentage of cycles when an L2 load miss requires a cache write	((PM_L2_LD_DISP - PM_L2_LD_HIT) * 2) / PM_RUN_CYC * 100
L2_ST_Wr_Util	Percentage of cycles when an L2 store requires a cache write	(PM_L2_ST_DISP * 2) / PM_RUN_CYC * 100
L2_Wr_Util(%)	Percentage of cycles when L2 Cache write is utilized	L2_LDMISS_Wr_Util + L2_ST_Wr_Util
L2_ST_Dispatch_Fail_Rate(%)	Percentage of L2 store dispatch attempts that fail per completed run instruction	((PM_L2_ST_DISP_FAIL_ADDR + PM_L2_ST_DISP_FAIL_OTHER) * 2) / PM_RUN_INST_CMPL * 100
L2_ST_Dispatch_Rate(%)	Percentage of L2 store dispatch attempts per completed run instructions	(PM_L2_ST_ATTEMPT * 2) / PM_RUN_INST_CMPL * 100



Table E-36. Metric Events and Formulas (Sheet 20 of 20)

Metric Name	Metric Description	Formula
L2_ST_Dispatch_Fail(%)	Percentage of L2 store dispatch attempts that fail	(PM_L2_ST_DISP_FAIL_ADDR + PM_L2_ST_DISP_FAIL_OTHER) / PM_L2_ST_ATTEMPT * 100
L2_ST_Dispatch_Addr_Fail(%)	Percentage of L2 store dispatch attempts that failed due to an address collision conflict with an L2 machine already working on this line	PM_L2_ST_DISP_FAIL_ADDR / PM_L2_ST_ATTEMPT * 100
L2_LD_Dispatch_Fail_Rate(%)	Percentage of L2 load dispatch attempts that fail per completed run instruction	((PM_L2_ISIDE_DSIDE_FAIL_ADDR + PM_L2_ISIDE_DSIDE_FAIL_OTHER) * 2) / PM_RUN_INST_CMPL * 100
L2_LD_Dispatch_Rate(%)	Percentage of L2 load dispatch attempts per completed run instructions	(PM_L2_ISIDE_DSIDE_ATTEMPT * 2) / PM_RUN_INST_CMPL * 100
L2_LD_Dispatch_Fail(%)	Percentage of L2 load dispatch attempts that fail	(PM_L2_ISIDE_DSIDE_FAIL_ADDR + PM_L2_ISIDE_DSIDE_FAIL_OTHER) / PM_L2_ISIDE_DSIDE_ATTEMPT * 100
L2_LD_Dispatch_Addr_Fail(%)	Percentage of L2 load dispatch attempts that failed due to an address collision conflict with an L2 machine already working on this line	PM_L2_ISIDE_DSIDE_FAIL_ADDR / PM_L2_ISIDE_DSIDE_ATTEMPT * 100
L2_RC_Usage	Average number of Read/Claim machines used per cycle	(PM_L2_RC_USAGE * 2) / PM_RUN_CYC * 16
L2_CO_Usage	Average number of Castout machines used per cycle	(PM_L2_CO_USAGE * 2) / PM_RUN_CYC * 16
L2_SN_Usage	Average number of Snoop machines used per cycle	(PM_L2_SN_USAGE) / PM_RUN_CYC * 8
L2_ST_Commands(%)	Percentage of dispatched L2 commands that were stores	PM_L2_ST * 100 / (PM_L2_ST + PM_L2_ISIDE_DSIDE)
L2_LD_Commands(%)	Percentage of dispatched L2 commands that were loads	PM_L2_LD * 100 / (PM_L2_ST + PM_L2_ISIDE_DSIDE)
L2_Instr_Commands(%)	Percentage of dispatched L2 commands that were instruction reads	PM_L2_INST * 100 / (PM_L2_ST + PM_L2_ISIDE_DSIDE)
L3_WI_Usage	Average number of Write Inject machines used per cycle	(PM_L3_WI_USAGE * 2) / PM_RUN_CYC * 8
L3_Id_hit_frequency	Average number of cycles between L3 load hits	PM_RUN_CYC / (PM_L3_LD_HIT * 2)
L3_Id_miss_frequency	Average number of cycles between L3 load misses	PM_RUN_CYC / (PM_L3_LD_MISS * 2)
L1_Prefetch_Rate(%)	Percentage of L1 prefetches per completed instruction	PM_L1_PREF / PM_RUN_INST_CMPL * 100
Prefetch_BytE	Average data bytes prefetched from memory per cycle	PM_MEM_PREF * 64 / PM_CYC
Demand_Read_BytE	Average demand data bytes read from memory per cycle	(PM_MEM_READ + PM_MEM_RWITM) * 64 / PM_CYC
Prefetch_BW(GB/s)	Total prefetch bandwidth seen by the chiplet for this partition, includes data/inst/xlate	PM_MEM_PREF * 64 * (proc_freq * 1E-9) / PM_CYC
Demand_Read_BW(GB/s)	Total demand read bandwidth seen by the chiplet for this partition, includes data/inst/xlate	(PM_MEM_READ + PM_MEM_RWITM) * 64 * (proc_freq * 1E-9) / PM_CYC

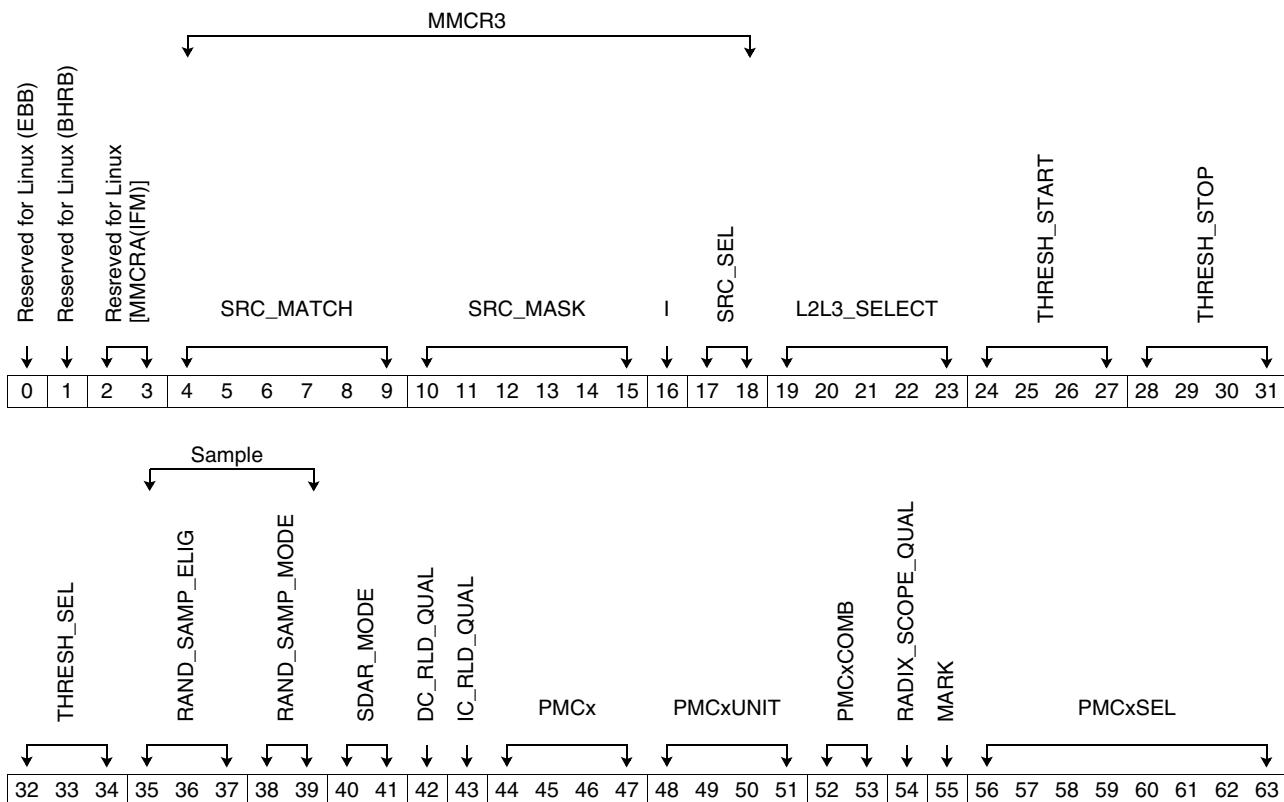
### E.5.15 Power10 Performance Monitor Event Selection

For each of the four programmable partition-level counters (PMCs), one event can be selected for monitoring at a given time. The monitored event is selected using the select event formation code shown in [Section E.5.15.1](#). The code sets the appropriate MMCR values, MMCR1[PMC1xUNIT] and MMCR1[PMCxSEL], to select the monitored event, and it sets additional MMCR values required for the monitored event (that is, MMCRA[MARK] for marked events and MMCR3 for events with specified sources).

#### E.5.15.1 Select Event Code Formation

The select event formation code shown in *Figure E-5* shows how the Power10 event codes are translated into MMCR settings.

*Figure E-5. Power10 Raw Event Coding*



Bit	Name	Description
0	EBB	Reserved for Linux.
1	BHRB	Reserved for Linux.
2:3	MMCRA(IFM)	Bits used to program MMCRA[IFM] field.



Bit	Name	Description
4:9	SRC_MATCH	
10:15	SRC_MASK	
16	I	Used to program MMCR3 SPR.
17:18	SRC_SEL	
19:23	L2L3_SELECT	Value to program in MMCR2 [56:60]. This is not an index.
24:27	THRESH_START	Value to program in MMCRA [ 48:51]. This is not an index.
28:31	THRESH_STOP	Value to program in MMCRA [ 52:55 ]. This is not an index.
32:34	THRESH_SEL	Value to program in MMCRA. This is not an index.
35:37	RAND_SAMP_ELIG	Value to program in MMCRA. This is not an index.
38:39	RAND_SAMP_MODE	Value to program in MMCRA. This is not an index.
40:41	SDAR_MODE	MMCRA [20:21].
42	DC_RLD_QUAL	MMCR1[16]
43	IC_RLD_QUAL	MMCR1[17]
44:47	PMCx	0      The event can be counted on any PMC 1      The event must be counted on PMC1 2      The event must be counted on PMC2 3      The event must be counted on PMC3 4      The event must be counted on PMC4 5      The event must be counted on PMC5 6      The event must be counted on PMC6
48:51	PMCxUNIT	MMCR1[0:3] for PMC1 MMCR1[4:7] for PMC2 MMCR1[8:11] for PMC3 MMCR1[12:15] for PMC4
52:53	PMCxCOMB	MMCR1[24:25] for PMC1 MMCR1[26:27] for PMC2 MMCR1[28:29] for PMC3 MMCR1[30:31] for PMC4
54	RADIX_SCOPE_QUAL	MMCR1[18]
55	MARK	MMCRA [63]
56:63	PMCxSEL	MMCR1[32:39] for PMC1 MMCR1[40:47] for PMC2 MMCR1[48:55] for PMC3 MMCR1[56:63] for PMC4



### E.5.16 Power10 Events Grouping

The Power10 PMU events are described in previous sections. This section describes some restrictions for grouping events so that the four PMCs can collect four events at a time. Grouping restrictions are as follows:

- Only one event per PMC
- PMC5 and PMC6 are not programmable
- L2 events can only be grouped with other L2 events. Likewise; L3 events can be grouped with other L3 events. Additionally, L2 and L3 events can only be grouped with other L2 and L3 events that have the same first 2 bits set in their select event codes.
- Reload source events that set MMCR1[DC\_RLD\_QUAL] or MMCR1[IC\_RLD\_QUAL] cannot be grouped with any events that do not have these bits set in their Select Event Code.

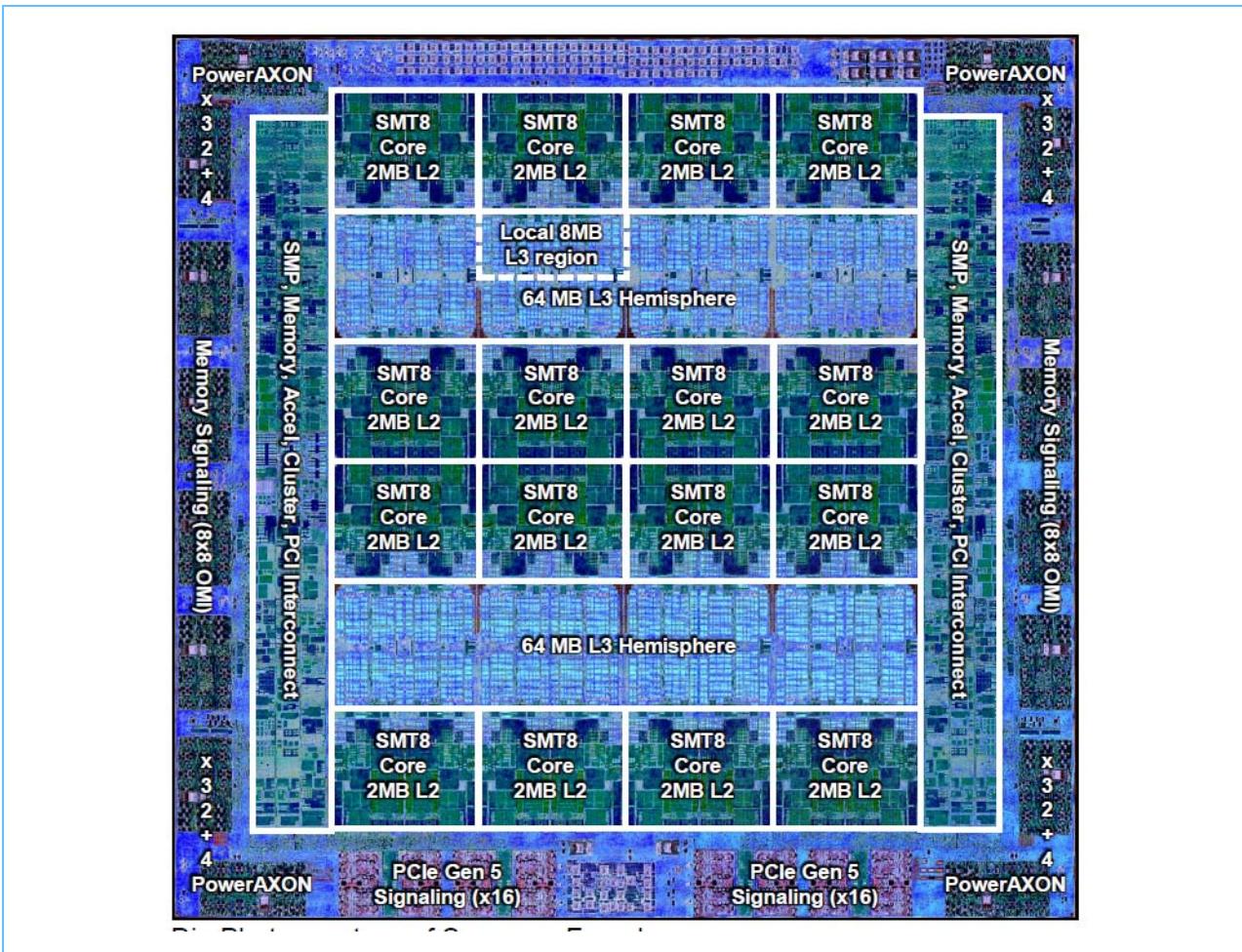
## E.6 Nest PMU Instrumentation and Performance Metrics

### E.6.1 Nest Performance Monitoring Unit Instrumentation

#### E.6.1.1 Power10 Chip Overview

Figure E-6 shows an overview of the Power10 chip and the corresponding Nest units.

Figure E-6. Power10 Block Diagram and Corresponding Nest Units (Shown with SMT8 Core Variant)





#### **E.6.1.2 Power10 Nest PMU Instrumentation**

The Power10 Nest PMU instrumentation spans chip-level units and OpenCAPI memory buffer chip. Performance events from the following Nest units are available for monitoring:

- SMP interconnect (PB)
- Memory controller (MC) – SMP interconnect interface
- OpenCAPI memory buffer (OCMB) – DIMM interface
- X links (links connecting across sockets)
- A links (links connecting across nodes)
- Processor Attached Functional Units (PAU) OpenCAPI Links
- PCIe controller (PEC) – SMP interconnect interface
- PCIe host bridge (PHB)
- On-chip accelerator (NX)

#### **E.6.1.3 Nest Instrumentation Counters (PMULEts)**

Nest PMU events are counted on dedicated PMU counters, which are different than thread-level PMU counters (PMC1 - PMC6). The Nest PMU counter (PMULEt) is a 64-bit register, which is partitioned internally as four 16-bit counters. Each 16-bit counter is an independent counter with individual control. Each PMULEt can be configured to freeze or free run on an overflow.

Unlike the thread-level (PMC1 - PMC6) counters, the PMULEts are not user accessible. The PMULEts are pre-programmed to count a specific set of events.

See *Appendix E.6.2 Nest Units and Performance Metrics Support* on page 661 for more information.

#### **E.6.1.4 Nest PMU Instrumentation Categories**

The Nest PMU instrumentation units are grouped into two categories based on whether they use shared counters or dedicated counters:

- Centralized nest performance monitor (CNPM) – units whose PMU events are synchronized to the PAU clock and share the centralized counters. Centralized counters have access to a 32-bit event bus that these units share along with counters (PMULEts) to access them.
- Distributed nest performance monitor (DNPM) – units whose PMU events might not be synchronous to the PAU clock and have independent counters. Distributed units have dedicated counters (PMULEts) with dedicated event buses.

*Table E-37* shows the Nest units in the CNPM and DNPM groupings. The clock domain column shows each of the Nest unit's PMU instrumentation design clock domain.

*Table E-37. Nest Unit Instrumentation and Grouping*

Instrumentation Type	Unit	Clock Domain	Number of Counters	Counter Width
Centralized Nest Performance Monitor	SMP interconnect (PB)	PAU clock at 2.014 - 2.588 GHz	64	16-bit with pre-scaler sizes (20, 16, 8, 4)
	Memory controller (MC)	PAU clock		
	PCIe Controller (PEC)	PAU clock		
Distributed Nest Performance Monitor	A/X/OpenCAPI Links	PAU clock	64	16-bit with pre-scaler sizes (20, 16, 8, 4)
	NX Accelerator	Nest clock 0.9 – 2.1 GHz	8	
	PCIe PHB	2 GHz	4 counters per PHB port	
	OCMB (DIMM interface)	DIMM frequency	4 counters per DDR port	

**Note:** On an FSP, the PAU frequency can be derived from the attribute, ATTR\_FREQ\_PAU\_MHZ. The DIMM frequency can be derived from the attribute, ATTR\_FREQ\_OMIC\_MHZ. The Nest frequency can be derived from the attribute, ATTR\_FREQ\_SYSTEM\_CORE\_FLOOR\_MHZ. The Nest frequency is one half of the core frequency. The PAU frequency is fixed for the system; however, core frequency on each socket is variable.

## E.6.2 Nest Units and Performance Metrics Support

The following sections provide a brief introduction to each of the Nest units and the events, performance metrics, and the formula for deriving performance metrics from those events.

A set of pre-defined important Nest PMU hardware events are configured to be counted and posted to memory for access as an in-memory collection (IMC). The interface to access these events is through the standard performance interface.

### E.6.2.1 Processor Fabric (SMP Interconnect)

The Power10 Fabric controller (FBC) is the underlying hardware used to create a scalable cache-coherent multiprocessor system. The Power10 Fabric controller provides coherent and non-coherent memory access, I/O operations, interrupt communication, and system controller communication. The FBC provides all of the interfaces, buffering, and sequencing of command and data operations within the storage subsystem. The FBC is integrated on the Power10 chip, with up to 32 processor cores and on the chip memory subsystem. The Power10 chip has up to eight SMP links that can be used to connect to other Power10 chips.

### E.6.2.2 Internal Fabric PMU Events and Performance Metrics

*Table E-38* lists the events and performance metrics derived from the internal Fabric events. Other events that are part of the Fabric profile exist but are not listed, because they cannot be used to derive any performance metrics by themselves.



Table E-38. Event and Performance Metrics for Fabric Events

Event Name	Event Description	Performance Metric Name and Formula
PM_PAU_CYC	Clock cycle count for the reference clock for Fabric, MCS, PEC, and Link events described in the following subsections.	
PM_PB_EVENT_VG_PUMP	Vector group scope operation (locally mastered) on port $n$ .	total_vector_group_pumps (per sec) = $((PM\_PB\_EVENT\_VG\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_LNS_PUMP	Local nodal scope operation (locally mastered) on port $n$ .	total_local_node_pumps (per sec) = $((PM\_PB\_EVENT\_LNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_GROUP_PUMP	Group scope operation (locally mastered) on port $n$ .	total_vector_group_pumps_retries (per sec) = $((PM\_PB\_EVENT\_GROUP\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_RNS_PUMP	Remote nodal scope operation (locally mastered) on port $n$ .	total_local_node_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_NNS_PUMP	Near nodal scope operation (locally mastered) on port $n$ .	total_near_node_pumps_retries (per sec) = $((PM\_PB\_EVENT\_NNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_LNS_PUMP	Remote nodal scope operation (locally mastered) on port $n$ .	total_local_node_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_RTY_VG_PUMP	Retry of a vector group scope operation (locally mastered). Retry due to (rtv_dropped_rcmd, rtv_lpc, rtv_other).	total_vector_group_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RTY\_VG\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_RTY_GROUP_PUMP	Retry of a group scope operation (locally mastered). Retry due to (rtv_dropped_rcmd, rtv_lpc, rtv_other).	total_group_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RTY\_GROUP\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_RTY_RNS_PUMP	Retry of a remote nodal scope operation (locally mastered). Retry due to (rtv_dropped_rcmd, rtv_lpc, rtv_other).	total_remote_node_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RTY\_RNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_RTY_NNS_PUMP	Retry of a near nodal scope operation (locally mastered). Retry due to (rtv_dropped_rcmd, rtv_lpc, rtv_other).	total_near_node_pumps_retries (per sec) = $((PM\_PB\_EVENT\_RTY\_NNS\_PUMP) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EVENT_INTERNAL_DATA_XFER	64 × 32-byte octword (OW) data transfer from local unit to local unit	total_int_pb_bw (GBps) = $((PM\_PB\_INT\_DATA\_XFER * 2048) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_EXT_DATA_XFER	64 × 32-byte OW data transfer sent/received on an internal Fabric X or A bus.	total_ext_pb_bw_sent (GB/s) = $((PM\_PB\_EXT\_DATA\_XFER * 2048) / PM\_PAU\_CYC) * PAU\_Freq$
PM_PB_PUMP_Pn	Total snoop bus operations (locally mastered) on port $n$ .	total_snoop_bus_utilization (%) = $((PM\_PB\_PUMP\_Pn) / PM\_PAU\_CYC) * (PAU\_Freq/Nest\_Freq) * 100$

### E.6.2.3 Memory Controller (MC – Fabric Interface)

The Power10 chip connects to an OpenCAPI 3.1 compliant memory buffer chip that attaches with DDR memory. Each Power10 processor chip has four memory controller units. Each memory controller unit supports two memory channels, and each memory channel supports two memory subchannels. Each subchannel connects to an OpenCAPI memory interface link that attaches with one memory buffer chip. The memory controller acts as an abstraction layer between masters on the internal fabric and the main memory subsystem, existent within an off-chip buffer.

*Table E-39* on page 663 lists the memory controller PMU events and the corresponding performance metrics.

*Table E-39. Memory Controller PMU Events and Performance Metrics*

Event	Description	Formula	Metric
PM_MCS_128B_RD_DATA_BLOCKS	Total 128-byte read data blocks.	$((PM\_MCS\_128B\_RD\_DATA\_BLOCKS * 128) / PM\_PAU\_CYC) * PAU\_Freq$	total_mcs_read_bw (GBps)
PM_MCS_64B_WR_DATA_BLOCKS	Total 64-byte write data blocks.	$((PM\_MCS\_64B\_WR\_DATA\_BLOCKS * 64) / PM\_PAU\_CYC) * PAU\_Freq$	total_mcs_write_bw (GBps)
PM_MCS_MDI_UPDATE_DONE	Total memory domain indicator (MDI) update done for read op	$((PM\_MCS\_MDI\_UPDATE\_DONE) / PM\_PAU\_CYC) * PAU\_Freq$	total mdi_updates (per second)

### E.6.2.4 OpenCAPI Memory Buffer (OCMB)

The Power10 chip connects to OpenCAPI 3.1-compliant memory buffer chip that attaches with DDR memory. The Power10 OCMB PMU instrumentation measures the DRAM activity per subchannel that attaches to the DRAM through OCMB chip.

*Table E-40* lists the OCMB PMU events and corresponding performance metrics.

*Table E-40. OCMB PMU Events and Performance Metrics*

Event	Description	Formula	Metric
PM_OCMB_COMMAND_CLOCKS	OCMB clock cycles event corresponding to memory frequency.		
PM_OCMB_ACT_ALL	Number of activations for either reads or writes.	$(PM\_OCMB\_ACT\_ALL / PM\_OCMB\_COMMAND\_CLOCKS) * Mem\_Freq$	total_ocmb_activation_rate (per second)
PM_OCMB_CAS_RD	Number of column address strobe (CAS) for read.	$(PM\_MBA\_CAS\_READ / PM\_OCMB\_COMMAND\_CLOCKS) * Mem\_Freq$ $(PM\_MBA\_CAS\_READ / PM\_OCMB\_COMMAND\_CLOCKS) * Mem\_Freq * 64$	total_ocmb_read_cas_rate (per second)
PM_OCMB_CAS_WR	Number of CAS for write.	$(PM\_MBA\_CAS\_WRITE / PM\_OCMB\_COMMAND\_CLOCKS) * Mem\_Freq$ $(PM\_MBA\_CAS\_WRITE / PM\_OCMB\_COMMAND\_CLOCKS) * Mem\_Freq * 64$	total_ocmb_write_cas_rate (per second) total_ocmb_write_bw (GBps)



### E.6.2.5 A/X Links – SMP Links

The off-chip Power10 Fabric supports up to eight concurrent logical SMP links (X0:X7 and/or A0:A3). In the 1-hop topology, the inter-group X-bus links connect up to eight processor chips. The X links carry coherency traffic as well as data; and are interchangeable as inter-group processor links. The X links can also be configured as aggregate data-only links. In the 2-hop topology, the inter-group A-bus links connect up to eight groups. The intra-group A-bus links connect up to four processor chips. The PMU instrumentation design is in the PAU clock domain. Therefore, all A and X link performance metric computations use the PAU frequency.

*Table E-41* on page 664 lists the X link PMU events and performance metrics.

*Table E-41. X Link PMU Events and Performance Metrics*

Event	Description	Formula	Metric
PM_XLINK_AVLBL_CYCLES	Available cycles for X link traffic on the link.		
PM_XLINK_ANY_RCMD	Number of any reflected command (rcmd) events received or sent through the X link.	(PM_XLINK_ANY_RCMD / PM_PAU_CYC)*PAU_freq	xlinks_rcmd_rate (per second)
PM_XLINK_DATA_COUNT	Number of 16-byte data blocks sent or received.	(PM_XLINK_DATA_EVENT * 16 / PM_PAU_CYC) *PAU_freq	xlinks_data_bw (GBps)
PM_XLINK_TOTAL_UTIL	Total utilization of the link due to all commands, header, data, rcmd, and presp.	(PM_XLINK_TOTAL_UTIL / PM_XLINK_AVLBL_CYCLES) * 100	xlink_total_utilization (%)

*Table E-42* lists the A link PMU events and performance metrics.

*Table E-42. A Link PMU Events and Performance Metrics*

Event	Description	Formula	Metric
PM_ALINK_AVLBL_CYCLES	Available cycles for A link traffic on the link.		
PM_ALINK_ANY_RCMD	Number of any reflected command (rcmd) events received or sent through the A link.	(PM_ALINK_ANY_RCMD / PM_PAU_CYC)*PAU_freq	alinks_rcmd_rate (per second)
PM_ALINK_DATA_COUNT	Number of 16-byte data blocks sent or received.	(PM_ALINK_DATA_EVENT * 16 / PM_PAU_CYC) *PAU_freq	alinks_data_bw (GBps)
PM_ALINK_TOTAL_UTIL	Total utilization of the link due to all commands, header, data, rcmd, and presp.	(PM_ALINK_TOTAL_UTIL / PM_ALINK_AVLBL_CYCLES) * 100	alink_total_utilization (%)

### E.6.2.6 PAU Links - OpenCAPI Transaction Layer (OTL)

The Open Coherently Attached Processor Interface (OpenCAPI) is used to allow an attached functional unit (AFU) to connect to the Power10 processor chip's fabric system bus in a high-speed and cache coherent manner. The PAU provides the transaction layer functionality for the OpenCAPI links on the Power10 chip. The Nest instrumentation provides PAU-OTL events and related metrics.

*Table E-43* lists the OTL PMU events and performance metrics.



Table E-43. OTL PMY Events and Performance Metrics

Event	Description	Formula	Metric
PM_OTL_CYC	Total number of clock cycles.		
PM_OTL_RX_DATA_FLIT	Number of 64-byte data flits received on an OTL link.	$(PM\_OTL\_RX\_DATA\_FLIT * 64 / PM\_OTL\_CYC) * PAU\_freq$	ocapi_rx_bw (GBps)
PM_OTL_TX_CNTRL_FLIT	Number of control flits transmitted on an OTL link.	$(PM\_OTL\_TX\_CNTRL\_FLIT / PM\_OTL\_CYC) * PAU\_freq$	ocapi_cmd_rate (per sec)
PM_OTL_TX_FLTRD_DATA_FLIT	Number of 64-byte data flits transferred on an OTL link,	$(PM\_OTL\_TX\_FLTRD\_DATA\_FLIT * 64 / PM\_OTL\_CYC) * PAU\_freq$	ocapi_tx_bw (GBps)



#### E.6.2.7 PCI Express Controller and PCIe Host Bridge

The PCI Express controller (PEC) provides a PCIe Root Complex port to connect to an adapter, slot, or as a link to a PCIe switch. It acts as a PCI Express Host Bridge (PHB) from the internal, coherent processor bus to the PCI Express I/O. The PHB implements all required functions to support the Power Systems and PCIe protocol.

*Table E-44* lists the PEC PMU events.

*Table E-44. PEC PMU Events*

Event	Description	Formula	Metric
PM_PCI[0-1]_32B_INOUT	Number of 32-byte data transferred over the internal fabric to and from the PEC. This event counts aggregate bandwidth used by three PHBs supported by selected PEC. There are two PECs in the Power10 processor chip. Hence, a similar event exists for PEC0 and PEC1	$(PM\_PCI[0-1]_{32B\_INOUT} * 32 / PM\_PAU\_CYC) * PAU\_Freq$	pec_fabric_bw (GBps)
PM_PCI[0-1]_DMA_32B_RD_DATA_STK[0-2]	Number of 32-byte read DMA data transferred on the internal fabric bus to this PHB. Similar events exist for PHB1 and PHB2 for both PECs.	$(PM\_PCI[0-1]_{DMA\_32B\_RD\_DATA\_STK[0-2]} * 32 / PM\_PAU\_CYC\_CNT) * PAU\_Freq$	phb_dma_read_bw (GBps)

**Note:** The formula for PEC PMU events are calculated for 32-byte granular data transfers, with smaller size transfers rounded up to 32-bytes.

*Table E-45* lists the PHB PMU events.

*Table E-45. PHB PMU Events*

Event	Description	Formula	Metric
PM_PHB_CYC	Count PHB clock cycles running at 2 GHz.		
PM_PHB0_IPD_BUFF_POP	Number of 16-byte DMA write data received from PHB0. Similar events exist for PHB1 and PHB2 for both PECs.	$(PM\_PHB0\_IPD\_BUFF\_POP * 16 / PM\_PHB\_CYC) * PHB\_Freq$	phb_dma_write_bw (GBps)
PM_PHB0_MMIO_WR_DATA_VALID	Number of 16-byte MMIO writes to PCIe link. Similar events exist for PHB1 and PHB2 for both PECs.	$(PM\_PHB0\_MMIO\_WR\_DATA\_VALID * 16 / PM\_PHB\_CYC) * PHB\_Freq$	phb_mmio_write_bw (GBps)
PM_PHB0_ICPLD_BUFF_POP	Number of 16-byte MMIO reads from a PCIe link. Similar events exist for PHB1 and PHB2 for both PECs.	$(PM\_PHB0\_ICPLD\_BUFF\_POP * 16 / PM\_PHB\_CYC) * PHB\_Freq$	phb_mmio_read_bw (GBps)

**Note:** The formula for PHB PMU events are calculated for 16-byte granular data transfers, with smaller size transfers rounded up to 16-bytes.

### E.6.3 Nest IMC Events Grouping

Table E-46 lists the predefined group of events to be monitored by the IMC and their corresponding memory offsets.

In the Nest IMC, a [DTS](#) file in conjunction with the [PCP](#) interface resolves the offset based on the event names. This eliminates the requirement that the user know the offset of each of the events relative to the base.

**Note:** Table E-46 lists all events that can exist. However, the actual active events depend on each system's configuration, as well as each processor chip configuration.

Table E-46. Nest PMU Events (Sheet 1 of 12)

Offset	Event
0x8	PM_MCS_MDI_UPDATE_DONE_MC0_CHAN01
0x10	PM_MCS_128B_RD_DATA_BLOCKS_MC0_CHAN01
0x18	PM_MCS_64B_WR_DATA_BLOCKS_MC0_CHAN01
0x20	PM_PCI0_32B_INOUT
0x28	PM_MCS_MDI_UPDATE_DONE_MC1_CHAN01
0x30	PM_MCS_128B_RD_DATA_BLOCKS_MC1_CHAN01
0x38	PM_MCS_64B_WR_DATA_BLOCKS_MC1_CHAN01
0x40	PM_PAU_CYC
0x48	PM_PAU_CYC
0x50	PM_PAU_CYC
0x58	PM_PAU_CYC
0x60	PM_PAU_CYC
0x68	PM_PCI0_DMA_32B_RD_DATA_STK0
0x70	PM_PCI0_DMA_32B_RD_DATA_STK1
0x78	PM_PCI0_DMA_32B_RD_DATA_STK2
0x80	PM_PAU_CYC
0x88	PM_MCS_MDI_UPDATE_DONE_MC2_CHAN01
0x90	PM_MCS_128B_RD_DATA_BLOCKS_MC2_CHAN01
0x98	PM_MCS_64B_WR_DATA_BLOCKS_MC2_CHAN01
0xa0	PM_PCI1_32B_INOUT
0xa8	PM_MCS_MDI_UPDATE_DONE_MC3_CHAN01
0xb0	PM_MCS_128B_RD_DATA_BLOCKS_MC3_CHAN01
0xb8	PM_PB_PUMP_P1
0xc0	PM_PB_PUMP_P3
0xc8	PM_PAU_CYC
0xd0	PM_MCS_64B_WR_DATA_BLOCKS_MC3_CHAN01
0xd8	PM_PB_PUMP_P0
0xe0	PM_PB_PUMP_P2



*Table E-46. Nest PMU Events (Sheet 2 of 12)*

Offset	Event
0xe8	PM_PCI1_DMA_32B_RD_DATA_STK0
0xf0	PM_PCI1_DMA_32B_RD_DATA_STK1
0xf8	PM_PCI1_DMA_32B_RD_DATA_STK2
0x100	PM_PAU_CYC
0x118	PM_PB_INT_DATA_XFER
0x120	PM_PB_EXT_DATA_XFER
0x128	PM_PB_NNS_PUMP23
0x130	PM_PB_RTY_NNS_PUMP23
0x138	PM_PB_LOCAL_DATA_X
0x140	PM_PB_LOCAL_DATA_A_S
0x148	PM_PAU_CYC
0x150	PM_PAU_CYC
0x158	PM_PB_VG_PUMP23
0x160	PM_PB_LNS_PUMP23
0x168	PM_PB_GROUP_PUMP23
0x170	PM_PB_RNS_PUMP23
0x178	PM_PB_RTY_VG_PUMP23
0x180	PM_PB_RTY_LNS_PUMP23
0x188	PM_PB_RTY_GROUP_PUMP23
0x190	PM_PB_RTY_RNS_PUMP23
0x198	PM_PB_VG_PUMP01
0x1a0	PM_PB_LNS_PUMP01
0x1a8	PM_PB_GROUP_PUMP01
0x1b0	PM_PB_RNS_PUMP01
0x1b8	PM_PB_RTY_VG_PUMP01
0x1c0	PM_PB_RTY_LNS_PUMP01
0x1c8	PM_PB_RTY_GROUP_PUMP01
0x1d0	PM_PB_RTY_RNS_PUMP01
0x1d8	PM_PAU_CYC
0x1e0	PM_PAU_CYC
0x1e8	PM_PB_NNS_PUMP01
0x1f0	PM_PB_RTY_NNS_PUMP01
0x1f8	PM_PAU_CYC
0x200	PM_PAU_CYC
0x208	PM_PAU_CYC
0x210	PM_PAU_CYC
0x228	PM_XLINK0_OUT_EVEN_AVLBL_CYCLES



Table E-46. Nest PMU Events (Sheet 3 of 12)

Offset	Event
0x230	PM_XLINK0_OUT_EVEN_ANY_RCMD
0x238	PM_XLINK0_OUT_EVEN_DATA
0x240	PM_XLINK0_OUT_EVEN_TOTAL_UTIL
0x248	PM_XLINK0_OUT_ODD_AVLBL_CYCLES
0x250	PM_XLINK0_OUT_ODD_ANY_RCMD
0x258	PM_XLINK0_OUT_ODD_DATA
0x260	PM_XLINK0_OUT_ODD_TOTAL_UTIL
0x268	PM_XLINK1_OUT_EVEN_AVLBL_CYCLES
0x270	PM_XLINK1_OUT_EVEN_ANY_RCMD
0x278	PM_XLINK1_OUT_EVEN_DATA
0x280	PM_XLINK1_OUT_EVEN_TOTAL_UTIL
0x288	PM_XLINK1_OUT_ODD_AVLBL_CYCLES
0x290	PM_XLINK1_OUT_ODD_ANY_RCMD
0x298	PM_XLINK1_OUT_ODD_DATA
0x2a0	PM_XLINK1_OUT_ODD_TOTAL_UTIL
0x2a8	PM_XLINK2_OUT_EVEN_AVLBL_CYCLES
0x2b0	PM_XLINK2_OUT_EVEN_ANY_RCMD
0x2b8	PM_XLINK2_OUT_EVEN_DATA
0x2c0	PM_XLINK2_OUT_EVEN_TOTAL_UTIL
0x2c8	PM_XLINK2_OUT_ODD_AVLBL_CYCLES
0x2d0	PM_XLINK2_OUT_ODD_ANY_RCMD
0x2d8	PM_XLINK2_OUT_ODD_DATA
0x2e0	PM_XLINK2_OUT_ODD_TOTAL_UTIL
0x2e8	PM_XLINK3_OUT_EVEN_AVLBL_CYCLES
0x2f0	PM_XLINK3_OUT_EVEN_ANY_RCMD
0x2f8	PM_XLINK3_OUT_EVEN_DATA
0x300	PM_XLINK3_OUT_EVEN_TOTAL_UTIL
0x308	PM_XLINK3_OUT_ODD_AVLBL_CYCLES
0x310	PM_XLINK3_OUT_ODD_ANY_RCMD
0x318	PM_XLINK3_OUT_ODD_DATA
0x320	PM_XLINK3_OUT_ODD_TOTAL_UTIL
0x338	PM_XLINK4_OUT_EVEN_AVLBL_CYCLES
0x340	PM_XLINK4_OUT_EVEN_ANY_RCMD
0x348	PM_XLINK4_OUT_EVEN_DATA
0x350	PM_XLINK4_OUT_EVEN_TOTAL_UTIL
0x358	PM_XLINK4_OUT_ODD_AVLBL_CYCLES
0x360	PM_XLINK4_OUT_ODD_ANY_RCMD



*Table E-46. Nest PMU Events (Sheet 4 of 12)*

Offset	Event
0x368	PM_XLINK4_OUT_ODD_DATA
0x370	PM_XLINK4_OUT_ODD_TOTAL_UTIL
0x378	PM_XLINK5_OUT_EVEN_AVLBL_CYCLES
0x380	PM_XLINK5_OUT_EVEN_ANY_RCMD
0x388	PM_XLINK5_OUT_EVEN_DATA
0x390	PM_XLINK5_OUT_EVEN_TOTAL_UTIL
0x398	PM_XLINK5_OUT_ODD_AVLBL_CYCLES
0x3a0	PM_XLINK5_OUT_ODD_ANY_RCMD
0x3a8	PM_XLINK5_OUT_ODD_DATA
0x3b0	PM_XLINK5_OUT_ODD_TOTAL_UTIL
0x3b8	PM_XLINK6_OUT_EVEN_AVLBL_CYCLES
0x3c0	PM_XLINK6_OUT_EVEN_ANY_RCMD
0x3c8	PM_XLINK6_OUT_EVEN_DATA
0x3d0	PM_XLINK6_OUT_EVEN_TOTAL_UTIL
0x3d8	PM_XLINK6_OUT_ODD_AVLBL_CYCLES
0x3e0	PM_XLINK6_OUT_ODD_ANY_RCMD
0x3e8	PM_XLINK6_OUT_ODD_DATA
0x3f0	PM_XLINK6_OUT_ODD_TOTAL_UTIL
0x3f8	PM_XLINK7_OUT_EVEN_AVLBL_CYCLES
0x400	PM_XLINK7_OUT_EVEN_ANY_RCMD
0x408	PM_XLINK7_OUT_EVEN_DATA
0x410	PM_XLINK7_OUT_EVEN_TOTAL_UTIL
0x418	PM_XLINK7_OUT_ODD_AVLBL_CYCLES
0x420	PM_XLINK7_OUT_ODD_ANY_RCMD
0x428	PM_XLINK7_OUT_ODD_DATA
0x430	PM_XLINK7_OUT_ODD_TOTAL_UTIL
0x448	PM_ALINK0_OUT_EVEN_AVLBL_CYCLES
0x450	PM_ALINK0_OUT_EVEN_ANY_RCMD
0x458	PM_ALINK0_OUT_EVEN_DATA
0x460	PM_ALINK0_OUT_EVEN_TOTAL_UTIL
0x468	PM_ALINK0_OUT_ODD_AVLBL_CYCLES
0x470	PM_ALINK0_OUT_ODD_ANY_RCMD
0x478	PM_ALINK0_OUT_ODD_DATA
0x480	PM_ALINK0_OUT_ODD_TOTAL_UTIL
0x488	PM_ALINK1_OUT_EVEN_AVLBL_CYCLES
0x490	PM_ALINK1_OUT_EVEN_ANY_RCMD
0x498	PM_ALINK1_OUT_EVEN_DATA



Table E-46. Nest PMU Events (Sheet 5 of 12)

Offset	Event
0x4a0	PM_ALINK1_OUT_EVEN_TOTAL_UTIL
0x4a8	PM_ALINK1_OUT_ODD_AVLBL_CYCLES
0x4b0	PM_ALINK1_OUT_ODD_ANY_RCMD
0x4b8	PM_ALINK1_OUT_ODD_DATA
0x4c0	PM_ALINK1_OUT_ODD_TOTAL_UTIL
0x4c8	PM_ALINK2_OUT_EVEN_AVLBL_CYCLES
0x4d0	PM_ALINK2_OUT_EVEN_ANY_RCMD
0x4d8	PM_ALINK2_OUT_EVEN_DATA
0x4e0	PM_ALINK2_OUT_EVEN_TOTAL_UTIL
0x4e8	PM_ALINK2_OUT_ODD_AVLBL_CYCLES
0x4f0	PM_ALINK2_OUT_ODD_ANY_RCMD
0x4f8	PM_ALINK2_OUT_ODD_DATA
0x500	PM_ALINK2_OUT_ODD_TOTAL_UTIL
0x508	PM_ALINK3_OUT_EVEN_AVLBL_CYCLES
0x510	PM_ALINK3_OUT_EVEN_ANY_RCMD
0x518	PM_ALINK3_OUT_EVEN_DATA
0x520	PM_ALINK3_OUT_EVEN_TOTAL_UTIL
0x528	PM_ALINK3_OUT_ODD_AVLBL_CYCLES
0x530	PM_ALINK3_OUT_ODD_ANY_RCMD
0x538	PM_ALINK3_OUT_ODD_DATA
0x540	PM_ALINK3_OUT_ODD_TOTAL_UTIL
0x558	PM_ALINK4_OUT_EVEN_AVLBL_CYCLES
0x560	PM_ALINK4_OUT_EVEN_ANY_RCMD
0x568	PM_ALINK4_OUT_EVEN_DATA
0x570	PM_ALINK4_OUT_EVEN_TOTAL_UTIL
0x578	PM_ALINK4_OUT_ODD_AVLBL_CYCLES
0x580	PM_ALINK4_OUT_ODD_ANY_RCMD
0x588	PM_ALINK4_OUT_ODD_DATA
0x590	PM_ALINK4_OUT_ODD_TOTAL_UTIL
0x598	PM_ALINK5_OUT_EVEN_AVLBL_CYCLES
0x5a0	PM_ALINK5_OUT_EVEN_ANY_RCMD
0x5a8	PM_ALINK5_OUT_EVEN_DATA
0x5b0	PM_ALINK5_OUT_EVEN_TOTAL_UTIL
0x5b8	PM_ALINK5_OUT_ODD_AVLBL_CYCLES
0x5c0	PM_ALINK5_OUT_ODD_ANY_RCMD
0x5c8	PM_ALINK5_OUT_ODD_DATA
0x5d0	PM_ALINK5_OUT_ODD_TOTAL_UTIL



*Table E-46. Nest PMU Events (Sheet 6 of 12)*

Offset	Event
0x5d8	PM_ALINK6_OUT_EVEN_AVLBL_CYCLES
0x5e0	PM_ALINK6_OUT_EVEN_ANY_RCMD
0x5e8	PM_ALINK6_OUT_EVEN_DATA
0x5f0	PM_ALINK6_OUT_EVEN_TOTAL_UTIL
0x5f8	PM_ALINK6_OUT_ODD_AVLBL_CYCLES
0x600	PM_ALINK6_OUT_ODD_ANY_RCMD
0x608	PM_ALINK6_OUT_ODD_DATA
0x610	PM_ALINK6_OUT_ODD_TOTAL_UTIL
0x618	PM_ALINK7_OUT_EVEN_AVLBL_CYCLES
0x620	PM_ALINK7_OUT_EVEN_ANY_RCMD
0x628	PM_ALINK7_OUT_EVEN_DATA
0x630	PM_ALINK7_OUT_EVEN_TOTAL_UTIL
0x638	PM_ALINK7_OUT_ODD_AVLBL_CYCLES
0x640	PM_ALINK7_OUT_ODD_ANY_RCMD
0x648	PM_ALINK7_OUT_ODD_DATA
0x650	PM_ALINK7_OUT_ODD_TOTAL_UTIL
0x668	PM_OTL0_0_CYC
0x670	PM_OTL0_0_RX_DATA_FLIT
0x678	PM_OTL0_0_TX_CNTRL_FLIT
0x680	PM_OTL0_0_TX_FLTRD_DATA_FLIT
0x688	PM_OTL0_1_CYC
0x690	PM_OTL0_1_RX_DATA_FLIT
0x698	PM_OTL0_1_TX_CNTRL_FLIT
0x6a0	PM_OTL0_1_TX_FLTRD_DATA_FLIT
0x6a8	PM_OTL3_0_CYC
0x6b0	PM_OTL3_0_RX_DATA_FLIT
0x6b8	PM_OTL3_0_TX_CNTRL_FLIT
0x6c0	PM_OTL3_0_TX_FLTRD_DATA_FLIT
0x6c8	PM_OTL3_1_CYC
0x6d0	PM_OTL3_1_RX_DATA_FLIT
0x6d8	PM_OTL3_1_TX_CNTRL_FLIT
0x6e0	PM_OTL3_1_TX_FLTRD_DATA_FLIT
0x6e8	PM_OTL4_0_CYC
0x6f0	PM_OTL4_0_RX_DATA_FLIT
0x6f8	PM_OTL4_0_TX_CNTRL_FLIT
0x700	PM_OTL4_0_TX_FLTRD_DATA_FLIT
0x708	PM_OTL4_1_CYC

Table E-46. Nest PMU Events (Sheet 7 of 12)

Offset	Event
0x710	PM_OTL4_1_RX_DATA_FLIT
0x718	PM_OTL4_1_TX_CNTRL_FLIT
0x720	PM_OTL4_1_TX_FLTRD_DATA_FLIT
0x728	PM_OTL5_0_CYC
0x730	PM_OTL5_0_RX_DATA_FLIT
0x738	PM_OTL5_0_TX_CNTRL_FLIT
0x740	PM_OTL5_0_TX_FLTRD_DATA_FLIT
0x748	PM_OTL5_1_CYC
0x750	PM_OTL5_1_RX_DATA_FLIT
0x758	PM_OTL5_1_TX_CNTRL_FLIT
0x760	PM_OTL5_1_TX_FLTRD_DATA_FLIT
0x778	PM_OTL6_0_CYC
0x780	PM_OTL6_0_RX_DATA_FLIT
0x788	PM_OTL6_0_TX_CNTRL_FLIT
0x790	PM_OTL6_0_TX_FLTRD_DATA_FLIT
0x798	PM_OTL6_1_CYC
0x7a0	PM_OTL6_1_RX_DATA_FLIT
0x7a8	PM_OTL6_1_TX_CNTRL_FLIT
0x7b0	PM_OTL6_1_TX_FLTRD_DATA_FLIT
0x7b8	PM_OTL7_0_CYC
0x7c0	PM_OTL7_0_RX_DATA_FLIT
0x7c8	PM_OTL7_0_TX_CNTRL_FLIT
0x7d0	PM_OTL7_0_TX_FLTRD_DATA_FLIT
0x7d8	PM_OTL7_1_CYC
0x7e0	PM_OTL7_1_RX_DATA_FLIT
0x7e8	PM_OTL7_1_TX_CNTRL_FLIT
0x7f0	PM_OTL7_1_TX_FLTRD_DATA_FLIT
0x7f8	FREE
0x800	FREE
0x808	FREE
0x810	FREE
0x818	FREE
0x820	FREE
0x828	FREE
0x830	FREE
0x838	FREE
0x840	FREE



*Table E-46. Nest PMU Events (Sheet 8 of 12)*

Offset	Event
0x848	FREE
0x850	FREE
0x858	FREE
0x860	FREE
0x868	FREE
0x870	FREE
0x888	PM_PHB0_0_CYC
0x890	PM_PHB0_0_MMIO_WR_DATA_VALID
0x898	PM_PHB0_0_IPD_BUFF_POP
0x8a0	PM_PHB0_0_ICPLD_BUFF_POP
0x8a8	PM_PHB0_1_CYC
0x8b0	PM_PHB0_1_MMIO_WR_DATA_VALID
0x8b8	PM_PHB0_1_IPD_BUFF_POP
0x8c0	PM_PHB0_1_ICPLD_BUFF_POP
0x8c8	PM_PHB0_2_CYC
0x8d0	PM_PHB0_2_MMIO_WR_DATA_VALID
0x8d8	PM_PHB0_2_IPD_BUFF_POP
0x8e0	PM_PHB0_2_ICPLD_BUFF_POP
0x8e8	PM_PHB1_0_CYC
0x8f0	PM_PHB1_0_MMIO_WR_DATA_VALID
0x8f8	PM_PHB1_0_IPD_BUFF_POP
0x900	PM_PHB1_0_ICPLD_BUFF_POP
0x908	PM_PHB1_1_CYC
0x910	PM_PHB1_1_MMIO_WR_DATA_VALID
0x918	PM_PHB1_1_IPD_BUFF_POP
0x920	PM_PHB1_1_ICPLD_BUFF_POP
0x928	PM_PHB1_2_CYC
0x930	PM_PHB1_2_MMIO_WR_DATA_VALID
0x938	PM_PHB1_2_IPD_BUFF_POP
0x940	PM_PHB1_2_ICPLD_BUFF_POP
0x948	PM_NX_IDLE_CH01
0x950	PM_NX_IDLE_CH23
0x958	PM_NX_IDLE_CH4
0x960	PM_NX_TIMEBASE_CYC
0x968	PM_NX_ERAT_LOOKUP
0x970	PM_NX_ERAT_MISS
0x978	PM_NX_ERAT_STALLED_CICO_BUFFERS



Table E-46. Nest PMU Events (Sheet 9 of 12)

Offset	Event
0x980	PM_NX_DMA_STALLED
0x998	PM_OCMB0_ACT_ALL
0xa00	PM_OCMB0_CAS_RD
0xa8	PM_OCMB0_CAS_WR
0xb0	PM_OCMB0_COMMAND_CLOCKS
0xb8	PM_OCMB1_ACT_ALL
0xc0	PM_OCMB1_CAS_RD
0xc8	PM_OCMB1_CAS_WR
0xd0	PM_OCMB1_COMMAND_CLOCKS
0xd8	PM_OCMB2_ACT_ALL
0xe0	PM_OCMB2_CAS_RD
0xe8	PM_OCMB2_CAS_WR
0xf0	PM_OCMB2_COMMAND_CLOCKS
0xf8	PM_OCMB3_ACT_ALL
0xa00	PM_OCMB3_CAS_RD
0xa08	PM_OCMB3_CAS_WR
0xa10	PM_OCMB3_COMMAND_CLOCKS
0xa18	TOD
0xa20	FREE
0xa28	FREE
0xa30	FREE
0xa38	FREE
0xa40	FREE
0xa48	FREE
0xa50	FREE
0xa58	FREE
0xa60	FREE
0xa68	FREE
0xa70	FREE
0xa78	FREE
0xa80	FREE
0xa88	FREE
0xa90	FREE
0xaa8	PM_OCMB4_ACT_ALL
0xab0	PM_OCMB4_CAS_RD
0xab8	PM_OCMB4_CAS_WR



Table E-46. Nest PMU Events (Sheet 10 of 12)

Offset	Event
0xac0	PM_OCMB4_COMMAND_CLOCKS
0xac8	PM_OCMB5_ACT_ALL
0xad0	PM_OCMB5_CAS_RD
0xad8	PM_OCMB5_CAS_WR
0xae0	PM_OCMB5_COMMAND_CLOCKS
0xae8	PM_OCMB6_ACT_ALL
0xaf0	PM_OCMB6_CAS_RD
0xaf8	PM_OCMB6_CAS_WR
0xb00	PM_OCMB6_COMMAND_CLOCKS
0xb08	PM_OCMB7_ACT_ALL
0xb10	PM_OCMB7_CAS_RD
0xb18	PM_OCMB7_CAS_WR
0xb20	PM_OCMB7_COMMAND_CLOCKS
0xb28	FREE
0xb30	FREE
0xb38	FREE
0xb40	FREE
0xb48	FREE
0xb50	FREE
0xb58	FREE
0xb60	FREE
0xb68	FREE
0xb70	FREE
0xb78	FREE
0xb80	FREE
0xb88	FREE
0xb90	FREE
0xb98	FREE
0xba0	FREE
0xbb8	PM_OCMB8_ACT_ALL
0xbc0	PM_OCMB8_CAS_RD
0xbc8	PM_OCMB8_CAS_WR
0xbd0	PM_OCMB8_COMMAND_CLOCKS
0xbd8	PM_OCMB9_ACT_ALL
0xbe0	PM_OCMB9_CAS_RD
0xbe8	PM_OCMB9_CAS_WR
0xbf0	PM_OCMB9_COMMAND_CLOCKS



Table E-46. Nest PMU Events (Sheet 11 of 12)

Offset	Event
0xbf8	PM_OCMB10_ACT_ALL
0xc00	PM_OCMB10_CAS_RD
0xc08	PM_OCMB10_CAS_WR
0xc10	PM_OCMB10_COMMAND_CLOCKS
0xc18	PM_OCMB11_ACT_ALL
0xc20	PM_OCMB11_CAS_RD
0xc28	PM_OCMB11_CAS_WR
0xc30	PM_OCMB11_COMMAND_CLOCKS
0xc38	FREE
0xc40	FREE
0xc48	FREE
0xc50	FREE
0xc58	FREE
0xc60	FREE
0xc68	FREE
0xc70	FREE
0xc78	FREE
0xc80	FREE
0xc88	FREE
0xc90	FREE
0xc98	FREE
0xca0	FREE
0xca8	FREE
0xcb0	FREE
0xcc8	PM_OCMB12_ACT_ALL
0xcd0	PM_OCMB12_CAS_RD
0xcd8	PM_OCMB12_CAS_WR
0xce0	PM_OCMB12_COMMAND_CLOCKS
0xce8	PM_OCMB13_ACT_ALL
0xcf0	PM_OCMB13_CAS_RD
0xcf8	PM_OCMB13_CAS_WR
0xd00	PM_OCMB13_COMMAND_CLOCKS
0xd08	PM_OCMB14_ACT_ALL
0xd10	PM_OCMB14_CAS_RD
0xd18	PM_OCMB14_CAS_WR
0xd20	PM_OCMB14_COMMAND_CLOCKS
0xd28	PM_OCMB15_ACT_ALL



*Table E-46. Nest PMU Events (Sheet 12 of 12)*

Offset	Event
0xd30	PM_OCMB15_CAS_RD
0xd38	PM_OCMB15_CAS_WR
0xd40	PM_OCMB15_COMMAND_CLOCKS
0xd48	FREE
0xd50	FREE
0xd58	FREE
0xd60	FREE
0xd68	FREE
0xd70	FREE
0xd78	FREE
0xd80	FREE
0xd88	FREE
0xd90	FREE
0xd98	FREE
0xda0	FREE
0xda8	FREE
0xdb0	FREE
0xdb8	FREE
0xdc0	FREE



## Appendix F. Power10 Processor Programming Model Bulletin

The Power10 processor implementation of stream prefetching is an evolutionary step toward an extension that is planned to appear in version 3.2 of the Power ISA. The *Power10 Processor Programming Model Bulletin* provides a preliminary view of the extension in the form of an update of the relevant portion of Section 4.3.2 of *Power ISA Virtual Environment Architecture - Book II (version 3.1B)*. Changes relative to version 3.1B are indicated by change bars. Fields of the effective address (EA) of the **dcbt** and **dcbtst** instructions that are not implemented by the Power10 processor are identified by red text. These fields should be coded as zeros for execution on the Power10 processor. Doing so will provide the functions described for their zero values. Failure to code the fields as zeros may produce undesirable stream prefetch behavior, and should be scrupulously avoided.

**Note:** Contact your IBM representative for access to the *Power10 Processor Programming Model Bulletin*.





## Appendix G. Errata

This section identifies differences between the actual operation of the IBM Power10 chip and what is described in the *Power10 User Manual* and applicable documents.

Differences that are not visible to the user are not described. If a difference is hidden or compensated for by IBM-supplied firmware or by IBM-supplied system software, it is not included in this document.

Check regularly with your IBM technical representative to verify that you have the most current version of the errata.

**Note:** Some of the workarounds described in this section might involve the use of IBM software or firmware modules to affect or to reduce the effect of the erratum. These modules are typically supplied with the Power10 devices. However, if you do not have the module required for a workaround, contact IBM to acquire it.

### G.1 Revision Levels Covered

This document includes information about errata that apply to the following design revision levels:

- Power10 processor DD2.0 design revision level

This document does not contain information about Power10 processor levels before DD2.0. Any statements in this document about other revision levels of these products are for reference only. For errata information about these revision levels, contact your IBM representative.

### G.2 Summary of Errata

*Table G-1.* summarizes the Power10 errata.

*Table G-1. Power10 Errata*

Errata ID	Description / Impact / Workaround	Workaround Available?
U1	When microrocoded instructions (see the “Expanded” column in <i>Appendix A Instruction Properties</i> on page 295) are subject to <b>cmodx</b> and replaced by a taken branch, there are cases where the instruction executed will be the sequential instruction after the branch and not the instruction at the target address. In addition, the <u>CFAR</u> is not updated with the branch target address.	No
U2	<u>BHRB</u> is incorrectly updated with the interrupt handler starting address if the branch is the subject of a CIABR match.	No
U3	BHRB is incorrectly updated for the target of a branch if it is the subject of a CIABR exception.	No



## Glossary

ABIST	Array built-in self test
AC	Alternating current. Alternating current mode refers to at-speed testing, which means run at a high frequency (GHz range).  Can also be defined as authentication code.
ACAM	Address content addressable memory
ADC	Analog-to-digital converter
AES	Advanced Encryption Standard
AIB	ASIC interface bus
ALI	Alignment interrupt
ALU	Arithmetic logic unit
AMO	Atomic memory operation
AMOR	Authority Mask Override Register
API	Application programming interface
ARF	Architected register file
ASIC	Application-specific integrated circuit
ASST	At-speed structure-test
ATC	Address translation cache or effective-to-real address translations
ATPG	Advanced test pattern generator
ATS	Address translation services
AVA	Abbreviated virtual address
AVS	Analog voltage scaling
AXON	A-bus/X-bus/OpenCAPI/Networking
BAR	Base Address Register
BCD	Binary coded decimal
BER	Bit error ratio
BFP	Binary floating-point
BFU	Binary floating-point unit
BHRB	Branch History Rolling Buffer

BHT	Branch history table
BIST	Built-in self-test
BMC	Baseboard management controller
BMI	Bit manipulation pipeline
BR	Branch register unit
BRU	Branch register unit
BTAC	Branch target address cache
CAM	Content-addressable memory
CAPI	Coherent accelerator processor interface
CAPP	Coherently attached processor proxy
CAR1	First cycle after the incoming snoop request
CBC	Cipher-block chaining
CBS	<u>CFAM</u> boot sequencer
CC	Completion code
CCM	Counter with CBC-MAC
CCS	Configured command sequencer
CDR	Clock and data recovery
CEC	Central electronics complex
CFAM	Common <u>FRU</u> access macro
CFAR	Come-From Address Register
cgc	Congruence class
CI	Cast-in
CIABR	Current Instruction Address Breakpoint Register
CIR	Chip information register
CIU	Core interface unit
CLB	Cache load buffer (IBuffer)
CME	Core management engine
CMOS	Complementary metal–oxide–semiconductor
CO	Cast-out



CPB	Coprocessor parameter block
CPI	Cycles per instruction
CPM	Critical path monitor
CPU	Central processing unit
CR	Condition Register
CRB	Coprocessor request block
CRC	Cyclic redundancy check
CRESP	Combined response
CRN	Conditioned random numbers
CT	Coprocessor type
CTLE	Continuous time linear equalizer
CTR	Counter
CTRL	Control Register
CTS	Convert-to-single
DAC	Digital-to-analog converter
DAR	Data Address Register
darn	Deliver a random number instruction
DARQ	Data and address recirculation queue
DAWR	Data Address Watch Register
DC	Direct current
	Direct current mode refers to low-speed testing, which means run at a low frequency (kHz to MHz range).
dcbt	Data cache block touch
dcbtst	Data cache block touch for store
dcbz	Data cache block zero
dcbst	Data cache block store
dcbst	Data cache block store
dcbf	Data cache block flush
dcbfl	Data cache block flush local
dcbflp	Data cache block flush local primary

DCP	Data co-processor
DDL	Data descriptor list
DDR	Double data rate
DDR4	Double data rate memory interface, 4th generation
DDS	Digital droop sensor
DECFP	Decimal floating-point unit
DEXCR	Dynamic Execution Control Register
DFE	Decision feedback equalizer
DFP	Decimal floating-point
DFT	Design for test
DFU	Decimal floating-point unit
DIMM	Dual in-line memory module
D-DIMM	Differential dual in-line memory module
DLL	Delay-locked loop
DLR	Dynamic lane reduction
DMA	Direct memory attach
DME	Dense math engine
DMI	Differential memory interface
DPC	DIMMs per channel or dynamic peaking control
DPDES	Directed privileged doorbell exception state
DPFD	Prefetch depth
DPLL	Digital phase-locked loop
DPTEG	Data page table entry group
DRAM	Dynamic random access memory
DRTM	Dynamic root of trust for measurement
DSEG	Data segment interrupt
DSI	Data storage interrupt
dss	Data stream stop
dst	Data stream touch

---

dstst	Data stream touch for store
DTS	Digital thermal sensor or direct-tree source
EA	Effective address
EADIR	Effective address directory
EAE	Event assignment entry
EAS	Event assignment structure
EASC	Event assignment structure cache
EAST	Event assignment structure table
EAT	Effective address translation or event assignment table
EBB	Event-based branch
ECB	Electronic codebook
ECC	Error correcting code
ECID	Electronic chip identification
ECO	Extended cache option
ECRC	End-to-end cyclic redundancy check
EDI	Elastic differential I/O
EDRAM	Enhanced dynamic random access memory
EEH	Enhance error handling
EEPROM	Electrically erasable programmable read-only memory
EMC	Extended memory controller
EMO	Extended memory <u>OMI</u>
EMQ	ERAT miss queue
END	Event notification descriptor
ENDC	Event notification descriptor cache
ENDE	Event notification descriptor entry
ENDT	Event notification descriptor table
EOI	End of interrupt
EPC	Enterprise Protected Container
EPF	Extended prefetch

EQ	Event queue
EQD	Event queue descriptor
EQDT	Event queue descriptor table
EQOC	Event queue page offset counter
ERAT	Effective-to-real address translation
ESB	Event state buffer
ESBC	Event state buffer cache
ESID	Effective segment identifier
FBC	SMP interconnect controller
FC	Function code
FFE	Feed-forward equalizer
FIFO	First-in, first-out
FIR	Fault Isolation Register
FLIT	Flow control digit
FPGA	Field-programmable gate array
FPR	Floating-point register
FPSCR	Floating-Point Status and Control Register
FPU	Floating-point unit
FRU	Field replaceable unit
FSI	Flexible service interface
FSP	Flexible service processor
FXU	Fixed-point units
GCM	Galois counter mode
GCT	Global completion table
GDDR	Graphics double data rate
GFW	Global firmware
GHV	Global history vector
GPE	General purpose engine
GPIO	General purpose input/output



GPR	General purpose register
GPS	Global Pstate
GPST	Global Pstates table
GPU	Graphics processor unit
GR	Guest Radix
GS	Group scope
GTps	Gigatransfers per second
GVA	Guest virtual address
HB	Hostboot
HBM	High-bandwidth memory
hcode	Hypervisor code
HDEC	Hypervisor decrementer
HDSI	Hypervisor data storage interrupt
hEA	Host effective address
HID	Hardware Implementation Dependent Register
HISI	Hypervisor instruction interrupt
HMAC	Hash message authentication code
HMEER	Hypervisor Maintenance Exception Enable Register
HMER	Hypervisor Maintenance Exception Register
HMI	Hypervisor maintenance interrupt or hardware maintenance interrupt
HPC	High-performance computing
HPT	Hashed page table
HR	Host Radix
hRA	Host real address
HRMOR	Hypervisor Real Mode Offset Register
HSS	High-speed serial
HTM	Hardware trace monitor
HVA	Host virtual address
I2C	Inter-integrated circuit

IBRTPDUS	Indirect Branch Recurrent Target Prediction Disable for MSR[HV, PR, TA, US]
IBUF	Instruction buffer
ICA	Instruction cache access
icbi	Instruction cache block invalidate
icbt	Instruction cache block touch
isync	Instruction cache synchronize
ICP	Interrupt control presenter
ICS	Interrupt controller source
ICT	Instruction completion table
IEEE	Institute of Electrical and Electronics Engineers
IFAR	Instruction fetch address register
IFB	Instruction fetch buffer
IFU	Instruction fetch and decode unit
IMA	In memory accumulate
IMC	In memory collection
IOO	OpenPOWER interface
IOP	Internal operation
IP	Intellectual property
IPB	Interrupt Pending Buffer
IPC	Instruction per cycle
IPL	Interrupt presenter layer; orinitial program load
IRL	Interrupt routing layer
IRR	Instruction retry and recovery
ISEG	Instruction segment interrupt
ISI	Instruction storage interrupt
ISU	Instruction sequencing unit
ISV	Independent software vendor
IVE	Interrupt vector entry
IVE	Interrupt Virtualization Entry



---

IVPE	Interrupt Virtualization Presentation Engine
IVRE	Interrupt Virtualization Routing Engine
iVRM	Internal voltage regulation
IVSE	Interrupt Virtualization Source Engine
IVT	Interrupt Virtualization Table
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
KiB	Kibibyte. The kibibyte is a multiple of the unit byte for quantities of digital information. The binary prefix kibi means 2 <sup>10</sup> , or 1024.
KVM	Kernal-based virtual machine
LBIST	Logic built-in self test
LCO	Lateral castout or lateral castout (cast out to another cache rather than memory)
LCRC	Link <a href="#">CRC</a>
LDBAR	Logical Domain Base Address Register
ld	Line delete
LDS	Load station
LE	Little-endian
LFSR	Linear Feedback Shift Register
LGA	Land grid array
LHR	Load-hit-reload
LLQ	Low-latency queue
LMQ	Load-miss queue
LNS	Load node scope
LPAR	Logical partition mobility
LPC	Lowest point of coherency or low-pin count
LPCR	Logical Partitioning Control Register
LPID	Logical partition ID
LPIDR	Logical Partition ID Register
LPM	Live partition mobility
LPST	Local Pstate table

LRDIMM	Load-reduced dual in-line memory module
LRQ	Load reorder queue
LRU	Least-recently used
LSAQ	Load store address queue
LSI	Level signaled interrupt
LSMFB	Logical Server Most Favored Backlog
LSSD	Level-sensitive scan design
LSU	Load store unit
LTE	Long-tail equalizer
LU	Load-only unit
MBA	Memory buffer asynchronous
MCA	Memory controller asynchronous
MCBIST	Memory card built in self-test
MCD	Memory cache-line domain
MCE	Machine check exception
MCS	Memory controller synchronous
MD5	Message Digest 5
MDI	Memory domain indicator
MDS	Memory domain status
MHCRO	Model hardware correlation ring oscillator
MI	Memory inception
MiB	Mebibyte. One mebibyte is equal to 1048576 bytes = 1024 kibibytes. A mebibyte is a power of two, appropriate for binary machines. Whereas, a megabyte (MB) is a power of ten.
MLE	Measured launch environment
MMA	Matrix math accelerator or matrix multiply assist
MMCRC	Core Mode Control Register
MMIO	Memory-mapped input/output
MMU	Memory management unit
MPG	Multi-protocol gateway

---

MPSS	Multiple page sizes per segment
MRQ	Master request controller
MRS	Mode register set
MRU	Most-recently used
MSI	Message signalled interrupt
MVPD	Module vital product data
NaN	Not a number
NCU	Noncacheable unit
NIA	Next instruction address
NMMU	Nest memory management unit
NPS	Nap Pstate
NTC	Next-to-complete
NUCA	Non-uniform cache access
NVC	Notification virtual crowd
NVG	Notification virtual group
NVP	Notification virtual processor
NVT	Notification virtual target
NVTS	Notification virtual target structure
NVTT	Notification Virtual Target Table
NxC	Notification virtual descriptor cache
OBS	Out-of-band signaling
OCC	On-chip controller
OCMB	OpenCAPI memory buffer
OCTS	On-chip thermal sensor
ODL	OpenCAPI datalink layer
OEM	Original equipment manufacturer
OHA	On-chiplet hardware assist
OpenCAPI	Open Coherent Accelerator Processor Interface
ODL	OpenCAPI datalink layer

OMI	OpenCAPI memory interface
OP	Optical <a href="#">PHY</a>
OPCG	On-chip clock generation
OTPROM	One-time programmable read-only memory
OTL	OpenCAPI transaction layer
P3CQ	Power10 fabric bus interface common queue
P3PC	Presentation Controller
P3SC	Source Controller
P3VC	Virtualization Controller
PACQ	PAU common queue
PAPR	Power Architecture Platform Reference
PATB	Partition Table Base field
PATS	Partition Table Size field
PAU	Power accelerator unit
PB	Processor bus
PBL	Packet buffer layer
PC	Pervasive core unit
PCB	Pervasive control bus
PCP	Performance co-pilot
PDL	SMP data link layer
PIB	Pervasive interconnect bus
PCIe	Peripheral component interconnect express
PCR	Processor Compatibility Register or Platform Configuration Register
PCS	Physical coding sublayer
PDE	Page directory entry
PE	Partitionable endpoints
PEC	PCI Express controller
PEF	Protected execution facility
PF	Prefetch machine



---

PFD	Phase-frequency detector
PFWI	Prefetch write inject
PGPE	P-State general purpose engine. GPE dedicated to processing P-state functions.
PHB	PCI host bridge
PHY	Physical layer
PID	Process ID
PIDR	Process ID Register
PIG	PCB-network interrupt generation
PIPR	Pending Interrupt Priority Register
PLL	Phase-locked loop
PMA	Physical media access or physical media attach
PMC	Power management control
PMCR	Performance Monitor Control Register
PMSR	Power Management Status Register
PMU	Performance monitor unit
POR	Power-on reset
PPE	Programmable PowerPC-lite engine
PPR	Program Priority Register
PRBS	Pseudo-random binary sequence
PRI	Private register interface
PRQ	Prefetch request queue
PSI	Processor serial interface
PSM5	PCIe speed match (fifth generation)
PSPB	Problem-state priority boost
PSRO	Process sensitive ring oscillator
PSSCR	Processor Stop State Control Register
Pstate	Performance state
PTCR	Partition Table Control Register
PTE	Page table entry

PTEG	Page table entry group
PTER	Physical Thread Enable Register
PTL	SMP transaction layer
PURR	Processor Utilization Resource Register
PVR	Processor Version Register
PWC	Page-walk cache
QME	Quad management engine
QNaN	Quiet Not a number
QoS	Quality of service
qpos	Queue position
RA	Read address
RAIM	Redundant array of independent memory
RAM	Random access memory
RAS	Reliability, availability, and serviceability
RAW	Read after write
RC	Root complex or read claim
RCD	Register clock driver
RCMD	Remote command or reflected command
RDIMM	Registered dual in-line memory module
RMA	Remote memory access
RMLS	Real Mode Limit Selector
RMOR	Real Mode Offset Register
RMSC	Real mode storage control
RNG	Random number generator
RNS	Remote node scope
ROB	Re-order buffer
ROP	Return oriented programming
ROT	Rollback-only transaction
RPR	Relative Priority Register



---

RPT	Radix page table
RRN	Raw random numbers
rtag	Routing tag
rVRM	Retention voltage regulation macro
RWMR	Region Weighted Mode Register
S2Q	Store drain queue
SAM	Store address machine
SAR	Second-level Architected Register
SBE	Self-boot engine or State bit entry
SBHE	Speculative Branch Hint Enable bit
Sc	Store clean (transactional memory value before a speculative store)
Schmoo	Shmoo plot is a technical term relating to the graphical display of test results in electrical engineering. The name most likely arose because the shape of the two-dimensional plots often resembled a shmoo. The term is also a verb; to shmoo means to run the test.
SCM	Single-chip module
SCOM	Scan communications
SDAR	Sampled Data Address Register
SDM	Store data machine
SDQ	Store data queue
SEC/DED	Single-error correction, double-error detection
SEEPROM	Serial electrically erasable programmable read-only memory
SER	Soft error
SERDES	Serializer/Deserializer
SETP	Set prediction directory
SFX	Simple fixed-point pipeline
SGPE	Stop general purpose engine. GPE dedicated to processing Stop functions.
SHA	Secure hash algorithm
SHR	Store-hit-reload
SIAAT	Shared instruction address alias table

SIAR	Sampled Instruction Address Register
SICQ	SMP interconnect common queue
SIMD	Single-instruction, multiple-data
SIU	SMP interconnect unit
Skitter	Skew and jitter
SLB	Segment lookaside buffer
SLR	Single Level Radix
SM	State machine
SMF	Secure memory facility
SMM	Selective memory mirroring
SMP	Symmetric multiprocessing
SMPI	SMP interconnect
SMT	Simultaneous multithreading
SN	Snoop machine
SOI	Silicon-on-insulator
SP	Single-precision
SPI	Serial peripheral interconnect
SPIVID	Serial Peripheral Interface - Voltage ID
SPR	Special purpose register
SPS	Sleep Pstate
SPURR	Scaled Processor Utilization Resource Register
SRAM	Static random access memory
SRAPDUS	Subroutine Return Address Prediction Disable for MSR[HV, PR, TA, US]
SRQ	Store reorder queue
SSE	Store-stream prefetch enable
SST	Series source terminated
ST	Single thread
ST-D	Store data production
STAG	Storage tag

---

STE	Send Window Table Entry (STE) or segment table entry
STEG	Segment table entry group
STF	Slice target file
STM	Synthetic transactional memory
SUE	Special uncorrectable error
SVM	Secure virtual machine
TADB	Database of translated addresses
TB	Time-base
TCAM	TAG content addressable memory
TCE	Translation control entry
TCTXT	Thread context
TDP	Thermal design point
TEXASRU	Transaction Exception And Summary Register Upper
TID	Thread ID
TIP	Tagged indirect predictor
TIMA	Thread interrupt management area
TIQ	Translation issue queue
TLB	Translation lookaside buffer
TLBI	Translation look-aside buffer invalidate
TLDLP	Transaction and data link layer
TLE	Transaction lock elision
TLP	Translation layer packet
TM	Transactional memory
TOD	Time of day
TOP	Tagged orientation predictor
TPM	Trusted platform module
TSCR	Thread Switch Control Register
TTR	Thread Timeout Register
UE	Uncorrectable error

UI	Unit interval
UMAC	User mode access control
UniQ	Unified issue queues
URR	User request register
VAS	Virtual Accelerator Switchboard
VCO	Voltage-controlled oscillator
VID	Voltage identification
VDM	Voltage droop monitor
VGA	Variable gain amplifier
VGS	Vectored group scope
VHDL	<a href="#"><u>VHSIC</u> Hardware Description Language</a>
VHSIC	Very High-Speed Integrated Circuit
VLE	Variable length encoding
VM	Virtual machine
VMX	Virtual machine extensions
VPD	Vital product data
VPD	Virtual Processor Descriptor
VPDT	Virtual Processor Descriptor Table
VPN	Virtual page number
VRF	Vector scalar register file
VRM	Voltage regulator module
VRMA	Virtualized real mode area
VS	Vector scalar
VSB	Voltage standby
VSCR	Vector Status and Control Register
VSD	Virtualization Structure Descriptor
VSR	Vector scalar register
VST	Virtual Structure Table
VSU	Vector and scalar unit



---

VSX	Vector-scalar extension
VTB	Virtual Time Base
WAT	Workaround trigger
WAW	Write after write
WB	Work buffer
WI	Write inject
WIMG	Storage control bits: Write-through access (W), cache-inhibited access (I), memory coherence (M), and Guarded (G).
WOF	Workload optimized frequency
WPS	Winkle Pstate
X bus	An X bus is the socket-to-socket SMP interconnect between 2 POWER9 processors.  This is not really an acronym but a name (X bus).
XER	Fixed-Point Exception Register
XIVE	External Interrupt Virtualization Engine
XMAC	XCBC-MAC-96
XPBC	Accelerator processed byte count
XSL	Address translation block
XTS	Extended translation services
XTS-AES	XOR-encrypt-xor-based tweaked-codebook mode with ciphertext stealing AES