MIPS RV64 P8700/P8700-F Multiprocessing System Programmer's Guide

Revision 1.83
April 9, 2025

MIPS Document Number: MD01502

Table of Contents

Chapter 1: Architecture Overview 11
1.1: Product Overview 12
1.1.1: Single-Cluster Configuration 13
1.1.2: Multi-Cluster Configuration 14
1.2: P8700-F Features 15
1.2.1: MIPS Out-of-Order Multithreading 15
1.2.2: Hybrid Debug 15
1.3: P8700/P8700-F Privileged Architecture 15
1.4: Functional Safety 16
1.5: System-level Features 16
1.6: CPU Core-Level Features 17
1.7: P8700-F Core Block Diagram 17
1.8: MIPS Software Tools 18
1.8.1: RISC-V Linux 18
1.8.2: MIPS RISC-V SDK 18
1.8.3: Compilers 19
1.8.4: Boot Loader 19
1.9: Performance Considerations 19
1.10: Instruction Set Architecture 19
1.10.1: RISC-V Unprivileged Architecture Extensions Implemented by the I8500 19
1.10.2: RISC-V Privileged Architecture Extensions Implemented by the I8500 20
1.10.3: RISC-V Debug Architecture Extensions Implemented by the I8500 21
1.10.4: RV64I Instruction Set Details 21
1.10.4.1: Endianess 21
1.10.4.2: misa[25:0] Extension Bits 22
1.10.4.3: A Extension 22
1.10.4.4: F and D Extension 22
1.10.4.5: Zicntr Extension 22
1.10.4.6: Zihintpause and Zawrs Extensions 22
1.10.4.7: Zihintntl Extension 22
1.10.4.8: Zkt Extension 23
1.10.4.9: Zfa Extension 23
1.10.4.10: Zicbom Extension 23
1.10.4.11: Zicbop Extension 23
1.10.4.12: Zicboz Extension 23
1.10.4.13: Svpbmt Extension 24
1.10.4.14: Rationale 24
1.10.4.15: Svinval Extension 24
1.11: Additional Information 24

Chapter 2: Memory Management Unit 25
2.1: Overview 25
2.1.1: TLB Types 25
2.1.2: Instruction TLB (ITLB) 26
2.1.3: Data TLB (DTLB) 27
2.1.4: Variable Page Size TLB (VTLB) 27
2.1.4.1: VTLB Organization 27
2.1.5: Fixed Page Size TLB (FTLB) 27
2.2: TLB ECC Errors 28
2.3: MIPS TLB Exception Handling 28

2.4: TLB Duplicate Entries ... 28
2.5: TLB Instructions ... 28
2.6: Shared FTLB Translations ... 28
2.7: Hardware Table Walker ... 29
2.8: MMU Programming ... 29

Chapter 3: Caches ... 30
3.1: Cache Configurations ... 30
3.2: LR and SC Instruction Considerations ... 31
3.3: Cache Subsystem Overview ... 31
3.3.1: L1 Instruction Cache ... 32
3.3.1.1: Level 1 Instruction Cache Error Detection ... 33
3.3.1.2: L1 Instruction Cache Organization ... 33
3.3.1.3: L1 Instruction Cache Error Types ... 33
3.3.1.4: L1 Instruction Cache Replacement Policy ... 33
3.3.1.5: L1 Instruction Cache Coherency Management ... 34
3.3.1.6: FENCE.I Instruction Usage ... 34
3.3.2: L1 Data Cache ... 34
3.3.3: Level 1 Data Cache Error Checking and Correction (ECC) ... 36
3.3.3.1: L1 Data Cache Organization ... 37
3.3.3.2: L1 Data Cache Load/Store Operations ... 37
3.3.3.3: L1 Data Cache Error Types ... 37
3.3.3.4: Store Operations Less than 32-bits ... 37
3.3.3.5: Examples of L1 Data Cache ECC Errors ... 37
3.3.4: L1 Data Cache Replacement Policy ... 38
3.3.5: L1 Data Cache Memory Coherence Protocol ... 38
3.3.6: Load/Store Bonding ... 39
3.3.7: L2 Cache ... 39
3.3.8: L2 Cache General Features ... 40
3.3.9: Overview of the AXI Interface ... 40
3.3.9.1: AXI Channels ... 40
3.3.9.2: Read Operations ... 41
3.3.9.3: Write Operations ... 41
3.3.9.4: AXI Memory Bus Ordering ... 41
3.3.10: Cache Instructions ... 41
3.4: Cache Coherency ... 42
3.5: L2 Cache Initialization Options ... 42
3.5.1: Automatic Hardware Cache Initialization ... 42
3.6: L2 Cache Flush, Burst, and Abort ... 43
3.6.1: L2 Cache Flush ... 43
3.6.2: L2 Cache Burst Operations ... 43
3.6.3: Abort Operations ... 44

Chapter 4: Exceptions and Interrupts ... 45
4.1: Exception Conditions ... 45
4.2: Selecting the Exception Address ... 46
4.3: Debug Exception Processing ... 46

Chapter 5: Coherence Manager ... 47
5.1: CM Overview ... 47
5.1.1: CM Interface — Register Ring Bus and Device ID's ... 47
5.1.2: Cluster to Cluster Accesses ... 50
5.2: Verifying Overall System Configuration ... 51
5.3: Programming the Base Addresses in Memory ... 52
5.3.1: CM GCR Register Interface ... 52
5.4: CM Register Access Permissions ... 52

5.4.1: Enabling Access Permissions 52
5.5: Coherency Enable 52
5.6: L2 Cache Prefetch 52
5.6.1: Prefetch Enable 53
5.6.2: Select Ports for L2 Prefetching 53
5.6.3: Enabling Code Prefetch 53
5.7: CM Uncached Semaphore Management 53
5.8: Custom GCR Implementation 54
5.9: IOCU Interface 54
5.10: MMIO Address Regions 55
5.10.1: CM GPR Register Interface 55
5.10.2: MMIO Region Control 55
5.11: Auxiliary Interfaces 56
5.12: Error Processing 57
5.12.1: Error Codes 1 and 3 — Tag ECC Error 59
5.12.1.1: Command Group Field Encoding 60
5.12.1.2: CCA Field Encoding 64
5.12.1.3: Type Field Encoding 65
5.12.2: Error Codes 1 and 3 — Data ECC Error 65
5.12.3: Error Code 2 — Request Decode Error 67
5.12.4: Error Code 4 — Parity Error 69
5.12.5: Error Code 5 — Fetch and Lock Error 70
5.12.6: Error Codes 6, 7, 8 — Bus Interface Unit (BIU) Errors 72
5.12.7: Error Code 10 — Ring Bus Error 73
5.12.8: Error Code 11 — IOCU Request Error 75
5.12.9: Error Code 12 — IOCU Parity Error 76
5.12.10: Error Code 13 — IOCU Response Error 76
5.12.11: Error Code 15 — RBI REGTC Bus Request Error 77
5.13: Memory Mapped Registers 78
5.14: Coherence Manager (CM) Memory Mapped Registers 79
5.14.1: GCR.Global Registers 82
5.14.1.1: GCR Global Configuration Register (offset = 0x0000) 82
5.14.1.2: Global GCR_BASE Register (offset = 0x0008) 83
5.14.1.3: GCR Global Control Register (offset = 0x0010) 84
5.14.1.4: Global Revision ID Register (offset = 0x0030) 86
5.14.1.5: GCR Global Error Control (ERR_CONTROL) Register (offset = 0x0038) 87
5.14.1.6: GCR Global Error Mask (ERR_MASK) Register (offset = 0x0040) 88
5.14.1.7: GCR Global Error Cause (ERR_CAUSE) Register (offset = 0x0048) 89
5.14.1.8: GCR Global Error Address (ERR_ADDR) Register (offset = 0x0050) 90
5.14.1.9: GCR Global Error Mult (ERR_MULT) Register (offset = 0x0058) 91
5.14.1.10: GCR Global Custom Status (CUSTOM_STATUS) Register (offset = 0x0068) 92
5.14.1.11: GCR Global Interrupt Status (AIA_STATUS) Register (offset = 0x00D0) 93
5.14.1.12: GCR Global Cache Revision (CACHE_REV) Register (offset = 0x00E0) 94
5.14.1.13: GCR Global CPC Status (CPC_STATUS) Register (offset = 0x00f0) 95
5.14.1.14: GCR Global Access (ACCESS) Register (offset = 0x0120) 96
5.14.1.15: GCR Global L2 Cache Configuration (L2_CONFIG) Register (offset = 0x0130) 98
5.14.1.16: GCR Global SDB Configuration (SDB_CONFIG) Register (offset = 0x0160) 100
5.14.1.17: GCR Global IOCU Revision (IOCU_REV) Register (offset = 0x0200) 101
5.14.1.18: GCR Global DBU Revision (DBU_REV) Register (offset = 0x0208) 102
5.14.1.19: GCR Global Interrupt Controller Revision (AIA_REV) Register (offset = 0x0208) 103
5.14.1.20: GCR Global L2 RAM Configuration (L2_RAM_CONFIG) Register (offset = 0x0240) 104
5.14.1.21: GCR Global Scratch0 (SCRATCH0) Register (offset = 0x0280) 105
5.14.1.22: GCR Global Scratch1 (SCRATCH1) Register (offset = 0x0288) 106
5.14.1.23: GCR Global L2 PFT Control (L2_PFT_CONTROL) Register (offset = 0x0300) 107
5.14.1.24: GCR Global L2 PFT Control B (L2_PFT_CONTROL_B) Register (offset = 0x0308) 108
5.14.1.25: GCR Global L2 Tag Address (L2_TAG_ADDR) Register (offset = 0x0600) 109

5.14.1.26: GCR Global L2 Tag State (L2_TAG_STATE) Register (offset = 0x0608) 110
5.14.1.27: GCR Global L2 Data (L2_DATA) Register (offset = 0x0610) 111
5.14.1.28: GCR Global L2 ECC (L2_ECC) Register (offset = 0x0618) 112
5.14.1.29: GCR Global L2SM CacheOp (L2SM_COP) Register (offset = 0x0620) 113
5.14.1.30: GCR Global L2SM Tag Address CacheOp (L2SM_TAG_ADDR_COP) Register (offset = 0x0628) 114
5.14.1.31: GCR Global Semaphore (SEM) Register (offset = 0x0640) 115
5.14.1.32: GCR Global Timeout Timer Limit (TIMEOUT_TIMER_LIMIT) Register (offset = 0x0650) 116
5.14.1.33: GCR Global MMIO Requests Limit (MMIO_REQ_LIMIT) Register (offset = 0x06F8) 117
5.14.1.34: GCR Global MMIO0 Bottom (MMIO0_BOTTOM) Register (offset = 0x0700) 118
5.14.1.35: GCR Global MMIO0 Top (MMIO0_TOP) Register (offset = 0x0708) 120
5.14.1.36: GCR Global MMIO1 Bottom (MMIO1_BOTTOM) Register (offset = 0x0710) 121
5.14.1.37: GCR Global MMIO1 Top (MMIO1_TOP) Register (offset = 0x0718) 123
5.14.1.38: GCR Global MMIO2 Bottom (MMIO2_BOTTOM) Register (offset = 0x0720) 124
5.14.1.39: GCR Global MMIO2 Top (MMIO2_TOP) Register (offset = 0x0728) 126
5.14.1.40: GCR Global MMIO3 Bottom (MMIO3_BOTTOM) Register (offset = 0x0730) 127
5.14.1.41: GCR Global MMIO3 Top (MMIO3_TOP) Register (offset = 0x0728) 129
5.14.1.42: GCR Global MMIO4 Bottom (MMIO4_BOTTOM) Register (offset = 0x0740) 130
5.14.1.43: GCR Global MMIO4 Top (MMIO4_TOP) Register (offset = 0x0748) 132
5.14.1.44: GCR Global MMIO5 Bottom (MMIO5_BOTTOM) Register (offset = 0x0750) 133
5.14.1.45: GCR Global MMIO5 Top (MMIO5_TOP) Register (offset = 0x0758) 135
5.14.1.46: GCR Global MMIO6 Bottom (MMIO6_BOTTOM) Register (offset = 0x0760) 136
5.14.1.47: GCR Global MMIO6 Top (MMIO6_TOP) Register (offset = 0x0768) 138
5.14.1.48: GCR Global MMIO7 Bottom (MMIO7_BOTTOM) Register (offset = 0x0770) 139
5.14.1.49: GCR Global MMIO7 Top (MMIO7_TOP) Register (offset = 0x0778) 141
5.14.2: GCR.Debug Registers 142
5.14.2.1: GCR Debug TCB ControlD (TCBCONTROLD) Register (offset = 0x0810) 142
5.14.2.2: GCR Debug TCB ControlE (TCBCONTROLE) Register (offset = 0x0820) 144
5.14.2.3: GCR Debug TCB Performance Counter Trace (TCBPERFCNTR) Register (offset = 0x0830) 145
5.14.2.4: GCR Debug Performance Counter Control (PC_CTL) Register (offset = 0x0900) 146
5.14.2.5: GCR Debug Performance Counter Overflowed (PC_OV) Register (offset = 0x0920) 148
5.14.2.6: GCR Debug Performance Counter Event (PC_EVENT) Register (offset = 0x0930) 149
5.14.2.7: GCR Debug Performance Counter Cycles (PC_CYCL) Register (offset = 0x0980) 150
5.14.2.8: GCR Debug Performance Counter Qualifier0 (PC_QUAL0) Register (offset = 0x0990) 151
5.14.2.9: GCR Debug Performance Counter Value0 (PC_CNT0) Register (offset = 0x0998) 152
5.14.2.10: GCR Debug Performance Counter Qualifier1 (PC_QUAL1) Register (offset = 0x09A0) . 153
5.14.2.11: GCR Debug Performance Counter Value1 (PC_CNT1) Register (offset = 0x09a8) 154
5.14.3: GCR.Core Registers 155
5.14.3.1: GCR HART Reset Exception Base (RESET_BASE) Register (offset = see below) 157
5.14.3.2: GCR Core Enables Coherence (COH_EN) Register (offset = see below) 158
5.14.4: CPC.Global Registers 159
5.14.4.1: CPC Global Sequencer (SEQDEL_REG) Register (offset = 0x8008) 160
5.14.4.2: CPC Global Rail (RAIL_REG) Register (offset = 0x8010) 161
5.14.4.3: CPC Global Reset Sequence (RESETLEN_REG) Register (offset = 0x8018) 162
5.14.4.4: CPC Global Revision (REVISION_REG) Register (offset = 0x8020) 163
5.14.4.5: CPC Global Clock Change Configuration, Control and Status. (CC_CTL_REG) Register (offset = 0x8028) 164
5.14.4.6: CPC Global Power Up Control (PWRUP_CTL_REG) Register (offset = 0x8030) 166
5.14.4.7: CPC Global Reset Release (RES_REL_REG) Register (offset = 0x8038) 167
5.14.4.8: CPC Global Core Rest Control (ROCC_CTL_REG) Register (offset = 0x8040) 168
5.14.4.9: CPC Global Controls Prescale Clock Changes Register (offset = 0x8048) 169
5.14.4.10: CPC Global RISC-V Mtime (MTIME_REG) Register (offset = 0x8050) 170
5.14.4.11: CPC Global RISC-V Mtime Control (TIMECTL_REG) Register (offset = 0x8058) 171
5.14.4.12: CPC Global Clock Gate Disabled (CLK_GATE_DIS_REG) Register (offset = 0x8060) .. 172
5.14.4.13: CPC Global Fault Status (FAULT_STATUS) Register (offset = 0x8068) 173

5.14.4.14: CPC Global Fault Supported (FAULT_SUPPORTED) Register (offset = 0x8070) 174
5.14.4.15: CPC Global Fault Enable (FAULT_ENABLE) Register (offset = 0x0078) 176
5.14.4.16: CPC Global High Resolution Timer (HRTIME_REG) Register (offset = 0x8090) 178
5.14.4.17: CPC Global Configuration (CONFIG) Register (offset = 0x8138) 179
5.14.5: CPC.Core Registers 180
5.14.5.1: CPC Power Command (CMD_REG) Register (offset = see below) 182
5.14.5.2: CPC Core Status and Domain Configuration (STAT_CONF_REG) Register (offset = see below) 183
5.14.5.3: CPC Control Clock Change (CC_CTL_REG) Register (offset = see below) 186
5.14.5.4: CPC Power VP Stop (VP_STOP_REG) Register (offset = see below) 188
5.14.5.5: CPC VP Run (VP_RUN_REG) Register (offset = see below) 189
5.14.5.6: CPC VP Running State (VP_RUNNING_REG) Register (offset = see below) 190
5.14.5.7: CPC Power Debug Interrupt (DBG_DBRK_REG) Register (offset = see below) 191
5.14.5.8: CPC Power Controls Deep Sleep (RAM_SLEEP_REG) Register (offset = see below) 192
5.14.5.9: CPC Power Fault Status (FAULT_STATUS) Register (offset = see below) 193
5.14.5.10: CPC Power Fault Set (FAULT_SET) Register (offset = see below) 195
5.14.5.11: CPC Power Fault Clear (FAULT_CLR) Register (offset = see below) 196
5.14.5.12: CPC Power Configuration (CONFIG) Register (offset = see below) 197
5.14.6: FDC.Global Registers 198
5.14.6.1: FDC Global Access Control and Status (FDACSR) Register (offset = 0x3F000) 199
5.14.6.2: FDC Global Configuration (FDCFG) Register (offset = 0x3F008 200
5.14.6.3: FDC Global Status (FDSTAT) Register (offset = 0x3F010) 201
5.14.6.4: FDC Global Receive (FDRX) Register (offset = 0x3F018) 202
5.14.6.5: FDC Global Transmit (FDTX[0-15]) Register (offset = 0x3F020) 203
5.14.7: Trace Funnel (TRF) Global Registers 204
5.14.7.1: TRF Global Trace Funnel Control (CONTROL) Register (offset = 0x3F100) 205
5.14.7.2: TRF Global Trace Funnel Configuration (CONFIG) Register (offset = 0x3F108) 208
5.14.7.3: TRF Global Trace Funnel Write Pointer (WRITEPTR) Register (offset = 0x3F110) 209
5.14.7.4: TRF Global Trace Funnel Read Pointer (READPTR) Register (offset = 0x3F118) 210
5.14.7.5: TRF Global Trace Data (DATA[0-7]) Register (offset = see below) 211
5.14.7.6: TRF Global System Trace User Control (STUSER) Register (offset = 0x3F160) 212
5.14.7.7: TRF Global System Trace Enable (STENABLE) Register (offset = 0x3F168) 213
5.14.8: GCR.U User Mode Registers 214
5.14.8.1: GCR.U User Mode Timer (MTIME_REG) Register (offset = 0x7F050) 215
5.14.8.2: GCR.U High Resolution Timer (L2_CONFIG) Register (offset = 0x7F090) 216

Chapter 6: Power Management 217
6.1: Overview 217
6.1.1: Power Domains 217
6.1.2: Clock Domains 218
6.1.3: Core and IOCU Selection 218
6.1.4: Overview of Power States 218
6.2: Individual Clock Gating 219
6.3: Global Control Block Register Map 220
6.4: Local Control Blocks 221
6.5: CPC Register Programming 222
6.5.1: Cluster Power Controller Register Address Map 222
6.5.2: Global Control Block Register Map 223
6.5.3: Local Control Blocks 223
6.5.4: Requestor Access to CPC Registers 223
6.5.4.1: Register Interface 223
6.5.5: Enabling Coherent Mode 223
6.5.6: Master Clock Prescaler 224
6.5.7: Individual Device Clock Ratio Modification 225
6.5.7.1: Clock Domain Change Example — Register Programming Sequence 225
6.5.7.2: Clock Change Delay 226

6.5.8: CM Standalone Powerup 226
6.5.8.1: Register Interface 226
6.5.9: Reset Detection 226
6.5.10: VP Run/Suspend 227
6.5.11: Local RAM Deep Sleep / Shutdown and Wakeup Delay 228
6.5.11.1: RAM Deep Sleep Mode 228
6.5.11.2: RAM Shut Down Mode 228
6.5.12: Fine Tuning Internal and External Signal Delays 229
6.5.12.1: Global Sequence Delay Count 229
6.5.12.2: Rail Delay 229
6.5.12.3: Reset Delay 230

Chapter 7: Control and Status Registers (CSR) 232
7.1: Machine Mode Registers 232
7.1.1: Machine Architecture ID Register (MarchID) — offset = 0xF12 233
7.1.2: Machine Cause Register (mcause) — offset = 0x342 234
7.1.3: Machine Hart ID Register (mhartID) — 0xF14 237
7.1.4: Machine Implementation ID Register (mimpid) — offset = 0xF13 238
7.1.5: Machine Vendor ID Register (mvendorid) — offset = 0xF11 239
7.2: User Mode Registers 240
7.2.1: Time Register (time) — offset = 0xC01 240
7.3: MIPS Custom Registers 241
7.3.1: MIPS Trap Vector Base Address Register (mipstvec) — offset = 0x7C0 243
7.3.2: MIPS Trap Value Register (mipstval) — offset = 0x7C3 244
7.3.3: MIPS Scratch Register (mipsscratch) — offset = 0x7C4 245
7.3.4: MIPS Cache Error Register (mipscacheerr) — offset = 0x7C5 246
7.3.5: MIPS Error Control Register (mipserrctrl) — offset = 0x7C6 248
7.3.6: MIPS Interrupt Control Register (mipsintctl) — offset = 0x7CB 249
7.3.7: MIPS DSPRAM Base Register (mipsdsprambase) — offset = 0x7CC 250
7.3.8: MIPS Configuration 1 Register (mipsconfig1) — offset = 0x7D1 251
7.3.9: MIPS Configuration 5 Register (mipsconfig5) — offset = 0x7D5 252
7.3.10: MIPS Configuration 6 Register (mipsconfig6) — offset = 0x7D6 254
7.3.11: MIPS Configuration 7 Register (mipsconfig7) — offset = 0x7D7 255
7.3.12: MIPS Configuration 8 Register (mipsconfig8) — offset = 0x7D8 258
7.3.13: MIPS Configuration 9 Register (mipsconfig9) — offset = 0x7D9 259
7.3.14: MIPS Configuration 10 Register (mipsconfig10) — offset = 0x7DA 260
7.3.15: MIPS Configuration 11 Register (mipsconfig11) — offset = 0x7DB 261
7.3.16: PMA Configuration Registers 262
7.3.16.1: PMA Configuration 0 Control and Status Register (PMACFG0) — offset = 0x7E0 263
7.3.16.2: PMA Configuration 1 Control and Status Register (PMACFG1) — offset = 0x7E1 264
7.3.16.3: PMA Configuration 2 Control and Status Register (PMACFG2) — offset = 0x7E2 265
7.3.16.4: PMA Configuration 3 Control and Status Register (PMACFG3) — offset = 0x7E3 266
7.3.16.5: PMA Configuration 4 Control and Status Register (PMACFG4) — offset = 0x7E4 267
7.3.16.6: PMA Configuration 5 Control and Status Register (PMACFG5) — offset = 0x7E5 268
7.3.16.7: PMA Configuration 6 Control and Status Register (PMACFG6) — offset = 0x7E6 269
7.3.16.8: PMA Configuration 7 Control and Status Register (PMACFG7) — offset = 0x7E7 270
7.3.16.9: PMA Configuration 8 Control and Status Register (PMACFG8) — offset = 0x7E8 271
7.3.16.10: PMA Configuration 9 Control and Status Register (PMACFG9) — offset = 0x7E9 272
7.3.16.11: PMA Configuration 10 Control and Status Register (PMACFG10) — offset = 0x7EA 273
7.3.16.12: PMA Configuration 11 Control and Status Register (PMACFG11) — offset = 0x7EB 274
7.3.16.13: PMA Configuration 12 Control and Status Register (PMACFG12) — offset = 0x7EC 275
7.3.16.14: PMA Configuration 13 Control and Status Register (PMACFG13) — offset = 0x7ED 276
7.3.16.15: PMA Configuration 14 Control and Status Register (PMACFG14) — offset = 0x7EE 277
7.3.16.16: PMA Configuration 15 Control and Status Register (PMACFG15) — offset = 0x7EF 278
7.3.16.17: PMA0CFG - PMA63CFG Bit Assignments 279
7.4: MIPS Hybrid Debug Registers 280

Chapter 8: Interrupt Controller 281
8.1: Overview 281
8.1.1: Multi-cluster Support 281
8.1.2: APLIC 282
8.1.3: ACLINT 283
8.1.3.1: MTIMER 283
8.1.3.2: MSWI and SSWI 284
8.1.4: Watchdog Timer 284
8.2: ACLINT Memory Mapped Registers 285
8.2.1: ACLINT Machine Mode Memory Map 285
8.2.1.1: ACLINT Machine Software Interrupt Pending (MSIP[0-4094]) Register (offset = see below) 286
8.2.1.2: ACLINT Machine Time Compare (MTIMECMP[0-4094]) Register (offset = see below) 287
8.2.1.3: ACLINT WatchDog ConFiG (WDCFG[0-1023]) Register (offset = see below) 288
8.2.1.4: ACLINT WatchDog Control and Status (WDCSR[0-1023]) Register (offset = see below) ... 289
8.2.2: Aclint Supervisor Mode Memory Map 290
8.2.2.1: ACLINT SET Supervisor Software Interrupt Pending (SETSSIP[0-4094]) Register (offset = see below) 291
8.3: APLIC Memory Mapped Registers 292
8.3.1: APLIC Machine Domain Memory Map 292
8.3.2: APLIC Supervisor Domain Memory Map 294
8.3.3: APLIC Custom Memory Map 296
8.3.3.1: APLIC Domain Configuration (DOMAINCFG) Register (offset = see below) 297
8.3.3.2: APLIC Source Configuration (SOURCECFG[1-1023]) Register (offset = see below) 298
8.3.3.3: APLIC SET Interrupt Pending (SETIP[0-31]) Register (offset = see below) 299
8.3.3.4: APLIC Input/Clear Interrupt Pending (IN_CLRIP[0-31]) Register (offset = see below) 300
8.3.3.5: APLIC Set Interrupt-Pending Number (SETIPNUM) Register (offset = see below) 301
8.3.3.6: APLIC Clear IP Number (CLRIPNUM) Register (offset = see below) 302
8.3.3.7: APLIC Set Interrupt Enable (SETIE[0-31]) Register (offset = see below) 303
8.3.3.8: APLIC Clear Interrupt Enable (CLRIE[0-31]) Register (offset = see below) 304
8.3.3.9: APLIC Set Interrupt Enable Number (SETIENUM) Register (offset = see below) 305
8.3.3.10: APLIC Clear Interrupt Enable Number (CLRIENUM) Register (offset = see below) 306
8.3.3.11: APLIC Set Interrupt-Pending Number (SETIPNUM_LE) Register (offset = see below) 307
8.3.3.12: APLIC Target (TARGET[1-1023]) Register (offset = see below) 308
8.3.3.13: APLIC Interrupt Delivery (HART[0-1023].IDELIVERY) Register (offset = see below) 309
8.3.3.14: APLIC Interrupt Force (HART[0-1023].IFORCE) Register (offset = see below) 310
8.3.3.15: APLIC Interrupt Threshold (HART[0-1023].ITHRESHOLD) Register (offset = see below) 311
8.3.3.16: APLIC Top Interrupt (HART[0-1023].TOPI) Register (offset = see below) 312
8.3.3.17: APLIC Claim Interrupt (HART[0-1023].CLAIMI) Register (offset = see below) 313
8.3.3.18: APLIC Set NMI Enable (SETNMIE[0-31]) Register (offset = see below) 314
8.3.3.19: APLIC Set NMI Number (SETNMIENUM) Register (offset = 0x4C0DC) 315
8.3.3.20: APLIC Clear NMI Enable (CLRNMIE[0-31]) Register (offset = see below) 316
8.3.3.21: APLIC Clear NMI Number (CLRNMIENUM) Register (offset = 0x4C1DC) 317

Chapter 9: Floating-Point Unit (FPU) 318
9.1: Features Overview 318
9.2: FPU Execution Units 318
9.2.1: Short Operations 318
9.2.2: Long Operations 319
9.3: Data Formats 319
9.3.1: Floating-Point Formats 319
9.3.1.1: Normalized and Denormalized Numbers 321
9.3.1.2: Reserved Operand Values—Infinity and NaN 321
9.3.1.3: Infinity and Beyond 321
9.3.1.4: Signalling Non-Number (SNaN) 322
9.3.1.5: Quiet Non-Number (QNaN) 322

9.3.2: Signed Integer Formats ... 323
9.4: Floating-Point General Registers ... 323
9.4.1: FPRs and Formatted Operand Layout ... 323

Chapter 10: Multithreading ... 324
10.1: Instruction Flow ... 324
10.2: Data Flow ... 325
10.3: Independent Exception Model ... 325

Chapter 11: Performance Counters ... 327
11.1: Core Performance Counters ... 327
11.1.1: Performance Event Masking ... 327
11.1.2: Core Performance Event Control Register (mhpmevent[6:3]) ... 328
11.1.3: Core Performance Counter Count Register (mhpmcounter[6:3]) ... 329
11.1.4: Core Performance Counter Events ... 329
11.2: CM3 Performance Counters ... 332
11.2.1: CM3 Performance Counter Functionality ... 332
11.2.2: CM3 Performance Counter Usage Models ... 333

Chapter 12: Instruction Latencies and Repeat Rates ... 344
12.1: Definition of Terms ... 344

Chapter 13: MIPS On-Chip Instrumentation ... 351
13.1: OCI Debug System Overview ... 351
13.1.1: Debug Unit (DBU) ... 351
13.1.1.1: APB Slave Port ... 353
13.1.1.2: JTAG TAP ... 353
13.1.1.3: Debug Monitor ... 353
13.1.1.4: RAM ... 353
13.1.2: Register Bus ... 353
13.1.3: Number of Breakpoints ... 353
13.1.4: Per Core/Hart Resources ... 353
13.1.4.1: Breakpoint Controller ... 353
13.1.4.2: Dseg ... 353
13.1.4.3: Dmseg ... 353
13.1.4.4: Drseg ... 353
13.1.4.5: CSR Registers ... 354
13.1.5: Coherence Devices ... 354
13.1.5.1: CPC (Cluster Power Controller) ... 354
13.1.5.2: GCR (Global Configuration Registers) ... 354
13.1.5.3: CGCR - (Custom Global Configuration Registers) ... 354
13.1.5.4: CM - (Coherence Manager) (v3) ... 354
13.1.5.5: IOCU (I/O Coherence Unit) ... 354
13.2: More Information ... 354

Appendix A: Revision History ... 355

Appendix B: MIPS Defined Instructions ... 358

Chapter 1

Architecture Overview

This document describes the software-programmable aspects of the 64-bit RV64GCZba_Zbb[1] P8700-F Multiprocessing System (MPS). The device consists of the logic blocks shown in Figure 1.1. The majority of blocks in the diagram have at least one dedicated chapter that describes how to control the hardware using registers and assembly code. The register-programming examples describe a programming sequence of how to set or change a programmable parameter using registers. The assembly code examples show how the MIPS instruction set can be used to perform the same function.

Each chapter provides the relevant background information required by the programmer in order to understand the examples. Common examples such as enabling and initialization are provided for each block, as well as more in depth examples relative to that block.

An overview of the material provided in this document is as follows:

- *Memory Management (MMU)*: This chapter describes the programmable elements of the Translation Lookaside Buffer or TLB of the P8700-F Multiprocessing System. The first section gives an overview of the TLB architecture, a description of its functionality and a description of the elements that go into programming the TLB. The sections that follow cover specific information on programming for the Translation Lookaside Buffer (TLB).
- *Caches*: This chapter provides an overview of the cache architecture, a description of its functionality, and a description of the elements that go into programing the caches. A description of the CSR register interface to each cache is provided, as well as initialization code for all three caches, setting up cache coherency, handling cache exceptions, and testing the cache RAM.
- *Exceptions*: This chapter describes an overview of exception processing and a definition of the interrupt modes. Information on how to program the reset, boot, and general exception vectors in memory is also covered. A list of exception priorities is provided, along with an assembly language example of an exception handler.
- *Coherence Manager* (CM): The P8700-F MPS contains a third generation Coherence Manager. This chapter provides an overview of the CM register ring bus and associated table that lists each device ID on the bus. The programmer uses this information to access these devices. An overview of the CM register address space is also provided. In addition, the chapter describes how to program the CM to perform various functions, including setting the base addresses in memory, accessing another Hart in the same core, accessing a Hart in another core, accessing the Interrupt Controller, Cluster Power Controller (CPC), and/or Debug Unit (DBU) registers via the CM, and setting the clock ratios between the various P8700-F system components. For the exact revision number of the Coherence Manager, refer to the Release Notes.

1. Zba and Zbb are bit manipulation extensions.

This chapter also introduces the multi-cluster configuration that allows multiple P8700/P8700-F Multiprocessing Systems to be connected through a Network-On-Chip (NOC) interface. The section describes the registers used to perform a cluster-to-cluster access.

- *Cluster Power Controller (CPC)*: This chapter provides an overview of how power is managed in the P8700-F Multiprocessing System and identifies the various power and clock domains the programmer can use to manage power consumption in the device. In addition, a procedure on how to set the CPC base address in memory is provided. Other programming principles include setting the device to coherent or non-coherent mode, requestor (core or IOCU) access of CPC registers, system power-up policy, programming examples of a clock domain change and clock delay change, powering up the CPC in standalone mode (no cores enabled), reset detection, Hart run/suspend mechanism, local RAM shutdown and wake-up procedure, accessing registers in another power domain, and fine tuning internal and external signal delays to help the programmer easily integrate the device into a system environment.
- *Interrupt Controller*: The Interrupt Controller processes internal and external interrupts in the P8700/P8700-F Multiprocessing System. It supports up to 512 external interrupts (configurable in multiples of 8), which are prioritized and routed to the selected hart for servicing. The Interrupt Controller is compliant with the Advanced Interrupt Architecture (AIA) standard.
- *Floating Point Unit (FPU)*: This chapter provides information on how to enable the FPU, how to handle floating point exceptions, how to set the rounding mode, and operation of the Flush-to-Zero (FS) function.
- *Multi-threading*: This chapter provides an overview of the hardware multi-threading mechanism in the P8700-F MPS.
- *On-Chip Instrumentation (OCI)*: This chapter provides a brief overview of the interface and external debugging environment required to debug MIPS processors that incorporate the MIPS On-Chip Instrumentation (OCI) debug system for multi-core designs.
- *Performance Counters*: The P8700 core contains four performance counters. Each counter has a Control register (mhpmevent) and an associated Count (mhpmcounter) register.
- *Implementation Specific Instructions*: This chapter describes the architectural definition for the following implementation-specific instructions in the P8700/P8700-F Multiprocessing System.
- *Latency and Repeat Rates*: This chapter provides the instructions latency and repeat rates

Throughout all of the aforementioned chapters, there are assembly language examples that describe how various programming elements are handled in software. These examples can be used by programmer's writing their own code to program a particular block, or for writing a low-level support library, RTOS, or their own tool chain.

This document is meant to be used with two other companion documents:

- *MIPS RV64 P8700-F Multiprocessing System Integrator's Guide* (MD01041), This companion document provides hardware details about the device, including functional verification, system integration, and system implementation.

1.1 Product Overview

The P8700-F series is a high performance multi-core microprocessor system that provides a best in class power efficiency for use in system-on-chip (SoC) applications. The P8700-F

Coherence Manager maintains Level 2 (L2) cache and system level coherency between all cores, main memory, and I/O devices. The P8700-F Multiprocessing System (MPS) can be configured with a variable number of cores, I/O coherent interfaces, and L2 cache size.

Each P8700-F core implements the Release 6 of the RV64GCZba_ZbbZba_Zbb Instruction Set Architecture with full hardware multithreading.

The P8700-F MPS supports both single-cluster and multi-cluster configurations as described in the following subsections.

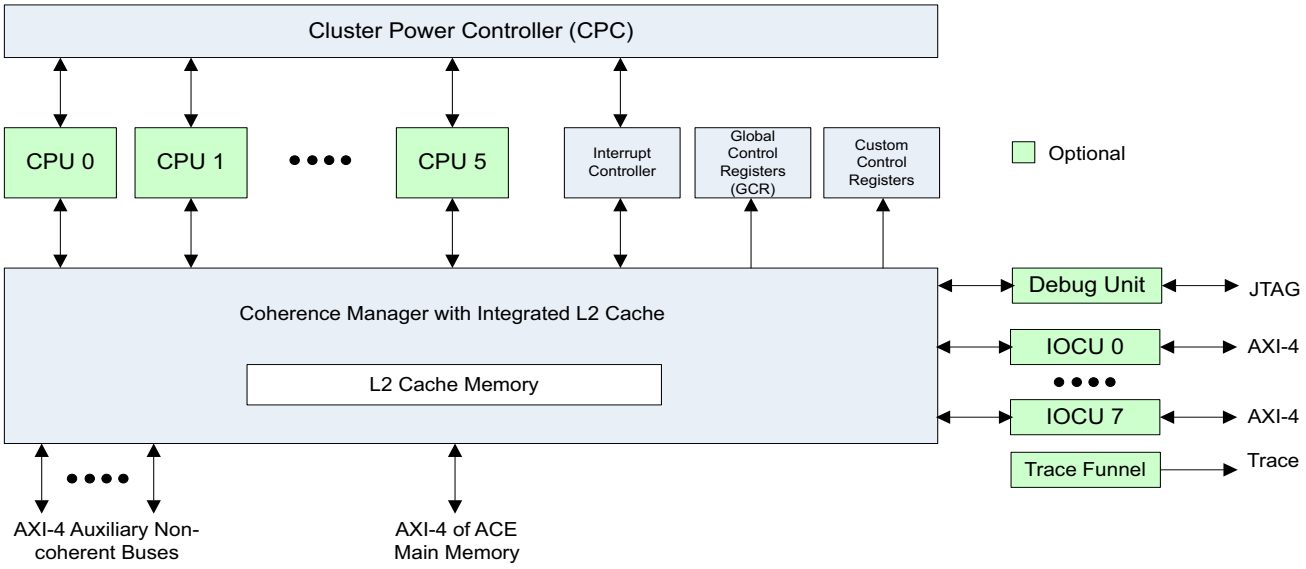
1.1.1 Single-Cluster Configuration

Figure 1.1 shows a block diagram of a single-cluster P8700-F Multiprocessing System. The P8700-F MPS contains the following logic blocks:

- Up to six cores
- Up to eight I/O Coherence Units (IOCU)
- Coherence Manager (CM) with integrated L2-cache
- Cluster Power Controller (CPC)
- Interrupt Controller
- Global Configuration Registers (GCR)
- Multiprocessor debug via in-system Debug Unit (DBU)

In the P8700-F MPS the total number of cores and IOCUs together must be less than or equal to eight. All cores and IOCU's are optional and can be configured in any combination of up to eight.

Figure 1.1 Block Diagram of Single Cluster P8700/P8700-F Multiprocessing System



For more information on the *Cluster Power Controller (CPC)* block, refer to the *Cluster Power Controller* chapter of this manual.

For more information on the *Interrupt Controller* block, refer to the *Interrupt Controller* chapter of this manual.

For more information on the *Coherence Manager (CM)*, refer to the *Coherence Manager* chapter of this manual.

For more information on the *L2 Cache Memory*, refer to the *Caches* chapter of this manual.

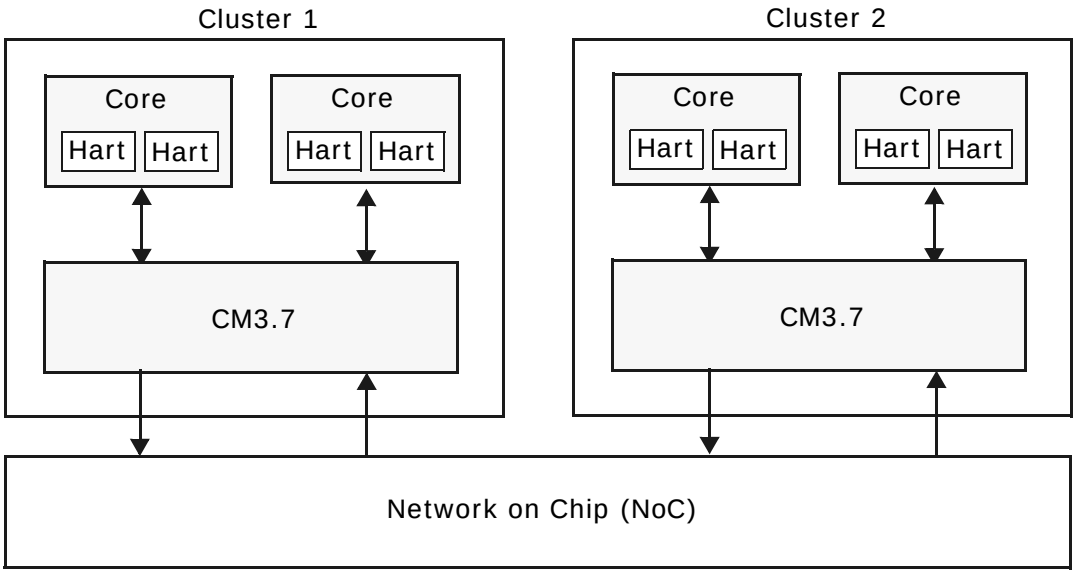
For more information on the *Global Configuration Registers* block, refer to the *Coherence Manager* chapter of this manual to provide various CM register programming examples.

For more information on the programmable blocks within the *Core*, such as *MMU*, *FPU*, etc. refer to the Figure 1.3.

1.1.2 Multi-Cluster Configuration

In addition to the single-cluster configuration shown above, the P8700-F also allows for cluster-to-cluster accesses. This allows a core or Hart in one cluster to access a core or Hart in another cluster through the Network-On-Chip (NOC) interface. This interface is shown in Figure 1.2.

Figure 1.2 Cluster-to-Cluster Accesses Using the NOC



For example, a Hart within a core in Cluster 1 can access and update a register in a Hart in Cluster 2 as shown. The access is processed by the CM3.7 and driven onto the NOC. The NOC then routes the request to the appropriate cluster where the access is scheduled by the CM3.7 in the destination cluster. The data is fetched and returned to the requesting Hart through the NOC.

For more information, refer to Chapter 4, Coherency Manager.

1.2 P8700-F Features

The P8700/P8700-F Multiprocessing System has four key architectural features as described in the following subsections.

- RISC-V RV64GCZba_Zbb architecture (Base ISA and Standard Extensions)
- User-Defined Custom Extensions
- MIPS Multithreading
- Hybrid Debug
- RISC-V Privileged Architecture
- Functional Safety

The P8700/P8700-F core is configured to support the RV64GCZba_Zbb (G = IMAFD) Standard ISA. It includes the RV64I base ISA, Multiply (M), Atomic (A), Single-Precision Floating Point (F), Double-Precision Floating Point (D), Compressed (C) RISC-V extensions, as well as the bit-manipulation extensions (Zba) and (Zbb).

The P8700/P8700-F provides memory management through on-chip configuration registers and enables real-time operating systems and application code to be implemented once and then reused.

1.2.1 MIPS Out-of-Order Multithreading

CPU performance depends on minimizing the latency to the system memory. Even with a cache hierarchy, the CPU still stalls while waiting for data. To avoid this scenario, MIPS out-of-order multithreading provides significant performance improvements by running additional instructions concurrently.

This hardware out-of-order multithreading enables execution of multiple instructions from multiple threads (harts) every clock cycle, providing higher utilization and CPU efficiency. In this way, out-of-order multi-threading is a more area efficient alternative to the use of additional cores and offers a typical 60% performance boost for the execution of two harts simultaneously instead of sequentially.

1.2.2 Hybrid Debug

The P8700/P8700-F offers proven MIPS EJTAG with RISC-V Trace and GDB support for Multi-Core/Cluster Debug.

1.3 P8700/P8700-F Privileged Architecture

The RISC-V privileged architecture covers all aspects of RISC-V systems beyond the unprivileged ISA, including privileged instructions as well as additional functionality required for running operating systems and attaching external devices.

The P8700/P8700-F implements the RISC-V compliant Privileged Architecture, as well as Custom CSRs. The P8700/P8700-F Privileged Architecture includes:

- Privileged operating modes (Supervisor-mode, Machine-mode, Debug-mode)
- M-mode: All Machine-level CSRs and Privileged Instructions

- S-mode: All Supervisor-level CSRs and Supervisor Instructions
- D-mode: All Debug/Trace CSRs

1.4 Functional Safety

The P8700/P8700-F IP is designed to support the ASIL-B(D) functional safety standard. In so doing, the P8700/P8700-F cluster includes the following fault detection features:

- Fault bus to report detected faults to external fault handling logic
- End-to-end parity protection on address and data buses
- Parity protection of software visible registers in the GCR, Interrupt Controller, and CPC blocks
- Programmable transaction time out detection on memory requests originating from a CPU or IOCU
- SRAM error detection and correction
- Protocol error detection on IOCU and REGTC AXI slave interfaces
- AXI/ACE interface parity protection of address and data compatible with third-party interconnects

1.5 System-level Features

- Up to six coherent RV64GCZba_Zbb CPU cores
- Multi-Cluster support: Cluster composed of up to 0 - 6 CPUs and 0 - 8 IOCUs (sum being no more than 8 agents) and a Level 2 cache connection to a coherent interconnect. Support for up to 64 clusters.
- Integrated L2 cache controller supporting a 8-way and 16-way set-associativity
  - Inclusive of the L1 data caches
  - 256 KB to 8 MB cache sizes
  - Single bit correction and double bit detection
- CPC to shut down idle cores for power efficiency
- Up to 8 I/O coherence units (total of cores + IOCUs must be no greater than 8)
- Cache-to-cache data transfers
- Out-of-order data return
- Hardware L2 cache prefetch controller significantly improves performance of workloads such as memory to memory data transfer/copy (memcpy)
- Independent clock ratios on core, memory, and IOCU ports
- SoC system interface supports AXI-4 (Advanced eXtensible Interface rev. 4, also known as AMBA 4 AXI) or ACE (AXI Coherency Extensions) protocol with 48-bit address and 256-bit data paths. This interface can be configured to support up to 96 outstanding requests.
- High bandwidth 128-bit data paths between each core and the Coherence Manager
- Software controlled core level and cluster level power management
- Debug port supporting multi-core debug (JTAG/APB)
- Program and Data trace (PDtrace) mechanism to debug software

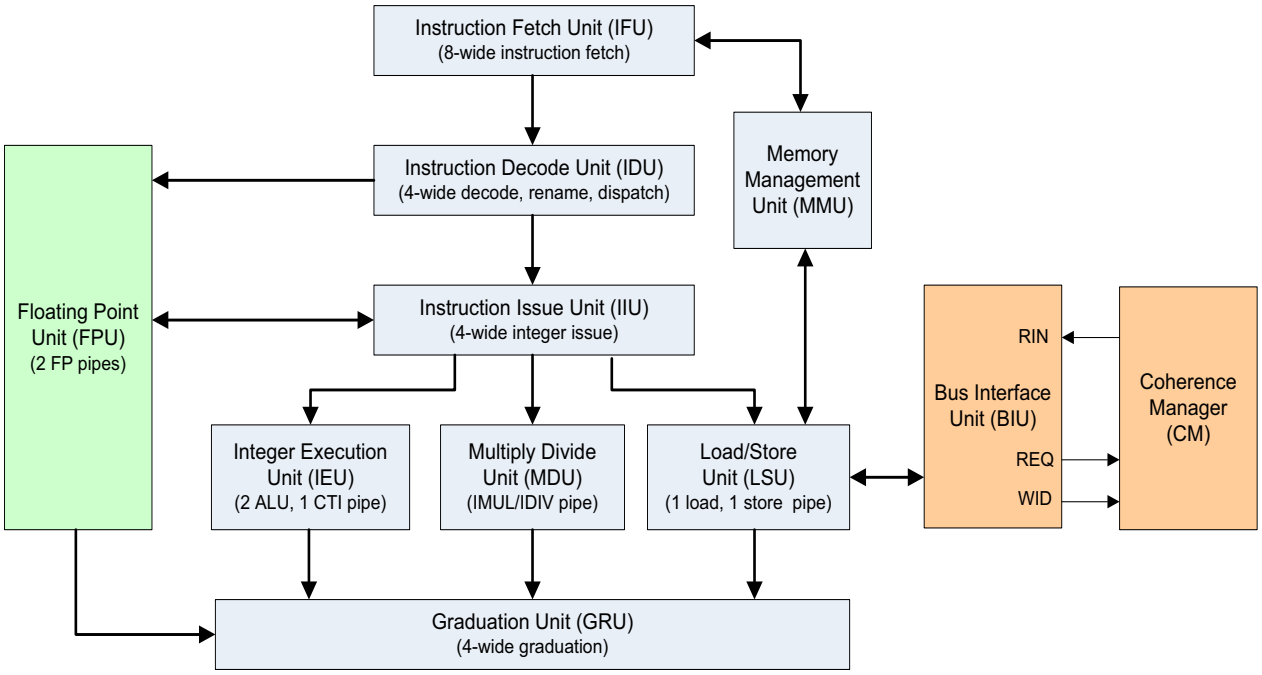
1.6 CPU Core-Level Features

- Full 64-bit Instruction Set Architecture with Compressed Instructions via RISC-V RV64GCZba_Zbb
- 48-bit virtual and physical addresses
- Power efficient design
- Quad issue instruction fetch, decode, issue, and graduate
- Hardware out-of-order multithreading
- L1 caches with Error Correction Code (ECC) protection
- L2 cache support — Implemented as shared L2 in the Coherence Manager
- Programmable Memory Management Unit with large first-level ITLB/DTLB backed by fast on-core second-level variable page size TLB (VTLB) and fixed page size TLB (FTLB)
- Shared FTLB across all hardware threads (harts) in a CPU
- MIPS DVM support through Global Instruction cache and TLB invalidation
- Load and store bonding support
- Unaligned load / store support in hardware
- Program and Data Trace (PDtrace) support for Instructions and Data (Virtual Addresses and Data Values)

1.7 P8700-F Core Block Diagram

Figure 1.3 shows a block diagram of a single P8700-F core.

Figure 1.3 P8700-F Core-level Block Diagram



For more information on the *Instruction TLB*, *Data TLB*, and *VTLB/FTLB* blocks shown in Figure 1.3, refer to the *Memory Management* chapter of this manual.

For more information on the *L1 Instruction Cache* and *L1 Data Cache* blocks, refer to the *Caches* chapter of this manual.

For more information on the *FPU* block, refer to the *FPU* chapter of this manual.

1.8 MIPS Software Tools

MIPS offers a complete portfolio of tools that address all stages of product development, including RISC-V Linux, RiscFree™ for RISC-V SDK, compilers, and MIPS boot loader. Some of the tools provided are described in the following subsections.

1.8.1 RISC-V Linux

MIPS actively supports, develops and improves the Linux kernel for the RISC-V architecture. Linux kernel and distributions that currently support the RISC-V architecture include Fedora, Debian, GENTOO, and Ubuntu.

For more information on the RISC-V Linux, refer to the RISC-V website at www.riscv.org/exchange/software.

1.8.2 MIPS RISC-V SDK

The RiscFree™ IDE is a third-party Integrated Development Environment (IDE) and Debugger for RISC-V based development.

1.8.3 Compilers

MIPS ports and maintains the GNU Compiler Collection (GCC) and provides prebuilt tool chains in the RiscFree™ for RISC-V SDK. A wide range of other industry leading compilers are also available for MIPS processors.

For more information on the MIPS Compilers, refer to the MIPS website at www.mips.com. Click on: *Developer* → *Developer Tools* → *Compilers*.

1.8.4 Boot Loader

MIPS offers a wide range of solutions for initializing MIPS cores and facilitating debugging. These include open-source and proprietary solutions to suit any requirement.

For more information on the MIPS Boot Loader, refer to the MIPS website at www.mips.com. Click on: *Developer* → *Developer Tools* → *Boot Loaders*.

1.9 Performance Considerations

MIPS recommends setting the following configuration registers to achieve maximum performance in most scenarios:

- Enable speculation bit in pmacfg register for all harts
- Enable L1 Prefetching by writing 0xFF to mipsconfig11 for all cores

1.10 Instruction Set Architecture

The Shogun CPU core implements the RISC-V RV64GCH base architecture. In addition to this base architecture, the I8500 CPU implements multiple MIPS Technologies custom extensions, including CorExtend. CorExtend provides customers the ability to add new compute capability to the I8500 in a well-defined manner. The tables in the following sections list the features and extensions supported by the I8500.

1.10.1 RISC-V Unprivileged Architecture Extensions Implemented by the I8500

Table 1.1 lists the supported extensions for the RISC-V unprivileged architecture.

Table 1.1 RISC-V Unprivileged Architecture 20240411 + RVB23U64 v0.1 Summary

Name	Version	Description
RV64I	2.1	Base Integer Instruction Set
Zicclsm	RVB23	Misaligned load/store
Zifencei	2.0	Instruction-Fetch Fence
Ziccif	RVB23	Atomic instruction fetch up to 32 bits
Zicsr	2.0	Control and Status Register Instructions
Zicntr	2.0	Base Counters and Timers
Zihpm	2.0	Hardware Performance Counters
Zihintntl	1.0	Non-Temporal Locality Hints
Zihintpause	2.0	Pause Hint
Zimop	1.0	May-Be-Operations

Table 1.1 RISC-V Unprivileged Architecture 20240411 + RVB23U64 v0.1 Summary *(continued)*

Name	Version	Description
Zcmop	1.0	Compressed May-Be-Operations
Zicond	1.0.0	Integer Conditional Instructions
M	2.0	Integer Multiplication and Division
A	2.1	Atomic Instructions
Zawrs	1.01	Wait on Reservation Set
Za64rs	RVB23	Reservation sets are 64 bytes
Ziccrse	RVB23	LR/SC progress guarantees (RsrvEventual)
Ziccamoa	RVB23	Main memory regions support AMO Arithmetic
RVWMO	2.0	RVWMO Memory Consistency Model
CMO	1.0.0	Base Cache Management Operations
		Zicbom: Basic Cache Maintenance
		Zicbop: Cache Prefetch
		Zicboz: Cache Block Zero
Zic64b	RVB23	Cache blocks must be 64 bytes.
F	2.2	Single Precision Floating Point
D	2.2	Double Precision Floating Point
Zfa	1.0	Additional Floating Point Instructions
C	2.0	Compressed Instructions
		Zca: Base compressed instruction set
		Zcd: Compressed double precision floating point load/store
Zcb	1.0.0	Additional Compressed Instructions
B	1.0.0	Bit manipulation instructions.
		Zba: Address arithmetic
		Zbb: General bit manipulation
		Zbs: Single bit manipulation
Zbc	1.0.0	Carryless Multiply
Zkt	1.0.1	Data Independent Execution Latency

1.10.2 RISC-V Privileged Architecture Extensions Implemented by the I8500

Table 1.2 lists the supported extensions for the RISC-V privileged architecture.

Table 1.2 RISC-V Privileged Architecture 20240411+ RVB23 S64 v0.1 Summary

Name	Version	Description
Ssstrict	RVB23	No non-conforming extensions present.
M mode	1.13	Machine-Level ISA
Ssmstateen	1.0.0	Machine state enable
Ssstateen	1.0.0	Supervisor state enable

Table 1.2 RISC-V Privileged Architecture 20240411+ RVB23 S64 v0.1 Summary

Name	Version	Description
Ss1p13	1.13	Supervisor-Level ISA
		Sv39: Page-based 39-bit virtual memory system
		Sv48: Page-based 48-bit virtual memory system
Sstvecd	RVB23	Supervisor trap vector (stvec) supports DIRECT
Sstvala	RVB23	Faulting address written to stval
Ssccptr	RVB23	Main memory supports hardware page-table reads
Svbare	RVB23	No translation or protection
Svade	RVB23	Manage A/D bits with page faults
Ssu64xl	RVB23	Supports 64-bit user mode (sstatus.UXL = 2)
Sscounterenw	RVB23	Implemented hpmcounter bits have corresponding scounteren bits.
Svnapot	1.0	Naturally Aligned Power-of-Two (NAPOT) Translation
Svpbmt	1.0	Page-Based Memory Types
Svinval	1.0	Fine-Grained Address-Translation Cache Invalidation
Sstc	1.0.0	Supervisor-mode Timer Interrupts
Sscofpmf	1.0.0	Count Overflow and Mode-Based Filtering
H	1.0	Hypervisor Support
Shcounterenw	RVB23	Implemented hpmcounter bits have corresponding hcounteren bits
Shvstvala	RVB23	Virt: writes vstval in all cases stval would be written
Shtvala	RVB23	Virt: writes hvtal with faulting guest physical address
Shvstvecd	RVB23	Virt: vstvec.MODE supports DIRECT w/ 4-byte aligned BASE
Shvstapa	RVB23	Virt: vsatp supports same translation modes as satp
Shgatpa	RVB23	Virt: hgatp supports ×4 versions of all supported satp modes

1.10.3 RISC-V Debug Architecture Extensions Implemented by the I8500

Table 1.1 lists the supported extensions for the RISC-V debug architecture.

Table 1.3 RISC-V Debug Architecture v1.0.0-rc2 ISA Extension Summary

Name	Version	Description
Sdext	1.0.0-rc2	RISC-V compliant external debug.
Sdtrig	1.0.0-rc2	RISC-V Trigger Module™.

1.10.4 RV64I Instruction Set Details

The following bullet items provide addition details on the I8500 implementation of the RV64I instruction set.

1.10.4.1 Endianess

The I8500 supports both little-endian and big-endian and boots into the mode selected by the pin input . The I8500 operates in a single, uniform endian mode at run time. Note that there is a GCR register that specifies the endianness of the I8500, and if software wants to alter the endianness on a system that sup-

ports both endian modes, it must write to that register, and then do a warm reset of the cores and the cluster for the endianness change to be observed.

1.10.4.2 misa[25:0] Extension Bits

The I8500 sets the misa extension bits listed below. misa is read only.

- A: Atomic extension.
- B: Bitmanip extension. Shogun implements the required Zba, Zbb, and Zbs extensions.
- C: Compressed instruction extension.
- D: Double-precision floating point extension.
- F: Single-precision floating point extension.
- H: Hypervisor extension.
- I: RV64I base ISA.
- M: Integer Multiply/Divide.
- S: Supervisor mode implemented.
- U: User mode implemented.
- X: Non-standard extensions present.

1.10.4.3 A Extension

The I8500 CPU implements LR/SC natively for both cacheable and uncacheable memory. For cacheable LR/SC, it implements one monitor per hart in the LSU. For uncacheable LR/SC, the I8500 relies on a monitor outside the core. The I8500 traps and emulates AMO arithmetic with LR/SC.

For LR/SC sequences, the I8500requires precise address and size matching; an LR of 8B and an SC of 4B within that 8B address will fail. Also, a reservation by one hart will be cleared by any ownership request by any other hart or core for the same 64B coherence granule.

1.10.4.4 F and D Extension

The I8500 implements both F and D extensions together. The I8500 does NOT provide a configuration option to remove either or both F and D extensions

1.10.4.5 Zicntr Extension

The I8500 should serialize reads to mcycle and minstret, as well as all the performance monitor counters, at issue. This is a change from Daimyo/current RTL.. The I8500 natively handles access to the time register. The I8500 provides four programmable performance monitor counters per hart.

1.10.4.6 Zihintpause and Zawrs Extensions

Shogun implements (RISC-V) pause, (MIPS) MPAUSE, WRS.STO, and WRS.NTO with variations on the behavior of the previous MIPS custom PAUSE instruction.

The WRS.NTO form should have a maximum delay of 1023 cycles. Note that the Zawrs specification has some rules about WRS.NTO when in VS or VU that need to also be honored.

1.10.4.7 Zihintntl Extension

The I8500 implements trivial support for Zihintntl: all Zihintntl HINTs are no-ops.

1.10.4.8 Zkt Extension

The I8500 MDU provides an OpCache intended to speed up operations with repeated arguments. This is (and must be) disabled for multiply instructions, to ensure compatibility with Zkt.

1.10.4.9 Zfa Extension

The I8500 implements the F and D extensions (single- and double-precision floating point). The I8500 does not support the Q and Vfh extensions (quad- and half-precision floating point). Any I8500 instantiation which supports F and D extensions also supports the single- and double-precision subsets of the Zfa extension. No I8500 configuration supports the quad-precision nor half-precision subset of the Zfa extension.

1.10.4.10 Zicbom Extension

The I8500 maps the RISC-V cache block operations to existing MCACHE behaviors as follows:

Table 1.4 Equivalent MCACHE Instructions

Zicbom	Equivalent MCACHE	Comments
cbo.clean	mcache L2HitWb	op[4:2] == 6 && op[1:0] == 2
cbo.flush	mcache L2HitWbInv	op[4:2] == 5 && op[1:0] == 2
cbo.inval	mcache L2HitWbInv	op[4:2] == 5 && op[1:0] == 2 if CBIE == 01b // Flush if U and not delegated to do invalidates via CBIE=11b
	mcache L2HitInv	op[4:2] == 4 && op[1:0] == 2 if CBIE == 11b // Inval

In the table above, CBIE refers to the effective CBIE value determined by menvcfg.CBIE, henvcfg.CBIE, senvcfg.CBIE, and the current privilege level.

1.10.4.11 Zicbop Extension

The I8500 maps the RISC-V prefetch operations to existing MIPS custom instruction behaviors as follows:

Table 1.5 Equivalent PREF Instructions

Zicbop	Equivalent MIPS Custom Instruction	Comments
prefetch.i	pref IcacheLoad	hint[4:0] == 0 ("Icache" "Load")
prefetch.r	pref DcacheLoad	hint[4:0] == 8 ("Dcache" "Load")
prefetch.w	pref DcacheStore	hint[4:0] == 9 ("Dcache" "Store")

The Zicbop extension does not provide a mechanism to specify which level of cache to prefetch into. HINTs defined in Zihintntl can provide this information; however, the i8500 implements Zihintntl as NOPs. For the I8500, the Zicbop prefetch operations prefetch to L1. Software can continue to use the MIPS custom PREF instructions to specify the target cache if desired.

1.10.4.12 Zicboz Extension

The I8500implements Zicboz as follows, based on the CCA encoding for the specified address. The CCA meanings are specified in the MIPS internal specification for the pma*cfg registers.

- CCA ≠ 1: Data cache
  - Miss in L1D cache: Commits a 64 byte write of zeros directly to L2.
  - Hit in L1D cache: Commits a 64 byte write of zeros to L1D cache.

Note: Hit vs. Miss is determined by the ordinary rules regarding CCA and page-based memory types (PBMT).

- CCA = 1: Buffer cache
  - Miss in L1B cache: Allocates line in L1B and fills the line with zeros.
  - Hit in L1B cache: Commits a 64 byte write of zeros to L1B cache.

1.10.4.13 Svpbmt Extension

The I8500 CPU honors Svpbmt PTE overrides, even for CCA = 1 buffer cache space. The PBMT encodings are as shown in the table below:

Table 1.6 Svpbmt Extensions

Binary Encoding	Mode name	Details	Maps to this PMACCA Encoding
00	PMA	Honor existing PMA attributes.	--
01	NC	Non-cacheable, idempotent, weakly-ordered (RVWMO), main memory.	PMACCA = 3 (UCA) and S = 1
10	IO	Non-cacheable, non-idempotent, strongly-ordered (I/O ordering), I/O.	PMACCA = 2 (UC) and S = 0
11	--	Reserved.	

1.10.4.14 Rationale

The RISC-V architecture defines Page-Based Memory Types (PBMTs) as overriding the memory type specified in the PMAs, unless the PMA specifies the address range as vacant.

In the I8500, the L1 buffer cache and L2 buffer pipe function as a separate, parallel memory hierarchy distinct from the normal RISC-V memory hierarchy. Therefore, the I8500 address ranges with CCA = 1 (buffer cache) should be treated as vacant with respect to PBMT.

1.10.4.15 Svinval Extension

The I8500 implements the Svinval implementation as described in the RISC-V Privileged Architecture Specification.

1.11 Additional Information

For additional information on the functional safety features of the P8700/P8700-F device, refer to the following documents and links:

MIPS RV64 P8700/P8700-F Multiprocessing System Functional Safety Technical Reference Manual

https://riscv.org/technical/specifications/

Chapter 2

Memory Management Unit

The MMU translates virtual addresses generated by the core, to physical addresses used to access caches, memory and other devices. Virtual-to-physical address translation is especially useful for operating systems that must manage physical memory to accommodate multiple tasks active in the same virtual address space. The MMU also enforces the protection of memory areas and defines the cache attributes. The P8700-F MMU implements a Translation Lookaside Buffer (TLB).

This chapter covers the programmable elements of the TLB in the P8700-F Multiprocessing System. The first section gives an overview of the TLB architecture, a description of its functionality and a description of the elements that go into programming the TLB. The sections that follow cover specific information on programming for the TLB.

2.1 Overview

The P8700-F TLB translates 48-bit virtual addresses to 48-bit physical addresses and provides access control for different page segments of memory. The core writes to internal CSR registers with the information used to initialize and modify entries in the TLB, then executes a TLB write instruction (MTLBWR) to move the data from the registers to the TLB.

2.1.1 TLB Types

The Memory Management Unit (MMU) in the P8700-F core consists of four address-translation lookaside buffers (TLB):

- Instruction TLB (ITLB)[1]. Number of ITLB entries may be configured for 8 or 16 entries. Supported page sizes are 4KB and 64KB. The Instruction TLB is fully associative, managed by hardware and is transparent to software. It acts as a cache of translations requested by the IFU.
- Data TLB (DTLB)[1]. The number of DTLB entries may be configured for 16, 32 or 64 dual entries. A dual entry is defined as two adjacent pages of the same page size, differing only in the bit above the page offset for page size of the entry. Supported page sizes are 4KB and 64KB. The Data TLB is fully associative, managed by hardware and is transparent to software. It acts as a cache to translations requested by the LSU.
- Programmable 16, 32, or 64 dual-entry VTLB. The VTLB is used as a shared second level cache for translations of both the ITLB and DTLB. The VTLB is primarily used to store translation request for page sizes of 2MB, 1 GB and 512G pages. It also supports 4k and 64k pages which are periodically diverted from the FTLB. The VTLB is a fully associative

1. The ITLB and DTLB perform address translations for the instruction and data caches respectively. These blocks are not software visible and are shown only for completeness.

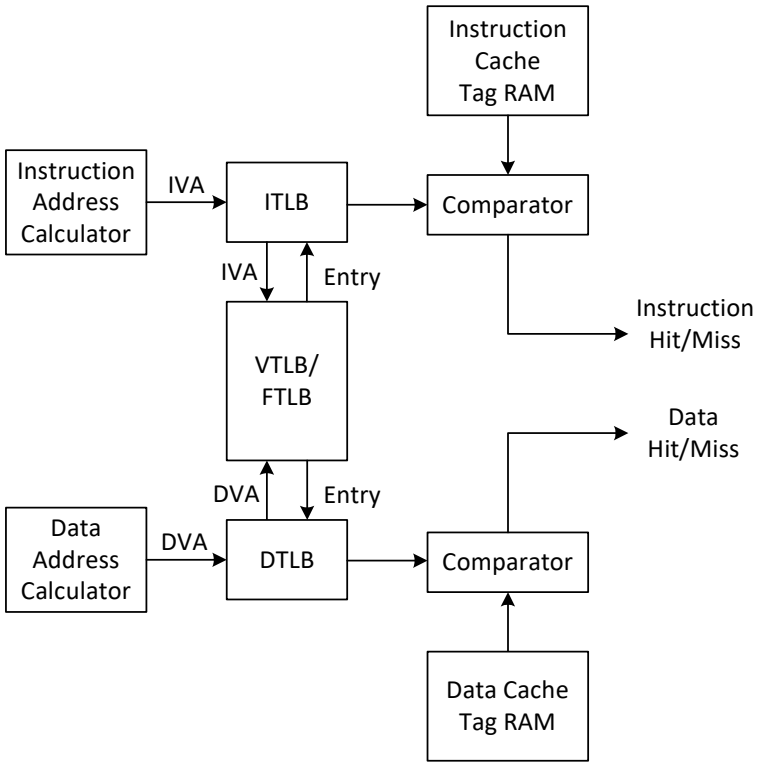
cache which uses a binary tree LRU for replacement selection.

If a fetch address cannot be translated by the ITLB or DTLB, the TLB attempts to translate it in the following clock cycle or when available. If successful, the translation information is copied into the ITLB/DTLB for future use. Entries are automatically refilled from the TLB when required, and automatically cleared whenever the associated TLB is updated.

- 512 dual-entry Fixed TLB (FTLB) that is also a shared backup storage for both ITLB and DTLB translation request. It functions as a second level cache of translation from the ITLB and DTLB. The FTLB differs from the VTLB in that it stores translation for 4K and 64k pages only. It is organized as a 128 way by 4 set associative way cache. Management of the 4 sets in each of the ways is by a modified round robin replacement policy. Both the VTLB and the FTLB are accessed in parallel. A specific translation can only reside in one of the two second level caches.

Figure 2.1 shows an overview of the P8700-F MMU architecture.

Figure 2.1 Overview of MMU Architecture in the P8700-F Core



When an instruction address is to be translated, the ITLB is accessed first. If the translation is not found, the VTLB/FTLB is accessed. If there is a miss in the VTLB/FTLB, an exception is taken. Similarly, when a data reference is to be translated, the DTLB is accessed first. If the address is not present in the DTLB, the VTLB/FTLB is accessed. If there is a miss in the VTLB/FTLB, an exception is taken. The OS should process the exception by overwriting a TLB entry from the appropriate VTLB or FTLB with the original translation requested.

2.1.2 Instruction TLB (ITLB)

The ITLB is a variable-entry high speed translation lookaside buffer dedicated to performing translations for the instruction stream. The number of ITLB entries is configurable and the entries are shared between harts. The ITLB maps only 4KB and 64KB pages. The ITLB is

managed by hardware and is transparent to software. The larger VTLB/FTLB is used as a backup structure for the ITLB as described in Section 2.1, "Overview".

2.1.3 Data TLB (DTLB)

The DTLB is an variable-entry high speed translation lookaside buffer dedicated to performing translations for the data stream. The number of DTLB entries is configurable and the entries are shared between harts. The DTLB maps only 4K and 64k pages. The DTLB is managed by hardware and is transparent to software. The larger VTLB/FTLB is used as a backup structure for the DTLB as described in Section 2.1, "Overview".

2.1.4 Variable Page Size TLB (VTLB)

The purpose of the VTLB is to translate virtual addresses and their corresponding ASID into a physical memory address. The VTLB is configurable for 16-, 32-, or 64 entries. Each entry is a dual entry containing a pair of even/odd translations. The translation is performed by comparing the upper bits of the virtual address (along with the ASID bits) against each of the entries in the *tag* portion of the VTLB structure. The VTLB is used to translate both instruction and data virtual addresses.

2.1.4.1 VTLB Organization

The VTLB supports the following page sizes:

- 4 KB, 64 KB, 2MB, 1 GB, 512G

The VTLB/FTLB is organized in pairs of page entries to minimize its overall size. Each virtual *tag* entry corresponds to two physical data entries, an even page entry and an odd page entry. The highest order virtual address bit not participating in the tag comparison is used to determine which of the two data entries is used. Since page size can vary on a page-pair basis, the determination of which address bits participate in the comparison and which bit is used to make the even-odd selection must be done dynamically during the TLB lookup.

The page size for a TLB entry is determined by the level in the page table from which the PTE was read. The level is part of the TLB tag, along with the VPN, ASID, and G bits.

2.1.5 Fixed Page Size TLB (FTLB)

The FTLB contains 512 dual-entries and is organized as 128-sets x 4-ways. Each set of each way contains dual data RAM entries and one tag RAM entry. If the tag RAM contents match the requested address, either the low or high RAM location of the dual data RAM is accessed depending on the state of the least-significant-bit (LSB) of the virtual address (VPN2). Each RAM location maps a fixed page size, which is configurable to either 4 KB or 64KB.

The FTLB64 bit in the MIPSConfig7 register selects between 4 KB and 64 KB page sizes as shown in Table 2.1.

Table 2.1 FTLB Organization

FTLB Parameter	Programmable Options	FTLB64 Bit
Page Size	4 KB	0
	64 KB	1

2.2 TLB ECC Errors

TLB ECC errors are reported using bits 31:26 of the CSR *mipscacheerr* register. Two fields are used; the STATE field (bits 31:30) and the ARRAY field (bits 29:26). These bits are set by hardware and are used to report errors within the L1 instruction and data caches, as well as the TLB. A TLB ECC error can be reported for either the tag portion or the data portion of the array as shown in Table 2.2. Note that this table only shows the TLB-specific errors. For a complete encoding of the *ARRAY* and *STATE* fields, refer to the *mipscacheerr* register.

Table 2.2 TLB ECC Error Reporting in the MIPSCacheErr Register

STATE (Bits 31:30)	ARRAY (Bits 29:26)	Condition
2'b00	4'b0100	No FTLB Tag RAM error
2'b01	4'b0100	Correctable FTLB Tag RAM error
2'b10	4'b0100	Uncorrectable FTLB Tag RAM error
2'b00	4'b0101	No FTLB Data RAM error
2'b01	4'b0101	Correctable FTLB Data RAM error
2'b10	4'b0101	Uncorrectable FTLB Data RAM error

2.3 MIPS TLB Exception Handling

The MIPS TLB's report exceptions as described in the *RISC-V Privileged Specification*

2.4 TLB Duplicate Entries

When writing to the TLB, all ways of a single set in the FTLB and all the entries of the VTLB are searched for duplicates. If a large page is written to the VTLB and multiple duplicates exist for that larger page in the FTLB (multiple sets in the FTLB), then not all the duplicates are detected (and invalidated).

2.5 TLB Instructions

This section defines the various types of instructions used when accessing the TLB.

- **MTLBWR** — The TLB Write Random instruction causes a random TLB entry selected by hardware to be written with the virtual address in mtval CSR and the leaf PTE value stored in integer register $rs1.

2.6 Shared FTLB Translations

The P8700-F core supports shared FTLB translations across all Harts in a core. In many applications, there can be multiple threads that are working cooperatively or running the same application on different data. In this situation, some translations are common across Harts and sharing the translations increases the FTLB capacity and reduces contention. Even under Linux, multiple threads can be associated with the same process and use the same translations on different Harts.

2.7 Hardware Table Walker

The Hardware Table Walker (HTW) performs a hardware page table walk to translate virtual addresses to physical. Its main functionality is supported by two state machines, called sequencers (sequencer 0 and sequencer 1).

The CSR registers that relate to the Hardware Table Walker (HTW) are as follows:

1. The PPN and the MODE fields in SATP, VSATP and HGATP registers. The PPN field holds the physical page number (PPN) of the root page table. The MODE field is the addressing mode. The programmer shall choose one of either the Sv39 or SV48 addressing mode.

2. The DHTW (Disable HTW) bit in the mipsconfig7 register. If this bit is set the HTW is disabled.

As mentioned above, the addressing mode that the HTW supports is either Sv39 or Sv48. Thus, the HTW supports 4KB, 64KB, 2MB, 1GB and 512GB page sizes. At the end of the table walk, the mapping is written to the FTLB or VTLB based on the page size.

- The 4KB page mappings are written to the FTLB if the FTLB64 bit of the MIPSConfig7 register is cleared. The 64KB page mappings are written to the FTLB if the FTLB64 bit of the MIPSConfig7 register is set.
- For the 2MB, 1GB and 512GB pages sizes, their virtual to physical mappings are written to the VTLB.

2.8 MMU Programming

The following subsections describe some of the programming options for the P8700-F MMU. Each section provides CSR register information listing the register and field(s) used to determine the required information, as well as an assembly code example.

This section is intended to provide examples of how to program the custom features of the MIPS RISC-V MMU implementation.

Chapter 3

Caches

The P8700-F Multiprocessing System contains the following caches: L1 instruction, L1 data, and shared L2. These caches provide on-chip temporary storage of information that can be retrieved much faster than accessing main memory. The dedicated L1 instruction and data caches have the fastest access times and are accessed first. If the data is not present in the L1 cache, the shared L2 cache is accessed. The L2 cache contains both data and instructions, hence the name 'shared'. If the requested data is not in the L2 cache, the main memory is accessed.

This chapter provides an overview of the cache architecture and a description of the elements that go into programming the caches. A description of the CSR register interface to each cache is provided, as well as cache initialization code. Other programmable elements include setting up cache coherency and handling cache exceptions.

3.1 Cache Configurations

The P8700-F Multiprocessing System contains three caches; L1 instruction, and L1 data, and shared L2. These caches are non-optional and are always present. The size of each cache can be configured as shown in Table 3.1.

Table 3.1 P8700-F Cache Configurations

Attribute	L1 Instruction Cache	L1 Data Cache	L2 Cache
Size	32 KB or 64 KB	32 KB or 64 KB	256 KB, 512 KB 1 MB, 2 MB, 4 MB, or 8 MB
Line Size	64-byte	64-byte	64-byte
Number of Cache Sets	128 or 256	128 or 256	512, 1024, 2048, 4096, 8192
Associativity	4-way	4-way	8-way (256 KB only) 16-way (all others)

The L1 instruction cache is attached to the Instruction Fetch Unit (IFU). The L1 data cache is attached to the Load/Store Unit (LSU). The L2 cache is embedded within the Coherence Manager (CM) and communicates with external memory via a 256-bit AXI interface.

For more information on the L1 instruction cache, refer to Section 3.3.1 "L1 Instruction Cache".

For more information on the L1 data cache, refer to Section 3.3.2 "L1 Data Cache".

3.2 LR and SC Instruction Considerations

The MIPS RISC-V CPUs support LR/SC to both Cached and Uncached addresses, but assume that both instructions in the pair address memory regions with same RISC-V Cache and Coherency PMA (CCA) type.

LR/SC to Uncached addresses with CCA set to MIPS UCA type will never succeed.

Atomic access is only supported for a memory location if all harts writing to that location use the same CCA. The CCA may not be changed while an atomic access (LR/SC sequence) is in progress.

MIPS King-V CPU depends on AXI exclusive access monitor for all Uncached LR/SC. If software attempts an uncached LR to an address that does not have an exclusive access monitor, a Bus Error will be generated to prevent software from relying on incorrect SC results.

For Cacheable LR/SC, the reservation set size is one cache line (aligned block of 64 bytes).

For uncached LR/SC, the reservation set size is determined by the AXI exclusive access monitor

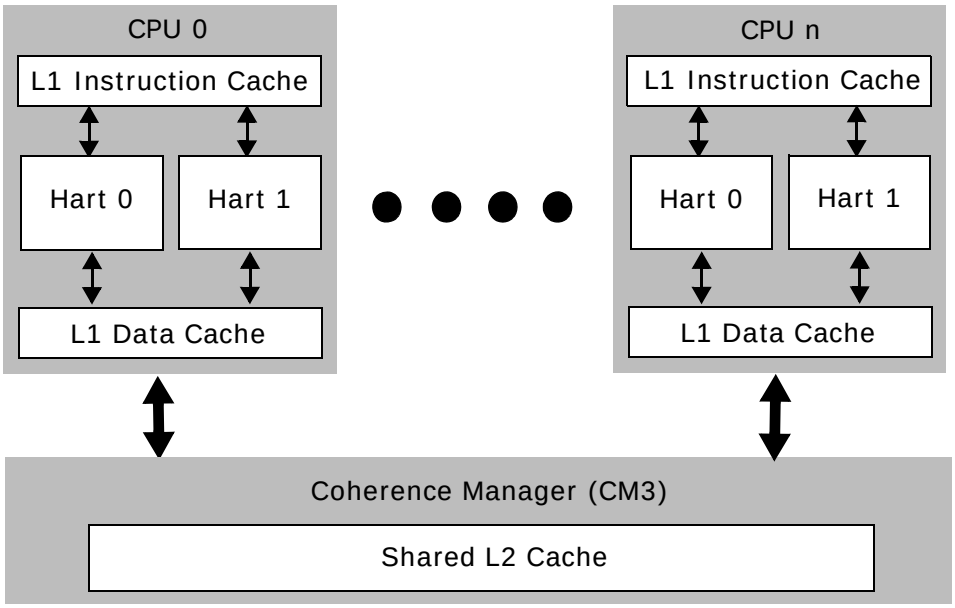
3.3 Cache Subsystem Overview

In the P8700-F MPS, the size of each cache can be configured as follows:

- L1 Instruction Cache: 32 KB or 64 KB
- L1 Data Cache: 32 KB or 64 KB
- L2 Cache: 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, or 8 MB

Figure 3.1 shows the relative location of the caches within the P8700-F Multiprocessing System. The L1 instruction and L1 data caches are shared by all Harts in the same core. The L2 cache is shared by all cores.

Figure 3.1 P8700-F Multiprocessing System Caches



3.3.1 L1 Instruction Cache

The L1 instruction cache contains two arrays: tag and data. The L1 instruction cache is virtually indexed and physically tagged.

Table 3.2 shows the key characteristics of the L1 instruction cache. Figure 3.2 shows the format of an entry in the three arrays comprising the instruction cache tag and data.

Table 3.2 L1 Instruction Cache Attributes

Attribute	With EDC
Size[1]	32 KB or 64 KB
Line Size	64-byte
Number of Cache Sets	128 or 256
Associativity	4-way
Replacement	LRU
Data Array	
Read Unit	(256b + 32-bit EDC) x number of ways
Write Unit	512b + 64-bit ECC
Tag Array	
Read Unit	(36-bit tag + 1-bit lock + 7-bit EDC + Valid bit) x 4-ways (32K and 64K)
Write Unit	36-bit tag + 1-bit lock + 7-bit EDC + Valid bit (32K and 64K)
Way-Select Array	
Read Unit	6-bits (4-way)
Write Unit	6-bits (4-way)

1. For Linux based applications, MIPS recommends a 64 KB L1 instruction cache size.

Figure 3.2 L1 Instruction Cache Read Unit — 32 KB and 64 KB Cache

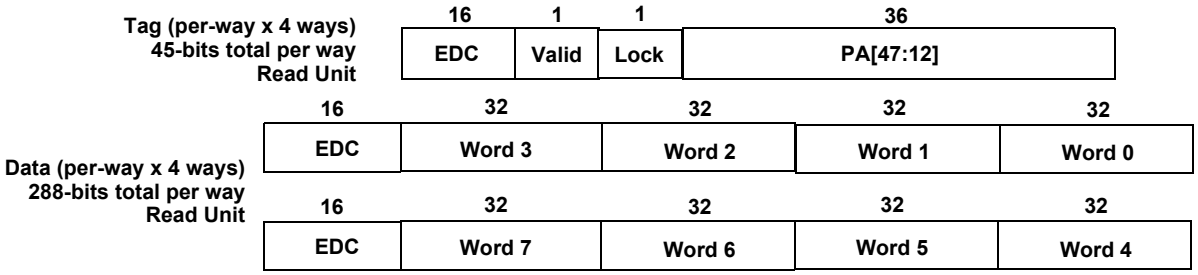
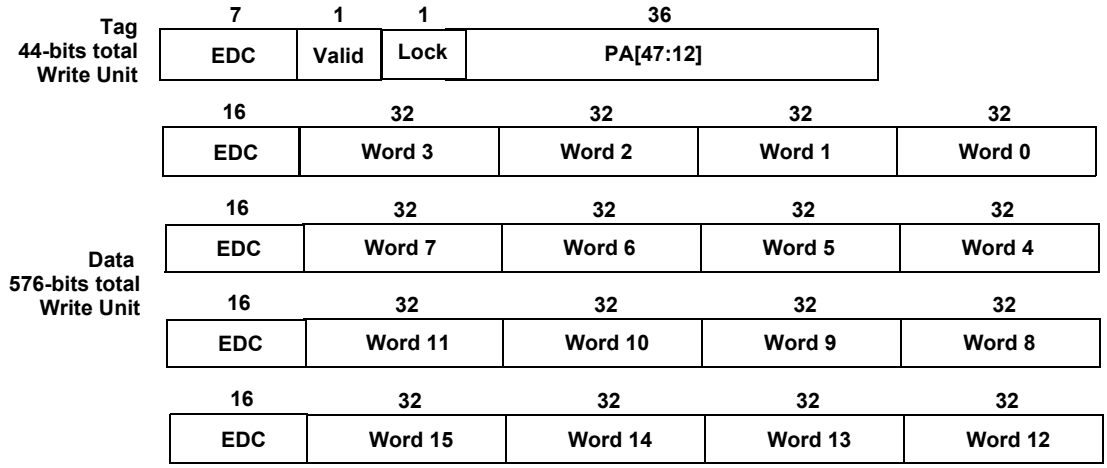


Figure 3.3 L1 Instruction Cache Write Unit — 32 KB and 64 KB Cache



3.3.1.1 Level 1 Instruction Cache Error Detection

The P8700-F core includes detection of single and double-bit errors in the Level 1 Instruction Cache. The error detection logic protects against data corruption caused by errors that may occur while data is stored in RAM. When an error is found, the code is refetched from memory. The error is handled entirely by hardware and is software-transparent.

3.3.1.2 L1 Instruction Cache Organization

The P8700-F core level 1 instruction cache comprises two logical RAM arrays: a tag array, a data array and a register-based way select array. With error detection, a 7-bit EDC is added to the 36-bit tag stored in the tag array; a 16-bit EDC is also added to each 64-bit data doubleword stored in the data array.

3.3.1.3 L1 Instruction Cache Error Types

On an L1 EDC error the Instruction Fetch Unit (IFU) re-fetches the data and bypasses the desired instruction while overwriting the instruction in error. The EDC error gets counted by the performance counters but the fetch continues. If the entire cache was to fail, the fetch would effectively proceed uncached by this method. The IFU raises cache errors from L2 as Cache Exceptions.

3.3.1.4 L1 Instruction Cache Replacement Policy

The L1 instruction cache replacement policy refers to how a way is chosen to hold an incoming cache line on a miss which will result in a cache fill. The replacement policy is least-

recently used (LRU). The LRU bit(s) in the way-select array encode the order in which ways on that line have been accessed.

On a cache miss, the LRU bits for the tag and way-select entries of the selected line may be used to determine the way which will be chosen. In the P8700-F core, the way select information is stored in registers and is not part of a memory array.

The LRU field in the way select array is updated as follows:

- On a cache hit, the associated way is updated to be the most recently used. The order of the other ways relative to each another is unchanged.
- On a cache refill, the filled way is updated to be the most recently used.

3.3.1.5 L1 Instruction Cache Coherency Management

In the P8700-F core, the hardware does not automatically keep the instruction cache coherent with the data cache, so code that modifies the instruction stream must invalidate stale instruction cache lines.

3.3.1.6 FENCE.I Instruction Usage

The **`FENCE.I`** instruction provides a mechanism available to user-level code for ensuring that previously written instructions are correctly presented for execution. Use of the **`FENCE.I`** instruction is preferred to the traditional alternative of a D-cache writeback followed by an I-cache invalidate.

3.3.2 L1 Data Cache

The L1 data cache contains two arrays: tag and data. The L1 Data cache is virtually indexed and physically tagged, but contains logic to correct virtual aliasing.

The tag and data arrays hold 4 ways of information per set, corresponding to the 4-way set associativity of the cache. A tag entry consists of the upper 34 or 35 bits of the physical address (depending on cache size), two coherent state bits, and some ECC bits. A data entry contains 64 bytes of data and associated ECC bits. All 64 bytes in the line are present in the data array together, hence the coherent state bits (2) stored with the tag.

After a valid line is resident in the cache, a store operation can update all or a portion of the words in that line depending on the type of store.

The data cache uses ECC so that single-bit errors can be corrected. ECC code is generated across a 32-bit word. Sub-word stores are handled by doing a read-modify-write sequence. The error checking and correction process is handled entirely by hardware and is transparent to kernel software.

A way-select register holds bits choosing the way to be replaced according to a Least Recently Used (LRU) algorithm. The LRU information applies to all the ways and there is one way-select register for all the ways in the set. Note that this information is stored in a register and is not part of a memory array.

Table 3.3 shows the key characteristics of the data cache. Figure 3.4 through Figure 3.7 shows the format of an entry in the arrays comprising the data cache: tag, data, and way-select for 32 KByte and 64 KByte read and write units.

Table 3.3 L1 Data Cache Organization

Attribute	With Parity
size	32 or 64KB
Line size	64-byte

Table 3.3 L1 Data Cache Organization *(continued)*

Attribute	With Parity
Number of Cache Sets	128 or 256
Associativity	4-way
Replacement	LRU
Data Array	
Read Unit	(128b + 28b ECC) x 4
Write Unit	512b + 112b ECC
Tag Array	
Read Unit	(34/35b PPN + 2b CohSt + 8b ECC) x 4
Write Unit	35b PPN + 2b CohSt + 8b ECC (32K) 34b PPN + 2b CohSt + 8b ECC (64K)
Way-Select	
Read Unit	6-bit register field
Write Unit	6-bit register field
Dirty Bits	
Read Unit	4-bit register field
Write Unit	1-bit register field

Figure 3.4 L1 Data Cache Read Unit — 32 KB Cache

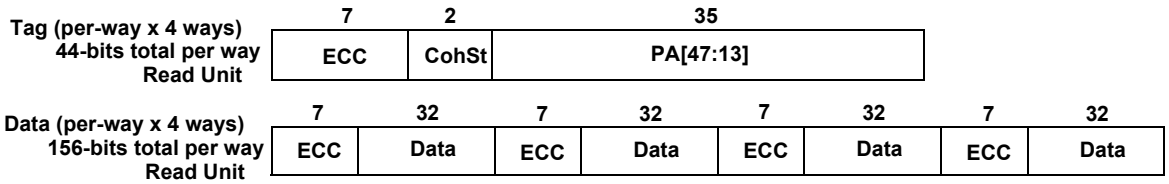


Figure 3.5 L1 Data Cache Write Unit — 32 KB Cache

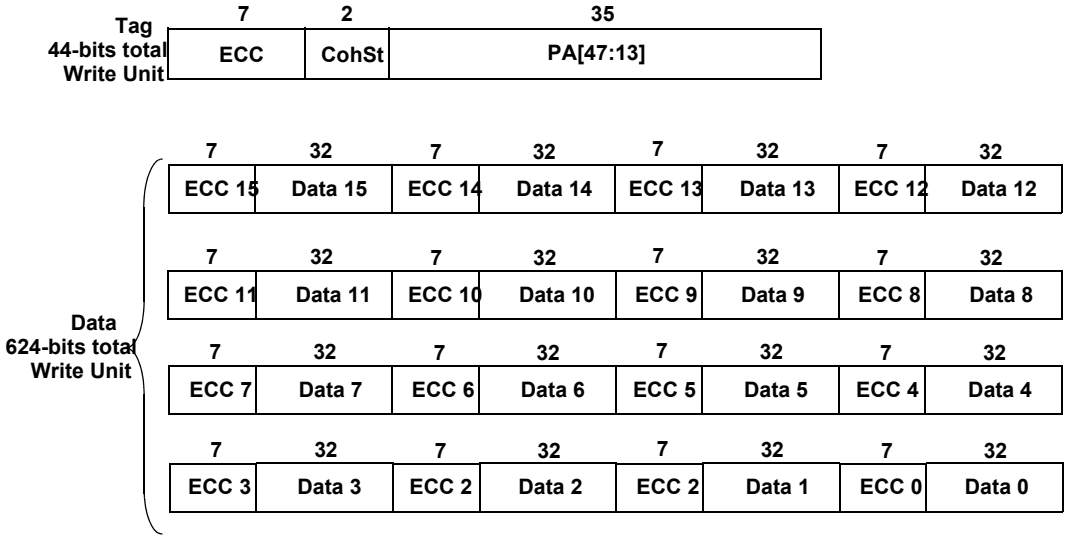


Figure 3.6 L1 Data Cache Read Unit — 64 KB Cache

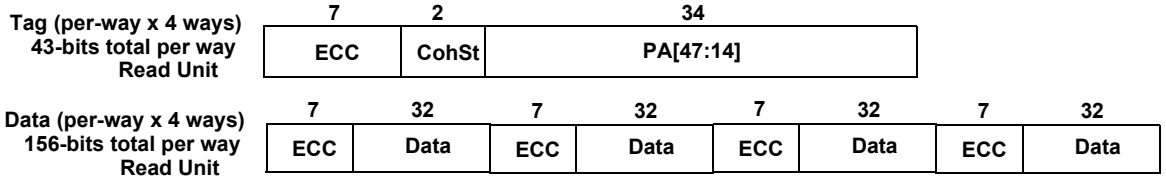
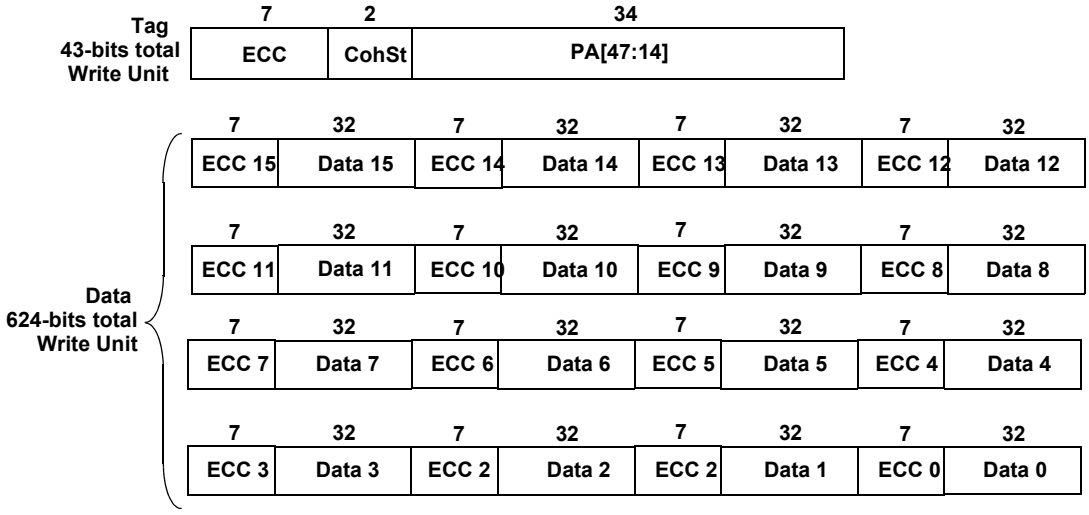


Figure 3.7 L1 Data Cache Write Unit — 64 KB Cache



3.3.3 Level 1 Data Cache Error Checking and Correction (ECC)

The P8700-F core includes error checking and correction (ECC) on the Level 1 Data Cache. Error correction codes are added to information stored in data-cache. The error detection and

correction logic protects against data corruption caused by single-bit transient errors that may occur while data is stored in RAM. The error codes allow for single-bit error correction and double-bit error detection. ECC generation and checking and error handling is done in the Load/Store Unit (LSU).

3.3.3.1 L1 Data Cache Organization

As shown in the above figures, the P8700-F core level 1 data cache comprises two logical RAM arrays: a tag array and a data array. With error detection and correction;

- A 7-bit ECC is added to each 34/35-bit tag stored in the tags array.
- A 7-bit ECC is added to each 32-bit data value stored in the data array.

3.3.3.2 L1 Data Cache Load/Store Operations

Cacheable loads and stores generate a data cache read to see if the memory operand is in the cache. If an error is detected, incoming loads and stores are halted by hardware and the LSU determines whether an ECC error is uncorrectable or correctable. Uncorrectable errors generate an exception. If the error is correctable, and the load/store is retried.

3.3.3.3 L1 Data Cache Error Types

L1 data cache ECC errors can be correctable or uncorrectable. Single-bit errors are correctable. Multiple-bit errors cannot be repaired. Multiple-bit errors in a data word of an invalid cache line are ignored. Note that a tag needs to be free of errors to affirm that a line is invalid. Hence, tag errors are processed before processing multiple-bit data errors. A multiple-bit error is uncorrectable if it occurs in (a) a tag, or (b) a data word in a dirty cache line.

3.3.3.4 Store Operations Less than 32-bits

The addition of ECC to the cache data array has special implications for stores into the data cache when the operand is smaller than a single 32-bit word, or the store operation is not 32-bit aligned. When partial-word stores hit in the cache, the LSU may need to perform a cache read-modify-write on the affected word because the ECC is a function of the entire 32-bit word.

The store buffer keeps track of valid bytes and allows multiple stores to merge together. If the entire word is valid, it can be written into the cache. If it is only partially valid, the data array is read to fill in the missing bytes.

3.3.3.5 Examples of L1 Data Cache ECC Errors

Consider some data cache ECC error scenarios:

Loads and Stores

During CPU loads and stores, single-bit errors in the primary tags array are scrubbed on detection. Multiple-bit errors in the tag array generate an exception. During CPU loads and stores, single-bit errors in the data array of valid lines are scrubbed on detection. Double-bit data errors generate an exception.

Evictions

During eviction of a dirty cache line, single-bit data errors are corrected on the fly as data is written back to the Bus Interface Unit (BIU). Multiple-bit errors in an evicted line are reported as an uncorrectable error to the BIU and generate an exception.

Interventions

During interventions, single-bit errors in the tag array are scrubbed on detection. Multiple-bit errors in the tag array generate an exception and return an ERROR response for the intervention.

During an intervention write-back of an modified line, single-bit data errors are corrected on the fly as data is forwarded to the BIU. Multiple-bit data errors during an intervention write-back are reported to the BIU and an exception is generated.

3.3.4 L1 Data Cache Replacement Policy

The replacement policy refers to how a way is chosen to hold an incoming cache line on a miss which results in a cache fill. The replacement policy is least-recently used (LRU). The LRU bit(s) in the way-select array encode the order in which ways on that line have been accessed.

On a cache miss, the LRU bits for the tag and way-select entries of the selected line may be used to determine the way which will be chosen. In the P8700-F core, the way select information is stored in registers and is not part of a memory array.

The LRU field in the way select array is updated as follows:

- On a cache hit, the associated way is updated to be the most recently used. The order of the other ways relative to each another is unchanged.
- On a cache refill, the filled way is updated to be the most recently used.

If the way selected for replacement has its dirty bit asserted in the dirty array, then that 64-byte line will be written back to memory before the new fill can occur.

3.3.5 L1 Data Cache Memory Coherence Protocol

The P8700-F core supports cache coherency in a multi-CPU system in conjunction with the directory-based coherence manger (CM).

The L1 data cache utilizes a standard MESI protocol. Each cache line will be in one of the following four states:

Invalid: The line is not present in this cache.

Shared: This cache has a read-only copy of the line. The line may be present in other L1 data caches, also in a Shared state. The line will have the same value as it does in the L2 cache.

Exclusive: This cache has a copy of the line with the right to modify. The line is not present in other L1 data caches. The line is still clean - consistent with the value in L2 cache.

Modified: This cache has a dirty copy of the line. The line is not present in other L1 data caches. This is the only up-to-date copy of the data in the system (the value in the L2 cache is stale).

Some of the basic characteristics of the coherence protocol are summarized below.

- Writeback cache - Uses a writeback cache to ensure high performance
- Cache-line based - Coherence and ownership is maintained per 64-byte cache line
- Invalidate - A line is invalidated from the cache (possibly with a writeback to memory) when a store from another processor is seen.

3.3.6 Load/Store Bonding

Bonding is a technique where adjacent loads or adjacent stores are merged into a single request in the IDU and sent to the LSU in one cycle.

Supported bonds:

- Only word and dword loads and stores.
- Only identical instruction (i.e., LW+LW and not LW+LD or LW+LWE).
- Only when using same base address register and the offset of the second instruction is +4 (word size ops) or +8 (dword) from the first.
- First load does not use same base and destination register must not be UC (CCA2, dseg).

IDU bonding is based on instruction decode. It does not know the base address value or the eventual alignment of operations. It attempts to bond any adjacent load/stores. If the operations turn out to not fall within an aligned double-quadword (LD/SD bonding) or double-word (LW/SW bonding), the LSU will signal the GRU for an instruction re-fetch.

Bonding is invisible to software other than improved performance.

3.3.7 L2 Cache

The L2 cache processes transactions that miss in the L1 caches. The L2 cache is larger than the L1 caches. In the P8700-F Multiprocessing System, the L2 cache is integrated into the Coherence Manager (*CMrev*). The L2 communicates with external memory via an AXI-4 interface. The L2 communicates with the cores through the proprietary MIPS Coherence Protocol (MCP) bus.

The associativity of the L2 cache can be either 8 or 16 ways. The 8-way option is used when the cache size is 256 KB. The 16-way option is used for all other cache sizes. The line size is fixed at 64 bytes. The number of sets and ways is selected during the build process and cannot be changed by the kernel software. Software can check the set size by reading the GCR_L2_CONFIG register, which is part of the *CMrev* register address space. Refer to the *Coherence Manager* chapter for more information.

Table 3.4 shows the list of possible L2 cache configurations.

Table 3.4 L2 Cache Configurations

Line Size	Sets per Way	Number of Ways	Total L2 Cache Size
64 bytes	512	8	256 KBytes
64 bytes	512	16	512 KBytes
64 bytes	1024	16	1 MByte
64 bytes	2048	16	2 MBytes
64 bytes	4096	16	4 MBytes
64 bytes	8192	16	8 MBytes

The L2 cache processes transactions that are not serviced by the L1 cache. The L2 cache is generally larger than the L1 cache, but slower, due to longer access latencies. In the P8700-F Multiprocessing System, the L2 cache is integrated into the Coherence Manager (CM). The L2 communicates with external memory via an AXI-4 interface.

The L2 also communicates with the CPU(s) through the proprietary MIPS Coherence Protocol (MCP) bus. In addition, the L2 has the clock, reset, and bypass signals as well as some static input signals which can be used to configure it for different operating modes.

3.3.8 L2 Cache General Features

- 5-stage pipeline.
- 48-bit address paths and 512-bit internal data paths
- Associativity: 8-way or 16-way
- Cache size: 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB
- Line Size: 64 bytes (8 doublewords)
- Locking Support: Yes
- Replacement Algorithm: Pseudo LRU
- Write policy: Write Back
- Write miss allocation policy: Write-Allocate
- Error Checking and Correction (ECC): 2-bit error detection and 1-bit error correction covering the tag and data arrays.
- Maximum read misses outstanding: 12 - 32. Build-time configuration option.
- 256-bit width on memory-side AXI-4 interface.
- AXI-4 Burst Size on the memory interface: 64-byte line size: 2 beats of 256-bit data
- Bypass Mode Support: In bypass mode, all processor requests are routed to the system. This mode is used only for debug purposes and should not be used during normal operation.
- Multi-cycle Data Rams: Configurable for either 2-cycle or 4-cycle latency
- Multi-cycle Tag Rams: Configurable for either 1-cycle or 2-cycle latency

Multi-cycle Way-Select Rams: 0, 1, 2, or 3 stalls can set the Way-Select RAM access times to 1, 2, 3, or 4 clocks. In the table above, the associativity of the L2 cache is fixed at 16 ways and the line size if fixed at 64 bytes. As a result, changes to the number of sets per way determine the overall size of the L2 cache. The only exception is the 256 KB cache option, which contains the same number of sets per way as the 512 KB option shown in Table 3.1, but is selected using 8 ways instead of 16.

3.3.9 Overview of the AXI Interface

In the P8700-F core, the L2 cache is integrated into the CM. The following are some features of the AXI interface to the CM.

- 256-bit AXI-4 data bus width between CM and memory.
- Each beat is at most 32 bytes
- Requests are either 1 beat of data or 2 beats of data
- Writes cannot receive an early response

3.3.9.1 AXI Channels

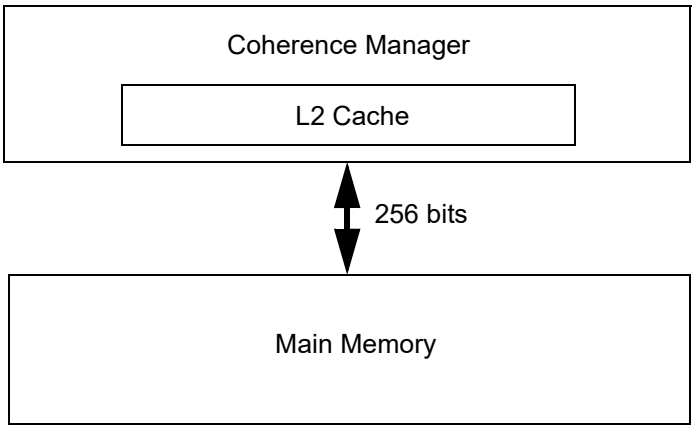
The AXI bus contains a 5-channel interface. Each channel is unidirectional and independent of the other channels:

- Read address
- Write address
- Write data
- Read response

- Write response

The AXI interface between the CM is 256-bits wide with a fixed 64-byte line size. This is shown in Figure 3.8.

Figure 3.8 AXI Interface Between CM and Memory



Flow control is similar to that in the previous generation OCP protocol.

3.3.9.2 Read Operations

On the AXI bus, each transaction is assigned an ID value. Depending on the type of transaction, transactions can have either the same ID, or a different ID. Read operations with different ID values can be processed and returned out of order. However, Read operations with the same ID value are processed and returned in order.

3.3.9.3 Write Operations

On the AXI bus, write operations, the order of the write data must be the same as that on the write address channel. However, the timing of the transactions can be different (transactions do not have to be latched on the exact same clock).

Write responses can be returned out of order.

3.3.9.4 AXI Memory Bus Ordering

In the AXI architecture, there is no relationship between a requests on read address bus and one driven on the write address bus, even for requests where the ID values or addresses match. The CM ensures the proper ordering between the read and write address requests.

Cacheable accesses use different ID values to allow out-of-order responses. The CM recognizes a Read/Write, Write/Read or Write/Write to the same cache line address. Hence, a 2nd request is not issued onto AXI until response to the first request has been received. Read/Read has no ordering constraints.

3.3.10 Cache Instructions

Operations are performed on the L1I, L1D, and L2 caches using the following instructions:

- **PREF** — This instruction causes data to be moved to or from the cache, to improve program performance. PREF does not cause addressing-related exceptions, including TLB exceptions.

- **FENCE.I** — This instruction synchronizes a data cache line with an instruction cache line. This instruction should be used when writing to the program image in memory to make the newly stored instruction opcodes visible to the instruction fetch logic via the I-Cache. The SYNCI instruction operates on all instruction caches in a cluster. In a multi-cluster system, this means all L1 instruction caches in all clusters.

3.4 Cache Coherency

The P8700-F core defines a set of Cache Coherency Attributes (CCA). The cache coherency is set using the PMA Configuration registers. For more information, refer to the *MIPS RISC-V Customizations* document that is part of the document suite.

The P8700-F core supports the following cacheability attributes:

- *Cacheable, coherent, write-back, write-allocate, read misses request shared. (code #0)*: Use coherent data. Load misses request data in the shared state (will get exclusive if the data is not being shared by another CPU). Multiple caches can contain data in the shared state. Stores bring data into the cache in an exclusive state - no other caches can contain that same line. If a store hits on a shared line in the cache, the line is updated to the exclusive state and any shared copies of the line in other L1 data caches are invalidated.
- *Uncached (code #2)*: Addresses in a memory area indicated as uncached are not read from the cache. Stores to such addresses are written directly to main memory, without changing cache contents.
- *Uncached Accelerated (code #3):* Uncached stores are gathered together for more efficient bus utilization.

3.5 L2 Cache Initialization Options

The P8700-F Multiprocessing System automatically selects hardware cache initialization at reset.

There are two types:

- L2 Tag array only (fast)
- L2 Tag and data arrays (slow)

Automatically selected hardware cache initialization (fast mode) initializes only the L2 tag array.

Each of these options are described in the following subsections.

3.5.1 Automatic Hardware Cache Initialization

The P8700-F MPS allows for the L2 cache to be automatically initialized by hardware when the following conditions are met at reset:

- The external input pin (si_cpc_l2_hw_init_inhibit) is driven low, indicating that automatic hardware initialization can proceed.
- Automatic hardware cache initialization is enabled by setting the L2_HW_INIT_EN bit in the *CPC Local Status and Configuration* register (CPC_Core_STAT_CONF_REG) located at offset 0x0008 in CPC CM-local address space.
- The L2 initialization delay has expired. Once this delay has expired, automatic hardware cache initialization can begin.

- MBIST is not enabled. If it is enabled, the cache initialization does not begin until the MBIST operation is complete. Even if the delay has expired, the cache initialization does not begin until the MBIST has completed.

Once all of these conditions are met, the L2 cache Tag RAM is automatically initialized by hardware. No initialization code is required. Once the initialization is complete, hardware sets the HCI_DONE bit in the *L2 RAM Configuration* register (GCR_L2_RAM_CONFIG) at offset address 0x0240 in GCR address space. Software can poll this bit to determine when the initialization is complete.

3.6 L2 Cache Flush, Burst, and Abort

This section describes the L2 cache flush, burst, and abort operations.

3.6.1 L2 Cache Flush

An L2 flush operation can only be initiated by software. To flush the entire L2 cache in one operation, perform the following steps:

1. Read the L2SM_COP_REG_PRESENT bit in the *L2 Cache Op State Machine Config/Control* register (GCR_L2SM_COP) at offset address 0x0620 in GCR address space to determine if this register is present. A '1' in this bit indicates that the flush cache operation is supported.

2. Read the L2SM_COP_MODE bit in the *L2 Cache Op State Machine Config/Control* register to determine the state of the L2 state machine. This bit must be 0, indicating the state machine is idle, in order for flush operation to proceed.

3. Program the L2SM_COP_TYPE field in bits 4:2 of the *L2 Cache Op State Machine Config/Control* register to a value of 0x0. This selects the full cache flush operation.

4. Program the L2SM_COP_CMD field in bits 1:0 of the *L2 Cache Op State Machine Config/Control* register to a value of 0x1. This starts the cache flush operation.

5. To determine the result of the flush operation, poll the L2SM_COP_RESULT field in bit 8:6 of the *L2 Cache Op State Machine Config/Control* register. A value of 0x0 indicates the process is still running. A value of 0x1 indicates that the process completed with no errors.

3.6.2 L2 Cache Burst Operations

The L2 Cache supports the following burst operations (CacheOps):

- Hit_Inv
- Hit_WB_Inv
- Hit_WB

These operations can be requested only by software and can be performed on a range of addresses in the cache. Burst operations can be executed using the following procedure. Note that the number of cache lines requested must be less than or equal to the available cache lines in the cache and also less than 65,536.

1. Program the starting address where the flush operation begins into the L2SM_COP_START_TAG_ADDR field in bits 47:6 of the *GCR L2 Cache Op State Machine*

Tag Address register (GCR_L2SM_TAG_ADDR_COP) at offset address 0x0628 in GCR address space.

2. Program the L2SM_COP_NUM_LINES field in bits 63:48 of the *GCR L2 Cache Op State Machine Tag Address* register to indicate the number of lines to be flushed from the starting address defined in step 1.

3. Program the type of operation to be performed on each line using the L2SM_COP_TYPE field in bits 4:2 of the *L2 Cache Op State Machine Config/Control* register. A value of 0x4 in this field indicates Hit Invalidate. A value of 0x5 indicates Hit Writeback Invalidate, and a value of 0x6 indicates Hit Writeback.

4. Read the L2SM_COP_MODE bit in the *L2 Cache Op State Machine Config/Control* register to determine the state of the L2 state machine. This bit must be 0, indicating the state machine is idle, in order for the CacheOp to proceed.

5. If the state machine is idle as determined in step 4, program the L2SM_COP_CMD field in bits 1:0 of the *L2 Cache Op State Machine Config/Control* register to a value of 0x1. This initiates the CacheOp starting from the address defined in step 1 and continuing for the number of lines defined in step 2. The operation to be performed in each of the selected cache lines is defined in step 3.

6. To determine the result of the flush operation, poll the L2SM_COP_RESULT field in bit 8:6 of the *L2 Cache Op State Machine Config/Control* register. A value of 0x0 indicates the process is still running. A value of 0x1 indicates that the process completed with no errors.

3.6.3 Abort Operations

During the automatic hardware initialization process described in the section entitled Automatic Hardware Cache Initialization, no coherent requests are permitted. Even if a coherent request is generated during the initialization procedure, it is not allowed to enter the pipeline until the procedure is complete.

Chapter 4

Exceptions and Interrupts

The P8700-F core receives exceptions from a number of sources, misses in the translation lookaside buffer (TLB), I/O interrupts, and environment calls. When the CPU detects an exception, the normal sequence of instruction execution is suspended and the processor enters machine mode, disables interrupts, loads the *Exception Program Counter* (*mepc*) register with the location where execution can restart after the exception has been serviced, and forces execution of a software exception handler located at a specific address.

The software exception handler saves the context of the processor, including the contents of the program counter, the current operating mode, and the status of the interrupts (enabled or disabled). This context is saved so it can be restored when the exception has been serviced.

Exceptions may be precise or imprecise. Precise exceptions are those for which the *mepc* can be used to identify the instruction that caused the exception. For precise exceptions, the restart location in the *mepc* register is the address of the instruction that caused the exception. LDA are examples of precise exceptions.

Imprecise exceptions, on the other hand, are those for which the instruction that caused the exception cannot be identified. Bus error exceptions are examples of imprecise exceptions. Imprecise exceptions are normally attached to the next instruction PC to be graduated. Basically uses the PC (program counter) of very next instruction to graduate as the return address. The instructions which caused imprecise exception may get graduated even before processing the exception, hence these are imprecise exceptions. STA related bus errors are imprecise exceptions.

4.1 Exception Conditions

When an exception condition occurs, the instruction causing the exception and all those that follow it in the pipeline are cancelled. Accordingly, any stall conditions and any later exception conditions that may have referenced this instruction are inhibited.

The term epc in RISC-V can be DEPC, SEPC, or MEPC, where D = Debug, S = Supervisor, and M = Machine.

When the exception condition is detected on an instruction fetch, the CPU aborts that instruction and all instructions that follow. When the instruction graduates, the exception flag causes it to write various CSR registers with the exception state, change the current program counter (PC) to the appropriate exception vector address, and clear the exception bits of earlier pipeline stages.

For most types of exceptions, this implementation allows all preceding instructions to complete execution and prevents all subsequent instructions from completing. Thus, the value in the DEPC/SEPC/MEPC is sufficient to restart execution. It also ensures that exceptions are

taken in program order. An instruction taking an exception may itself be aborted by an instruction further down the pipeline that takes an exception in a later cycle.

The Error PC or Exception PC of the instruction which raised the exception is updated to one of the mepc registers available, based on the mode in which exception is being processed.

Imprecise exceptions are taken after the instruction that caused them has completed and potentially after following instructions have completed.

4.2 Selecting the Exception Address

In the baseline RISC-V RV64 ISA, the exception vector address for several types of exceptions are provided by the trap vector address CSR. The processor mode (Machine, Supervisor) in which exceptions or interrupts are processed will decide the trap vector address CSR. It could be from mtvec CSR or stvec CSR.

MIPS custom exception trap vector address is provided by mipstvec CSR.

In the baseline MIPS RV64 ISA, the exception vector for several types of exceptions is constrained to the lower 512MB of memory. The mtvec, stvec, vstvec, or mipstvec CSR registers can be used to position the base address anywhere in the 256TB 48-bit address space. The GCR.HART.RESET_BASE register also supports specifying a separate reset vector for each thread.

4.3 Debug Exception Processing

All debug exceptions are described in the *MIPS Hybrid Debug Specification*, which is part of the document suite provided.

Chapter 5

Coherence Manager

The Coherence Manager (CM) communicates with all cores and other devices in the P8700-F Multiprocessing System (MPS), as well as coherent devices external to the P8700-F MPS, to achieve system-wide coherence. In a multi-cluster system, the CM also interfaces to an external Network-on-Chip (NOC) controller, which facilitates communication between clusters.

The CM includes an integrated low-latency shared L2 cache. A directory-based coherence protocol is used to efficiently maintain coherence among the L1 data caches of each P8700-F core, with up to eight I/O coherence units (IOCUs), providing the I/O subsystem coherent access to the L1 Data and L2 caches.

This chapter provides an overview of the CM register ring bus and associated table that lists each device ID on the bus. The programmer uses this information to access these devices. An overview of the CM register address space is also provided. In addition, the chapter describes how to program the CM to perform various functions, including setting the base addresses in memory, accessing another Hart in the same core, accessing a Hart in another core, accessing the Interrupt Controller, Cluster Power Controller (CPC), and/or Debug Unit (DBU) registers via the CM, and setting the clock ratios between the various P8700-F system components. For the exact revision number of the Coherence Manager, refer to the Release Notes.

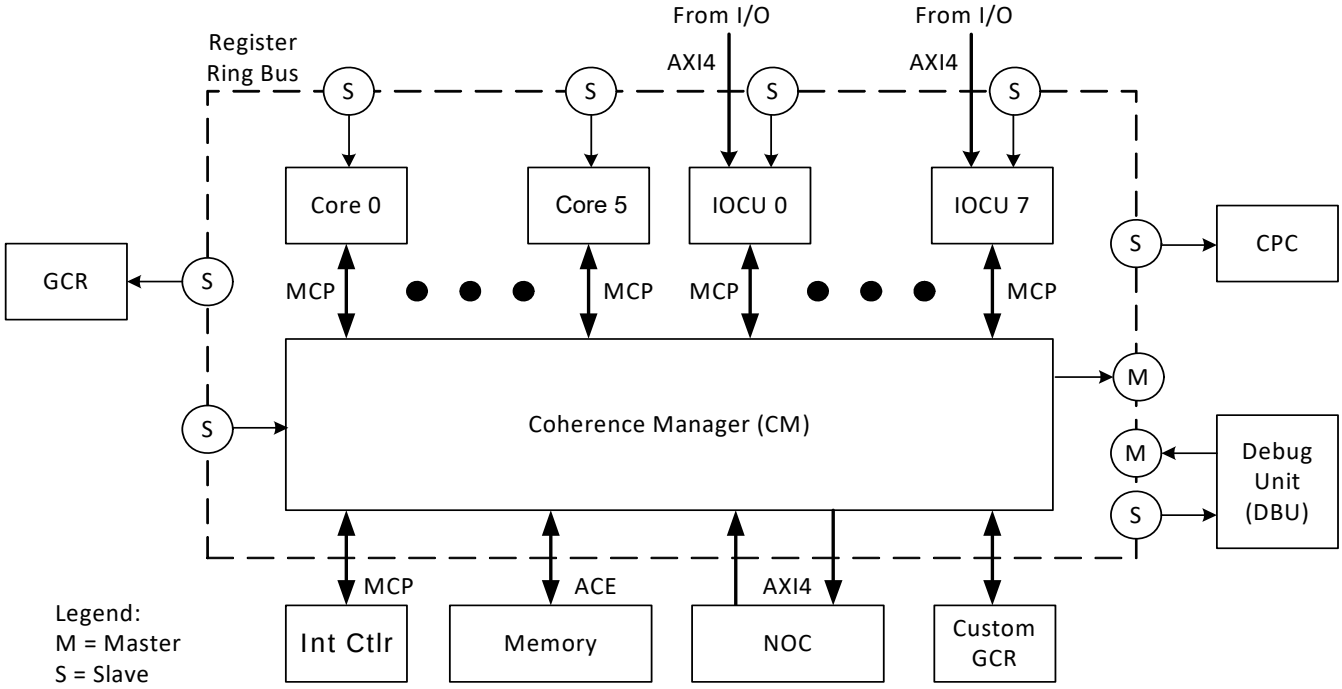
5.1 CM Overview

This section provides an overview of the CM and describes information necessary for programming, including the register ring bus and device ID information, and the CM register map. For details of the CM register MAP, refer to Chapter 3 CM Memory Mapped Registers of MIPS® Technologies Coherence Manager and Advanced Interrupt Controller for RISC-V Cores document.

5.1.1 CM Interface — Register Ring Bus and Device ID’s

The CM communicates with the various system devices via a register ring bus. The devices connected to the CM are shown in Figure 5.1. The P8700-F Multiprocessing System can have up to 6 cores per cluster.

Figure 5.1 Interface Ports and Register Ring Bus Interface to the CM



Certain devices such as the cores and IOCU's connect to the CM via an internal proprietary bus called the MIPS Coherence Protocol (MCP) bus. This bus consists of three unidirectional channels used to maximize throughput. The bus implements a credit-based protocol to allow multiple simultaneous in-flight operations. In the above figure, note that the P8700-F supports up to a total of eight cores and IOCUs together. For example, if there are four cores, there can only be up to four IOCUs.

The CM accesses the registers of the various devices shown in Figure 5.1 using a register ring bus, indicated by the dotted line. As shown above, the CM and DBU can function as both Master (M) and Slave (S). All other devices, including the cores, are slave devices. Each device on the ring bus is assigned a 6-bit ID value stored in the destination ID (dest_id) or source ID (src_id) fields of the packet being sent. When a device initiates an access to the registers of another device, the corresponding ID is attached to the packet. Only the device whose ID number matches that in the packet accepts the transaction. Table 5.1 lists the ID values for each logic block shown in Figure 5.1. These values are used to write to registers in these blocks as described in the following subsections. All values not shown are reserved.

Table 5.1 Register Ring Bus Device ID Values

dest_id / src_id (Decimal value)	dest_id / src_id (Hexadecimal value)	Device Accessed
0	0x00	Core 0
1	0x01	Core 1
2	0x02	Core 2
3	0x03	Core 3
4	0x04	Core 4
5	0x05	Core 5
16	0x10	IOCU0

Table 5.1 Register Ring Bus Device ID Values *(continued)*

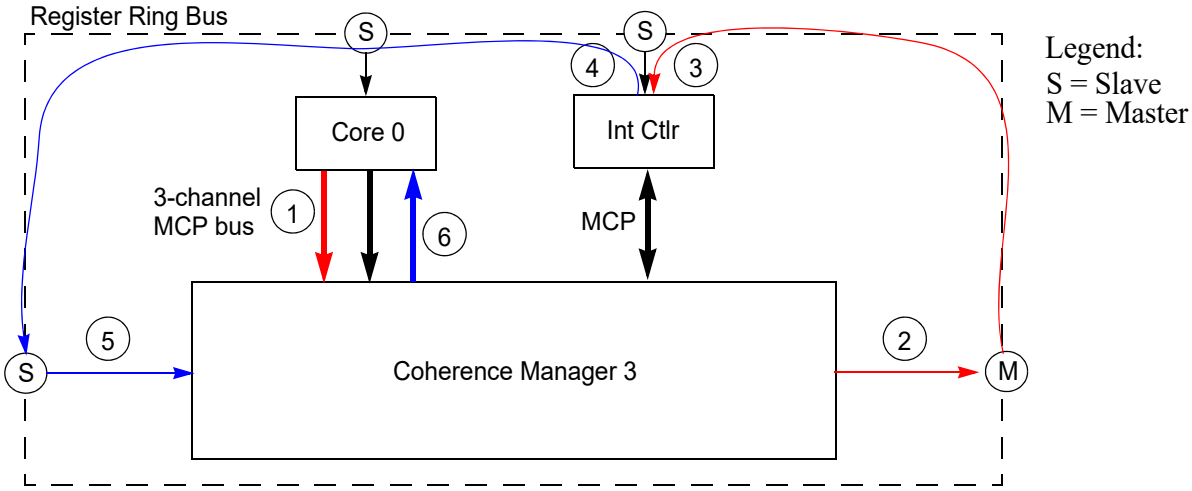
dest_id / src_id (Decimal value)	dest_id / src_id (Hexadecimal value)	Device Accessed
17	0x11	IOCU1
18	0x12	IOCU2
19	0x13	IOCU3
20	0x14	IOCU4
21	0x15	IOCU5
22	0x16	IOCU6
23	0x17	IOCU7
24	0x18	Interrupt Controller
25	0x19	User Defined GCR's
26	0x1A	Memory
32	0x20	CM
33	0x21	CPC
34	0x22	GCR
35	0x23	DBU Master
36	0x24	DBU dmxseg_normal
37	0x25	DBU dmxseg_debug
40	0x28	AUX 0
41	0x29	AUX 1
42	0x2A	AUX 2
43	0x2B	AUX 3
62	0x3E	No Destination Error
63	0x3F	No Destination OK

The following example shows the path taken in order for core 0 to read a register from the Interrupt Controller. The data path for this access is shown in Figure 5.2. This figure is similar to Figure 5.1, except only those devices involved in the example transaction are shown. The red color indicates the access request path, and the blue color indicates the data return path. The following sequence is enumerated in Figure 5.2. In this example the following actions would occur.

1. Core 0 sends a request to the CM over the MCP 'Request' bus. Note that Core 0 cannot access the Interrupt Controller registers directly because it is only a Slave on the ring bus as indicated.
2. The CM processes this request, assigns the appropriate ID number as defined in Table 5.1, and drives this request onto the register ring bus through its Master port.
3. The Interrupt Controller decodes the ID on the bus and gets a match.
4. The Interrupt Controller then fetches the requested data and drives the data onto the ring bus.
5. Data is returned to the CM through its dedicated register ring bus Slave port.

6. The CM sends the requested data back to Core 0 over the dedicated MCP 'Response' bus.

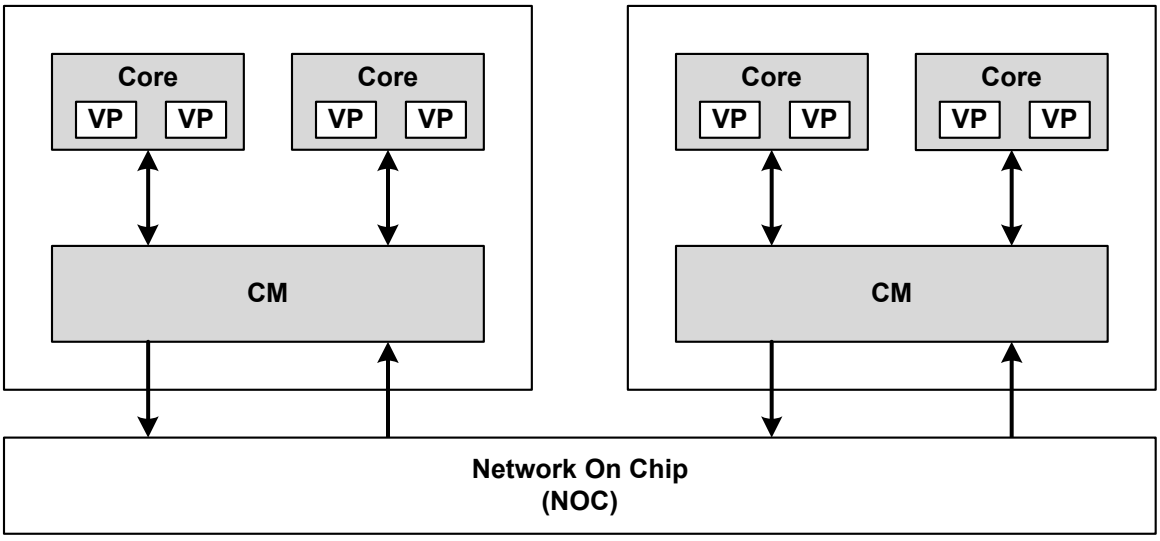
Figure 5.2 Data Path of Core 0 Access of IOCU0 Registers



5.1.2 Cluster to Cluster Accesses

In addition to facilitating core-to-core and Hart-to-Hart accesses within the same cluster, the P8700-F also allows for cluster-to-cluster accesses. This allows a core or Hart (VP) in one cluster to access the registers in a core or Hart of another cluster through the Network-On-Chip (NOC) interface. This interface is shown in Figure 5.3.

Figure 5.3 Cluster-to-Cluster Register Accesses Using the NOC



For example, a Hart within a core in Cluster 1 can access and update a register in a Hart in Cluster 2 as shown. The access is processed by the CM and driven onto the NOC. The NOC then routes the request to the appropriate cluster where the access is scheduled by the CM in the destination cluster.

If a register access is within a given cluster as shown above, the NOC is not used and the access is placed onto the Register Ring Bus (RRB) described in the section entitled CM

Interface — Register Ring Bus and Device ID's. If the register access is to another cluster, the NOC is used to transfer the access request where it is placed onto the RRB of the destination cluster. There are dedicated unidirectional AXI bus interfaces that move the access from the cluster to the NOC, and from the NOC to the cluster. A separate bidirectional bus is used to manage coherence as shown above.

5.2 Verifying Overall System Configuration

At IP configuration time, the customer selects the number of cores in the system, the number of I/O coherency units (IOCU's), and the number of address regions. When the device is built, these values are hard wired into the *Global Configuration* register at offset address 0x0000. All of these fields are read-only and allow kernel software to quickly determine the system configuration.

CM GCR Register Interface

Reading the *Global Configuration* register provides the following information:

- Bits 7:0 — Number of cores in the system (up to 6)
- Bits 11:8 — Number of IOCU's (up to 8)
- Bits 19:16 — Number of MMIO address regions
- Bits 22:20 — Number of auxiliary memory ports
- Bits 29:23 — Number of clusters in the system
- Bit 31 — Indicates if an Inter-Thread Communication Unit is present
- Bits 39:32 — Indicates the ID number for the current cluster. Each cluster has a unique ID number.
- Bit 40 — Indicates if a Debug Unit is present
- Bits 43:41 — Indicates the type of hardware interface to the Network-On-Chip (NOC) coherent interconnect.

5.3 Programming the Base Addresses in Memory

This section describes how to set the base address of the CM logic.

5.3.1 CM GCR Register Interface

The address map is programmable through the GCR_BASE register as summarized in Table 5.2.

Table 5.2 Setting the Base Address for the GCR_BASE Register

Block	Register Name	Offset Address	Field Name	Bits	Description
GCR	GCR_BASE	0x0008	GCR_BASE_ADDR	47:19	GCR Base Address register. Sets the base address of the GCR registers. Note that this region must reside on a 512 KB boundary.

5.4 CM Register Access Permissions

A requestor can request access to selected CM registers. A requestor can be either a core or an IOCU. The CM allows up to eight requestors in a system in any combination of cores and IOCU's, from 6 cores and up to two IOCU's, to 8 IOCU's and no cores, or anywhere in between. Note that there can only be a combined total of 8 cores/IOCU's.

5.4.1 Enabling Access Permissions

Access permissions to the CM GCR registers follows the memory access permission rules as defined in the Physical Memory Protection (PMP) section of the *RISC-V Privileged Architecture Manual*. Privileged code can program the PMP registers to control which CM registers are accessible from each privileged mode on each Hart.

5.5 Coherency Enable

The P8700-F Multiprocessing System allows each power domain to be placed in either a coherent or non-coherent mode. Because the P8700-F implements a directory-based coherence protocol, MIPS recommends that each domain be placed in coherent mode during normal operation. The non-coherent mode should only be used during boot-up and power-down. Software should not execute any cacheable memory accesses (instruction fetch or load/store) while coherence is disabled.

In the CM, coherency is either enabled or disabled using the *Coherence Enable (COH_EN)* register. There is one of these registers per core. Each register can be accessed at address: 0x020f8 + 0x100 * CORENUM + GCR_BASE for Core 0 through 5.

5.6 L2 Cache Prefetch

The coherence manager in the P8700-F MPS contains an L2 prefetcher used to enhance L2 performance. The L2 prefetcher is managed using two CM GCR registers.

- L2 Prefetch Control register (GCR_L2_PFT_CONTROL) at offset 0x0300

- L2 Prefetch 2nd Control register (GCR_L2_PFT_CONTROL_B) at offset 0x0308

These registers control the following L2 capabilities:

- Minimum operating system page size (supports 4K - 64K pages in multiples of two)
- Prefetch enable
- Coherent invalidate requests
- Code prefetch enable
- L2 prefetching port ID. Each bit corresponds to a CM port ID. If the bit is set, the corresponding CM port is monitored for prefetching.

5.6.1 Prefetch Enable

The number of prefetch units implemented in the P8700-F Multiprocessing System is determined by the user during IP configuration. This value is programmed by hardware into the NPFT field (bits 7:0) of the L2 Prefetch Control register (GCR_L2_PFT_CONTROL) located at offset address 0x0300 in the GCR Global register space. This read-only field allows kernel software a convenient way to determine the number of prefetch units implemented.

CM GCR Register Interface

Prefetching is enabled by setting the PFTEN bit in the GCR_L2_PFT_CONTROL register. Note that the number of prefetch units implemented as described above must be greater than 0 in order for this bit to have meaning.

5.6.2 Select Ports for L2 Prefetching

The CM allows up to 8 ports to be selected for L2 prefetching. These ports correspond to the (up to) six cores and (up to) eight IOCU's as shown in Figure 5.1. L2 prefetching can be selected for some of all of these ports using the 8-bit PORT_ID field in the GCR_L2_PFT_CONTROL_B register. Each bit of this field corresponds to a single port. There can be any number of cores and IOCU's up to the maximum or eight. For example, if there are 6 cores, then there can only be up to 2 IOCU's to make a total or 8, or 4 cores and 4 IOCU's, etc. If a given bit is set, L2 prefetching is monitored for that port. If the bit is cleared, L2 prefetching does not occur.

The field is organized as cores followed by IOCU's starting from bit 0. So in a 4-core and 2-IOCU system, bits 0 - 3 of the field would represent cores 0 - 3 respectively. Bits 4 - 5 of the field would represent IOCU 0 - 1 respectively. Bits 6 - 7 would not be used in this example.

5.6.3 Enabling Code Prefetch

In addition to data prefetching, the CM allows prefetching of the code stream. Code prefetching is enabled by setting the CEN bit in the GCR_L2_PFT_CONTROL_B register.

5.7 CM Uncached Semaphore Management

The P8700-F CM provides a mechanism for managing uncached semaphores. This mechanism is managed by the *Global CM Semaphore* (GCR_SEM) register located at offset address 0x0640.

A write to this register with write data bit 31 = 1 is inhibited if the SEM_LOCK bit is already 1. A write to this register proceeds normally if the write data has bit 31 = 0 or if the SEM_LOCK bit is currently 0.

CM GCR Register Interface

To acquire the semaphore:

1. Write this register with bit 31 = 1 and the lower bits with the threads VPID.
2. Read the register.
3. If the value read in step #2 is the same as the value as written in step #1, then a semaphore has been acquired, else go to step #1.

To release the semaphore:

1. Write the register with bit 31 = 0.

5.8 Custom GCR Implementation

The CM provides the ability for the system designer to implement a 64 KB block of custom registers that can be used to control system level functions. These registers are defined by the system designer and then instantiated into the design.

The existence of a custom GCR implementation in the system is selected during IP Configuration. If this option is selected, the GGU_EX bit is set in the *Global Custom Status* register at offset address 0x0068 in GCR Global address space. This bit indicates that a custom GCR block is connected to the CM.

5.9 IOCU Interface

The P8700-F CM contains up to eight I/O Coherency Units (IOCU) for managing cache coherency between the CM and external devices. The IOCU is a hardware block and is not directly programmable. However, the IOCU can be indirectly controlled using the following register fields:

- The read-only NUMIOCU field in bits 11:8 of the *Global Config* register (GCR_CONFIG) located at offset 0x0000 of CM GCR address space and indicates the number of IOCUs instantiated in the design. This field is filled by hardware during IP configuration.
- IOCU requests are prevented from being issued to MMIO regions by setting the bit 13 of the *Global CM Control* register (GCR_CONTROL) at offset 0x0010 in CM GCR address space.
- IOCU requests to external devices are counted toward the outstanding request limit when bit 12 of the *Global CM Control* register (GCR_CONTROL) at offset 0x0010 in CM GCR address space. If this bit is set, IOCU accesses to MMIO regions are blocked once the MMIO outstanding limit is reached. Note that bit 13 of this register must be 0 for this bit to have meaning as described above.
- Software can select which IOCUs are allowed to access the CM GCR registers by programming bits 23:16 of the *Global CSR Access Privilege* register (GCR_ACCESS) at offset 0x0120 in CM GCR address space. Each bit corresponds to one of eight IOCUs. If the corresponding bit is set, accesses from that IOCU are allowed to write the GCR and Cluster Power Controller (CPC) registers.

5.10 MMIO Address Regions

As described in the section entitled Verifying Overall System Configuration, the number of MMIO address regions is determined at IP configuration time. The P8700-F supports up to four MMIO regions. Each region is assigned an upper and lower address bound.

The MMIO regions are intended to be used with communicating with external PCIe devices. The MMIO registers allow for counting of number of non-speculative code fetches of uncached requests in order to avoid potential deadlock condition by having too many requests outstanding. This is accomplished by programming the MMIO_REQ_LIMIT field.

5.10.1 CM GPR Register Interface

Software can set the number of MMIO requests that can be in-flight at any given time by programming the MMIO_REQ_LIMIT field of the MMIO Request Limit register (GCR_MMIO_REQ_LIMIT) at offset 0x6F8.

In addition, the address range of each MMIO region is defined using the Upper and Lower Bound MMIO region registers. A pair of registers are used for each MMIO region, with each register containing a 32-bit address bound value. These registers are located at:

- Lower bound of MMIO region 0 (GCR_MMIO0_BOTTOM) at offset 0x0700
- Upper bound of MMIO region 0 (GCR_MMIO0_TOP) at offset 0x0708
- Lower bound of MMIO region 1 (GCR_MMIO1_BOTTOM) at offset 0x0710
- Upper bound of MMIO region 1 (GCR_MMIO1_TOP) at offset 0x0718
- Lower bound of MMIO region 2 (GCR_MMIO2_BOTTOM) at offset 0x0720
- Upper bound of MMIO region 2 (GCR_MMIO2_TOP) at offset 0x0728
- Lower bound of MMIO region 3 (GCR_MMIO3_BOTTOM) at offset 0x0730
- Upper bound of MMIO region 3 (GCR_MMIO3_TOP) at offset 0x0738

5.10.2 MMIO Region Control

Each of the four MMIO regions listed above can be enabled or disabled by programming the MMIO_EN bit that resides in the Lower Bound register for each MMIO region (GCR_MMIO[0-3]_BOTTOM). If the MMIO region is enabled, then the request address and CCA are used to determine if the request falls into an MMIO Region. The decoded address is used to determine if the access is to a MMIO region as shown in the following equation:

MMIO_BOTTOM_ADDR[47:16] <= phys_address[47:16] <= MMIO_TOP_ADDR[47:16]

If bits 47:16 of the physical address fall between the value in MMIO_BOTTOM_ADDR[47:16] and MMIO_TOP_ADDR[47:16], then the access is to the corresponding MMIO region.

If MMIO_CCA is set to 0x0, just the request address is used to determine whether the request is to an MMIO region as shown above. If MMIO_CCA is set to 0x01, then the address comparison above is further qualified by whether the request has CCA = UC. In other words, only UC requests will be considered eligible to hit the MMIO region. If MMIO_CCA is set to 0x2, then the request is qualified by CCA = UCA. If MMIO_CCA = 0x3, then the request is qualified by CCA = UC or CC = UCA. In other words, either UC or UCA requests can match the MMIO region.

If an address hits in multiple MMIO register address regions, then the lowest-numbered enabled MMIO region hit takes precedence for determining which MMIO region the request

matches. Once a request is determined to reside in an MMIO region, that region MMIO_PORT field in the Lower Bound register determines where the request will be routed. Options are the main memory port or an Auxiliary interface. See section 5.13.

The user can limit the total number of MMIO requests issued by the CM, which can be useful to avoid deadlock when accessing PCIe bridges that also service incoming coherent requests. The limit is defined by the MMIO_REQ_LIMIT field in bits 7:0 of the MMIO Request Limit (GCR_MMIO_REQ_LIMIT) register at offset 0x06F8 in GCR address space. Once the limit is reached, the CM stops serializing uncached and code fetches until a response to an MMIO request has been received. For example, a value of 0x01 in this field indicates one outstanding MMIO request is permitted. Setting this value to 0x00 disables the MMIO limiting feature, allowing any amount of outstanding requests to occur. The MMIO_DISABLE_REQ_LIMIT bit in the region's Lower Bound Register can be set to indicate that requests to the particular MMIO region should not be limited.

By default, IOCU uncached requests are never considered part of the MMIO limit (to allow for forward progress). However, this is controllable via the GCR_CONTROL.CM_MMIO_IOCU_ENABLE_REQ_LIMIT. When this bit is set, IOCU uncached requests are counted as outstanding MMIO requests. In this case, IOCU uncached requests are blocked if the MMIO request limit has been reached.

5.11 Auxiliary Interfaces

The CM supports up to four non-coherent Auxiliary AXI4 buses, called AUX0 - AUX3. The AUX master ports are intended to be used for lower latency access to peripherals or instruction SRAM. Each cluster supports up to four AUX ports. Each AUX interface has a configurable data width. Values of 32, 64, 128, 256 and 512 are supported. The data width is determined during IP configuration. Each AUX address width is 48 bits. The number of AUX ports is stored in the 3-bit NUMAUX field of the *Global Configuration register* (GCR_CONFIG) at offset 0x0000 in GCR address space.

The clock for each AUX interface can be provided internally by the cluster or provided externally from outside the cluster. Each internally provided AUX clock can have an independent clock ratio. An externally provided clock can be provided on the external AUX clock pin. An externally provided clock is assumed to be asynchronous to the cluster. Selection between an internal versus external clock is done during IP configuration.

The AUX ports are memory mapped by the MMIO GCR control registers. There are up to 4 MMIO regions. Each GCR_MMIO<x>_BOTTOM register listed above contains an MMIO_PORT field in bits 5:2 that indicates which auxiliary port the request should be routed to. This field is encoded as shown in Table 5.3.

Table 5.3 Encoding of MMIO_PORT Field

Field Name	Register Bits	Encoding	Port Accessed
MMIO_PORT	5:2	0x0	Main memory
		0x8	AUX port 0
		0x9	AUX port 1
		0xA	AUX port 2
		0xB	AUX port 3

5.12 Error Processing

The CM detects, reports, and handles several types of errors that may be caused by errant software or hardware soft or hard errors. When an error is detected, information that may be useful in debugging the error is captured in the *Global CM Error Cause Register* and *Global CM Error Address Register*.

When an error occurs, hardware updates the read-only ERR_TYPE field in bits 63:58 of the *Global CM Error Cause* register with one of the values listed in Table 5.3 above. When this field is written, hardware also updates the 58-bit ERROR_INFO field that provides additional information about the error. The organization of this field varies depending on the value in the ERR_TYPE field.

When a second error is detected, it will overwrite the first error if the first error was an L2 ram correctable error (MP_CORRECTABLE_ECC_ERR). Otherwise, the second error is captured in the *CM Error Multiple Register*. Note that for the second error, only the error type is captured, not the associated error address or error information.

The *GCR_ERROR_CAUSE.ERR_TYPE* field and the *GCR_ERROR_MULT.ERR_TYPE* fields can be cleared by either a reset or by writing the current value of *GCR_ERROR_CAUSE.ERR_TYPE* to the *GCR_ERROR_CAUSE. ERR_TYPE* register.

When the *Global CM Error Cause Register* is loaded, an interrupt may be generated if the corresponding bit for that type of error is set in the *Global CM Error Mask Register* located at offset address 0x0040.

One distinction between error management in the CM and the previous generation CM2-based products is in error responses when the Error Mask register is set. In CM2-based products;

- If the error was generated by a request that requires a response and the corresponding *Global CM2 Error Mask Register* bit is 0, then the CM2 issues an ERROR response.
- If the corresponding *Global CM2 Error Mask Register* bit is 1, then the CM2 issues a normal response and an interrupt is generated instead.

In the CM version in the P8700-F, the error response is independent of the mask setting. If the normal response should be an ERROR, then an ERROR response is returned regardless of the *Error Mask Register* setting. The mask setting controls whether an interrupt is generated in addition to the normal error response.

Table 5.4 lists the errors detected by the CM. The following subsections describe each type of error in more detail and provide the encoding of the ERR_INFO field for each error type.

Table 5.4 CM Error Types

Error Type	Error Name	Description	Action
0	-	Reserved	-
1	*MP_CORRECTABLE_ECC_ERR*	A correctable ECC error occurred during an L2 cache access.	The error is corrected. Signal an interrupt if CM_ERROR_MASK[1] = 1
2	*MP_REQUEST_DECODE_ERR*	A decoding error was detected in the request.	Respond with an error to the original requestor. Signal an interrupt if CM_ERROR_MASK[2] = 1

Table 5.4 CM Error Types *(continued)*

Error Type	Error Name	Description	Action
3	*MP_UNCORRECTABLE_ECC_ERR*	An uncorrectable ECC error occurred during an L2 cache access.	Signal an interrupt if *CM_ERROR_MASK[3]* = 1
4	*MP_PARITY_ERR*	A parity error was detected in the L2 data coming from either the core or the memory.	Signal an interrupt if *CM_ERROR_MASK[4]* = 1
5	*MP_FNL_ERR*	If an L2 fetch and lock (FNL) cacheop is processed when only one or zero ways of the cache are unlocked, including pseudo-locks, then the FNL fails.	Signal an interrupt if *CM_ERROR_MASK[5]* = 1
6	*CMBIU_REQUEST_DECODE_ERR*	A decoding error was detected during a request on the BIU.	Signal an interrupt if *CM_ERROR_MASK[6]* = 1
7	*CMBIU_PARITY_ERR*	The BIU detected a parity error.	Signal an interrupt if *CM_ERROR_MASK[7]* = 1
8	*CMBIU_AXI_RESP_ERR*	The BIU detected a response error was detected on the AXI bus.	Signal an interrupt if *CM_ERROR_MASK[8]* = 1
9	*Reserved*	Reserved	Reserved
10	*RBI_BUS_ERR*	An error occurred during a register ring bus during a register access.	Signal Interrupt if *CM_ERROR_MASK[10]* = 1
11	*IOC_REQUEST_ERR*	An error occurred on an IOCU request on the AXI bus.	Signal Interrupt if *CM_ERROR_MASK[11]* = 1
12	*IOC_PARITY_ERR*	The IOCU detected a parity error.	Signal Interrupt if *CM_ERROR_MASK[12]* = 1
13	*IOC_RESP_ERR*	The IOCU detected a response error.	Signal Interrupt if *CM_ERROR_MASK[13]* = 1
14	*Reserved*	Reserved	Reserved
15	*RBI_REGTC_REQ_ERR*	An illegal request was received by the REGTC bus during a NOC access.	Signal Interrupt if *CM_ERROR_MASK[15]* = 1

5.12.1 Error Codes 1 and 3 — Tag ECC Error

If the decimal value in the ERR_TYPE field is either 1 or 3 and there is a Tag ECC error, the ERROR_INFO field in the *Global CM Error Cause* register is organized as shown in Table 5.5

Table 5.5 State of ERR_INFO Field for Tag Error Types 1 or 3

Bit	Meaning
57	Error type 0: Tag error 1: Data error
56:45	Reserved
44:29	Indicates the way of the cache that caused the error. There is one bit per way as follows: Bit 29: way 0 Bit 30: way 1 Bit 31: way 2 ... Bit 44: way 15
28	Bank in which the error occurred. 0: Bank 0 1: Bank 1
27:22	Core ID value. The first IOCU encoding is always directly after the last core encoding. For example, in a system with four cores and two IOCU's, the cores would occupy encoding 0x0 - 0x3, and the IOCU's would occupy encoding 0x4 - 0x5. So 0x0 - 0x[n] = cores, and 0x[n+1] - 0x[m] = IOCU's. The following example shows the encoding for a system with six cores and two IOCU's. 0x0: core 0 0x1: core 1 0x2: core 2 0x3: core 3 0x4: core 4 0x5: core 5 0x6: IOCU 0 0x7: IOCU 1
21:18	Hart ID value. 0x0: Hart 0 0x1: Hart 1 0x2: Hart 2 0x3: Hart 3
17:14	Command. This field indicates the command type. Refer to Table 5.7 through Table 5.10 for the encoding of this field.
13:11	Command Group. This field indicates the command group. Refer to Table 5.6 for the encoding of this field.
10:8	Cache Coherency Attribute (CCA) value. This field indicates the CCA value corresponding to the transaction. Refer to Table 5.11 for the encoding of this field.

Table 5.5 State of ERR_INFO Field for Tag Error Types 1 or 3 *(continued)*

Bit	Meaning
7:5	MCP bus transfer size. Indicates the size of the transfer on the bus. This field is encoded as $2^{(MCP\ size)}$. 0x0: 1 byte 0x1: 2 bytes 0x2: 4 bytes 0x3: 8 bytes 0x4: 16 bytes 0x5: 32 bytes (Reserved. Not used in the P8700-F) 0x6: 64 bytes 0x7: 128 bytes (Reserved. Not used in the P8700-F)
4:1	Transaction type. This field indicates the type of bus transaction that caused the error. Refer to Table 5.12 for the encoding of this field.
0	Scheduler. The P8700-F core can be configured at build time with either 1 or 2 pipeline schedulers. If the build is configured with one scheduler, this bit is always 0. If configured with two schedulers, this bit can be either 0 or 1 and indicates the scheduler involved in the error.

5.12.1.1 Command Group Field Encoding

Bits 13:11 indicate the type of command group. The command group is used along with the command to specify the operation to be performed. Memory reads and writes (cacheable as well as non-cacheable) usually use the "NORM" command group. Some special cache maintenance operations (L1I, L1D, L2, L3) must be able to target a specific cache level as well as specify the operation to be performed. The encoding for the different values is given in the table below.

This field is decoded as shown in Table 5.6. The encoding table for each of these command group types are described in the following subsections.

Table 5.6 Command Group Field Encoding

Encoding	Mnemonic	Description	Usage
0	NORM	Normal loads and stores use this space.	Normal loads and stores
1	REGS	Register reads / writes and sync operations.	Register access and sync
2	GBL	Globalized (to local and other clusters) I-cache and TLB invalidates.	Global instruction cache and TLB maintenance
3		Reserved	N/A
4	L1I	The command is targeted at the level 1 instruction cache.	Cache maintenance operations.
5	L1D	The command is targeted at the level 1 data cache.	
6	L3	The command is targeted at the level 3 cache.	
7	L2	The command is targeted at the level 2 cache.	

NORM Command Field Encoding

Bits 17:14 in Table 5.6 indicate the type of command to be performed. When the Command Group field in bits 13:11 is set to 3'b000, indicating the NORM field encoding, the Command field in bits 17:14 is decoded as shown in Table 5.7.

Table 5.7 NORM Command Field Encoding

Encoding	Mnemonic	Description
0	Read	Legacy read.
1	Write	Legacy write.
2	CohReadOwn	Requests an exclusive copy of the cache line.
3	CohReadShare	Requests a shared copy of the cache line.
4	CohReadDiscard	Request the latest copy of the cache line and is leaving the coherent domain.
5	CohEvict	The line has been evicted from the cache without a change. The directory can be updated.
6	CohUpgrade	Request ownership of a shared cache line.
7	CohUpgradeSC	Request ownership of a shared cache line for the purpose of executing a Store Conditional instruction.
8	CohWriteBack	Transfers ownership of a cache line back to the next level along with the new copy of the line data.
9	CohWriteInvalidate	Injects new, possibly sub cache line data into a coherent system. This command is only valid from the L1 to the L2 and is also called a Commit to L2.
10	CohReadDiscardAlloc	Request the latest copy of the cache line and is leaving the coherent domain. The next level cache should allocate the line if no present. This command is expected to be used for cacheable instruction fetches.
11	CohPrefOwn	This command attempts to pre-fetch the specified line in to the L2 cache in the "exclusive" state. If the line already exists in the cache in the exclusive or modified states, then this command does not change the line. Otherwise, a command needs to get sent to the next level to gain ownership of the line. No data is returned to the requestor.
12	CohPrefShr	This command attempts to pre-fetch the specified line in to the L2 cache in the "shared" state. If the line already exists in the cache, then this command does not change the line. Otherwise, a command needs to get sent to the next level to obtain a shared copy of the line. No data is returned to the requestor.
13	CohPrefWriteInv	This prefetch command is similar to the CohPrefOwn command but in addition to bringing the cache line in to the L2 in one of the 'exclusive' states, it makes sure that the line is not currently owned by any L1. This command is not expected to be issued by a core but can be used by the L2 prefetcher within the CM main pipeline.
14	CohGetOwn	This command is used to get ownership of the cache line from the next level without asking for the data. This command can only be issued when the entire cache line is being overwritten and is not expected to be issued by the core.

Table 5.7 NORM Command Field Encoding *(continued)*

Encoding	Mnemonic	Description
15	TagErr	This command is used to indicate that a tag error has been detected by the requestor as it tried to send out a command. This command is typically used on a write type command where the data has already been sent out on the WID channel and an error is detected while trying to generate the address for the request. This command is sent to the next level so that the SDB Id is not left hanging. The receiver just frees up the resources as it processes the command sending back a response without data.

REGS Command Field Encoding

Bits 17:14 in Table 5.6 indicate the type of command to be performed. When the Command Group field in bits 13:11 is set to 3'b001, indicating the REGS field encoding, the Command field in bits 17:14 is decoded as shown in Table 5.8.

Table 5.8 REGS Command Field Encoding

Encoding	Mnemonic	Description
0	DbgRead	Debug Read. This is used by the core (and CM to CMBIU) for debug register reads (DMXSEG, DRSEG and CSR).
1	DbgWrite	Debug Write. This is used by the core (and CM to CMBIU) for debug register writes (DMXSEG, DRSEG and CSR).
2	RegRead	Register Read. This is used by the core for Fast Debug Channel (FDC) reads. This is used by CM to CMBIU for both FDC reads and memory mapped register reads.
3	RegWrite	Register Write. This is used by the core for Fast Debug Channel (FDC) writes. This is used by CM to CMBIU for both FDC writes and memory mapped register writes.
4 - 7		Reserved.
8	MemSync0	This is used for memory synchronization operations and has a type of 0. This value does not correspond to the "stype" field of a SYNC instruction.
9	MemSync1	This is used for memory synchronization operations and has a type of 1. This value does not correspond to the "stype" field of a SYNC instruction.
10	MemSync2	This is used for memory synchronization operations and has a type of 2. This value does not correspond to the "stype" field of a SYNC instruction.
11	MemSync3	This is used for memory synchronization operations and has a type of 3. This value does not correspond to the "stype" field of a SYNC instruction.
12 - 15		Reserved.

GBL Command Field Encoding

Bits 17:14 in Table 5.6 indicate the type of command to be performed. When the Command Group field in bits 13:11 is set to 3'b010, indicating the GBL field encoding, the Command field in bits 17:14 is decoded as shown in Table 5.9.

Table 5.9 GBL Command Field Encoding

Encoding	Mnemonic	Description
0	GBL_HIT_INVI	Invalidate the specified Physical Address (PA) in all I-caches.
1	GBL_ONE_INVI	Invalidate all addresses in one I-cache, selected by the General Number Register (GNR).
2	GBL_ALL_INVI	Invalidate all addresses in all I-cache.
3		Reserved
4	GBL_GINVGT	Guest invoked, Invalidate one or many lines except for wired in matching Guest TLB.
5	GBL_RINVGT	Root invoked, Invalidate one or many lines including wired in matching Guest TLB
6	GBL_INVT	Invalidate one or many lines in Root TLB, except for wired entries.
7	GBL_SYNC	Sync and return only when all previous Global group commands have completed their tasks.
8 - 15		Reserved.

Cache Maintenance (L1I, L1D, L2, L3) Command Field Encoding

Bits 17:14 in Table 5.6 indicate the type of command to be performed. When the Command Group field in bits 13:11 is set to 3'b100 through 3'b111, indicating the Cache Maintenance encodings, the Command field in bits 17:14 is decoded as shown in Table 5.10.

The first set of encodings correspond to the encoding of bits [20:18] of the CACHE instruction. The last encoding is only valid for the L1I command group.

Table 5.10 Cache Maintenance Command Field Encoding

Encoding	Mnemonic	Description
0	IdxWbInval	This command corresponds to the "Index invalidate / Index write-back invalidate" CacheOp. Write-back caches flush out the data to the next level if the line was dirty. All caches invalidate the line at the end of the operation.
1	IdxLdTag	This command corresponds to the "Index load tag / data" type CacheOp. The tag and data RAMs are read out at the location specified by the index and returned with the response.
2	IdxStTag	This command corresponds to the "Index store tag" type CacheOp. This command is accompanied with write data that contains the tag bits to be written.

Table 5.10 Cache Maintenance Command Field Encoding *(continued)*

Encoding	Mnemonic	Description
3	Impl / Reserved	This command corresponds to the "Implementation Dependent" CacheOp. This command is currently unsupported and considered reserved.
4	ConInvalidate / HitInvl	This command corresponds to the "Hit invalidate" type CacheOp or the "Coherent Invalidate" command on the OCP 3.0 bus protocol. It indicates that the addressed line needs to be invalidated irrespective of its ownership status.
5	CohCopyBackInval / HitWbInvl	This command corresponds to the "Hit Write Back Invalidate" type CacheOp or the "Coherent Copy Back Invalidate" command on the OCP 3.0 bus protocol. It indicates to the system that the addressed line needs to be flushed from the system if in a dirty state and invalidated.
6	CohCopyBack / HitWb	This command corresponds to the "Hit Write Back" type CacheOp or the "Coherent Copy Back" command on the OCP 3.0 bus protocol. It indicates that the addressed line needs to be written out to memory if in a dirty state. The line can continue to stay valid in the caches if already present.
7	FetchNLock	This command corresponds to the "Fetch and Lock" type CacheOp. The line should be brought in to the cache and locked so that it does not get evicted due to random replacement.
8 - 15		Reserved.

5.12.1.2 CCA Field Encoding

Bits 10:8 indicate the cache coherency attribute. This field is decoded as shown in Table 5.11.

Table 5.11 Cache Coherency Attributes Field Encoding

CCA[10:8]	Attribute
3'b000	Mapped to '3b101 (Cached Coherent Read-Share).
3'b001	Mapped to '3b101 (Cached Coherent Read-Share).
3'b010	Uncached.
3'b011	Mapped to '3b101 (Cached Coherent Read-Share).
3'b100	Mapped to '3b101 (Cached Coherent Read-Share).
3'b101	Cached Coherent Read-Share.
3'b110	Mapped to '3b101 (Cached Coherent Read-Share).
3'b111	Uncached Accelerated.

5.12.1.3 Type Field Encoding

Bits 4:1 indicate the type of transaction when the error occurred. This field is decoded as shown in Table 5.11.

Table 5.12 Type Field Encoding

Encoding	Mnemonic	Description
0	ReqNoData	Normal request with no associated data. Used for most requests.
1		Reserved
2	ReqWData	Normal request with associated data. Used for stores & write back requests.
3		Reserved
4	IReqNoResp	Intervention request with no response required.
5	IReqWResp	Intervention request with a response required.
6	IReqNoRespDat	Intervention request with associated data and no response required.
7	IReqWRespDat	Intervention request with associated data and response required.
8	RespNoData	Normal response with no data returned.
9	RespDataFol	Normal response with data to follow on a different transaction.
10	RespWData	Normal response with data being returned (3 clocks later).
11	RespDataOnly	Normal response with data being returned (3 clocks later) as a consequence of a "data-to-follow" response.
12	IRespNoData	Intervention response with no data returned.
13	IRespDataFol	Intervention response with data to follow on a different transaction.
14	IRespWData	Intervention response with data being returned (3 clocks later).
15	IRespDataOnly	Intervention response with data being returned (3 clocks later) as a consequence of a "data-to-follow" response.

5.12.2 Error Codes 1 and 3 — Data ECC Error

If the decimal value in the ERR_TYPE field is either 1 or 3 and there is a Data ECC error, the ERROR_INFO field in the *Global CM Error Cause* register is organized as shown in Table 5.13

Table 5.13 State of ERR_INFO Field for Data Error Types 1 or 3

Bit	Meaning
57	Error type 0: Tag error 1: Data error
56:49	DWORD with error. This field indicates the DWORD that caused the error.
48:45	Indicates the way of the cache that caused the error. This field is encoded as follows. Note that this field is handled differently from the Tag error shown in Table 5.5, where the field is one bit per way. 0x0: way 0 0x1: way 1 0x2: way 2 ... 0xF: way 15

Table 5.13 State of ERR_INFO Field for Data Error Types 1 or 3 *(continued)*

Bit	Meaning
44:29	Indicates which one of up to 8K sets of the cache that caused the error. This field is encoded as follows: 0x0000: set 0 0x0001: set 1 0x0002: set 2 ... 0x1FFE: set 8,190 0x1FFF: set 8,191
28	Bank in which the error occurred. 0: Bank 0 1: Bank 1
27:22	Core ID value. The first IOCU encoding is always directly after the last core encoding. For example, in a system with four cores and two IOCU's, the cores would occupy encoding 0x0 - 0x3, and the IOCU's would occupy encoding 0x4 - 0x5. So 0x0 - 0x[n] = cores, and 0x[n+1] - 0x[m] = IOCU's. The following example shows the encoding for a system with six cores and two IOCU's. 0x0: core 0 0x1: core 1 0x2: core 2 0x3: core 3 0x4: core 4 0x5: core 5 0x6: core 6 0x7: core 7 0x8: IOCU 0 0x9: IOCU 1 0xA: IOCU 2 0xB: IOCU 3 0xC: IOCU 4 0xD: IOCU 5 0xE: IOCU 6 0xF: IOCU 7
21:18	Hart ID value. 0x0: Hart 0 0x1: Hart 1 0x2: Hart 2 0x3: Hart 3
17:14	Command. This field indicates the command type. Refer to Table 5.7 for more information.
13:11	Command Group. This field indicates the command group. Refer to Table 5.6 for the encoding of this field.
10:8	Cache Coherency Attribute (CCA) value. This field indicates the CCA value corresponding to the transaction. Refer to Table 5.11 for the encoding of this field.

Table 5.13 State of ERR_INFO Field for Data Error Types 1 or 3 *(continued)*

Bit	Meaning
7:5	MCP bus transfer size. Indicates the size of the transfer on the bus. This field is encoded as $2^{(MCP\ size)}$. 0x0: 1 byte 0x1: 2 bytes 0x2: 4 bytes 0x3: 8 bytes 0x4: 16 bytes 0x5: 32 bytes (Reserved. Not used in the P8700-F) 0x6: 64 bytes 0x7: 128 bytes (Reserved. Not used in the P8700-F)
4:1	Transaction type. This field indicates the type of bus transaction that caused the error. Refer to Table 5.12 for the encoding of this field.
0	Scheduler. The P8700-F core can be configured at build time with either 1 or 2 pipeline schedulers. If the build is configured with one scheduler, this bit is always 0. If configured with two schedulers, this bit can be either 0 or 1 and indicates the scheduler involved in the error.

5.12.3 Error Code 2 — Request Decode Error

If the decimal value in the ERR_TYPE field is 2, indicating a decode request error, the ERROR_INFO field in the *Global CM Error Cause* register is organized as shown in Table 5.14.

Table 5.14 State of ERR_INFO Field for Data Error Type 2

Bit	Meaning
57	Reserved.
56	GIC access error. Hardware sets this bit to indicate a code fetch was sent GIC address space.
55	Non-Coherent MMIO error. Hardware sets this bit to indicate if an invalid MMIO access was made to MMIO address space.
54	Coherent MMIO error. Hardware sets this bit to indicate that coherent access was made to MMIO address space.
53	Reserved.
52	CCA or LL/SC error. Hardware sets this bit to indicate that the error occurred in the decoding of the CCA field, either a register access with CCA not equal to UC was attempted, or or an LLSC request was made to a register.
51	Size error. Hardware sets this bit to indicate that the error occurred in the decoding of the Size field. A register access with size not equal to 4 or 8 bytes was attempted.
50	Multiple regions error. Hardware sets this bit to indicate that the error occurred in the decoding of multiple regions.
49	Coherency request or redirect error. Hardware sets this bit to indicate that a coherent request was made to either a register-mapped address, or a redirect access was made to a block redirect that does not exist.
48	Debug register access error. Hardware sets this bit to indicate a Debug register access.
47	FDC Register Access. Hardware sets this bit to indicate a Fast Debug Channel (FDC) access.

Table 5.14 State of ERR_INFO Field for Data Error Type 2 *(continued)*

Bit	Meaning
46	Normal Register Access. Hardware sets this bit to indicate a normal register mapped access.
45	GCR Hit. Hardware sets this bit to during a hit to the GCR registers.
44	User GCR Hit. Hardware sets this bit to during a hit to the User GCR registers.
43	CPC Hit. Hardware sets this bit to during a hit to the Cluster Power Controller (CPC).
42	GIC Hit. Hardware sets this bit to during a hit to the Global Interrupt Controller (GIC).
41	IOCU Hit. Hardware sets this bit to during a hit to the I/O Coherence Unit (IOCU).
40:37	Decode CMD. This field indicates the command sent to memory on a register request. This field has the same encoding as the Command field. The bit orientation of this field depends on the type of error as listed in Table 5.6 through Table 5.10.
36:34	Decode CMD Group. This field indicates the indicates the Command Group sent to memory on a register request. The field has the same encoding as Table 5.6.
33:28	Decode Destination ID. This field indicates the destination ID sent to memory on a register request.
27:22	Port ID value. This field indicates the port ID value of all cores and IOCU's in the system. The first IOCU encoding is always directly after the last core encoding. For example, in a system with four cores and two IOCU's, the cores would occupy encoding 0x0 - 0x3, and the IOCU's would occupy encoding 0x4 - 0x5. So 0x0 - 0x[n] = cores, and 0x[n+1] - 0x[m] = IOCU's. The example below shows the encoding for a six core and two IOCU system. 0x0: core 0 0x1: core 1 0x2: core 2 0x3: core 3 0x4: core 4 0x5: core 5 0x6: IOCU 0 0x7: IOCU 1
21:18	Hart ID value. 0x0: Hart 0 0x1: Hart 1 0x2: Hart 2 0x3: Hart 3
17:14	Command. This field indicates the command type. Refer to Refer to Table 5.7 for the encoding of this field.
13:11	Command Group. This field indicates the command group. Refer to Table 5.6 for the encoding of this field.
10:8	Cache Coherency Attribute (CCA) value. This field indicates the CCA value corresponding to the transaction. Refer to Table 5.11 for the encoding of this field.

Table 5.14 State of ERR_INFO Field for Data Error Type 2 *(continued)*

Bit	Meaning
7:5	MCP bus transfer size. Indicates the size of the transfer on the bus. This field is encoded as $2^{(\text{MCP size})}$. 0x0: 1 byte 0x1: 2 bytes 0x2: 4 bytes 0x3: 8 bytes 0x4: 16 bytes 0x5: 32 bytes (Reserved. Not used in the P8700-F) 0x6: 64 bytes 0x7: 128 bytes (Reserved. Not used in the P8700-F)
4:1	Transaction type. This field indicates the type of bus transaction that caused the error. Refer to Table 5.12 for the encoding of this field.
0	Scheduler. The P8700-F core can be configured at build time with either 1 or 2 pipeline schedulers. If the build is configured with one scheduler, this bit is always 0. If configured with two schedulers, this bit can be either 0 or 1 and indicates the scheduler involved in the error.

5.12.4 Error Code 4 — Parity Error

If the decimal value in the ERR_TYPE field is 4, indicating a parity error, the ERROR_INFO field in the *Global CM Error Cause* register is organized as shown in Table 5.15.

Table 5.15 State of ERR_INFO Field for Data Error Type 4

Bit	Meaning
57:36	Reserved.
35:28	DWORD with error. This field indicates the DWORD that caused the error.
27:22	Port ID value. This field indicates the port ID value of all cores and IOCU's in the system. The first IOCU encoding is always directly after the last core encoding. For example, in a system with four cores and two IOCU's, the cores would occupy encoding 0x0 - 0x3, and the IOCU's would occupy encoding 0x4 - 0x5. So 0x0 - 0x[n] = cores, and 0x[n+1] - 0x[m] = IOCU's. The example below shows the encoding for a six core and two IOCU system. 0x0: core 0 0x1: core 1 0x2: core 2 0x3: core 3 0x4: core 4 0x5: core 5 0x6: IOCU 0 0x7: IOCU 1

Table 5.15 State of ERR_INFO Field for Data Error Type 4 *(continued)*

Bit	Meaning
21:18	Hart ID value. 0x0: Hart 0 0x1: Hart 1 0x2: Hart 2 0x3: Hart 3
17:14	Command. This field indicates the command type. The encoding of this field depends on the type of error. Refer to Table 5.7 through Table 5.10 for the encoding of this field.
13:11	Command Group. This field indicates the command group. Refer to Table 5.6 for the encoding of this field.
10:8	Cache Coherency Attribute (CCA) value. This field indicates the CCA value corresponding to the transaction. Refer to Table 5.11 for the encoding of this field.
7:5	MCP bus transfer size. Indicates the size of the transfer on the bus. This field is encoded as $2^{(\text{MCP size})}$. 0x0: 1 byte 0x1: 2 bytes 0x2: 4 bytes 0x3: 8 bytes 0x4: 16 bytes 0x5: 32 bytes (Reserved. Not used in the P8700-F) 0x6: 64 bytes 0x7: 128 bytes (Reserved. Not used in the P8700-F)
4:1	Transaction type. This field indicates the type of bus transaction that caused the error. Refer to Table 5.12 for the encoding of this field.
0	Scheduler. The P8700-F core can be configured at build time with either 1 or 2 pipeline schedulers. If the build is configured with one scheduler, this bit is always 0. If configured with two schedulers, this bit can be either 0 or 1 and indicates the scheduler involved in the error.

5.12.5 Error Code 5 — Fetch and Lock Error

If the decimal value in the ERR_TYPE field is 5, indicating a fetch and lock error, the ERROR_INFO field in the *Global CM Error Cause* register is organized as shown in Table 5.16.

Table 5.16 State of ERR_INFO Field for Data Error Type 5

Bit	Meaning
57:29	Reserved.
28	Bank in which the error occurred. 0: Bank 0 1: Bank 1

Table 5.16 State of ERR_INFO Field for Data Error Type 5 *(continued)*

Bit	Meaning
27:22	Port ID value. This field indicates the port ID value of all cores and IOCU's in the system. The first IOCU encoding is always directly after the last core encoding. For example, in a system with four cores and two IOCU's, the cores would occupy encoding 0x0 - 0x3, and the IOCU's would occupy encoding 0x4 - 0x5. So 0x0 - 0x[n] = cores, and 0x[n+1] - 0x[m] = IOCU's. The example below shows the encoding for a six core and two IOCU system. 0x0: core 0 0x1: core 1 0x2: core 2 0x3: core 3 0x4: core 4 0x5: core 5 0x6: IOCU 0 0x7: IOCU 1
21:18	Hart ID value. 0x0: Hart 0 0x1: Hart 1 0x2: Hart 2 0x3: Hart 3
17:14	Command. This field indicates the command type. The encoding of this field depends on the type of error. Refer to Table 5.7 through Table 5.10 for the encoding of this field.
13:11	Command Group. This field indicates the command group. Refer to Table 5.6 for the encoding of this field.
10:8	Cache Coherency Attribute (CCA) value. This field indicates the CCA value corresponding to the transaction. Refer to Table 5.11 for the encoding of this field.
7:5	MCP bus transfer size. Indicates the size of the transfer on the bus. This field is encoded as $2^{(\text{MCP size})}$. 0x0: 1 byte 0x1: 2 bytes 0x2: 4 bytes 0x3: 8 bytes 0x4: 16 bytes 0x5: 32 bytes (Reserved. Not used in the P8700-F) 0x6: 64 bytes 0x7: 128 bytes (Reserved. Not used in the P8700-F)
4:1	Transaction type. This field indicates the type of bus transaction that caused the error. Refer to Table 5.12 for the encoding of this field.
0	Scheduler. The P8700-F core can be configured at build time with either 1 or 2 pipeline schedulers. If the build is configured with one scheduler, this bit is always 0. If configured with two schedulers, this bit can be either 0 or 1 and indicates the scheduler involved in the error.

5.12.6 Error Codes 6, 7, 8 — Bus Interface Unit (BIU) Errors

If the decimal value in the ERR_TYPE field is between 6 and 8, the ERR_INFO field in the *Global Error Cause* register is organized as shown in Table 5.17.

Table 5.17 State of ERR_INFO Field for Error Types 6 through 8

Bit	Meaning
57:54	Subcode. This field indicates the type of bus error and is encoded as follows: 0: Internal MCP request decode error 1: AXI parity error 2: Internal MCP parity error 3: AXI xRESP error (SLVERR or DECERR) 4: Unexpected AXI RID 5: Unexpected AXI BID 6: Reserved 7: AXI CD parity error 8: MMIO port error 9: NOC_REG_ACCESS error
53:49	Reserved
48:41	AXI ID value. Valid if TYPE = 8. This value applies to subcodes 1, 3, 4, and 5 in bits 57:54 above. Refer to Table 5.3 for a list of error types.
40:37	RRESP/BRESP. Valid if TYPE = 8. This value applies to subcodes 1, 3, 4, and 5 in bits 57:54 above. Refer to Table 5.3 for a list of error types.
36	Request data buffer lock (rdb_lock). This field is valid for subcodes 0 - 3, 6, 8 and 9.
35:31	Request data buffer thread ID (req_thrd_id). This field is valid for subcodes 0 - 3, 6, 8 and 9.
30:27	Request port (req_port). This field is valid for subcodes 0 - 3, 6, 8 and 9. See the table below for encoding.
26	Request data buffer write (rdb_wr). This field is valid for subcodes 0 - 3, 6, 8 and 9.
25	Request data buffer uncached accelerated (rdb_uca). This field is valid for subcodes 0 - 3, 6, 8 and 9.
24	Request data buffer uncached (rdb_uc). This field is valid for subcodes 0 - 3, 6, 8 and 9.
23:0	Reserved.

Table 5.18 BIU Error Request Port (req_port) Field Encoding — Bits 30:27

Bits 30:27	Output Channel
0x0	C_MEM_AR (memory read)
0x1	C_MEM_AW (memory write)
0x2	AUX0_AR (Aux port 0 read)
0x3	AUX0_AW (Aux port 0 write)
0x4	AUX1_AR (Aux port 1 read)
0x5	AUX1_AW (Aux port 1 write)
0x6	AUX2_AR (Aux port 2 read)

Table 5.18 BIU Error Request Port (req_port) Field Encoding — Bits 30:27*(continued)*

Bits 30:27	Output Channel
0x7	AUX2_AW (Aux port 2 write)
0x8	AUX3_AR (Aux port 3 read)
0x9	AUX3_AW (Aux port 3 write)
0x10	REGTN_AR (inter-cluster Reg-to-NOC register read)
0x11	REGTN_AW (inter-cluster Reg-to-NOC register write)
0x12	ITU_AR (ITU read)
0x13	ITU_AW (ITU write)
0x14	Reserved
0x15	RBI local registers read and write

5.12.7 Error Code 10 — Ring Bus Error

If the decimal value in the *ERR_TYPE* field is 10, the *ERR_INFO* field in the *Global CM Error Cause* register is organized as shown in Table 5.19.

Table 5.19 State of ERR_INFO Field for Error Type 10

Bit	Meaning
57:54	Sub-code 0: reserved 1: Master endpoint response error (see CMD[1:0] field for error type) 2: Register ring bus error 3. Byte enable error
53:48	Reserved
47	cmd[3]. In the P8700-F, this bit is always 0. 0: Standard packets 1: Extended packets (reserved for future implementations)
46	cmd[2]. Identifies the packet as a read or write packet. This field is encoded as follows: 0: Read 1: Write
45:44	cmd[1:0] 0: No error (packet is valid) 1: Endpoint not available. When an endpoint is powered down or in the clock-off state, the slave node responds with an "Endpoint Unavailable Error". 2: Destination not found or byte enable error on MCP/REGTC requests. If the master acting as the request terminator finds an unclaimed request, it turns the packet into a response packet swapping the src/dest ID's and signal a "Destination Not Found Error". This error can also indicate that a byte enable error has occurred attempting to not write all bytes of the word or double-word transaction 3: Parity error on RRB. If a bus parity error occurs, the endpoint responds with a "Bus Parity Error". Normal request packets created by the master endpoints set this field to zero.

Table 5.19 State of ERR_INFO Field for Error Type 10 *(continued)*

Bit	Meaning
43:38	Destination ID. Indicates the destination of the operation when the error occurred. This field is encoded in decimal as follows: 0 - 5: Core 0 through Core 5. Values 6 - 7 are reserved 8 - 15: Reserved 16 - 23: IOCU 0 through IOCU 7 24: GIC 25: User-defined GCR block 26: Memory 27 - 31: Reserved 32: CM master 33: CPC 34: GCR block 35: DBU master 36: DBU dmxseg normal 37: DBU dmxseg debug 38 - 39: Reserved 40: AUX 0 41: AUX 1 42: AUX 2 43: AUX 3 44 - 61: Reserved 62: No destination error 63: No destination OK The values 36-37 accommodate DBU dmxseg normal and debug mode accesses. The slave node connected to the Debug Unit slave block allows multiple dest_id's to match the slave node and be forwarded to the Debug Unit slave interface. This allows access the Debug Unit dmxseg block memory mapping using two modes of operation (normal and debug/privileged). The values 62-63 allow the address decode block of the CM to indicate to the register bus interface that there is no destination for an enabled memory mapped register area or that a write from a requestor has been blocked by global access control. The register bus interface returns a response packet to the initiator without sending a packet over the register bus. If the register bus interface decodes a dest_id of "No Dest Err", an error response packet is returned. If the register bus interface decodes a dest_id of "No Dest OK", a normal response packet is returned. Read responses for dest_ids of "No Dest Err" and "No Dest OK" will return data that is all zeros.
37:32	Destination cluster ID. This field indicates the destination ID number of the cluster where the error occurred. Each register bus cluster request node and cluster response node is enumerated with a CLUSTER_ID. The CLUSTER_ID input is hardwired to its associated identifier when it is instantiated. This value of the CLUSTER_ID is compared against the value in this field to determine if the register bus cluster node should send the packet along its own cluster ring or sent it to the multi-cluster ring.
31:24	Shared data buffer ID (sbd_id) This identifier is used to match the response to the original request. This field is determined by the originating master of the transaction, i.e. CM or Debug Unit, and will be returned to that master.
23:18	Source ID. Indicates the source of the operation when the error occurred. This field is encoded the same as the destination ID field in bits 43:38.

Table 5.19 State of ERR_INFO Field for Error Type 10 *(continued)*

Bit	Meaning
17:12	Source cluster ID. This field indicates the source ID number of the cluster where the error occurred. Each register bus cluster request node and cluster response node is enumerated with a CLUSTER_ID. The CLUSTER_ID is hardwired to its associated identifier when it is instantiated. This value of the CLUSTER_ID is compared against the value in this field to determine if the register bus cluster node should send the packet along its own cluster ring or sent it to the multi-cluster ring.
11:6	Address (reads only) This field gives the byte address for the register bus transaction.
5:3	Size (reads only). The data byte length is interpreted as 2^{size}. The protocol supports 1 to 64 bytes of data in powers of two. For register transactions only 32-bit (4 byte) and 64-bit (8-byte) sizes are supported. This field is encoded as follows: 3'b000: Byte 3'b001: Half -word (2 bytes) 3'b010: Word (4 bytes) 3'b011: Double-word (8 bytes) 3'b100: Quad-word (16 bytes) 3'b101: Reserved (32 bytes) 3'b110: Cache line (64 bytes) 3'b111: Reserved (128 bytes)
2:0	Reserved

5.12.8 Error Code 11 — IOCU Request Error

If the decimal value in the *ERR_TYPE* field is 11, the *ERR_INFO* field in the *Global CM Error Cause* register is organized as shown in Table 5.20.

Table 5.20 State of ERR_INFO Field for Error Type 11

Bit	Meaning
57:54	Sub-code 0x0: FIXED mode. AXI burst is set to FIXED mode. This mode is not supported by the IOCU. 0x1: WRAP mode. On a read request, if burst mode is set to WRAP, then the LEN field must be either 0 or 3. If the LEN field is neither 0 or 3, an error is generated. 0x2: LEN > 0 and SIZE < 128. If the LEN field is greater than 0 and the SIZE field is <128, an error is generated. 0x3: For a write request with the Burst mode set to WRAP and the LEN field set to 3, the starting offset must be 0. If the offset is non-zero, an error is generated. 0x4 - 0xF: Reserved
53:0	Reserved

5.12.9 Error Code 12 — IOCU Parity Error

If the decimal value in the *ERR_TYPE* field is 12, the *ERR_INFO* field in the *Global CM Error Cause* register is organized as shown in Table 5.21.

Table 5.21 State of ERR_INFO Field for Error Type 12

Bit	Meaning
57:56	Reserved
55:54	IOCU command 0: Reserved 1: Write 2: Read 3: Reserved
53:52	IOCU Cache coherency attribute 0: Reserved 1: Coherent 2: Non-coherent 3: Reserved
51:50	Reserved
49:44	AXI device ID. This field is configurable and can be any value up to a maximum of 64 device ID's. 49:44 = 6'b000000: AXI device ID 0 .... 49:44 = 6'b111111: AXI device ID 63
43	Reserved
42:39	AXI request ID. This field is configurable and can be any value up to a maximum of 16 request ID's. Note that there can be up to 16 read requests and 16 write requests. 42:39 = 0x0: AXI request ID 0 .... 42:39 = 0xF: AXI request ID 15
38:0	Reserved

5.12.10 Error Code 13 — IOCU Response Error

If the decimal value in the *ERR_TYPE* field is 13, the *ERR_INFO* field in the *Global CM Error Cause* register is organized as shown in Table 5.22.

Table 5.22 State of ERR_INFO Field for Error Type 13

Bit	Meaning
57:56	Error type. 0: No RIN error 1: Bus error 2. Cache error
55:54	IOCU command 0: Reserved 1: Write 2: Read 3: Reserved

Table 5.22 State of ERR_INFO Field for Error Type 13 *(continued)*

Bit	Meaning
53:52	IOCU Cache coherency attribute 0: Reserved 1: Coherent 2: Non-coherent 3: Reserved
51:50	Reserved
49:44	AXI device ID. This field is configurable and can be any value up to a maximum of 64 device ID's. 49:44 = 6'b000000: AXI device ID 0 .... 49:44 = 6'b111111: AXI device ID 63
43	Reserved
42:39	AXI request ID. This field is configurable and can be any value up to a maximum of 16 request ID's. Note that there can be up to 16 read requests and 16 write requests. 42:39 = 0x0: AXI request ID 0 .... 42:39 = 0xF: AXI request ID 15
38:0	Reserved

5.12.11 Error Code 15 — RBI REGTC Bus Request Error

If the decimal value in the *ERR_TYPE* field is 15, the *ERR_INFO* field in the *Global CM Error Cause* register is organized as shown in Table 5.23.

Table 5.23 State of ERR_INFO Field for Error Type 15

Bit	Meaning
57:56	Subcode. 0: Burst error 1: Size error 2. Length error
53	Reserved
52:42	AxID. This field stores the REGTC AxID of the REGTC request that generated the error.
41	Read/write. 0: Write 1: Read
40:20	AxUSER. This field stores the AxUSER of the REGTC request that generated the error.
19:0	AxADDR. This field stores the AxADDR of the REGTC request that generated the error.

5.13 Memory Mapped Registers

Each cluster in a MIPS® Technologies multi-core RISC-V system contains a MIPS® Coherence Manager (CM) and a MIPS® Technologies implementation of a RISC-V Advanced Interrupt Controller (AIA), also known as the Interrupt Controller. The CM includes the MIPS® Cluster Power Controller (CPC) and the MIPS® Fast Debug Channel (FDC).

The CM and Interrupt Controller blocks are controlled by a set of memory mapped registers which we will refer to a GCRs (General Control Registers) arranged in a 512KB block of memory starting at a physical address which we refer to as GCR_BASE. At reset, GCR_BASE is set to the first naturally aligned 512KB block of memory below the reset PC of the cluster's core number 0, unless the GCR_BASE reset value has been overridden in the system configuration. GCR_BASE can be reprogrammed by writing to the GCR.Global.GCR_BASE register within the GCR block.

A cluster accesses its own GCRs at the physical address in GCR.Global.GCR_BASE. In addition, a system level NoC may be present that can be programmed to intercept memory accesses to specific addresses and direct them to access to the GCR blocks of any cluster in the system.

Within the GCR block, the GCRs are arranged in the following sections, which are described in detail in the subsequent sections of this chapter:

Table 1: Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x00000 - 0x01FFF	GCR.Global	Per-cluster CM registers.
0x02000 - 0x05FFF	GCR.Core	Per-core CM registers.
0x06000 - 0x07FFF		Reserved.
0x08000 - 0x09FFF	CPC.Global	Per-cluster CPC registers.
0x0A000 - 0x0EFFF	CPC.Core	Per-core/Per-device CPC registers.
0x0F000 - 0x0FFFF		Reserved.
0x10000 - 0x1FFFF	uGCR	Reserved for user defined CM registers.
0x20000 - 0x3EFFF		Reserved.
0x3F000 - 0x3F0FF	FDC.Global	FDC.Global registers.
0x3F100 - 0x3FFFF	TRF.Global	TRF.Global registers
0x40000 - 0x4BFFF	APLIC.M	APLIC Machine registers.
0x4C000 - 0x4CFFF	APLIC.custom	APLIC custom registers.
0x4D000 - 0x4FFFF		Reserved.
0x50000 - 0x5FFFF	ACLINT.M	ACLINT Machine registers.
0x60000 - 0x6BFFF	APLIC.S	APLIC Supervisor registers.
0x6C000 - 0x6FFFF	ACLINT.S	ACLINT Supervisor registers.
0x70000 - 0x7EFFF		Reserved.
0x7F000 - 0x7FFFF	GCR.U	User Mode GCRs.

5.14 Coherence Manager (CM) Memory Mapped Registers

The GCR.Global region contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 2: CM Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x00000	GCR.Global.CONFIG	CM global configuration
0x00008	GCR.Global.GCR_BASE	Base address of GCR block
0x00010	GCR.Global.CONTROL	Control bits for the Coherence Manager
0x00030	GCR.Global.REV	RevisionID of the GCR hardware
0x00038	GCR.Global.ERR_CONTROL	Controls Error Checking/Correct logic within the CM3
0x00040	GCR.Global.ERR_MASK	
0x00048	GCR.Global.ERR_CAUSE	Captures info when an error occurs within the CM3
0x00050	GCR.Global.ERR_ADDR	Captures address which caused the CM3 error.
0x00058	GCR.Global.ERR_MULT	Captures information for subsequent CM3 errors.
0x00068	GCR.Global.CUSTOM_STATUS	Existence and status of the custom user-defined GCR
0x000D0	GCR.Global.AIA_STATUS	Existence and status of Interrupt Controller.
0x000E0	GCR.Global.CACHE_REV	Revision of cache attached to the coherent Cluster
0x000F0	GCR.Global.CPC_STATUS	Existence and status of CPC
0x00120	GCR.Global.ACCESS	Controls which Cores/IOCUs can modify the GCR and CPC Registers
0x00130	GCR.Global.L2_CONFIG	Provides L2 cache configuration
0x00160	GCR.Global.SDB_CONFIG	Defines the Memory, Intervention, PFU and total SDB for the cluster
0x00200	GCR.Global.IOCU_REV	Revision of IOCU
0x00208	GCR.Global.DBU_REV	Revision of Debug Unit
0x00210	GCR.Global.AIA_REV	Revision of the Interrupt Controller.
0x00240	GCR.Global.L2_RAM_CONFIG	Configuration of the L2 cache and control the dynamic L2 RAM low power states
0x00280	GCR.Global.SCRATCH0	General Purpose Read/Write Register
0x00288	GCR.Global.SCRATCH1	General Purpose Read/Write Register
0x00300	GCR.Global.L2_PFT_CONTROL	Controls the L2 prefetcher
0x00308	GCR.Global.L2_PFT_CONTROL_B	L2 prefetch 2nd control register
0x00600	GCR.Global.L2_TAG_ADDR	Holds Address Portion of CACHE L2 Load or Store Tag & Data CACHE instruction

Table 2: CM Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x00608	GCR.Global.L2_TAG_STATE	
0x00610	GCR.Global.L2_DATA	Holds results of CACHE L2 Load or Store Tag & Data instruction
0x00618	GCR.Global.L2_ECC	Holds State Portion of CACHE L2 Load or Store Tag & Data CACHE
0x00620	GCR.Global.L2SM_COP	Holds CMD, TYPE, MODE, RESULT and PRESENT bit info of L2 Cache Op State machine
0x00628	GCR.Global.L2SM_TAG_ADDR_COP	Holds Tag address details L2 CacheOp State Machine
0x00640	GCR.Global.SEM	Provides Uncached Semaphore
0x00650	GCR.Global.TIMEOUT_TIMER_LIMIT	Time-out limit for transaction time-out timer in number of CM clocks
0x006F8	GCR.Global.MMIO_REQ_LIMIT	Number of MMIO requests that the CM3 will allow to be in flight.
0x00700	GCR.Global.MMIO0_BOTTOM	Lowest address of MMIO Region 0
0x00708	GCR.Global.MMIO0_TOP	Highest address of MMIO Region 0
0x00710	GCR.Global.MMIO1_BOTTOM	Lowest address of MMIO Region 1
0x00718	GCR.Global.MMIO1_TOP	Highest address of MMIO Region 1
0x00720	GCR.Global.MMIO2_BOTTOM	Lowest address of MMIO Region 2
0x00728	GCR.Global.MMIO2_TOP	Highest address of MMIO Region 2
0x00730	GCR.Global.MMIO3_BOTTOM	Lowest address of MMIO Region 3
0x00738	GCR.Global.MMIO3_TOP	Highest address of MMIO Region 3
0x00740	GCR.Global.MMIO4_BOTTOM	Lowest address of MMIO Region 4
0x00748	GCR.Global.MMIO4_TOP	Highest address of MMIO Region 4
0x00750	GCR.Global.MMIO5_BOTTOM	Lowest address of MMIO Region 5
0x00758	GCR.Global.MMIO5_TOP	Highest address of MMIO Region 5
0x00760	GCR.Global.MMIO6_BOTTOM	Lowest address of MMIO Region 6
0x00768	GCR.Global.MMIO6_TOP	Highest address of MMIO Region 6
0x00770	GCR.Global.MMIO7_BOTTOM	Lowest address of MMIO Region 7
0x00778	GCR.Global.MMIO7_TOP	Highest address of MMIO Region 7
0x00810	GCR.Debug.TCBCONTROLD	Controls CM3 PDTrace
0x00820	GCR.Debug.TCBCONTROLE	Status reg for CM3 PDTrace
0x00830	GCR.Debug.TCBPERFCNTR	Config reg for CM3 PDTrace
0x00900	GCR.Debug.PC_CTL	Starting/stopping of Performance Counters
0x00920	GCR.Debug.PC_OV	Which performance counters have overflowed
0x00930	GCR.Debug.PC_EVENT	Select event type of each CM3 performance counter

Table 2: CM Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x00980	GCR.Debug.PC_CYCL	Counts Cycles
0x00990	GCR.Debug.PC_QUAL0	Performance counter 0 event qualifiers
0x00998	GCR.Debug.PC_CNT0	Performance Counter 0 value
0x009A0	GCR.Debug.PC_QUAL1	Performance counter 1 event qualifiers
0x009A8	GCR.Debug.PC_CNT1	Performance Counter 1 value

5.14.1 GCR.Global Registers

The GCR global space registers are defined below.

5.14.1.1 GCR Global Configuration Register (offset = 0x0000)

Indicates the number of processor cores, number of interrupts, number of IOCUs, etc.

This register provides information on the overall system configuration. These fields are read-only and their reset state is determined at IP configuration time.

Figure 5.4 GCR Global Configuration Register Bit Assignments

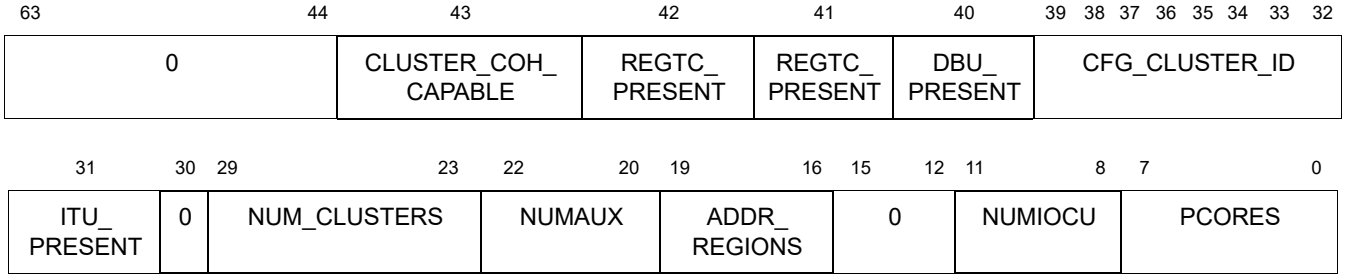


Table 3: GCR Global Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:44	Reserved.	R	0
CLUSTER_COH_ CAPABLE	43	Set to 1 if this cluster supports ACE coherent interconnect.	R	From configuration
REGTC_PRESENT	42	Set to 1 if REGTC is present in this cluster.	R	From configuration
REGTN_PRESENT	42	Set to 1 if REGTN is present in this cluster.	R	From configuration
DBU_PRESENT	41	Set to 1 if DBU is present in this cluster.	R	From configuration
CFG_CLUSTER_ID	39:32	Indicates the cluster_id of current cluster.	R	Cluster ID
ITU_PRESENT	31	Set to 1 if ITU is present in the design.	R	From configuration
0	30	Reserved.	R	
NUM_CLUSTERS	29:23	Indicates total number of clusters present in the design.	R	From configuration
NUMAUX	33:20	Indicates the number of auxiliary memory ports in this cluster.	R	From configuration
ADDR_REGIONS	19:16	Indicates the number of MMIO address region registers. This value is determined by the IP configuration.	R	From configuration
0	15:12	Reserved.	R	
NUMIOCU	11:8	Indicates the total number of IOCUs in this cluster.	R	From configuration
PCORES	7:0	Total number of CPU Cores - 1 in this cluster, not including the IOCUs.	R	From configuration

5.14.1.2 Global GCR_BASE Register (offset = 0x0008)

The default GCR_BASE reset value is the first naturally aligned address which allows the GCR region to fit below the CPU reset PC. For example, if the reset PC is 0x1fc00000, then the reset value of GCR_BASE is 0x1fbf8000 for pre-AIA CM implementations, and 0x1fb80000 for AIA CM implementations. A different default value may be specified at IP configuration time.

The number of writable bits in GCR_BASE depends on the number of supported physical address bits, and the size of the GCR address region. For pre-AIA CM implementations, 48 physical address bits are supported, and the GCR address region is 0x8000 bytes, meaning that bits 47:15 of GCR_BASE are writable. For AIA CM implementations, 48 physical address bits are supported, and the GCR address region is 0x80000 bytes, meaning that bits 47:19 of GCR_BASE are writable.

When writing this register with a 64b write the register acts normally and all bits are updated immediately. However, when this register is written with 32b writes, then the bits 63:32 must be written first, followed by the write to the lower 32b. A 32b write to the upper portion of the register does not immediately change the GCR_BASE value. Instead, the write data is stored in an internal shadow register. A subsequent 32b write to the lower portion of this register causes GCR_BASE[63:32] to be loaded with the value stored in the internal shadow register and GCR_BASE[31:0] to be loaded with the value being written. Note that GCR[63:32] is only updated on a 32b write if there was a previous 32b write to GCR_BASE[63:32].

Figure 5.5 Global GCR Base Register Bit Assignments

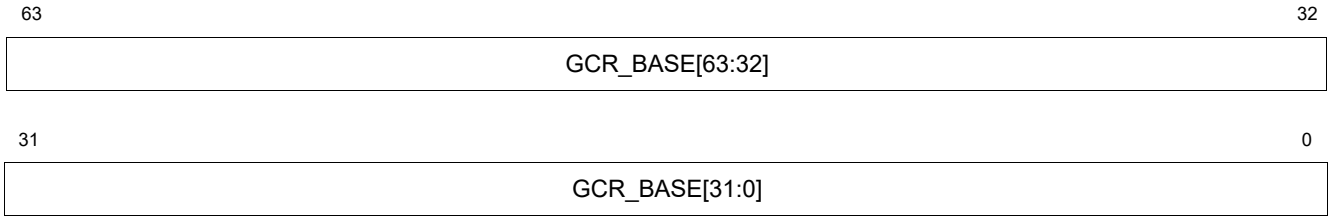


Table 4: Global GCR Base Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
GCR_BASE[63:32]	63:32	Upper 32 bits of GCR_BASE.	R/W	0
GCR_BASE[31:0]	31:30	Lower 32 bits of GCR_BASE.	R/W	0

5.14.1.3 GCR Global Control Register (offset = 0x0010)

This register contains the control bits for the Coherence Manager (CM).

Figure 5.6 Global GCR Control Register Bit Assignments

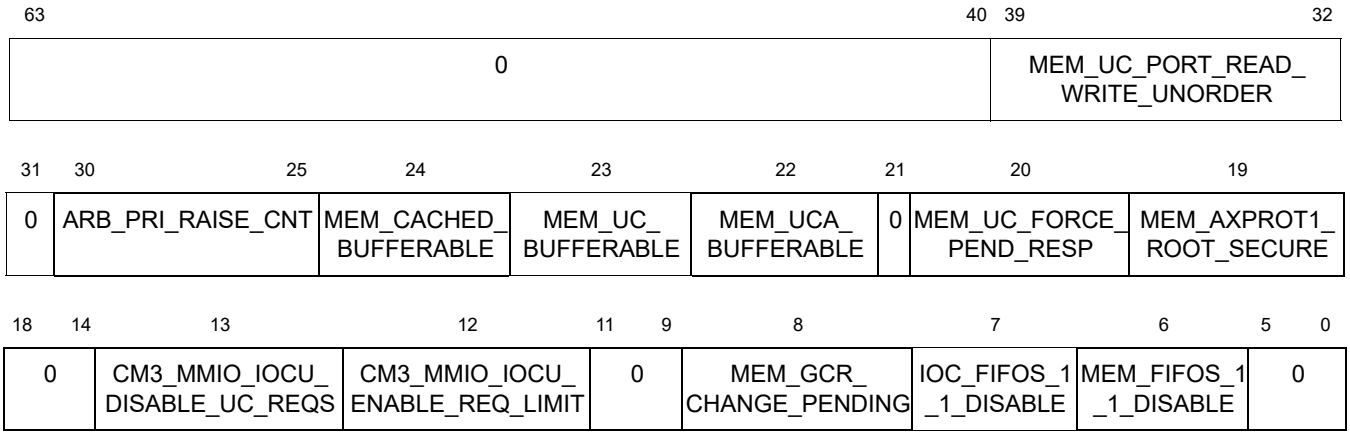


Table 5: Global GCR Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:40	Reserved	R	0
MEM_UC_PORT_READ_WRITE_UNORDER	39:32	When set, uncached (UC) requests enforce order between reads and writes by waiting for request responses on the AXI bus before issuing the next request. When cleared, read and write requests are allowed to be issued on the AXI bus independent from each other. This may cause read followed by write, or write followed by read errors unless order is protected elsewhere in the system. Bits 0 to (n-1) for n cores, n to (n+m-1) for m iocus.	R/W	0
ARB_PRI_RAISE_CNT	30:25	This field determines how the main arbiter treats low priority requests. Normally, high priority requests are always selected for serialization ahead of low priority requests. However, setting ARB_PRI_RAISE_CNT to a non-zero value ensures that a low priority request will be serviced after waiting ARB_PRI_RAISE_CNT cycles.	R/W	32
MEM_CACHED_BUFFERABLE	24	Sets the BUFFERABLE bit on the memory AXI port for cached requests.	R/W	0
MEM_UC_BUFFERABLE	23	Sets the BUFFERABLE bit on the memory AXI port for uncached requests.	R/W	0
MEM_UCA_BUFFERABLE	22	Sets the BUFFERABLE bit on the memory AXI port for uncached accelerated requests.	R/W	0
0	21	Reserved.	R	0
MEM_UC_FORCE_PEND_RESP	20	Causes UC requests not be issued on AXI bus until previous UC response has been received.	R/W	0

Table 5: Global GCR Control Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MEM_AXPROT1_ ROOT_SECURE	19	When set, this bit causes AxPROT[1] to be 0 (secure) for any access from a zero guestID.	R/W	0
0	18:14	Reserved	R	0
CM3_MMIO_IOCU_ DISABLE_UC_REQS	13	When set, incoming IOCU uncached requests are prevented from being issued to MMIO regions and will receive a BUSERR response. (This can be enabled by software to assist in MMIO debugging if required).	R/W	0
CM3_MMIO_IOCU_ ENABLE_REQ_LIMIT	12	When set, this bit allows IOCU uncached requests to be counted in MMIO outstanding request limit and to have its UC requests blocked if the MMIO outstanding request limit is reached. This field only has an effect if CM3_MMIO_IOCU_DISABLE_UC_REQS = 0.	R/W	0
0	11:9	Reserved	R	0
MEM_GCR_CHANGE _PENDING	8	Indicates that a change to one of the MEM_* bits changed it that CM has not yet observed the change.	R/W	0
IOC_FIFOS_1_1_ DISABLE	7	When set, this bit disables the IOC clock-crossing FIFO's ability to use 1:1 mode. Typically this bit should be programmed to 0.	R/W	0
MEM_FIFOS_1_ 1_DISABLE	6	When set, this bit disables the mem clock-crossing FIFOs ability to use 1:1 mode. Typically this should be programmed to 0.	R/W	0
0	5:0	Reserved.	R	0

5.14.1.4 Global Revision ID Register (offset = 0x0030)

This register contains the Revision ID value for the GCR hardware.

Figure 5.7 Global Revision ID Register Bit Assignments

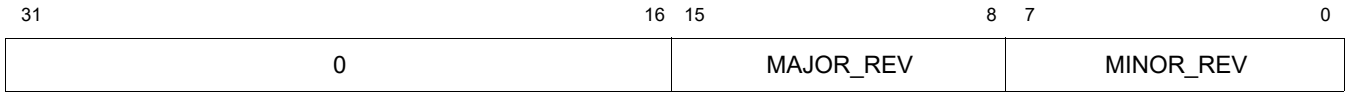


Table 6: Global Revision ID Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved.	R	0
MAJOR_REV	15:8	CM3 Major revision number. This field reflects the major revision of the GCR block. A major revision might reflect the changes from one product generation to another. This value changes based on the processor revision. Refer to the errata sheet for the exact value of this field.	R	From configuration
MINOR_REV	7:0	CM3 Minor revision number. This field reflects the minor revision of the GCR block. A minor revision might reflect the changes from one release to another. This value changes based on the processor revision. Refer to the errata sheet for the exact value of this field.	R	From configuration

5.14.1.5 GCR Global Error Control (ERR_CONTROL) Register (offset = 0x0038)

This register controls the Error Checking/Correction logic within the CM3.

Figure 5.8 Global Error Control Register Bit Assignments

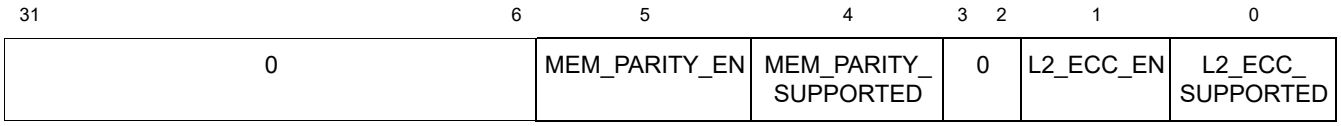


Table 7: Global Error Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:6	Reserved.	R	0
MEM_PARITY_EN	5	Parity on CM3 Memory bus is enabled.	R/W	0
MEM_PARITY_ SUPPORTED	4	Parity on CM3 Memory Bus is supported.	R	From configuration
0	3:2	Reserved.	R	0
L2_ECC_EN	1	Enables L2 ECC checking and error reporting.	R/W	1
L2_ECC_ SUPPORTED	0	L2 ECC is supported. Currently L2 ECC is always available.	R	1

5.14.1.6 GCR Global Error Mask (ERR_MASK) Register (offset = 0x0040)

This register controls what errors are reported as interrupts. It is used in conjunction with the Global CM3 Error Cause and Global CM3 Error Address registers to determine the type of error and the address which caused the error.

Figure 5.9 Global Error Mask Register Bit Assignments

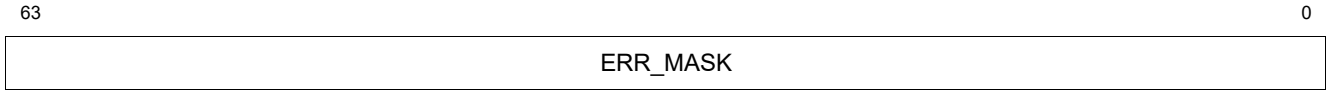


Table 8: Global Error Mask Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
ERR_MASK	63:0	CM3 Error Mask field. Each bit in this field represents an Error Type. If the bit is set, an interrupt is generated if an error of that type is detected. If the bit is set, the transaction for Read-Type Errors completes with OK response to avoid double reporting of the error.	R/W	0

5.14.1.7 GCR Global Error Cause (ERR_CAUSE) Register (offset = 0x0048)

This register captures info when an error occurs within the CM3. It is used in conjunction with the Global CM3 Error Mask and Global CM3 Error Address registers to determine the type of error and the address which caused the error. NOTE: this register is reset on a cold reset only.

Figure 5.10 Global Error Cause Register Bit Assignments

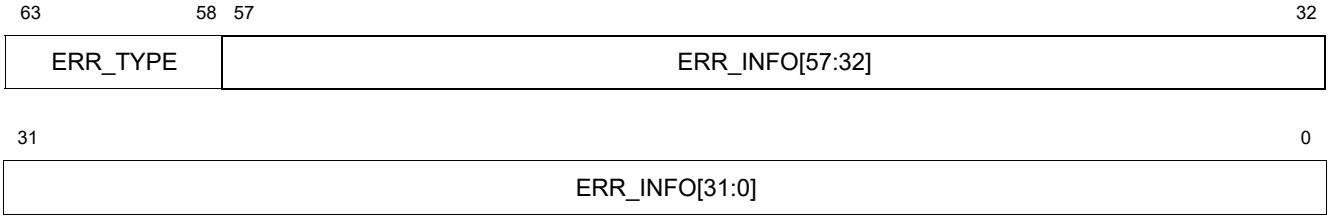


Table 9: Global Error Cause Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
ERR_TYPE	63:58	Indicates type of error detected. When CM3_ERROR_TYPE is zero, no errors have been detected. When CM3_ERROR_TYPE is non-zero, another error will not be reloaded until a power-on reset or this field is written to current value of ERR_TYPE.	R/W	0
ERR_INFO	57:0	Information about the error. Refer to the System Programmer's Reference for more information.	R	0

5.14.1.8 GCR Global Error Address (ERR_ADDR) Register (offset = 0x0050)

This register captures address which caused the CM3 error. It is used in conjunction with the Global CM3 Error Mask and Global CM3 Error Address registers to determine the type of error and the address which caused the error. NOTE: this register is reset on a cold reset only.

Figure 5.11 Global Error Address Register Bit Assignments

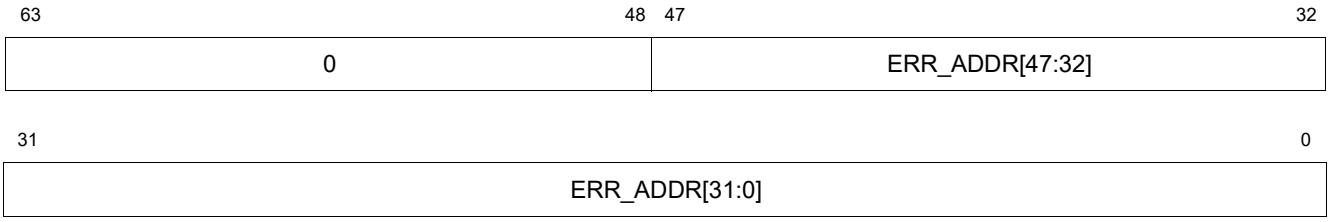


Table 10: Global Error Address Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
ERR_ADDR	47:0	Request address which caused error. Loaded when the Global Error Cause Register is loaded.	R	Undefined

5.14.1.9 GCR Global Error Mult (ERR_MULT) Register (offset = 0x0058)

This register captures information for subsequent CM3 errors.

The Global CM3 Error Cause, Global CM3 Error Address, and Global CM3 Error Mask registers described above provide information on the type of error, and the address which caused the error. This register is used to log the type of secondary error. NOTE: this register is reset on a cold reset only.

Figure 5.12 Global Error Mult Register Bit Assignments

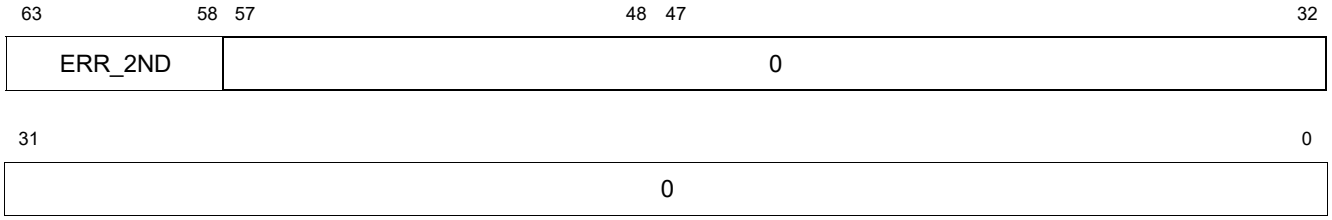


Table 11: Global Error Mult Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
ERR_2ND	63:58	Type of second error. Loaded when the Global CM3 Error Cause Register has valid error information and a second error is detected. When ERR_2ND is zero, a second error has not been detected. When ERR_2ND is non-zero, this field will not be reloaded until a power-on reset or this field is written to current value of ERR_TYPE.	R	0
0	57:0	Reserved	R	0

5.14.1.10 GCR Global Custom Status (CUSTOM_STATUS) Register (offset = 0x0068)

This register describe the existence and status of the custom user-defined GCR block.

Figure 5.13 Global Custom Status Register Bit Assignments

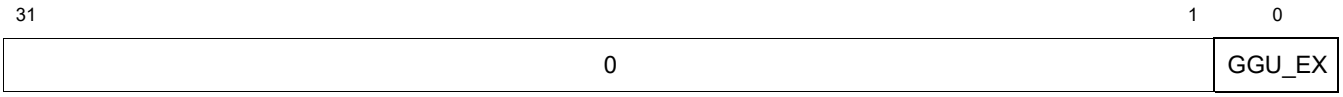


Table 12: Global Custom Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
GGU_EX	0	If this bit is set, the Custom GCR block is connected to the CM3. The state of this bit is set based on whether or not this block is implemented at build time as determined by the state of the GU_Present signal. If a Custom GCR block is not present, the GU_Present pin is driven to 0. If there is a custom GCR block present, then the user must drive GU_Present = 1 inside their custom GCR module.	R	0

5.14.1.11 GCR Global Interrupt Status (AIA_STATUS) Register (offset = 0x00D0)

This register describe the existence and status of the Interrupt Controller.

Figure 5.14 Global Interrupt Status Register Bit Assignments

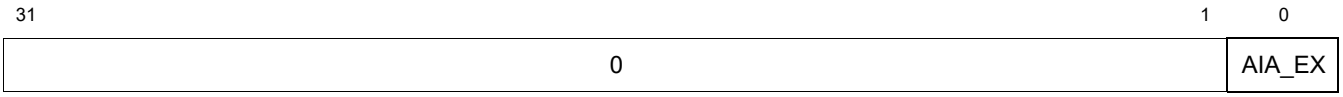


Table 13: Global Interrupt Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
AIA_EX	0	If this bit is set, the Interrupt Controller is present in the CM3.	R	1

5.14.1.12 GCR Global Cache Revision (CACHE_REV) Register (offset = 0x00E0)

This register describe the revision of cache attached to the coherent cluster.

Figure 5.15 Global Cache Revision Register Bit Assignments

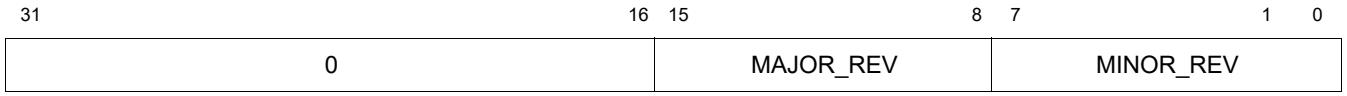


Table 14: Global Cache Revision Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
MAJOR_REV	15:8	This field reflects the major revision of the Cache block inside the CM3.	R	From configuration
MINOR_REV	7:0	This field reflects the minor revision of the Cache block inside the CM3.	R	From configuration

5.14.1.13 GCR Global CPC Status (CPC_STATUS) Register (offset = 0x00f0)

This register describe the existence and status of CPC.

Figure 5.16 Global CPC Status Register Bit Assignments

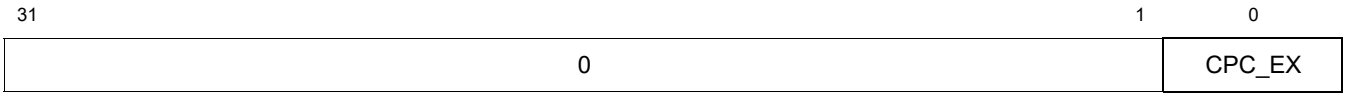


Table 15: Global CPC Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
CPC_EX	0	This bit is always 1 as the CPC is always connected to the CM3.	R	1

5.14.1.14 GCR Global Access (ACCESS) Register (offset = 0x0120)

This register controls which Cores/IOCUs can modify the GCR and CPC Registers.

It can be used to inhibit specific cores or IOCUs from writing GCR and CPC registers. Each bit in this registers controls the access from a particular requestor. Bits 7:0 control access for cores 7-0 and bits 23:16 control access from IOCU7 to IOCU0. If the bit is set, the corresponding requester is able to write to the GCR, CPC registers (this includes all registers within the Global, Core-Local, Core-Other, and Global Debug control blocks. The GIC is always writable by all requestors). If the bit is clear, any write request from that requestor to the GCR registers (Global, Core-Local, Core-Other, or Global Debug control blocks) will be dropped. NOTE: Care must be taken to not write a 0 to all fields in this register. Writing all zeros inhibit writes from all requestors to all registers until reset.

Figure 5.17 Global Access Register Bit Assignments

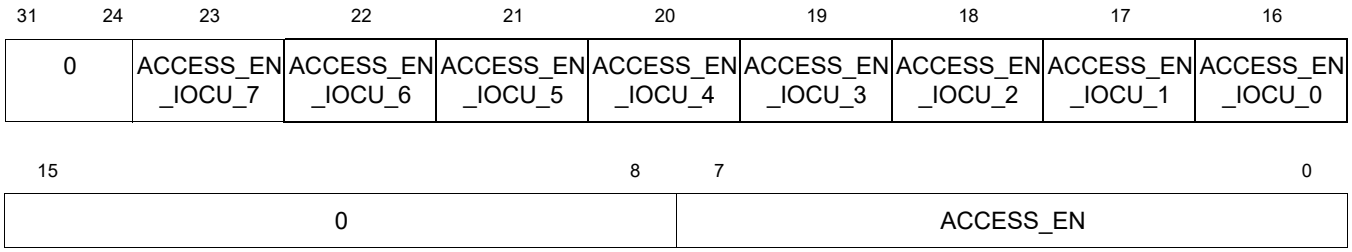


Table 16: Global Access Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:24	Reserved	R	0
ACCESS_EN_IOCU_7	23	When this bit is 1 accesses from IOCU7 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU7 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_6	22	When this bit is 1 accesses from IOCU6 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU6 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_5	21	When this bit is 1 accesses from IOCU5 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU5 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_4	20	When this bit is 1 accesses from IOCU4 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU4 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_3	19	When this bit is 1 accesses from IOCU3 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU3 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_2	18	When this bit is 1 accesses from IOCU2 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU2 (if implemented) are inhibited.	R/W	1

Table 16: Global Access Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
ACCESS_EN_IOCU_1	17	When this bit is 1 accesses from IOCU1 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU1 (if implemented) are inhibited.	R/W	1
ACCESS_EN_IOCU_0	16	When this bit is 1 accesses from IOCU0 (if implemented) are enabled to write GCR and CPC registers. When this bit is 0 accesses from IOCU0 (if implemented) are inhibited.	R/W	1
0	15:8	Reserved.	R	0
ACCESS_EN	7:0	When bit i is 1 accesses from Core i (if implemented) are enabled to write GCR and CPC registers. When bit is 0 accesses from Core i (if implemented) are inhibited.	R/W	255

5.14.1.15 GCR Global L2 Cache Configuration (L2_CONFIG) Register (offset = 0x0130)

This register provides the L2 cache configuration. The L2 cache size (in bytes) can be computed as associa-tivity * line_size * sets_per_way. For example, if SET_SIZE = 4 (1K), LINE_SIE = 5 (64 Bytes), and ASSOC = 15 (16-ways), the L2 cache is 1024 * 64 * 16 = 1MB.

Figure 5.18 Global L2 Cache Configuration Register Bit Assignments

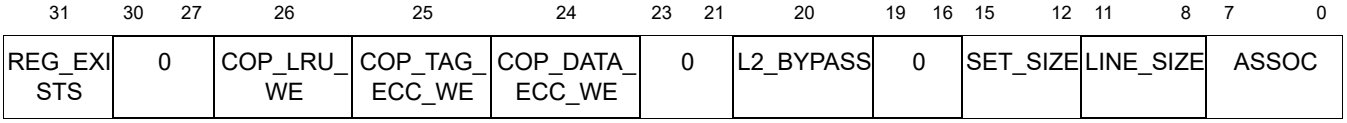


Table 17: Global L2 Cache Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
REG_EXISTS	31	This bit is hardwired to "1" to indicate the presence of the L2_CONFIG register.	R	1
0	30:27	Reserved	R	0
COP_LRU_WE	26	When set to 1, the TAG_LRU field of GCR_L2_TAG_STATE field is written into the L2 LRU RAM when an L2 Store Tag & Data Cache Op is executed. When set to 0, the L2 LRU RAM is not updated when an L2 Store Tag & Data Cache Op is executed.	R/W	1
COP_TAG_ECC_WE	25	When set to 1, the TAG_ECC field of GCR_L2_ECC register is written into the ECC portion of the L2 Tag RAM when an L2 Store & Data Tag Cache Op is executed. When set to 0, the ECC written is computed for the values in GCR_L2_TAG_ADRR and GCR_L2_TAG_STATE when the L2 Store Tag & Data Cache Op is executed.	R/W	0
COP_DATA_ECC_WE	24	When set to 1, the DATA_ECC field of GCR_L2_ECC register is written into the ECC portion of the L2 Data RAM when an L2 Store Tag & Data Cache Op is executed. When set to 0, the ECC written is computed for the values in GCR_L2_DATA and GCR_L2_ when the L2 Store Tag & Data Cache Op is executed.	R/W	0
0	23:21	Reserved	R	0
L2_BYPASS	20	When set to 1 the L2 is placed in bypass mode.	R/W	0
0	19:16	Reserved	R	0
SET_SIZE	15:12	Set Size. L2 cache number of sets per way. SET_SIZE sets/way: 2 256 3 512 4 1024 5 2048 6 4096 7 8192 8 16K 9 32K 10 64K	R	From configuration

Table 17: Global L2 Cache Configuration Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
LINE_SIZE	11:8	L2 data cache line size. 0x5 indicates 64 byte cache line size.	R	5
ASSOC	7:0	L2 cache associativity. 0xF indicates 16-way associativity.	R	From configuration

5.14.1.16 GCR Global SDB Configuration (SDB_CONFIG) Register (offset = 0x0160)

This register Defines the Memory, Intervention, PFU and total SDB for the cluster.

Figure 5.19 Global SDB Configuration Register Bit Assignments

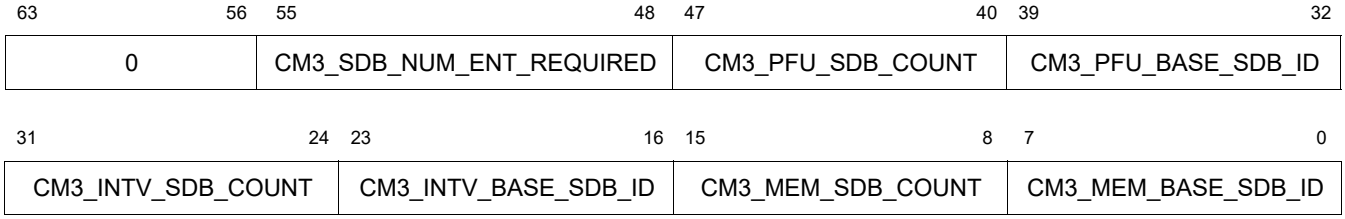


Table 18: Global SDB Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:56	Reserved	R	0
CM3_SDB_NUM_ENT_REQUIRED	55:48	Provides total number of SDBs required.	R	From configuration
CM3_PFU_SDB_COUNT	47:40	Provides SDB count for PFU.	R	From configuration
CM3_PFU_BASE_SDB_ID	39:32	Provides BASE_SDB_ID for PFU SDBs.	R	From configuration
CM3_INTV_SDB_COUNT	31:24	Provides SDB count for Intervention.	R	From configuration
CM3_INTV_BASE_SDB_ID	23:16	Provides BASE_SDB_ID for Intervention SDBs.	R	From configuration
CM3_MEM_SDB_COUNT	15:8	Provides SDB count for Memory.	R	From configuration
CM3_MEM_BASE_SDB_ID	7:0	Provides BASE_SDB_ID for Memory SDBs.	R	From configuration

5.14.1.17 GCR Global IOCU Revision (IOCU_REV) Register (offset = 0x0200)

This register reflects the revision of IOCU.

Figure 5.20 Global IOCU Revision Register Bit Assignments

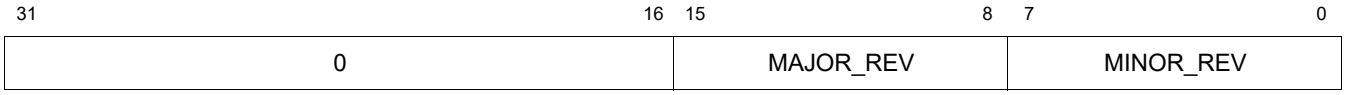


Table 19: Global IOCU Revision Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
MAJOR_REV	15:8	This field reflects the major revision of the IOCU attached to the CM3. A major revision might reflect the changes from one product generation to another. The value of 0x0 means that no IOCU is attached.	R	From configuration
MINOR_REV	7:0	This field reflects the minor revision of the IOCU attached to the CM3. A minor revision might reflect the changes from one release to another.	R	From configuration

5.14.1.18 GCR Global DBU Revision (DBU_REV) Register (offset = 0x0208)

This register reflects the revision of Debug Unit.

Figure 5.21 Global DBU Revision Register Bit Assignments

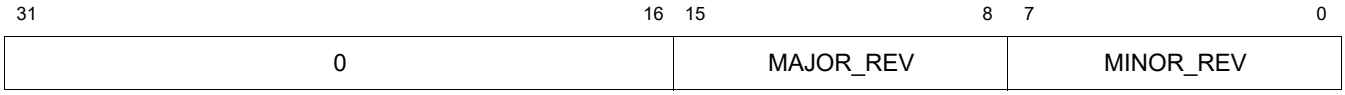


Table 20: Global DBU Revision Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
MAJOR_REV	15:8	This field reflects the major revision of the DBU attached to the CM3. A major revision might reflect the changes from one product generation to another. The value of 0x0 means that no DBU is attached.	R	From configuration
MINOR_REV	7:0	This field reflects the minor revision of the DBU attached to the CM3. A minor revision might reflect the changes from one release to another.	R	From configuration

5.14.1.19 GCR Global Interrupt Controller Revision (AIA_REV) Register (offset = 0x0208)

This register reflects the revision of Interrupt Controller.

Figure 5.22 Global Interrupt Controller Revision Register Bit Assignments

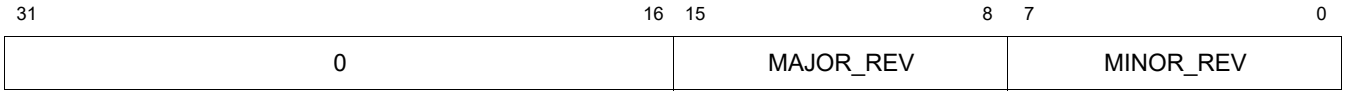


Table 21: Global Interrupt Controller Revision Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
MAJOR_REV	15:8	This field reflects the major revision of the Interrupt Controller implemented in the CM. A major revision might reflect the changes from one product generation to another. The value of 0x0 means that no Interrupt Controller is attached.	R	From configuration
MINOR_REV	7:0	This field reflects the minor revision of the Interrupt Controller implemented in the CM. A minor revision might reflect the changes from one release to another.	R	From configuration

5.14.1.20 GCR Global L2 RAM Configuration (L2_RAM_CONFIG) Register (offset = 0x0240)

This register provides information about the configuration of the L2 cache and controls the dynamic L2 RAM low power states.

Figure 5.23 Global L2 RAM Configuration Register Bit Assignments

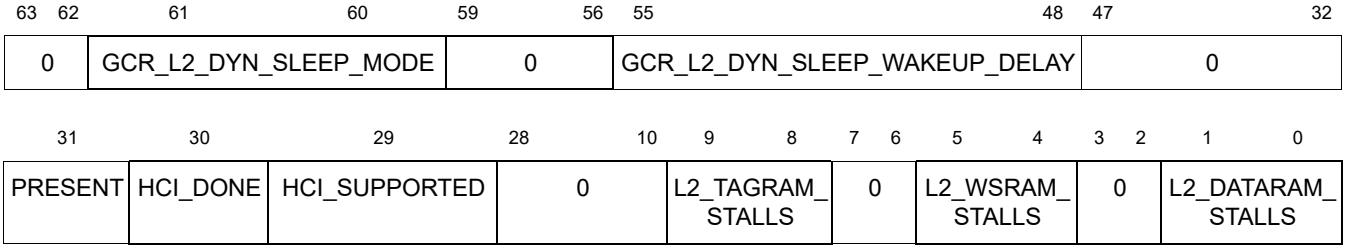


Table 22: Global L2 RAM Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:62	Reserved	R	0
GCR_L2_DYN_SLEEP_MODE	61:60	Controls the L2 cache RAM low power mode entered when all cores are in "sleep" mode and the IOCUs are idle. 0: No low power mode 1: Light Sleep 2: Reserved 3: Reserved	R/W	From configuration
0	59:56	Reserved	R	0
GCR_L2_DYN_SLEEP_WAKEUP_DELAY	55:48	Indicates number of CM clock cycles it takes to wake up the L2 Cache RAMs upon wakeup.	R/W	From configuration
0	47:32	Reserved	R	0
PRESENT	31	Indicates this register exists.	R	1
HCI_DONE	30	Indicates Hardware Cache Initialization is complete.	R	1
HCI_SUPPORTED	29	Indicates Hardware Cache Initialization is supported.	R	0
0	28:10	Reserved	R	0
L2_TAGRAM_STALLS	9:8	Indicates the number of waitstates assumed when accessing the L2 Tag RAMS.	R	From configuration
0	7:6	Reserved	R	0
L2_WSRAM_STALLS	5:4	Indicates the number of waitstates assumed when accessing the L2 Way Select RAMS.	R	From configuration
0	3:2	Reserved	R	0
L2_DATARAM_STALLS	1:0	Indicates the number of waitstates assumed when accessing the L2 Data RAMS.	R	From configuration

5.14.1.21 GCR Global Scratch0 (SCRATCH0) Register (offset = 0x0280)

This register general purpose read/write register.

Figure 5.24 Global Scratch0 Register Bit Assignments

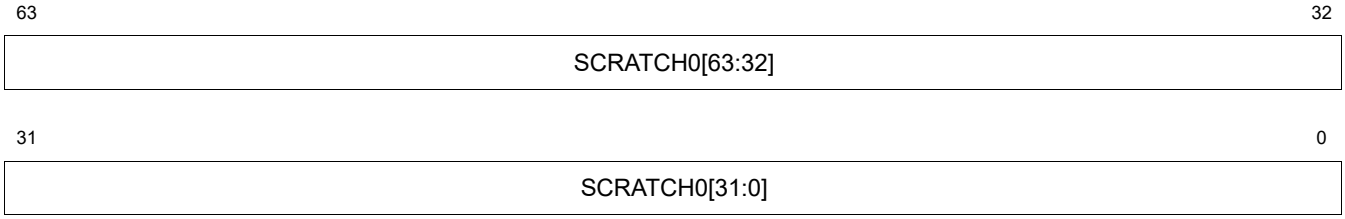


Table 23: Global Scratch0 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SCRATCH0	63:0	General purpose read/write register.	R/W	0

5.14.1.22 GCR Global Scratch1 (SCRATCH1) Register (offset = 0x0288)

This register general purpose read/write register.

Figure 5.25 Global Scratch1 Register Bit Assignments

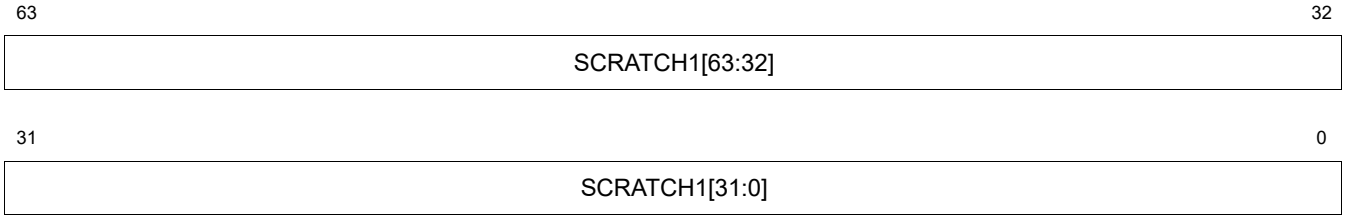


Table 24: Global Scratch1 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SCRATCH1	63:0	General purpose read/write register.	R/W	0

5.14.1.23 GCR Global L2 PFT Control (L2_PFT_CONTROL) Register (offset = 0x0300)

This register controls the L2 hardware prefetcher.

Figure 5.26 Global L2 PFT Control Register Bit Assignments

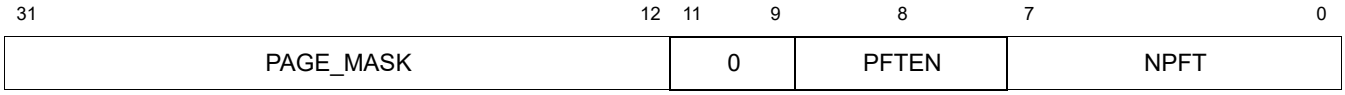


Table 25: Global L2 PFT Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PAGE_MASK	31:12	This field is a mask that indicates the minimum operating system page size. Address bits larger than 31 default to a bit mask of 1. The following settings are supported: 4K page = 0xFFFFF 8K page = 0xFFFFE 16K page = 0xFFFFC 32K page = 0xFFFF8 64K page = 0xFFFF0	R/W	From configuration
0	11:9	Reserved	R	0
PFTEN	8	Prefetch enable. This bit should be set by software only if the number of prefetch units in the NPFT field is greater than zero.	R/W	From configuration
NPFT	7:0	Number of prefetch units. Note that if this field contains a value greater than 0, the PFTEN bit must be set in order for prefetching to occur.	R	From configuration

5.14.1.24 GCR Global L2 PFT Control B (L2_PFT_CONTROL_B) Register (offset = 0x0308)

This register L2 prefetch 2nd control register.

Figure 5.27 Global L2 PFT Control B Register Bit Assignments

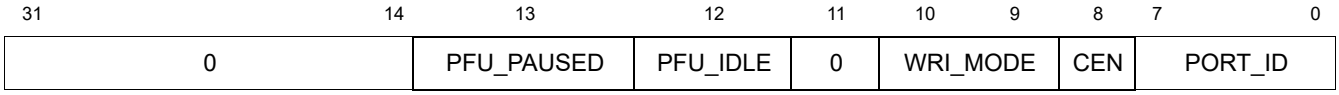


Table 26: Global L2 PFT Control B Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:14	Reserved	R	0
PFU_PAUSED	13	Indicates that the L2 Prefetcher is paused.	R	0
PFU_IDLE	12	Indicates that all Prefetch trackers have aged out and the Prefetcher is idle.	R	0
0	11	Reserved	R	0
WRI_MODE	10:9	Determines how the Prefetch unit handles Coherent Write Invalidate requests. 00: No prefetch 01: Prefetch by reading memory data. 10: Prefetch optimized for ownership when possible, else read memory data 11: Reserved.	R/W	2
CEN	8	Code Prefetch enable.	R/W	From configuration
PORT_ID	7:0	Enable port ID for L2 prefetching. Each bit in this field corresponds to a CM3 port ID. Each bit of this field is encoded as follows: 0: Requests from the corresponding CM3 port are not monitored for L2 prefetching. 1: Requests from the corresponding CM3 port are monitored for L2 prefetching.	R/W	255

5.14.1.25 GCR Global L2 Tag Address (L2_TAG_ADDR) Register (offset = 0x0600)

This register holds address portion of CACHE L2 Load or Store Tag & Data CACHE instruction.

It is loaded with the address information from the L2 Tag RAMs when the L2 Load Tag & Data CACHE instruction is executed. The value of this register is written to the address portion of L2 Tag RAM when an L2 Store Tag and Data CACHE instruction is executed.

Figure 5.28 Global L2 Tag Address Register Bit Assignments

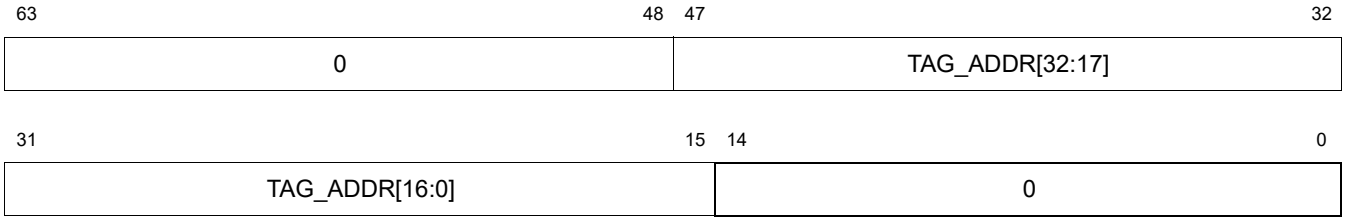


Table 27: Global L2 Tag Address Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
TAG_ADDR	47:15	This field holds the 32-bit address portion of L2 Tag RAM entry. The format of this field changes depending up the cache configuration as described in the System Programmer's Reference.	R/W	0
0	14:0	Reserved	R	0

5.14.1.26 GCR Global L2 Tag State (L2_TAG_STATE) Register (offset = 0x0608)

This register holds State portion of cache L2 Load or Store Tag and Data CACHE instruction.

This register is loaded with the state information from the L2 Tag RAMs when the L2 Load Tag and Data CACHE instruction is executed. The value of this register is written to the state portion of L2 Tag RAM when an L2 Store Tag and Data CACHE instruction is executed.

Figure 5.29 Global L2 Tag State Register Bit Assignments

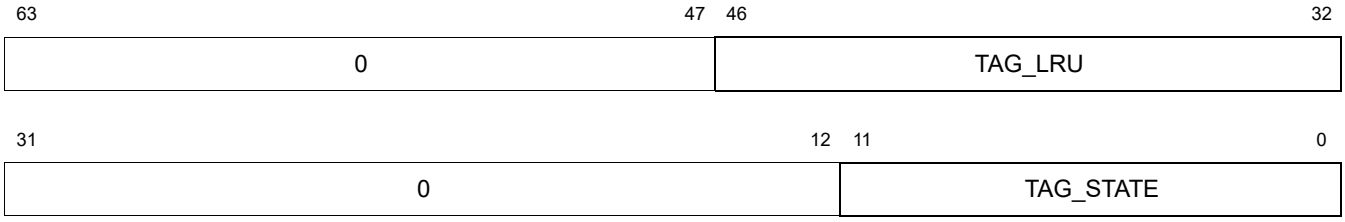


Table 28: Global L2 Tag State Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:47	Reserved	R	0
TAG_LRU	46:32	This field holds the LRU state.	R/W	0
0	31:12	Reserved	R	0
TAG_STATE	11:0	This field holds the L2/L1 state for the L2 Tag RAM entry. The format of this field changes depending up the value of L1_SHARED and the number of CPU cores in the cluster	R/W	0

5.14.1.27 GCR Global L2 Data (L2_DATA) Register (offset = 0x0610)

This register holds results of CACHE L2 Load or Store Tag & Data instruction.

This register is loaded with the information from the L2 Data RAMs when the L2 Load Tag & Data CACHE instruction is executed. The value of this register is written to the L2 Data RAM when a L2 Store Tag and Data CACHE instruction is executed.

Figure 5.30 Global L2 Data Register Bit Assignments

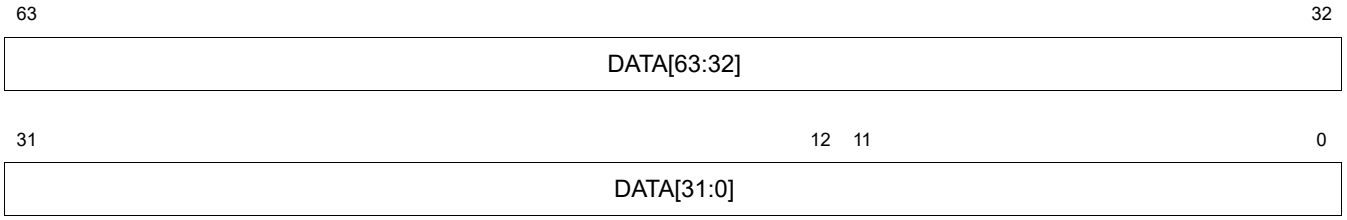


Table 29: Global L2 Data Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
DATA	63:0	This register is loaded with the information from the L2 Data RAMs when the L2 Load Tag and Data CACHE instruction is executed. This value in this register is stored in the L2 Data RAMs when the L2 Store Tag & Data CACHE instruction is executed.	R/W	0

5.14.1.28 GCR Global L2 ECC (L2_ECC) Register (offset = 0x0618)

This register holds Tag and Data ECC field of CACHE L2 Load Tag & Data or Store Tag & Data CACHE instructions.

It is loaded with the ECC information from the L2 Tag and Data RAMs when the L2 Load Tag & Data CACHE instruction is executed. If the GCR_L2_CONFIG.COP_DATA_ECC_WE bit is set then value of the DATA_ECC register is written to the ECC portion of the L2 Data RAM when a L2 Store Tag and Data CACHE instruction is executed.

If the GCR_L2_CONFIG.COP_TAG_ECC_WE bit is set then value of the TAG_ECC register is written to the ECC portion of the L2 Tag RAM when a L2 Store Tag and Data CACHE instruction is executed.

Figure 5.31 Global L2 ECC Register Bit Assignments

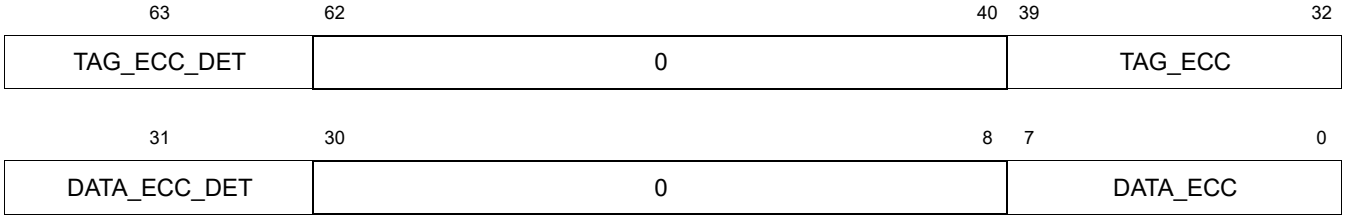


Table 30: Global L2 ECC Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
TAG_ECC_DET	63	Tag ECC Error was detected during the most recent L2 CacheOp load Tag and Data CACHE Instruction.	R/W	0
0	62:40	Reserved	R	0
TAG_ECC	39:32	This register is loaded with the ECC information from the L2 Tag RAMs when the L2 Load Tag & Data CACHE instruction is executed. If the GCR_L2_CONFIG.COP_TAG_ECC_WE bit is set then the value in this register is stored in the ECC portion L2 Tag RAMs when the L2 Store Tag and Data CACHE instruction is executed.	R/W	0
DATA_ECC_DET	31	Data ECC Error was detected during the most recent L2 CacheOp load Tag & Data CACHE Instruction.	R/W	0
0	30:8	Reserved	R	0
DATA_ECC	7:0	This register is loaded with the ECC information from the L2 Data RAMs when the L2 Load Tag and Data CACHE instruction is executed. If the GCR_L2_CONFIG.COP_DATA_ECC_WE bit is set then the value in this register is stored in the ECC portion L2 Data RAMs when the L2 Store Tag and Data CACHE instruction is executed.	R/W	0

5.14.1.29 GCR Global L2SM CacheOp (L2SM_COP) Register (offset = 0x0620)

This register holds CMD, TYPE, MODE, RESULT and PRESENT bit info of the L2 Cache Op State machine.

Figure 5.32 Global L2SM Cache Op Register Bit Assignments

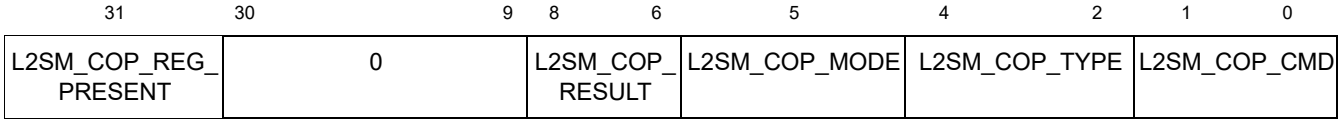


Table 31: Global L2SM Cache Op Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
L2SM_COP_REG_PRESENT	31	Present. Indicates that this register is present.	R	1
0	30:9	Reserved	R	0
L2SM_COP_RESULT	8:6	0x0: DON'T CARE [During RUNNING mode or after reset] 0x1: DONE - NO_ERR [When completes the COP and switches to IDLE mode] 0x2: DONE - ERR [When completes the COP and switches to IDLE mode] 0x3: ABORT - NO_ERR [When completes the COP and switches to IDLE mode] 0x4: ABORT- ERR [When completes the COP and switches to IDLE mode]	R	0
L2SM_COP_MODE	5	0x0: IDLE 0x1: RUNNING	R	0
L2SM_COP_TYPE	4:2	0x0: Index WB inv/Index Inv [Full cache Flush] 0x1: Index Store Tag [Full Cache Init - Fast - Only Tag RAM] 0x2: Index Store Tag [Full cache init - Norm - Tag & Data RAM] 0x3: Reserved 0x4: Hit Inv 0x5: Hit WB Inv 0x6: Hit WB 0x7: Fetch & Lock This field can only be written when the COP SM is in IDLE mode.	R/W	0
L2SM_COP_CMD	1:0	0x0: NOP 0x1: START [START can only be issued in IDLE mode] 0x2: Reserved 0x3: ABORT [ABORT can only be issued in RUNNING mode] Note: It may take few cycles for SM to be IDLE after ABORT is issued	R/W	0

5.14.1.30 GCR Global L2SM Tag Address CacheOp (L2SM_TAG_ADDR_COP) Register (offset = 0x0628)

This register holds Tag address details L2 CacheOp State Machine.

Figure 5.33 Global L2SM Tag Address CacheOp Register Bit Assignments

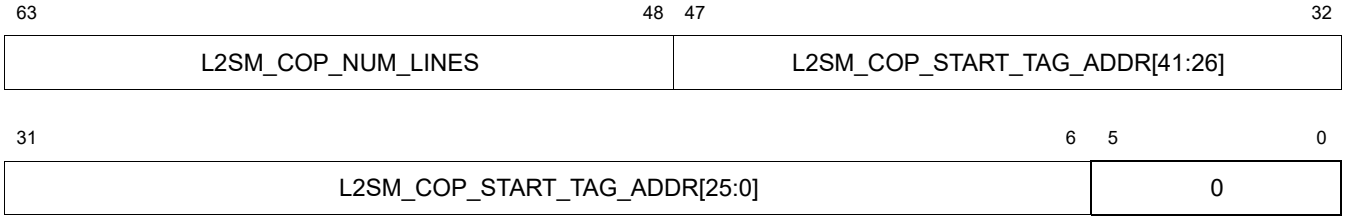


Table 32: Global L2SM Tag Address CacheOp Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
L2SM_COP_NUM_LINES	63:48	• Number of lines (from starting address) to be operated for Requested burst COP. • Max supported number is 65536 (2^16) • This field can only be written when the COP SM is in IDLE mode. • Not valid for index type cache ops.	R/W	0
L2SM_COP_START_TAG_ADDR	47:6	• Starting address (tag) of Burst COP • This field can only be written when the COP SM is in IDLE mode. • Not valid for index type cache ops.	R/W	0
0	5:0	Reserved	R	0

5.14.1.31 GCR Global Semaphore (SEM) Register (offset = 0x0640)

This register provides an uncached semaphore mechanism. A write to this register with write data bit 31 = 1 is inhibited if the SEM_LOCK bit is already 1. A write to this register proceeds normally if the write data has bit 31 = 0 or if the SEM_LOCK bit is currently 0.

To acquire the semaphore:

1. Write this register with bit 31 = 1 and the lower bits with the threads VPID.
2. Read the register.
3. If the value read in step #2 is the same as the value as written in step #1, then semaphore has been acquired, else go to step #1.

To release the semaphore:

1. Write the register with bit 31 = 0.

Figure 5.34 Global Semaphore Register Bit Assignments

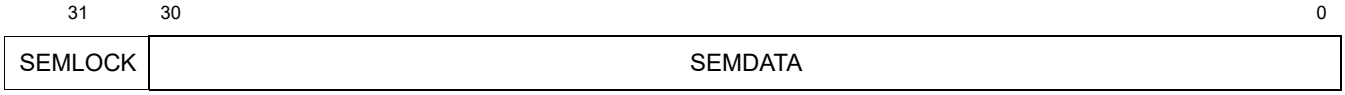


Table 33: Global Semaphore Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SEMLOCK	31	Lock bit on semaphore. A value of 1 indicates that this register is locked. In which case, subsequent writes trying to set this bit to 1 will be inhibited, i.e., the SEMDATA field will not be updated.	R	0
SEMDATA	30:0	Data value on semaphore.	R	0

5.14.1.32 GCR Global Timeout Timer Limit (TIMEOUT_TIMER_LIMIT) Register (offset = 0x0650)

This register provides the timeout limit for the transaction timeout timer in number of CM clocks.

Figure 5.35 Global Timeout Timer Limit Register Bit Assignments

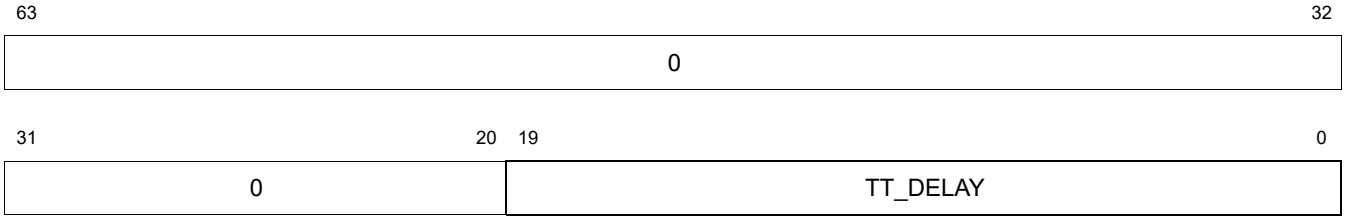


Table 34: Global Timeout Timer Limit Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:20	Reserved	R	0
TT_DELAY	19:0	Timeout limit for transaction timeout timer in number of CM clocks.	R/W	From configuration

5.14.1.33 GCR Global MMIO Requests Limit (MMIO_REQ_LIMIT) Register (offset = 0x06F8)

This register determines the number of MMIO requests that the CM3 will allow to be in flight.

Figure 5.36 Global MMIO Requests Limit Register Bit Assignments

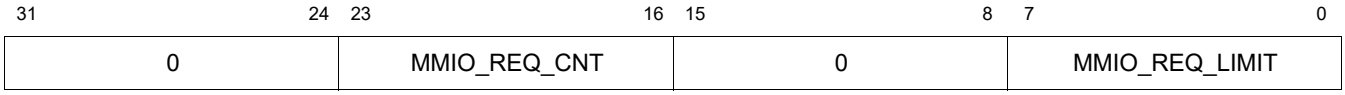


Table 35: Global MMIO Requests Limit Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:24	Reserved	R	0
MMIO_REQ_CNT	23:16	Provides current count of requests in flight to MMIO regions that have REQ_LIMIT request limitations enabled.	R	0
0	15:8	Reserved	R	0
MMIO_REQ_LIMIT	7:0	Determines the number of requests to the regions with request limits enabled that the CM3 will allow to be in flight. Setting a value of 1 allows one outstanding MMIO request. Setting a value of 0 disables the MMIO limiting feature.	R/W	From configuration

5.14.1.34 GCR Global MMIO0 Bottom (MMIO0_BOTTOM) Register (offset = 0x0700)

This register holds lowest address of MMIO Region 0.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 0.

Figure 5.37 Global MMIO0 Bottom Register Bit Assignments

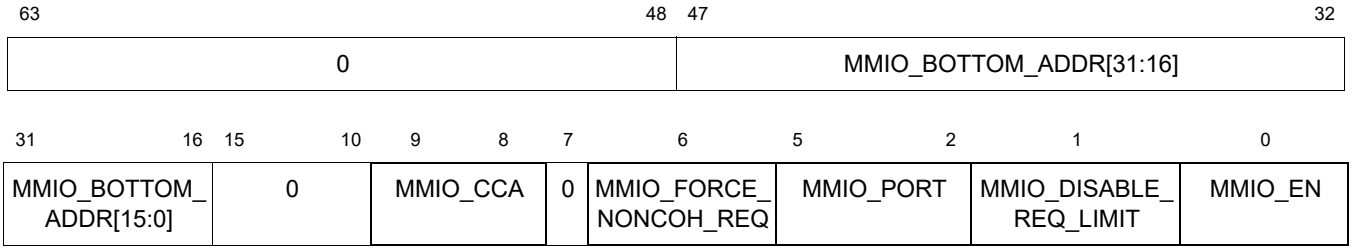


Table 36: Global MMIO0 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of Address bits 47:16 for MMIO region 0.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration

Table 36: Global MMIO0 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 0.	R/W	From configuration

5.14.1.35 GCR Global MMIO0 Top (MMIO0_TOP) Register (offset = 0x0708)

This register holds highest address of MMIO Region 0.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 0.

Figure 5.38 Global MMIO0 Top Register Bit Assignments

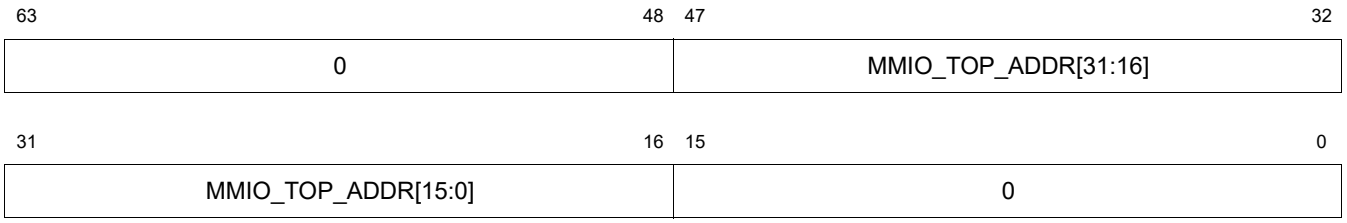


Table 37: Global MMIO0 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 31:0 for MMIO region 0.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.36 GCR Global MMIO1 Bottom (MMIO1_BOTTOM) Register (offset = 0x0710)

This register holds lowest address of MMIO Region 1.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 1.

Figure 5.39 Global MMIO1 Bottom Register Bit Assignments

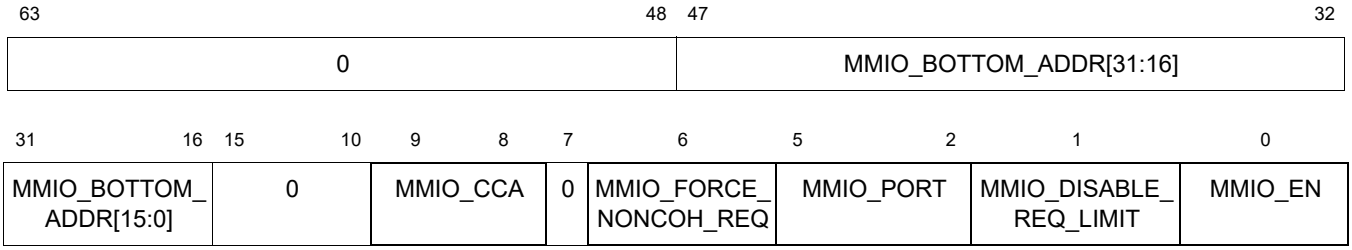


Table 38: Global MMIO1 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of Address bits 31:0 for MMIO region 1.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration

Table 38: Global MMIO1 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to cmd type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable cmd type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 1.	R/W	From configuration

5.14.1.37 GCR Global MMIO1 Top (MMIO1_TOP) Register (offset = 0x0718)

This register holds highest address of MMIO Region 1.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 1.

Figure 5.40 Global MMIO1 Top Register Bit Assignments

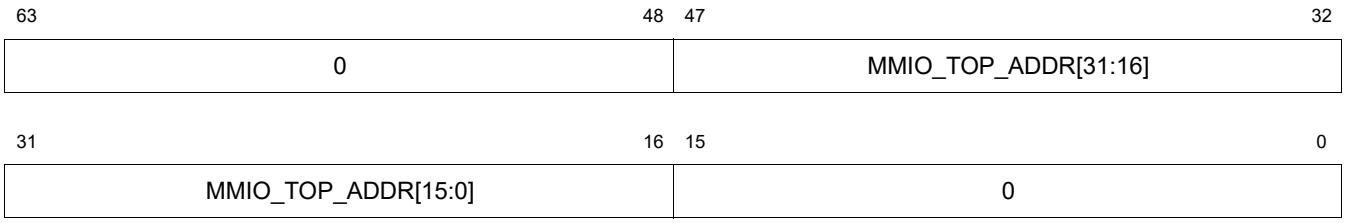


Table 39: Global MMIO1 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 31:0 for MMIO region 1.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.38 GCR Global MMIO2 Bottom (MMIO2_BOTTOM) Register (offset = 0x0720)

This register holds lowest address of MMIO Region 2.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 2.

Figure 5.41 Global MMIO2 Bottom Register Bit Assignments

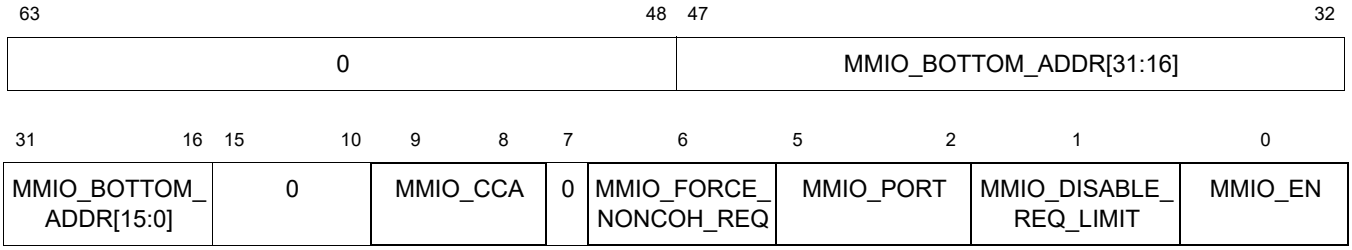


Table 40: Global MMIO2 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of Address bits 31:0 for MMIO region 2.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration

Table 40: Global MMIO2 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 2.	R/W	From configuration

5.14.1.39 GCR Global MMIO2 Top (MMIO2_TOP) Register (offset = 0x0728)

This register holds highest address of MMIO Region 2.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 2.

Figure 5.42 Global MMIO2 Top Register Bit Assignments

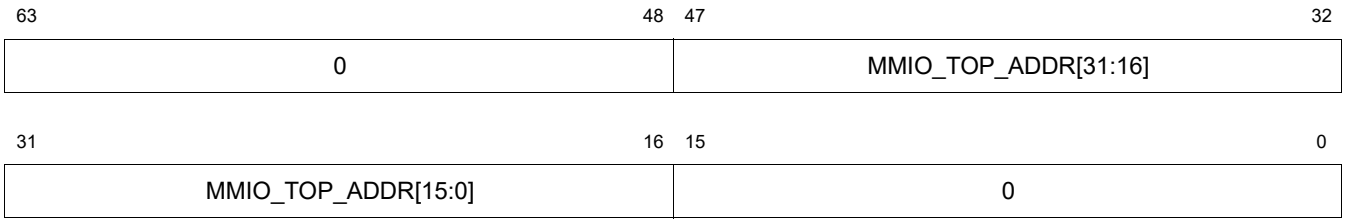


Table 41: Global MMIO2 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of address bits 31:0 for MMIO region 2.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.40 GCR Global MMIO3 Bottom (MMIO3_BOTTOM) Register (offset = 0x0730)

This register holds lowest address of MMIO Region 3.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 3.

Figure 5.43 Global MMIO3 Bottom Register Bit Assignments

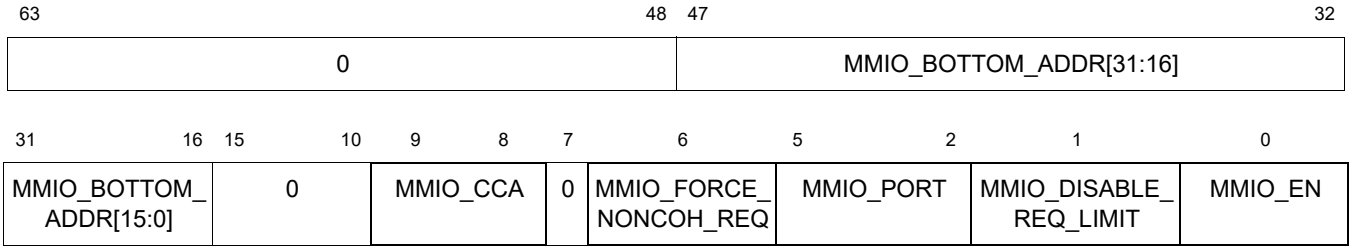


Table 42: Global MMIO3 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of address bits 31:0 for MMIO region 3.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match. MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC). MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA). MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA).	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration

Table 42: Global MMIO3 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 3.	R/W	From configuration

5.14.1.41 GCR Global MMIO3 Top (MMIO3_TOP) Register (offset = 0x0728)

This register holds highest address of MMIO Region 3.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 3.

Figure 5.44 Global MMIO3 Top Register Bit Assignments

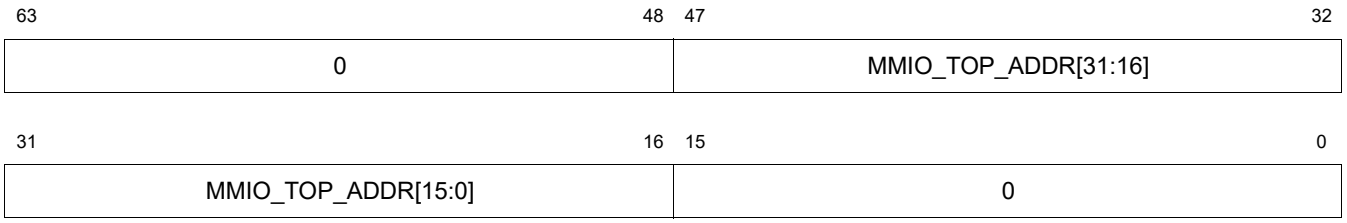


Table 43: Global MMIO3 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved.	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 47:16 for MMIO region 3.	R/W	From configuration
0	15:0	Reserved.	R	0

5.14.1.42 GCR Global MMIO4 Bottom (MMIO4_BOTTOM) Register (offset = 0x0740)

This register holds lowest address of MMIO Region 4.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 4.

Figure 5.45 Global MMIO4 Bottom Register Bit Assignments

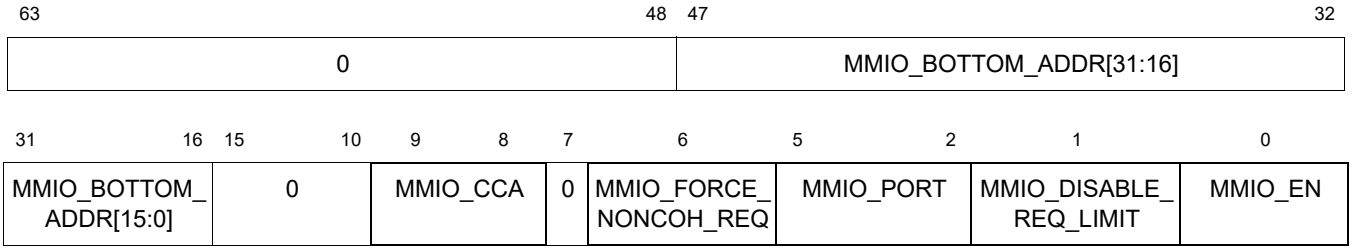


Table 44: Global MMIO4 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved.	R	0
MMIO_BOTTOM_ ADDR	47:16	Lower limit of Address bits 47:16 for MMIO region 4.	R/W	From configuration
0	15:10	Reserved.	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved.	R	0
MMIO_FORCE_ NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration

Table 44: Global MMIO4 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 4.	R/W	From configuration

5.14.1.43 GCR Global MMIO4 Top (MMIO4_TOP) Register (offset = 0x0748)

This register holds highest address of MMIO Region 4.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 4.

Figure 5.46 Global MMIO4 Top Register Bit Assignments

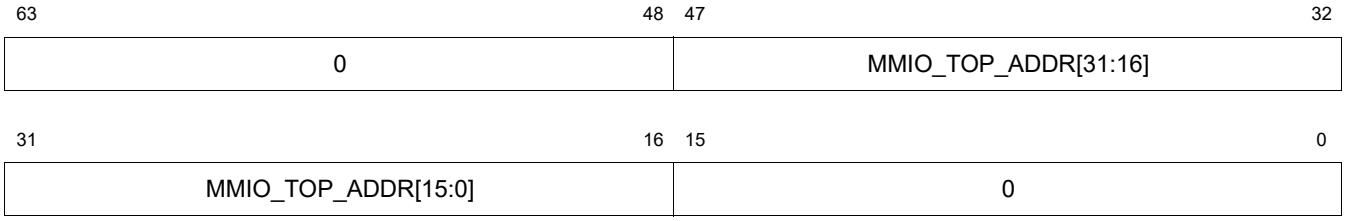


Table 45: Global MMIO4 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 47:16 for MMIO region 4.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.44 GCR Global MMIO5 Bottom (MMIO5_BOTTOM) Register (offset = 0x0750)

This register holds lowest address of MMIO Region 5.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 5.

Figure 5.47 Global MMIO5 Bottom Register Bit Assignments

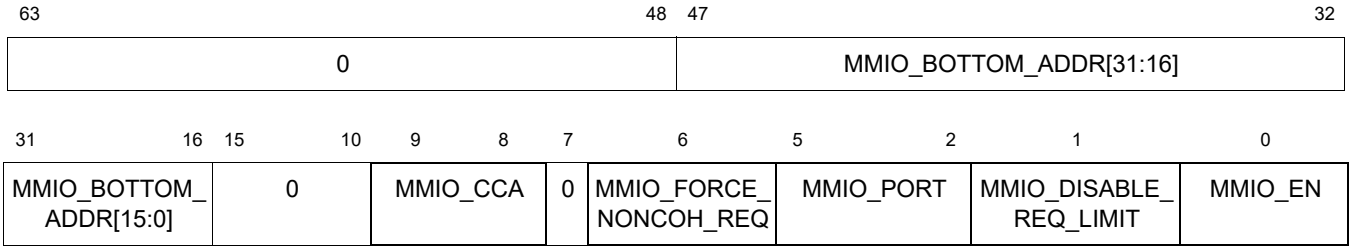


Table 46: Global MMIO5 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of Address bits 47:16 for MMIO region 5.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration

Table 46: Global MMIO5 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 5.	R/W	From configuration

5.14.1.45 GCR Global MMIO5 Top (MMIO5_TOP) Register (offset = 0x0758)

This register holds highest address of MMIO Region 5.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 5.

Figure 5.48 Global MMIO5 Top Register Bit Assignments

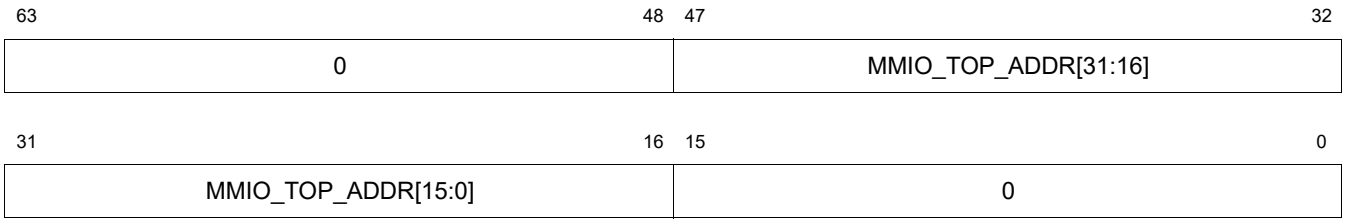


Table 47: Global MMIO5 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 47:16 for MMIO region 5.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.46 GCR Global MMIO6 Bottom (MMIO6_BOTTOM) Register (offset = 0x0760)

This register holds lowest address of MMIO Region 6.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 6.

Figure 5.49 Global MMIO6 Bottom Register Bit Assignments

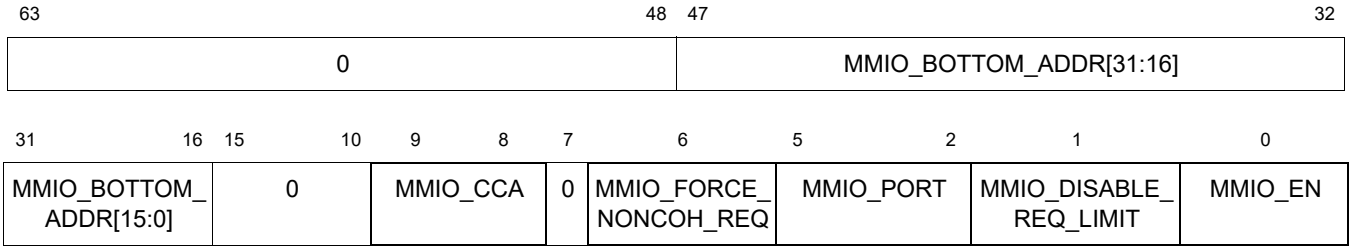


Table 48: Global MMIO6 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ ADDR	47:16	Lower limit of Address bits 47:16 for MMIO region 6.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_ NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration

Table 48: Global MMIO6 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to cmd type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable cmd type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 6.	R/W	From configuration

5.14.1.47 GCR Global MMIO6 Top (MMIO6_TOP) Register (offset = 0x0768)

This register holds highest address of MMIO Region 6.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 6.

Figure 5.50 Global MMIO6 Top Register Bit Assignments

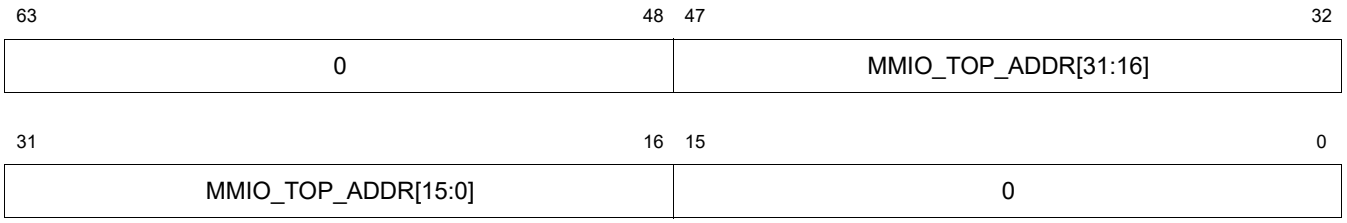


Table 49: Global MMIO6 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 47:16 for MMIO region 6.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.1.48 GCR Global MMIO7 Bottom (MMIO7_BOTTOM) Register (offset = 0x0770)

This register holds lowest address of MMIO Region 7.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 7.

Figure 5.51 Global MMIO7 Bottom Register Bit Assignments

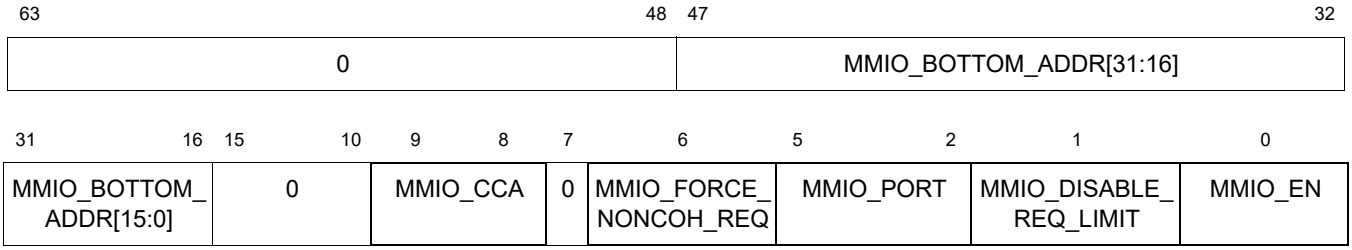


Table 50: Global MMIO7 Bottom Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_BOTTOM_ADDR	47:16	Lower limit of Address bits 47:16 for MMIO region 7.	R/W	From configuration
0	15:10	Reserved	R	0
MMIO_CCA	9:8	Allows MMIO region hit to be qualified by CCA in addition to address. If this field is zero, then all CCA types may fall into this MMIO region. MMIO region hit is determined based upon just address hit. If bits in MMIO_CCA are set, then MMIO qualification is based upon address and CCA. MMIO_CCA = 2'b00: CCA is not considered as part of the match MMIO_CCA = 2'b01: This region will only match if the CCA is Uncached (UC) MMIO_CCA = 2'b10: This region will only match if the CCA is Uncached Accelerated (UCA) MMIO_CCA = 2'b11: This region will only match if the CCA is Uncached (UC) or Uncached Accelerated (UCA)	R/W	From configuration
0	7	Reserved	R	0
MMIO_FORCE_NONCOH_REQ	6	If a transaction that hits this region generates a request out to a coherent interconnect, force the request to be non-coherent. The request will be externalized to ACE as ReadNoSnoop/WriteNoSnoop.	R/W	From configuration

Table 50: Global MMIO7 Bottom Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
MMIO_PORT	5:2	Specify which port issues requests to: 15:12 - reserved 11 - AUX3 10 - AUX2 9 - AUX1 8 - AUX0 7:1 - reserved 0 - Main memory port; MEM	R/W	From configuration
MMIO_DISABLE_ REQ_LIMIT	1	Determines whether this MMIO region is subject to CMD type,CCA and number of requests outstanding limits imposed by MMIO_REQ_LIMIT. Set to 1 to disable CMD type, CCA and MMIO REQ_LIMIT limitations.	R/W	From configuration
MMIO_EN	0	Enable MMIO region 7.	R/W	From configuration

5.14.1.49 GCR Global MMIO7 Top (MMIO7_TOP) Register (offset = 0x0778)

This register holds highest address of MMIO Region 7.

NOTE: This register only exists if GCR_CONFIG.ADDR_REGIONS is greater than 7.

Figure 5.52 Global MMIO7 Top Register Bit Assignments

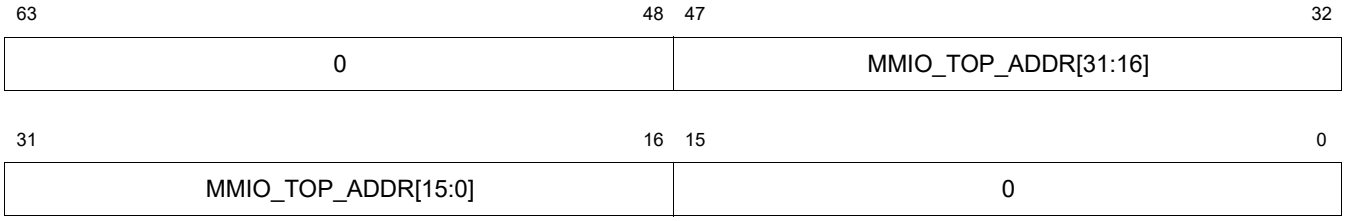


Table 51: Global MMIO7 Top Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:48	Reserved	R	0
MMIO_TOP_ADDR	47:16	Upper limit of Address bits 47:16 for MMIO region 7.	R/W	From configuration
0	15:0	Reserved	R	0

5.14.2 GCR.Debug Registers

5.14.2.1 GCR Debug TCB ControlD (TCBCONTROLD) Register (offset = 0x0810)

This register controls CM3 PDTrace. It only exists if the CM3 is configured with PDTrace.

Figure 5.53 Debug TCB ControlD Register Bit Assignments

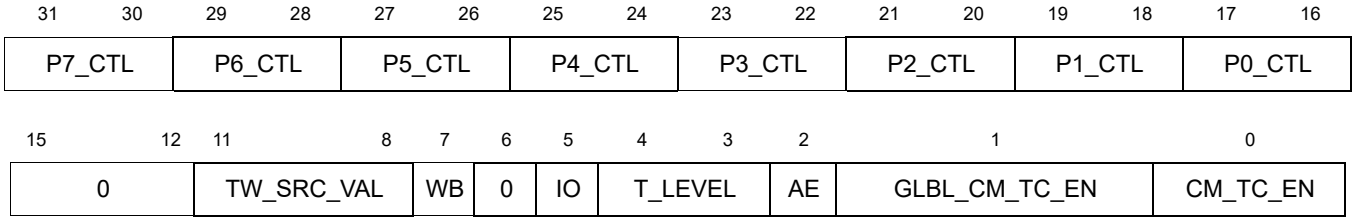


Table 52: Debug TCB ControlD Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
P7_CTL	31:30	Provides specific control over tracing transactions on Port 7 of the CM3. 00: Tracing Enabled, no Address Tracing 01: Tracing Enabled with Address Tracing 10: Reserved 11: Tracing Disabled	R/W	0
P6_CTL	29:28	Provides specific control over tracing transactions on Port 6 of the CM3. See Encoding for P7_CTL.	R/W	0
P5_CTL	27:26	Provides specific control over tracing transactions on Port 5 of the CM3. See Encoding for P7_CTL.	R/W	0
P4_CTL	25:24	Provides specific control over tracing transactions on Port 4 of the CM3. See Encoding for P7_CTL.	R/W	0
P3_CTL	23:22	Provides specific control over tracing transactions on Port 3 of the CM3. See Encoding for P7_CTL.	R/W	0
P2_CTL	21:20	Provides specific control over tracing transactions on Port 2 of the CM3. See Encoding for P7_CTL.	R/W	0
P1_CTL	19:18	Provides specific control over tracing transactions on Port 1 of the CM3. See Encoding for P7_CTL.	R/W	0
P0_CTL	17:16	Provides specific control over tracing transactions on Port 0 of the CM3. See Encoding for P7_CTL.	R/W	0
0	15:12	Reserved	R	0
TW_SRC_VAL	11:8	The source ID inserted into the Trace Word by the CM3. NOTE: When disabling trace by setting GLBL_CM_TC_EN to 0, the value in TW_SRC_VAL continues to be used until all trace messages have been flushed from the CM. Therefore, when writing to this register to disable CM, the correct value must still be written into the TW_SRC_VAL field.	R/W	15
WB	7	When this bit is set, Coherent Writeback requests are traced. If this bit is not set, all Coherent Writeback requests are suppressed from the CM3 PDTrace Stream.	R/W	0

Table 52: Debug TCB ControlD Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
0	6	Reserved	R	0
IO	5	Inhibit Overflow on the CM3 PDTrace FIFO full condition. When set to 0, the CM3 will drop a new PDTrace message if the internal PDTrace FIFOs are full. When set to 1, the CM3 will not drop PDTrace messages, but may stall transactions within the CM3 when the internal PDTrace FIFOs are full.	R/W	0
T_LEVEL	4:3	This defines the current trace level being used by the CM3 PDTrace: 00: Minimal 01: Medium 10: Full 11: Reserved	R/W	0
AE	2	When set to 1, address tracing is always enabled for the CM3. When set to 0, address tracing may be enabled on a per-port basis through the Px_CTL bits.	R/W	0
GLBL_CM_TC_EN	1	Setting this bit to 1 enables tracing from the CM3 as long as the CM_EN bit is also enabled.	R/W	0
CM_TC_EN	0	This is the master trace enable for the CM3. When zero, tracing from the CM3 is always disabled. When set to one, tracing is enabled if GLBL_CM_TC_EN is set or at least one of the cores asserts cm_trace_on bit	R/W	0

5.14.2.2 GCR Debug TCB ControlE (TCBCONTROLE) Register (offset = 0x0820)

This register status reg for CM3 PDTrace.It only exists if the CM3 is configured with PDTrace.

Figure 5.54 Debug TCB ControlE Register Bit Assignments

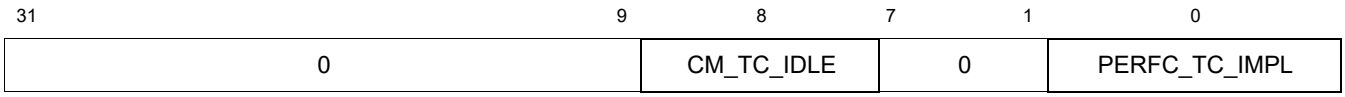


Table 53: Debug TCB ControlE Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:9	Reserved	R	0
CM_TC_IDLE	8	Indicates when CM trace unit is idle. This bit is updated by HW and is read by software whenever the trace is disabled.	R	1
0	7:1	Reserved	R	0
PERFC_TC_IMPL	0	When set indicates tracing of performance counters is implemented.	R	1

5.14.2.3 GCR Debug TCB Performance Counter Trace (TCBPERFCNTR) Register (offset = 0x0830)

This register config reg for CM3 PDTrace. It only exists if the CM3 is configured with PDTrace.

Figure 5.55 Debug TCB Performance Counter Trace Register Bit Assignments

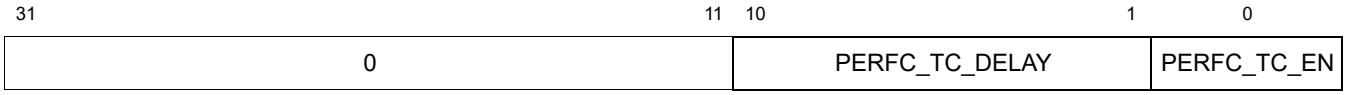


Table 54: Debug TCB Performance Counter Trace Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:11	Reserved	R	0
PERFC_TC_DELAY	10:1	Delay between two perf counter trace message generation i.e. snapshot delay.	R/W	From configuration
PERFC_TC_EN	0	When set indicates tracing of performance counters is enabled.	R/W	From configuration

5.14.2.4 GCR Debug Performance Counter Control (PC_CTL) Register (offset = 0x0900)

This register controls starting/stopping of Performance Counters.

Figure 5.56 Debug Performance Counter Control Register Bit Assignments

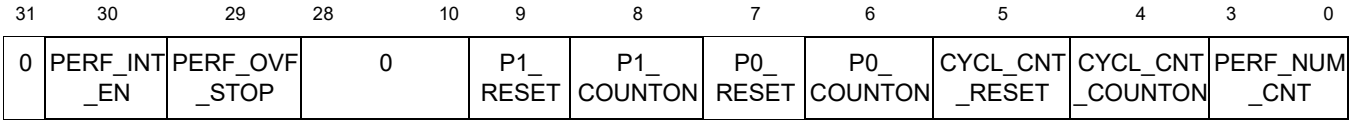


Table 55: Debug Performance Counter Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31	Reserved	R	0
PERF_INT_EN	30	Enable Interrupt on counter overflow. If set to 1, a CM3 performance counter interrupt is generated when any enabled CM3 performance counter overflows.	R/W	0
PERF_OVF_STOP	29	Stop Counting on overflow. If set to 1, all CM3 Performance counters stop counting when any enabled CM3 performance counter overflows i.e., the counter has reached 0xFFFF_FFFF.	R/W	0
0	28:10	Reserved	R	0
P1_RESET	9	If P1_RESET is written to 1 when P1_COUNTON is written to 1, then CM3 Performance Counter 1 and the P1_OF bit is reset before counting is started. If P1_RESET is written to 0 when P1_COUNTON is written to 1, then counting is resumed from previous value. This bit is automatically set to 0 when the counter is reset, so P1_RESET is always read as 0.	R/W	0
P1_COUNTON	8	Start/Stop Counting. If this bit is set to 1 then CM3 Performance Counter 1 starts counting the specified event. If this bit is set to 0 then CM3 Performance Counter 1 is disabled. This bit is automatically set to 0 if any counter overflows and Perf_Ovf_Stop is set to 1.	R/W	0
P0_RESET	7	If P0_RESET is written to 1 when P0_COUNTON is written to 1, then CM3 Performance Counter 0 and the P0_OF bit is reset before counting is started. If P0_RESET is written to 0 when P0_COUNTON is written to 1, then counting is resumed from previous value. This bit is automatically set to 0 when the counter is reset, so P0_RESET is always read as 0.	R/W	0
P0_COUNTON	6	Start/Stop Counting. If this bit is set to 1 then CM3 Performance Counter 0 starts counting the specified event. If this bit is set to 0 then CM3 Performance Counter 0 is disabled. This bit is automatically set to 0 if any counter overflows and Perf_Ovf_Stop is set to 1.	R/W	0

Table 55: Debug Performance Counter Control Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
CYCL_CNT_RESET	5	If CYCL_CNT_RESET is written to 1 when CYCL_CNT_COUNTON is written to 1, then CM3 Cycle Counter and the Cycl_Cnt_OF bit is reset before counting is started. If CYCL_CNT_RESET is written to 0 when CYCL_CNT_COUNTON is written to 1, then counting is resumed from previous value. This bit is automatically set to 0 when the counter is reset, so CYCL_CNT_RESET is always read as 0.	R/W	0
CYCL_CNT_ COUNTON	4	Start/Stop the Cycle Counter. If this bit is set to 1 then CM3 Cycle Counter starts counting. If this bit is set to 0 then CM3 Cycle Counter is disabled. This bit is automatically set to 0 if any Counter Overflows and Perf_Ovf_Stop is set to 1.	R/W	0
PERF_NUM_CNT	3:0	The number of performance counters implemented (not including the cycle counter). The CM3 has 2 performance counters.	R/W	2

5.14.2.5 GCR Debug Performance Counter Overflowed (PC_OV) Register (offset = 0x0920)

This register indicates which performance counters have overflowed.

Figure 5.57 Debug Performance Counter Overflowed Register Bit Assignments

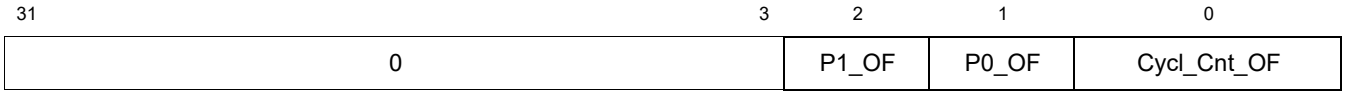


Table 56: Debug Performance Counter Overflowed Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:3	Reserved	R	0
P1_OF	2	If this bit is set to 1, CM3 Performance Counter 1 has overflowed i.e., the counter has reached 0xFFFF_FFFF.	RW1C	0
P0_OF	1	If this bit is set to 1, CM3 Performance Counter 0 has overflowed i.e., the counter has reached 0xFFFF_FFFF.	RW1C	0
Cycl_Cnt_OF	0	If this bit is set to 1, CM3 Cycle Counter 0 has overflowed i.e., the counter has reached 0xFFFF_FFFF.	RW1C	0

5.14.2.6 GCR Debug Performance Counter Event (PC_EVENT) Register (offset = 0x0930)

This register select event type of each CM3 performance counter.

Figure 5.58 Debug Performance Counter Event Register Bit Assignments

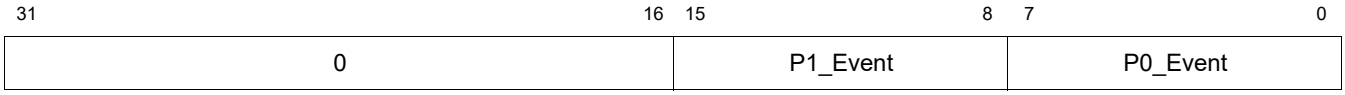


Table 57: Debug Performance Counter Event Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
P1_Event	15:8	Event Selection for CM3 Performance Counter 1. Refer to the System Programmer's Reference for more information.	R/W	0
P0_Event	7:0	Event Selection for CM3 Performance Counter 0. Refer to the System Programmer's Reference for more information.	R/W	0

5.14.2.7 GCR Debug Performance Counter Cycles (PC_CYCL) Register (offset = 0x0980)

This register counts cycles.

Figure 5.59 Debug Performance Counter Cycles Register Bit Assignments

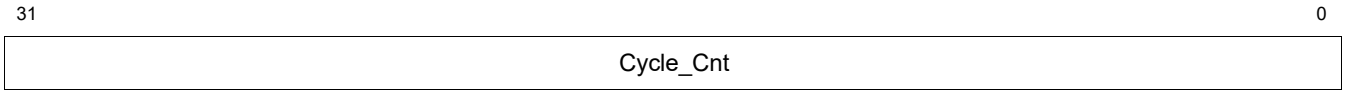


Table 58: Debug Performance Counter Cycles Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
Cycle_Cnt	31:0	32-bit count of CM3 clock cycles.	R/W	0

5.14.2.8 GCR Debug Performance Counter Qualifier0 (PC_QUAL0) Register (offset = 0x0990)

This register performance counter 0 event qualifiers.

Figure 5.60 Debug Performance Counter Qualifier0 Register Bit Assignments

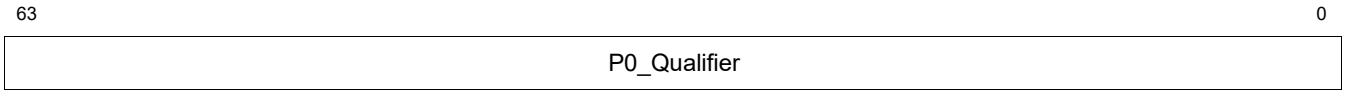


Table 59: Debug Performance Counter Qualifier0 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
P0_Qualifier	63:0	CM3 Performance Counter 0 Event Qualifier. The qualifier corresponds to the event configured through the Performance Counter 0 Event Select Register.	R/W	0

5.14.2.9 GCR Debug Performance Counter Value0 (PC_CNT0) Register (offset = 0x0998)

This register performance counter a value.

Figure 5.61 Debug Performance Counter Value0 Register Bit Assignments

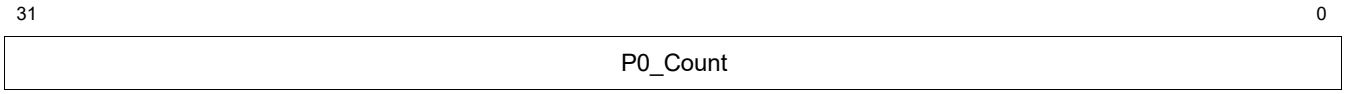


Table 60: Debug Performance Counter Value0 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
P0_Count	31:0	32-bit Performance Counter. The event counted is specified in the CM3 Performance Counter Event Select Register and by the corresponding Qualifier Register.	R/W	0

5.14.2.10 GCR Debug Performance Counter Qualifier1 (PC_QUAL1) Register (offset = 0x09A0)

This register performance counter 1 event qualifiers.

Figure 5.62 Debug Performance Counter Qualifier1 Register Bit Assignments

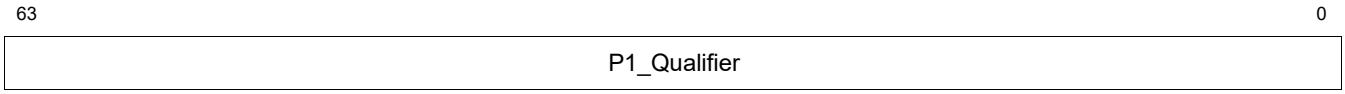


Table 61: Debug Performance Counter Qualifier1 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
P1_Qualifier	63:0	CM3 Performance Counter 1 Event Qualifier. The qualifier corresponds to the event configured through the Performance Counter 1 Event Select Register.	R/W	0

5.14.2.11 GCR Debug Performance Counter Value1 (PC_CNT1) Register (offset = 0x09a8)

This register performance counter a value.

Figure 5.63 Debug Performance Counter Value1 Register Bit Assignments

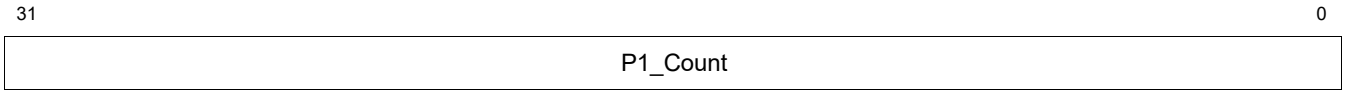


Table 62: Debug Performance Counter Value1 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
P1_Count	31:0	32-bit Performance Counter. The event counted is specified in the CM3 Performance Counter Event Select Register and by the corresponding Qualifier Register.	R/W	0

5.14.3 GCR.Core Registers

The GCR.Core region contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 63: GCR.Core Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x02000 0x02100 0x02200 ..... 0x05F00	GCR.Core[0-63] .H0_RESET_BASE	Core[0-63] Hart0 Reset Program Counter (PC)
0x02008 0x02108 0x02208 ..... 0x05F08	GCR.Core[0-63] .H1_RESET_BASE	Core[0-63] Hart1 Reset Program Counter (PC)
0x02010 0x02110 0x02210 ..... 0x05F10	GCR.Core[0-63] .H2_RESET_BASE	Core[0-63] Hart2 Reset Program Counter (PC)
0x02018 0x02118 0x02218 ..... 0x05F18	GCR.Core[0-63] .H3_RESET_BASE	Core[0-63] Hart3 Reset Program Counter (PC)
0x02020 0x02120 0x02220 ..... 0x05F20	GCR.Core[0-63] .H4_RESET_BASE	Core[0-63] Hart4 Reset Program Counter (PC)
0x02028 0x02128 0x02228 ..... 0x05F28	GCR.Core[0-63] .H5_RESET_BASE	Core[0-63] Hart5 Reset Program Counter (PC)
0x02030 0x02130 0x02230 ..... 0x05F30	GCR.Core[0-63] .H6_RESET_BASE	Core[0-63] Hart6 Reset Program Counter (PC)
0x02038 0x02138 0x02238 ..... 0x05F38	GCR.Core[0-63] .H7_RESET_BASE	Core[0-63] Hart7 Reset Program Counter (PC)

Table 63: GCR.Core Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x02040 0x02140 0x02240 ..... 0x05F40	GCR.Core[0-63] .H8_RESET_BASE	Core[0-63] Hart8 Reset Program Counter (PC)
0x02048 0x02148 0x02248 ..... 0x05F48	GCR.Core[0-63] .H9_RESET_BASE	Core[0-63] Hart9 Reset Program Counter (PC)
0x02050 0x02150 0x02250 ..... 0x05F50	GCR.Core[0-63] .H10_RESET_BASE	Core[0-63] Hart10 Reset Program Counter (PC)
0x02058 0x02158 0x02258 ..... 0x05F58	GCR.Core[0-63] .H11_RESET_BASE	Core[0-63] Hart11 Reset Program Counter (PC)
0x02060 0x02160 0x02260 ..... 0x05F60	GCR.Core[0-63] .H12_RESET_BASE	Core[0-63] Hart12 Reset Program Counter (PC)
0x02068 0x02168 0x02268 ..... 0x05F68	GCR.Core[0-63] .H13_RESET_BASE	Core[0-63] Hart13 Reset Program Counter (PC)
0x02070 0x02170 0x02270 ..... 0x05F70	GCR.Core[0-63] .H14_RESET_BASE	Core[0-63] Hart14 Reset Program Counter (PC)
0x02078 0x02178 0x02278 ..... 0x05F78	GCR.Core[0-63] .H15_RESET_BASE	Core[0-63] Hart15 Reset Program Counter (PC)
0x020F8 0x021F8 0x022F8 ..... 0x05FF8	GCR.Core[0-63] .COH_EN	Core[0-63] Coherence Enable

5.14.3.1 GCR HART Reset Exception Base (RESET_BASE) Register (offset = see below)

Offset: 0x2000 + 0x100 * CORENUM + 0x008 * HARTNUM from GCR_BASE.

This register sets the Reset Exception Base for the local Hart.

It is instantiated for each Hart in the cluster. This register is used to drive the core_exception_base[31:12] input to the local Hart.

Figure 5.64 HART Reset Exception Base Register Bit Assignments

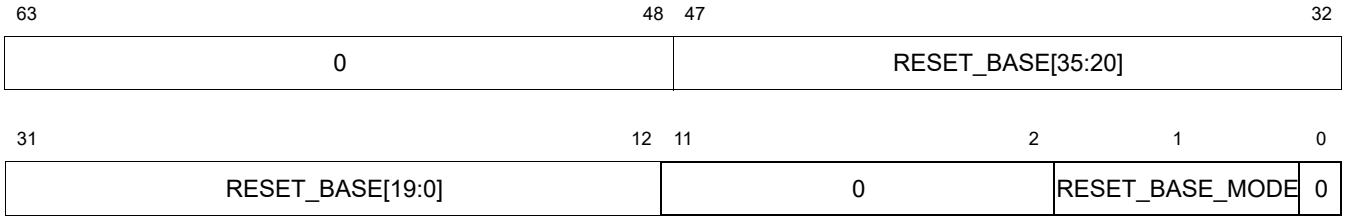


Table 64: HART Reset Exception Base Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63.:48	Reserved	R	0
RESET_BASE	47:12	Bits [47:12] of the virtual address that the local core will use as the exception base.	R/W	From configuration
0	11:2	Reserved	R	0
RESET_BASE_MODE	1	Legacy field, always 1 for MIPS implementations of RISCV cores.	R	1
0	0	Reserved	R	0

5.14.3.2 GCR Core Enables Coherence (COH_EN) Register (offset = see below)

Offset: 0x020f8 + 0x100 * CORENUM from GCR_BASE for Core[0..63]

This register enables coherence for this core.

Setting this bit has 2 effects: First, the CPC will not transition power states for this core; Second, the CM3 may send interventions to this core. Note that the software must follow the appropriate procedure when setting/clearing this bit as outlined in the System Programmer's Reference. This register is instantiated for each Core domain.

Figure 5.65 Core Enables Coherence Register Bit Assignments

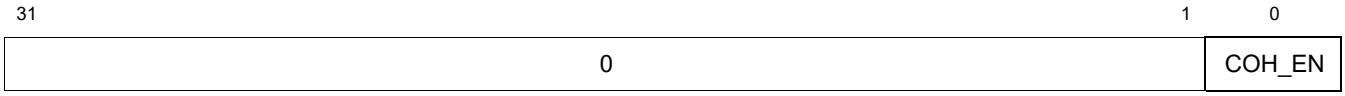


Table 65: Core Enables Coherence Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
COH_EN	0	Enables Coherence for the core.	R/W	0

5.14.4 CPC.Global Registers

The CPC.Global region contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 66: CPC.Global Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x08008	CPC.Global.SEQDEL_REG	Time between microsteps of CPC domain sequencer
0x08010	CPC.Global.RAIL_REG	Rail power-up timer to delay
0x08018	CPC.Global.RESETLEN_REG	Duration of domain reset sequence
0x08020	CPC.Global.REVISION_REG	RTL Revision of CPC
0x08028	CPC.Global.CC_CTL_REG	CPC global clock change configuration, control and status.
0x08030	CPC.Global.PWRUP_CTL_REG	Control CM Power independent of Cores' power states
0x08038	CPC.Global.RES_REL_REG	Reset release and Clock Enable timing
0x08040	CPC.Global.ROCC_CTL_REG	Which cores have been reset
0x08048	CPC.Global.PRESCALE_ CC_CTL_REG	Controls Prescale Clock changes
0x08050	CPC.Global.MTIME_REG	RISC-V mtime register
0x08058	CPC.Global.TIMECTL_REG	Control RISC-V mtime and htime registers.
0x08060	CPC.Global.CLK_GATE_DIS_REG	Disable module level clock gaters.
0x08068	CPC.Global.FAULT_STATUS	OR of all fault domain status registers. FUSA CPUs only.
0x08070	CPC.Global.FAULT_SUPPORTED	Design configured for each fault type? FUSA CPUs only.
0x08078	CPC.Global.FAULT_ENABLE	Is each fault type enabled? FUSA CPUs only.
0x08090	CPC.Global.HRTIME_REG	High resolution timer register
0x08138	CPC.Global.CONFIG	Indicates the number of processor cores, number of interrupts, number of IOCUs, etc.
0x08140	CPC.Global.SYS_CONFIG	System level configuration

5.14.4.1 CPC Global Sequencer (SEQDEL_REG) Register (offset = 0x8008)

This register describe the time between microsteps of a CPC domain sequencer in CPC clock cycles.

The CPC_SEQDEL_REG describes globally the number of clock cycles each domain micro-sequencer will take to advance. It describes a set of worst-case timing of the physical implementation and is used to ensure electrical and bus protocol integrity. Mainly, buffer tree delays on *_isolate and/or*_rail_enable can be used to set proper micro sequencer delay values. Typically, the CPC_SEQDEL_REG contents would be defined at IP configuration time. However, runtime write capability allows fine tuning to optimize sequencer timing.

Figure 5.66 Global Sequencer Register Bit Assignments

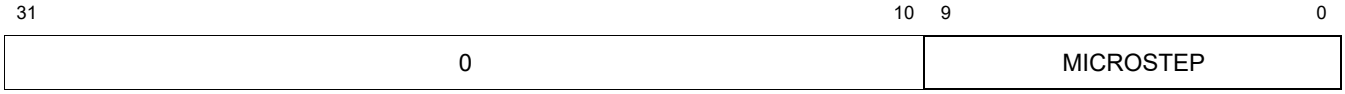


Table 67: Global Sequencer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:10	Reserved	R	0
MICROSTEP	9:0	This field reflects the delay in clock cycles, taken by each power domain micro-sequencer to advance between atomic micro steps. Cycles/Step = MICROSTEP[9:0] value + 1; 0 = > 1 cycle, 1 = > 2 cycles Physical implementation might not allow power sequence micro steps to advance with full cluster speed. At cluster cold start, the counter divides cluster frequency by a hard-coded IP configuration value to derive a micro step width.	R/W	From configuration

5.14.4.2 CPC Global Rail (RAIL_REG) Register (offset = 0x8010)

This register rail power-up timer to delay CPS sequencer progress until the gated rail has stabilized.

The CPC_RAIL_REG represents a 10-bit counter register to schedule delayed start of domain operation after the RailEnable signal has been activated by the CPC. This allows to compensate for slew rates at the gated rail, since hardware interlocks such as ci_core_vdd_ok are either unavailable or do not reflect to complete power up time of a domain.At IP configuration time, the contents of CPC_RAIL_REG is preset. However, for fine tuning, the register can be written at run time.

Figure 5.67 Global Rail Register Bit Assignments

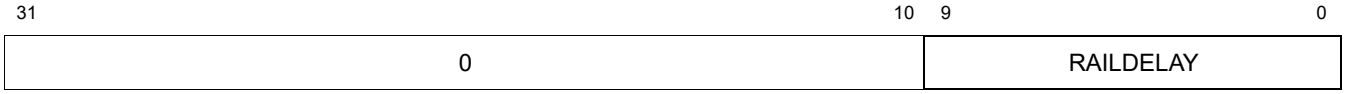


Table 68: Global Rail Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:10	Reserved	R	0
RAILDELAY	9:0	10-bit counter value to delay power-up sequence per domain after RailStable and VddOK signals became active. The power-up micro-sequence starts after RAILDELAY has been loaded into the internal counter and a counted down to zero has concluded. After completion of the domain power-up micro-sequence, the DomainReady signal is raised and can be used for domain daisy-chaining.	R/W	From configuration

5.14.4.3 CPC Global Reset Sequence (RESETLEN_REG) Register (offset = 0x8018)

This register duration of any domain reset sequence.

Within the power-up micro-sequence, reset is applied. This register defines the timing of the reset signal. NOTE: this register is reset on a cold-reset only.

Figure 5.68 Global Reset Sequence Register Bit Assignments

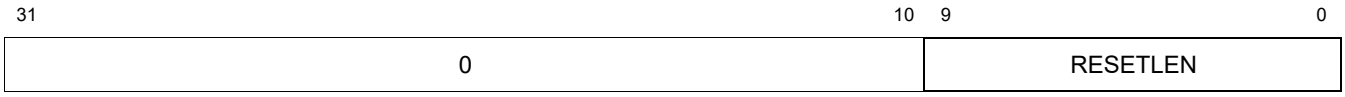


Table 69: Global Reset Sequence Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:10	Reserved	R	0
RESETLEN	9:0	Determines timing of the domain reset. The value of RESET_LEN is always greater than or equal to CM3_CPC_RESET_LEN_MIN. If IP Config value or the value written to the reg is less than min, default value of CM3_CPC_RESET_LEN_MIN is assumed.	R/W	From configuration

5.14.4.4 CPC Global Revision (REVISION_REG) Register (offset = 0x8020)

This register RTL revision of CPC.

Figure 5.69 Global Revision Register Bit Assignments

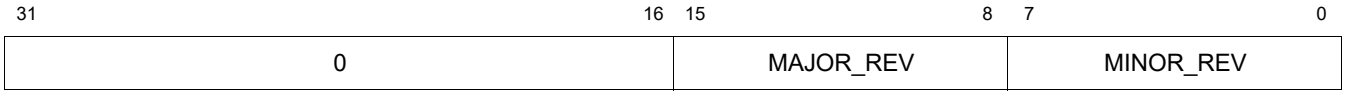


Table 70: Global Revision Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:16	Reserved	R	0
MAJOR_REV	15:8	This field reflects the major revision of the CPC block. A major revision might reflect the changes from one product generation to another.	R	From configuration
MINOR_REV	7:0	This field reflects the minor revision of the CPC block. A minor revision might reflect the changes from one release to another.	R	From configuration

5.14.4.5 CPC Global Clock Change Configuration, Control and Status. (CC_CTL_REG) Register (offset = 0x8028)

This register CPC global clock change configuration, control and status.

Enables clock change for all clock change enable domains of the cluster.

Figure 5.70 Global Clock Change Configuration, Control and Status Register Bit Assignments

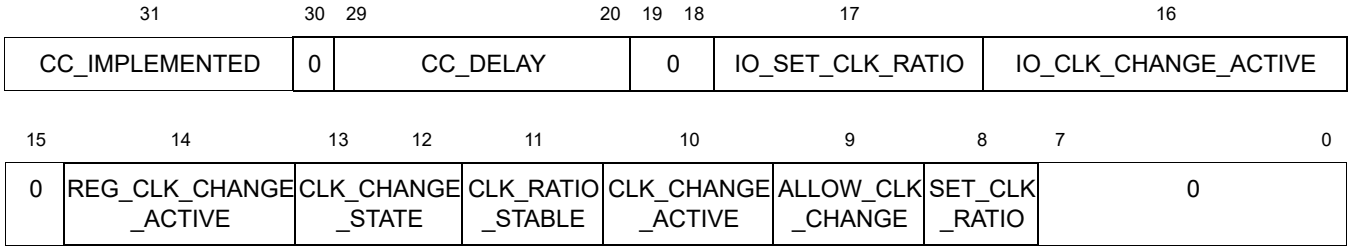


Table 71: Global Clock Change Configuration, Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CC_IMPLEMENTED	31	Clock change functionality implemented.	R	1
0	30	Reserved	R	0
CC_DELAY	29:20	Clock change delay value. This specifies the number clocks after the prescaler to wait in the C1 - CC_StartChange state to allow all CM3_CDC_FIFOs to prepare for a clock domain change.	r/w	From configuration
0	19:18	Reserved	R	0
IO_SET_CLK_RATIO	17	Indicates the current state of the io_set_clk_ratio input.	R	0
IO_CLK_CHANGE_ACTIVE	16	Readback of io_clk_change_active signal indicating that an IO based clock change is still in progress.	R	0
0	15	Reserved	R	0
REG_CLK_CHANGE_ACTIVE	14	Readback of reg_clk_change_active signal indicating that a register based clock change is still in progress.	R	0
CLK_CHANGE_STATE	13:12	Current domain clock change sequence state 0 C0 - CC_Idle 1 C1 - CC_StartChange 2 C2 - CC_SetRatio 3 C3 - CC_WaitStable	R	0
CLK_RATIO_STABLE	11	Indicates the AND of the clk_ratio_stable signal for all CPC local state machines which are participating in the clock change.	R	0
CLK_CHANGE_ACTIVE	10	Readback of clk_change_active signal indicating the global clock change is still in progress.	R	0
ALLOW_CLK_CHANGE	9	Indicates the AND of the cpc_allow_clk_change signal for all CPC local state machines which are participating in the clock change.	R	0

Table 71: Global Clock Change Configuration, Control and Status Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
SET_CLK_RATIO	8	Writing a 1 to this field initiates a register based clock change for all participating domains (domains which have SET_CLK_RATIO_EN set to 1 in their CPC.Global.CC_CTL_REG) and prescaler. Writing a 0 is ignored. A readback value of 1 for this field indicates the register based clock change is still pending or in progress. A readback value of 0 for this field indicates that a register based clock change has been taken. The register clock change is completed when both this field and CLK_CHANGE_ACTIVE are both low.	RW1S	0
0	7:0	Reserved	R	0

5.14.4.6 CPC Global Power Up Control (PWRUP_CTL_REG) Register (offset = 0x8030)

This register controls power of CM even independent of the cores power states.

Normally, the CM is automatically powered-up if any core is powered-up and the CM is automatically powered-down if no core is powered-up. This register powers-up the CM even if no core is powered-up. This may be useful for system debug/setup via the DBU.

Figure 5.71 Global Power Up Control Register Bit Assignments

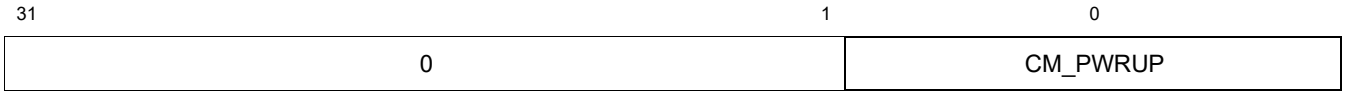


Table 72: Global Power Up Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
CM_PWRUP	0	If CM_PWRUP is 1, then the CM will be powered-on if not already powered-on. If the CM is already powered-on then it will stay that way even if all the cores are powered-down.	R/W	0

5.14.4.7 CPC Global Reset Release (RES_REL_REG) Register (offset = 0x8038)

This register control reset release and clock enable timing.

Figure 5.72 Global Reset Release Register Bit Assignments

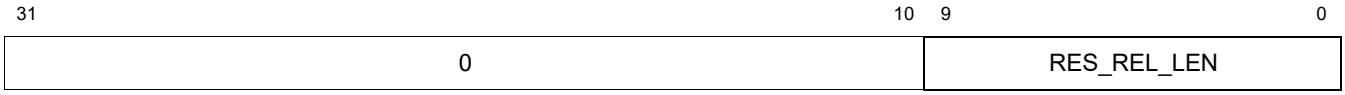


Table 73: Global Reset Release Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:10	Reserved	R	0
RES_REL_LEN	9:0	Used to control 3 timing parameters in the CPC state machines: 1) U4ER: Number of CPC clocks that the core reset signal is deasserted to CM before the reset is deasserted to the corresponding core. 2) U4R: Number of CPC clocks between releasing reset to a core before enabling non-Coherent Mode 3) D2R: The Number of CPC clocks between enabling clocks and enabling non-Coherent Mode.	R/W	From configuration

5.14.4.8 CPC Global Core Rest Control (ROCC_CTL_REG) Register (offset = 0x8040)

This register is to indicate which cores have been reset.

NOTE: This register is reset on a cold reset only.

Figure 5.73 Global Core Rest Control Register Bit Assignments

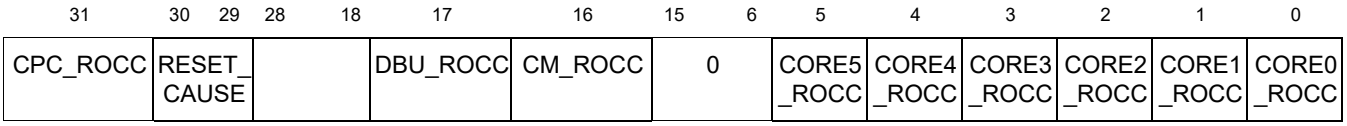


Table 74: Global Core Rest Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CPC_ROCC	31	Indicates CPC has been reset. This bit can be cleared by writing 1 to it.	RW1C	1
RESET_CAUSE	30:29	Indicates source for latest CPC reset. 00 reserved 01 cold reset 10 external warm reset 11 Watchdog warm reset	R	0
0	28:18	Reserved	R	0
DBU_ROCC	17	Indicates DBU has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CM_ROCC	16	Indicates CM has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
0	15:6	Reserved	R	0
CORE5_ROCC	5	Indicates Core 5 (if implemented) has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CORE4_ROCC	4	Indicates Core 4 (if implemented) has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CORE3_ROCC	3	Indicates Core 3 (if implemented) has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CORE2_ROCC	2	Indicates Core 2 (if implemented) has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CORE1_ROCC	1	Indicates Core 1 (if implemented) has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0
CORE0_ROCC	0	Indicates Core 0 has been reset. This bit can be cleared by writing 1 to it. Set to 0 on cold_reset.	RW1C	0

5.14.4.9 CPC Global Controls Prescale Clock Changes Register (offset = 0x8048)

This register controls prescale clock changes.

Figure 5.74 Global Controls Prescale Clock Changes Register Bit Assignments

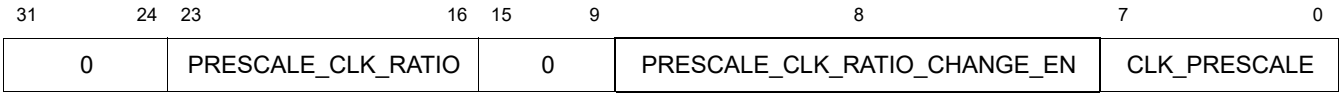


Table 75: Global Controls Prescale Clock Changes Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:24	Reserved	R	0
PRESCALE_CLK_RATIO	23:16	Indicates the current prescaler clock ratio.	R	From configuration
0	15:9	Reserved	R	0
PRESCALE_CLK_RATIO_CHANGE_EN	8	Setting this to 1 enables prescaler register based clock change. Setting this to 0 disables prescaler register based clock change. When the clock change completes this field is cleared.	R/W	0
CLK_PRESCALE	7:0	This field sets a new prescaler value for all clock domains. The value of prescaling is (CLK_PRESCALE + 1). A value of 0 indicates no prescaling (divide input clock by 1). A value of 1 indicates prescaling by 2 (divide input clock by 2), etc. Supports prescaling the input clk by 1 through 256.	R/W	From configuration

5.14.4.10 CPC Global RISC-V Mtime (MTIME_REG) Register (offset = 0x8050)

This register RISC-V mtime register.

Figure 5.75 Global RISC-V Mtime Register Bit Assignments

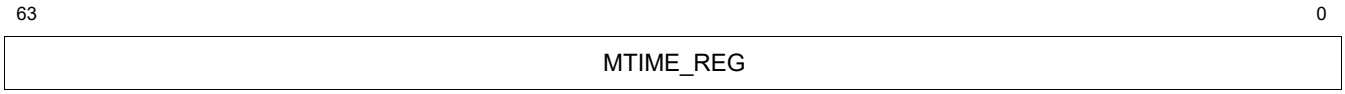


Table 76: Global RISC-V Mtime Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MTIME_REG	63:0	RISC-V mtime register.	R	0

5.14.4.11 CPC Global RISC-V Mtime Control (TIMECTL_REG) Register (offset = 0x8058)

This register Control register for RISC-V mtime and hrtime registers. Support software-assisted multi-cluster timer synchronization.

Figure 5.76 Global RISC-V Mtime Control Register Bit Assignments

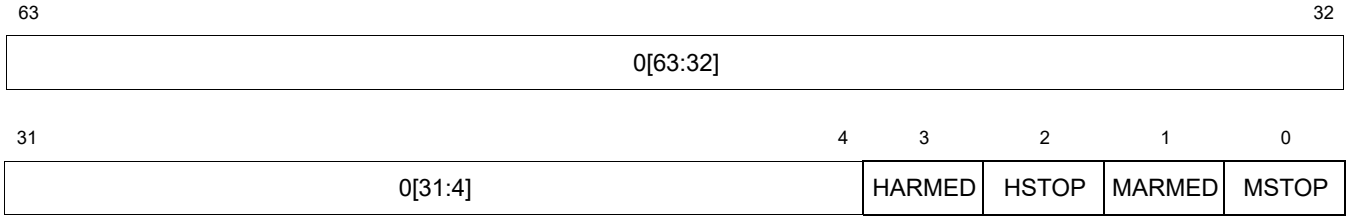


Table 77: Global RISC-V Mtime Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:4	Reserved	R	0
HARMED	3	Arm the HRTIME counter for hardware synchronization. When the HARMED and HSTOP bits are both asserted, the counter can be restarted by assertion of a cluster-level input pin (which will also clear the HARMED and HSTOP bits).	R/W	0
HSTOP	2	Stop the HRTIME counter.	R/W	0
MARMED	1	Arm the MTIME counter for hardware synchronization. When the MARMED and MSTOP bits are both asserted, the counter can be restarted by assertion of a cluster-level input pin (which will also clear the MARMED and MSTOP bits).	R/W	0
MSTOP	0	Stop the MTIME counter.	R/W	0

5.14.4.12 CPC Global Clock Gate Disabled (CLK_GATE_DIS_REG) Register (offset = 0x8060)

This register allows for module level clock gaters to be disabled.

Figure 5.77 Global Clock Gate Disabled Register Bit Assignments

Table 78: Global Clock Gate Disabled Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLK_GATE_DIS	31:0	Each bit disables one or more clock gaters. Refer to the CM3 micro-arch spec for the mapping of these bits to clock gaters.	R/W	0

5.14.4.13 CPC Global Fault Status (FAULT_STATUS) Register (offset = 0x8068)

This register allow the Global fault status contains the OR of all the fault domain status registers. This register is valid for FUSA CPUs only.

Figure 5.78 Global Fault Status Register Bit Assignments

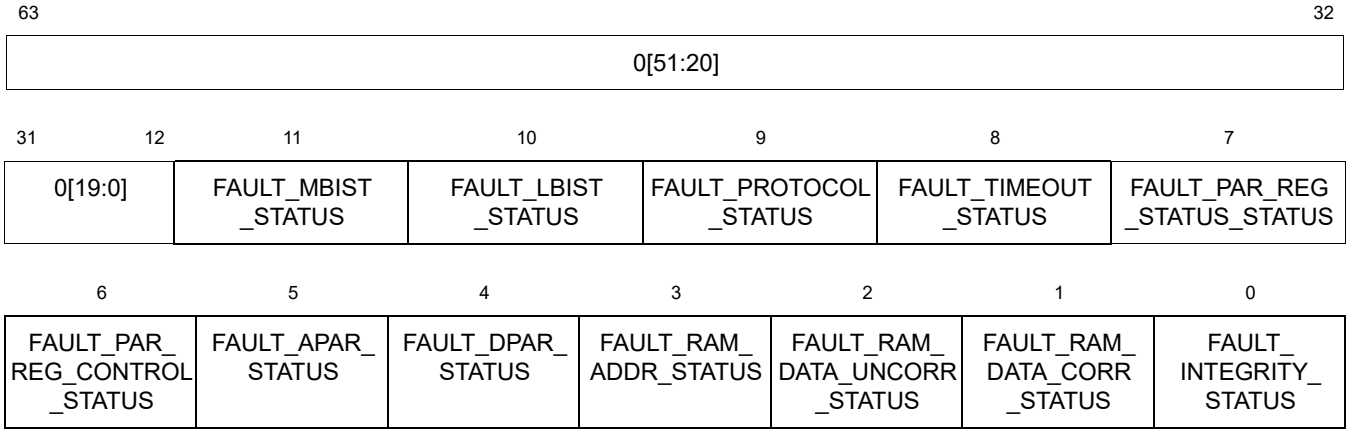


Table 79: Global Fault Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_ STATUS	11	An MBIST error was detected.	R	0
FAULT_LBIST_ STATUS	10	An LBIST error was detected.	R	0
FAULT_PROTOCOL_ STATUS	9	An interface protocol fault was detected.	R	0
FAULT_TIMEOUT_ STATUS	8	A transaction timeout error was detected.	R	0
FAULT_PAR_REG_ STATUS_STATUS	7	A parity error in a status register detected.	R	0
FAULT_PAR_REG_ CONTROL_STATUS	6	A parity error in a control register was detected.	R	0
FAULT_APAR_ STATUS	5	An address path parity error was detected.	R	0
FAULT_DPAR_ STATUS	4	A data path parity error was detected.	R	0
FAULT_RAM_ADDR_ STATUS	3	An SRAM address fault was detected.	R	0
FAULT_RAM_DATA_U NCORR_STATUS	2	An uncorrectable data fault in SRAM was detected.	R	0
FAULT_RAM_DATA_C ORR_STATUS	1	A correctable data fault in SRAM detected.	R	0
FAULT_INTEGRITY_ STATUS	0	An integrity check fault was detected.	R	0

5.14.4.14 CPC Global Fault Supported (FAULT_SUPPORTED) Register (offset = 0x8070)

This register indicates whether the design is configured to detect each type of fault. This register is only valid when FUSA features are implemented.

Figure 5.79 Global Fault Supported Register Bit Assignments

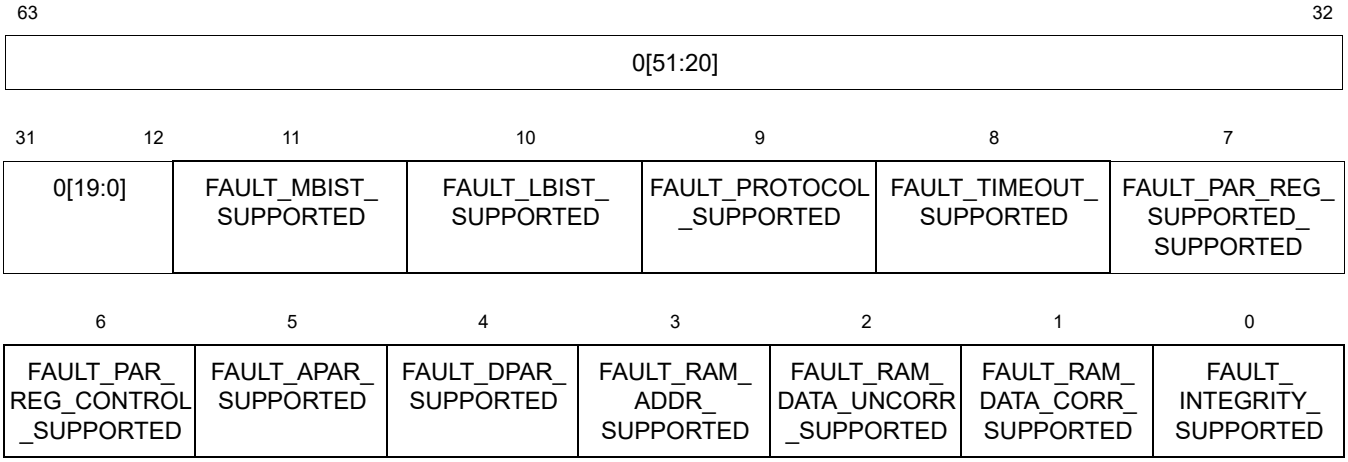


Table 80: Global Fault Supported Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_ SUPPORTED	11	MBIST error detection supported.	R	From configuration
FAULT_LBIST_ SUPPORTED	10	LBIST error detection supported.	R	From configuration
FAULT_PROTOCOL_ SUPPORTED	9	Interface protocol fault detection supported.	R	From configuration
FAULT_TIMEOUT_ SUPPORTED	8	Transaction timeout error detection supported.	R	From configuration
FAULT_PAR_REG_ SUPPORTED_ SUPPORTED	7	Parity error in a status register detection supported.	R	From configuration
FAULT_PAR_REG_ CONTROL_ SUPPORTED	6	Parity error in a control register detection supported.	R	From configuration
FAULT_APAR_ SUPPORTED	5	Address path parity error detection supported.	R	From configuration
FAULT_DPAR_ SUPPORTED	4	A data path parity error was detected.	R	From configuration
FAULT_RAM_ADDR_ SUPPORTED	3	SRAM address fault detected supported.	R	From configuration
FAULT_RAM_DATA_U NCORR_ SUPPORTED	2	Uncorrectable data fault in SRAM detection is supported.	R	From configuration
FAULT_RAM_DATA_C ORR_SUPPORTED	1	Correctable data fault in SRAM detection is supported.	R	From configuration

Table 80: Global Fault Supported Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
FAULT_INTEGRITY_SUPPORTED	0	Integrity check fault detected is supported.	R	From configuration

5.14.4.15 CPC Global Fault Enable (FAULT_ENABLE) Register (offset = 0x0078)

This register indicates whether detection each type of fault is enabled. This register is only valid when FUSA features are implemented.

Figure 5.80 Global Fault Enable Register Bit Assignments

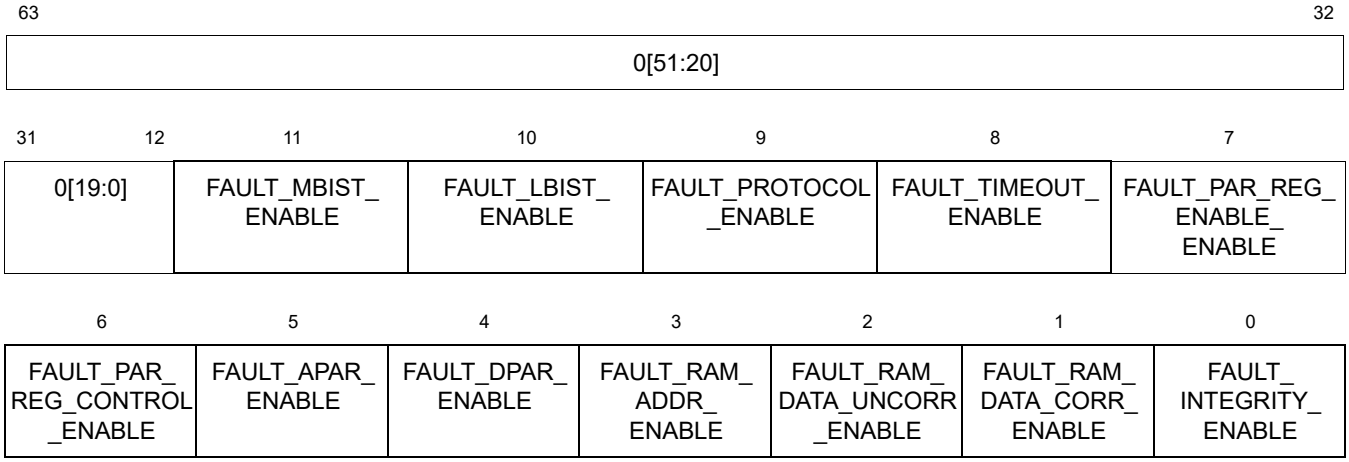


Table 81: Global Fault Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_ ENABLE	11	MBIST error detection enabled.	R	From configuration
FAULT_LBIST_ ENABLE	10	LBIST error detection enabled.	R	From configuration
FAULT_PROTOCOL_ ENABLE	9	Interface protocol fault detection enabled.	R	From configuration
FAULT_TIMEOUT_ ENABLE	8	Transaction timeout error detection enabled.	R	From configuration
FAULT_PAR_REG_ ENABLE_ENABLE	7	Parity error in a status register detection enabled.	R	From configuration
FAULT_PAR_REG_ CONTROL_ENABLE	6	Parity error in a control register detection enabled.	R	From configuration
FAULT_APAR_ ENABLE	5	Address path parity error detection enabled.	R	From configuration
FAULT_DPAR_ ENABLE	4	A data path parity error was enabled.	R	From configuration
FAULT_RAM_ADDR_ ENABLE	3	SRAM address fault detected enabled.	R	From configuration
FAULT_RAM_DATA_U NCORR_ ENABLE	2	Uncorrectable data fault in SRAM detection is enabled.	R	From configuration
FAULT_RAM_DATA_C ORR_ENABLE	1	Correctable data fault in SRAM detection is enabled.	R	From configuration

Table 81: Global Fault Enable Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
FAULT_INTEGRITY_ENABLE	0	Integrity check fault detected is enabled.	R	From configuration

5.14.4.16 CPC Global High Resolution Timer (HRTIME_REG) Register (offset = 0x8090)

This register describe the high resolution timer register.

Figure 5.81 Global High Resolution Timer Register Bit Assignments

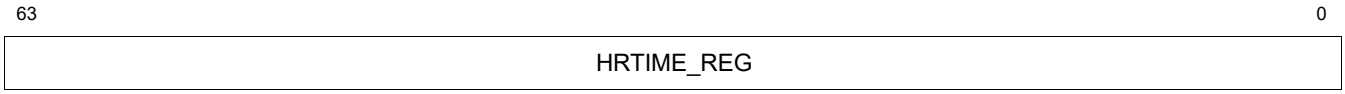


Table 82: Global High Resolution Timer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
HRTIME_REG	63:0	High resolution timer register.	R	0

5.14.4.17 CPC Global Configuration (CONFIG) Register (offset = 0x8138)

This register indicates the number of processor cores, number of interrupts, number of IOCUs, etc.

This register provides information on the overall system configuration. These fields are read-only and their reset state is determined at IP configuration time.

Figure 5.82 Global Configuration Register Bit Assignments

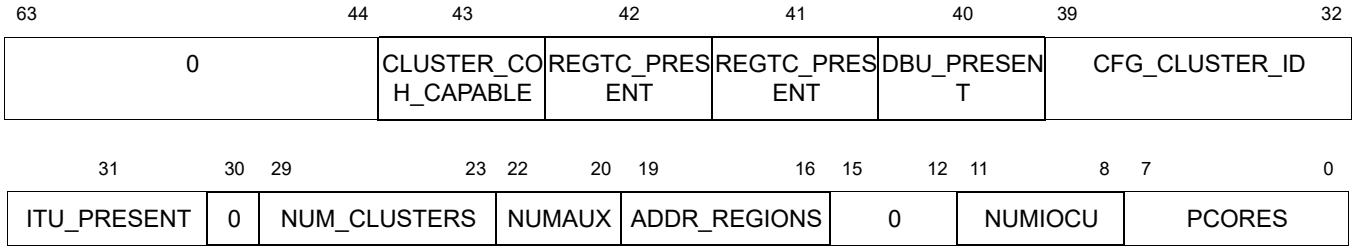


Table 83: Global Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:44	Reserved	R	0
CLUSTER_COH_CAPABLE	43	Set to 1 if this cluster supports ACE coherent interconnect.	R	From configuration
REGTC_PRESENT	42	Set to 1 if REGTC is present in this cluster.	R	From configuration
REGTC_PRESENT	41	Set to 1 if REGTN is present in this cluster.	R	From configuration
DBU_PRESENT	40	Set to 1 if DBU is present in the design.	R	From configuration
CFG_CLUSTER_ID	39:32	Indicates the cluster_id of current cluster.	R	From configuration
ITU_PRESENT	31	Set to 1 if ITU is present in the design.	R	From configuration
0	30	Reserved	R	0
NUM_CLUSTERS	29:23	Indicates total number of clusters present in the design.	R	From configuration
NUMAUX	22:20	Number of auxiliary memory ports in this cluster.	R	From configuration
ADDR_REGIONS	19:16	Number of MMIO address region registers. This value is determined by the IP configuration.	R	From configuration
0	15:12	Reserved	R	0
NUMIOCU	11:8	Total number of IOCUs in this cluster.	R	From configuration
PCORES	7:0	Total number of CPU Cores - 1 in this cluster, not including the IOCU's.	R	From configuration

5.14.5 CPC.Core Registers

The CPC.Core region contains the following per-core and per-device memory mapped registers, which are described in detail in the subsequent per-register descriptions:

Table 84: CPC Core Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x08200 0x08208 ..... 0x083F8	CPC.IOCU[0..63].CC_CTL_REG	Controls clock changes on corresponding IOCU.
0x08400	CPC.MEM.CC_CTL_REG	Controls clock changes on AXI/ACE memory port.
0x08440 0x08448 0x08450 0x08458	CPC.AUX[0..3].CC_CTL_REG	Controls clock changes on CM AXI AUX[0..3] port.
0x09008	CPC.CM.STAT_CONF_REG	CM domain power status and domain configuration register.
0x09018	CPC.CM.CC_CTL_REG	Controls clock changes on corresponding device.
0x09050	CPC.CM.RAM_SLEEP_REG	Controls Deep Sleep and Shut Down power state of RAMs for CM power domain.
0x09068	CPC.CM.FAULT_STATUS	CM Domain fault status, for FUSA CPUs only.
0x09070	CPC.CM.FAULT_SET	Any bit written to 1 will set the corresponding fault bit.
0x09078	CPC.CM.FAULT_CLR	Any bit written to 1 will clear the corresponding fault bit.
0x09090	CPC.CM.CONFIG	Configuration parameters for the CM domain.
0x09100	CPC.DBU.CMD_REG	Places a new CPC domain state command into this individual domain sequencer for the DBU power domain.
0x09108	CPC.DBU.STAT_CONF_REG	Debug domain power status and domain configuration register.
0x09118	CPC.DBU.CC_CTL_REG	Controls clock changes on corresponding device.
0x09150	CPC.DBU.RAM_SLEEP_REG	Controls Deep Sleep and Shut Down power state of RAMs for DBU power domain.
0x09190	CPC.DBU.CONFIG	Configuration parameters for the DBU domain.
0x0A000 0x0A100 ..... 0x0DF00	CPC.Core[0-63].CMD_REG	Places a new CPC domain state command into this individual domain sequencer for the core power domain.

Table 84: CPC Core Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x0A008 0x0A108 ..... 0x0DF08	CPC.Core[0-63].STAT_CONF_REG	Core domain power status and domain configuration register.
0x0A018 0x0A118 ..... 0x0DF18	CPC.Core[0-63].CC_CTL_REG	Controls clock changes on corresponding device.
0x0A020 0x0A120 ..... 0x0DF20	CPC.Core[0-63].VP_STOP_REG	Stops execution of harts.
0x0A028 0x0A128 ..... 0x0DF28	CPC.Core[0-63].VP_RUN_REG	Starts execution of harts.
0x0A030 0x0A130 ..... 0x0DF30	CPC.Core[0-63].VP_RUNNING_REG	Indicates which harts are in the Running State.
0x0A040 0x0A140 ..... 0x0DF40	CPC.Core[0-63].DBG_DBRK_REG	Used to send a DINT (debug interrupt) to a specific hart.
0x0A050 0x0A150 ..... 0x0DF50	CPC.Core[0-63].RAM_SLEEP_REG	Controls Deep Sleep and Shut Down power state of RAMs for core power domain.
0x0A068 0x0A168 ..... 0x0DF68	CPC.Core[0-63].FAULT_STATUS	Core domain fault status, for FUSA CPUs only.
0x0A070 0x0A170 ..... 0x0DF70	CPC.Core[0-63].FAULT_SET	Any bit written to 1 will set the corresponding fault bit.
0x0A078 0x0A178 ..... 0x0DF78	CPC.Core[0-63].FAULT_CLR	Any bit written to 1 will clear the corresponding fault bit.
0x0A090 0x0A190 ..... 0x0DF90	CPC.Core[0-63].CONFIG	Configuration parameters for the Core domain.

5.14.5.1 CPC Power Command (CMD_REG) Register (offset = see below)

This register places a new CPC domain state command into this individual domain sequencer.

This register is not available within the CM sequencer. Writes to the CM CMD register are ignored while reads will return zero. This register is instantiated for each power domain.

Offset: 0x9100 from GCR_BASE for DBU power domain.

0xa000 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.83 Power Command Register Bit Assignments

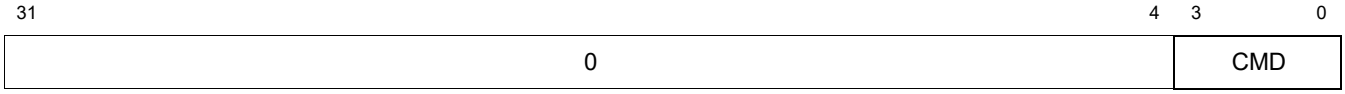


Table 85: Power Command Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:4	Reserved	R	0
CMD	3:0	Requests a new power sequence execution for this domain. Read value is the last executed command. Not available in the CM domain. 1) ClockOff This command causes the domain to cycle into clock-off mode.It disables the clock to this power domain. Only successful if core_coherence_enable and other protocol interlocks are observed. If not, the command remains inactive until the protocol barriers subside. After that, the command is executed. Depending on the current sequencer state, the command either causes power-up of a domain, or a domain leaves active duty to become inactive. A power-up leads to sequencer state U2, which will require the execution of a subsequent Reset or PwrUp command to make this domain operational. 2) PwrDown this domain using setup values in CPC_STAT_CONF_REG. Only successful if core_coherence_enable inactive and all protocol interlocks are observed. If not, the command remains inactive until the protocol barriers subside. Then, the command is executed. 3) PwrUp this domain using setup values in CPC_STAT_CONF_REG. Usable only for Core- Others access. It is the software equivalent to ci_*_pwr_up hardware signal. 4) Reset This domain is reset if in non-coherent mode. After the domain has been reset, the domain becomes operational and the CMD field reads as PwrUp cmd. Other values Reserved - Writes with reserved values will not be loaded into this register.	R/W	Derived from value driven on si_core_cold_pwr_up / si_dbu_cold_pwr_up when CPC is reset

5.14.5.2 CPC Core Status and Domain Configuration (STAT_CONF_REG) Register (offset = see below)

This register individual domain power status and domain configuration.

Reflects domain micro-sequencer execution. Initiates microsequencer after status register programming. Reflects command execution status. This register is instantiated for each Pwr domain.

Offset: 0x9008 from GCR_BASE for CM power domain.

0x9108 from GCR_BASE for DBU power domain.

0xa008 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.84 Core Status and Domain Configuration Register Bit Assignments

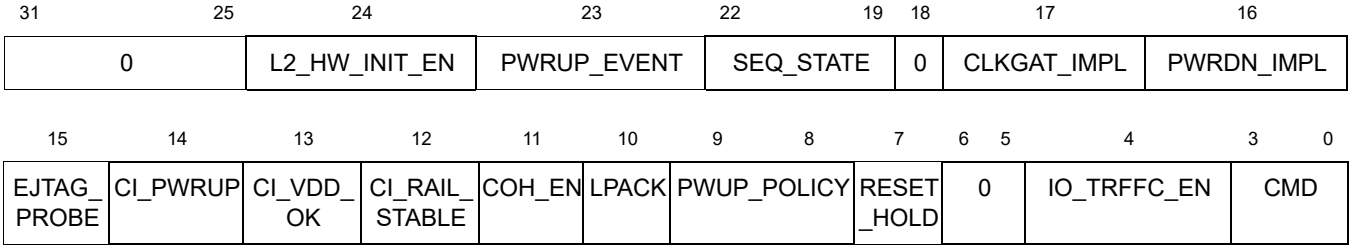


Table 86: Core Status and Domain Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:25	Reserved	R	0
L2_HW_INIT_EN	24	If this bit is set L2 state machine is allowed to start L2 hardware initialization after waiting for CM3_CPC_L2_HW_INIT_DELAY once reset is de-asserted, provided that BIST is not on and si_cpc_l2_hw_init_inhibit is tied low. The initialization request is generated only once after every reset if all the above conditions are met. This bit is only valid for CM power domain for rest of the domains this bit always reads out 0x0	R/W	From configuration
PWRUP_EVENT	23	The ci_*_pwr_up pin had been activated and caused the sequencer to cycle into power up state. The event also caused the sequencer to place a PwrUp command into the CMD field. Writing a 0 into the PWRUP_EVENT field will clear this bit.	RW0C	0

Table 86: Core Status and Domain Configuration Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
SEQ_STATE	22:19	Current domain sequencer state. State Description 0 D0 - PwrDwn 1 U0 - RailStable 2 U1 - UpDelay (VddOK/rail delay state) 3 U2 - UClkOff 4 U3 - Reset 5 U4 - ResetDly 6 U5 - nonCoherent execution 7 U6 - Coherent execution 8 D1 - Isolate 9 D3 - ClrBus 10 D2 - DclkOff 12 U4ER - early reset release delay state 13 U4R - Reset release delay state 15 D2R - DClkOff clk enable state	R	0
0	18	Reserved	R	0
CLKGAT_IMPL	17	If set, this domain is implemented with clock tree root gating. If cleared, the CPC will still execute power-down/ clock-off sequences if commanded; however, no physical clock gating is performed.	R	From configuration
PWRDN_IMPL	16	If set, this domain is implemented as power-gated. If cleared, the CPC will still execute power-down sequences if commanded; however, no physical power switching is performed.	R	From configuration
EJTAG_PROBE	15	A Debug probe connection event has been seen. The domain powers up if required and observes a reset sequence. Thereafter the core transitions into clock-off mode. After a probe has been seen once, the power domain will not assume power-off mode until this bit is written to zero or the CPC experiences a cold reset. NOTE: This bit is only used for the local CPC registers corresponding to the DBU. The CM and Core CPC local sections always set this set to 0.	RW0C	0
CI_PWRUP	14	Reads the synchronized value of ci_pwr_up for this domain.	R	Pin Value
CI_VDD_OK	13	Reads the synchronized value of ci_vdd_ok for this domain.	R	Pin Value
CI_RAIL_STABLE	12	Reads the synchronized value of ci_rail_stable for this domain.	R	Pin Value
COH_EN	11	Reads the synchronized value of Coherence Enable for this domain.	R	Pin Value
LPACK	10	Reads the synchronized value of LPACK for this domain.	R	Pin Value

Table 86: Core Status and Domain Configuration Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
PWUP_POLICY	9:8	Each CPC domain sequencer is hardwired through the ci_*_cold_pwr_up signal to either power up, remain power-gated, go into clock-off mode, or become operational. To influence the cold start behavior of the domain, three distinct policies can be wired for this domain: 0 - This CPU remains powered down after a system cold start. A later PwrUp or Reset command, or ci_*_pwr_up signal assertion will make this domain operational. 1 - Go into Clock-Off mode. Disables domain clock after power-up sequence. Core will wake up through a CPC PwrUp or Reset command or a ci_*_pwr_up signal assertion. In this Clock-Off mode, the core will not be initialized and its boundary isolation will be maintained. 2 - Power up this domain after system cold start. The CPU will be reset and become operational based on its boot vector contents. 3 - Reserved Within a processor cluster, CPU 0 would power-up, while peer CPU's 1 - 3 would remain unpowered until released through a PwrUp commands. The PWUP_POLICY field reflects the hardwired ci_*_cold_pwr_up bus.	R	Value driven on si_core_cold_pwr_up / si_dbu_cold_pwr_up when CPC is reset
RESET_HOLD	7	Reads the synchronized value of the RESET_HOLD signal for this domain.	R	Pin Value
0	6:5	Reserved	R	0
IO_TRFFC_EN	4	Enable CM for stand alone IOCU traffic. Setting this bit changes the low power state of the CM power domain from PwrDwn to ClkOff. The si_cm_pwr_up signal can be used by an external device to re-enable the CM to perform IOCU data transfers without CPU activities. If si_cm_pwr_up is deasserted, deselecting IO_TRFFC_EN will power down the CM if all CPUs are powered down. A powered down CM domain will clear all preset CM/IOCU control registers. Powering up from powered down due to CPU power-up or si_cm_pwr_up will send the CM/IOCU through a reset sequence, together with the CPU. This bit is only used on the CM registers and is read-only 0 for the CPUs and DBU.	R/W	0
CMD	3:0	Reflects most recent placed sequencer command. See definition in CPC_CMD_REG The sequencer will overwrite the field after a Reset command, or ci_*_pwr_up signal caused power up of the domain. The command reads then as PwrUp.	R	0

5.14.5.3 CPC Control Clock Change (CC_CTL_REG) Register (offset = see below)

This register controls clock changes on corresponding device. It is instantiated for each Clk domain.

Offset: 0 x08200 + 8 * IOCUNUM from GCR_BASE for IOCU[0..63] power domain.

0x08400 from GCR_BASE for memory port power domain.

0x4040 + 8 * AUXNUM from GCR_BASE for AUX[0..3] power domain.

0x9018 from GCR_BASE for CM power domain.

0x9118 from GCR_BASE for DBU power domain.

0xa018 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.85 Control Clock Change Register Bit Assignments

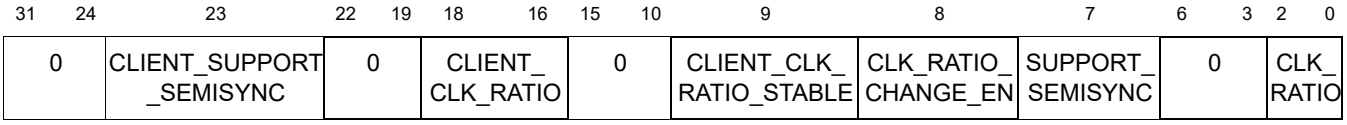


Table 87: Control Clock Change Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:24	Reserved	R	0
CLIENT_SUPPORT_SEMISYNC	23	Indicates the current domain support semisync value. This bit is unused for the CM local register.	R	From configuration
0	22:19	Reserved	R	0
CLIENT_CLK_RATIO	18:16	Indicates the current domain clock ratio. This value is unpredictable when a clock ratio is in process, i.e. CLIENT_CLK_RATIO_STABLE is 0.	R	From configuration
0	15:10	Reserved	R	0
CLIENT_CLK_RATIO_STABLE	9	Readback of client_clk_ratio_stable signal for this domain. Indicates that the clock ratio for this domain is stable, i.e., there is not a clock ratio change in process.	R	0
CLK_RATIO_CHANGE_EN	8	Setting this to 1 enables this domain for register based clock change. Setting this to 0 disables this domain for register based clock change. When the clock change completes this field is cleared.	R/W	0
SUPPORT_SEMISYNC	7	Specifies that the domain supports semisync operation. This requires the clock trees to be latency matched between the cm clock domain and client domain. if CLIENT_SUPPORT_SEMISYNC is 0, indicating that the design does not support semi-synchronous clocks, then SUPPORT_SEMISYNC is read-only with a value of 0. This bit is unused for the CM local register.	R/W	From configuration
0	6:3	Reserved	R	0

Table 87: Control Clock Change Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
CLK_RATIO	2:0	Specifies the requested clock ratio (with respect to the pre-scaled clock) for this domain when using register-based clock changes. For all domains except the CM, clock ratios of 1:1 through 1:8 are supported. For the CM clock domain only ratios of 1:1 and 1:2 are supported. This field is encoded as follows: 000: CM clock ratio = 1:1, other clock ratio = 1:1 001: CM clock ratio = 2:1, other clock ratio = 2:1 010: CM clock ratio = reserved, other clock ratio = 3:1 011: CM clock ratio = reserved, other clock ratio = 4:1 100: CM clock ratio = reserved, other clock ratio = 5:1 101: CM clock ratio = reserved, other clock ratio = 6:1 110: CM clock ratio = reserved, other clock ratio = 7:1 111: CM clock ratio = reserved, other clock ratio = 8:1	R/W	From configuration

5.14.5.4 CPC Power VP Stop (VP_STOP_REG) Register (offset = see below)

This register stops execution of harts. It is instantiated for each core power domain.

Offset: 0xa020 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.86 VP Stop Register Bit Assignments

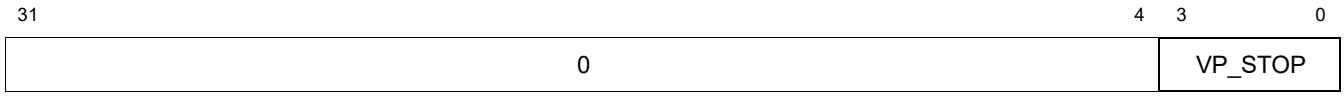


Table 88: VP Stop Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:4	Reserved	R	0
VP_STOP	3:0	If bit x is written 1 and bit x of CPC.Pwr.VP_RUNNING_REG is 1, then hart x on the local core will stop execution.	W1C	Undefined

5.14.5.5 CPC VP Run (VP_RUN_REG) Register (offset = see below)

This register start execution of harts. It is instantiated for each core power domain.

Offset: 0xa028 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.87 VP Run Register Bit Assignments

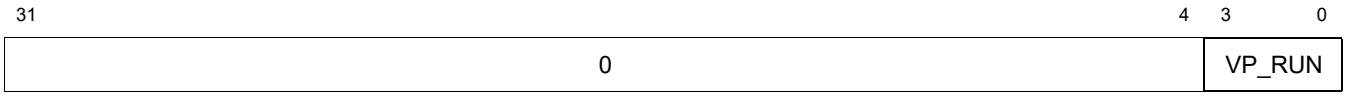


Table 89: VP Run Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:4	Reserved	R	0
VP_RUN	3:0	If bit x is written to 1 and bit x of CPC.Pwr.VP_RUNNING_REG is 0, then hart x on the local core will start execution from the reset vector, which is specified in the GCR.HART.RESET_BASE Register. Note that the Run state of each hart after a core reset is determined by the corresponding si_core_vc_run_init_ pin (if the pin is 1 then the corresponding hart will be start execution from the reset vector, as defined above, once the reset sequence is complete).	W1S	Undefined

5.14.5.6 CPC VP Running State (VP_RUNNING_REG) Register (offset = see below)

This register indicates which harts are in the running state. It is instantiated for each core power domain.

Offset: 0xa030 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.88 Core VP Running State Register Bit Assignments

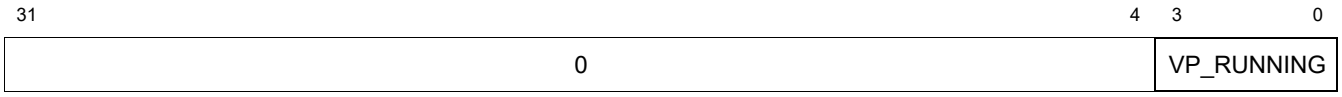


Table 90: Core VP Running State Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:4	Reserved	R	0
VP_RUNNING	3:0	If bit x is 1, then hart x on the local core is in the Running state. If bit x is a 0, then hart x on the local core is not in the Running state. On reset the running state is determined by the value driven on the si_core_vc_run_init_. The running state of a hart can be changed by writing a 1 to the corresponding bit in the CPC.Pwr.VP_RUN_REG or CPC.Pwr.VP_STOP_REG, or by asserting the si_vc_run_init_load_en signal, which sets the running state for each hart based on the value driven on si_core_vc_run_init.	R	From configuration

5.14.5.7 CPC Power Debug Interrupt (DBG_DBRK_REG) Register (offset = see below)

This register used to send a DINT (debug interrupt) to a specific hart, and to show which harts are in debug mode. It is instantiated for each core power domain.

Offset: 0xa040 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.89 Power Debug Interrupt Register Bit Assignments

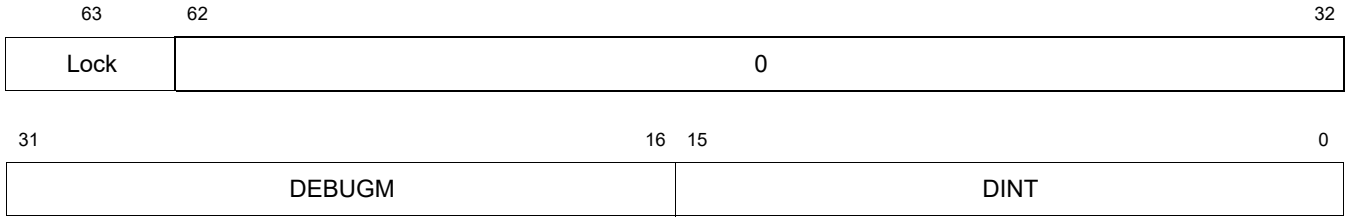


Table 91: Power Debug Interrupt Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
Lock	63	Writing 1 makes this register reserved until the CPC is reset.	RW1	0
0	62:32	Reserved	R	0
DEBUGM	31:16	Bit i of DEBUGM is 1 if the corresponding hart in this core is in debug mode, 0 otherwise.	R	0
DINT	15:0	Writing any bit to 1 sends a DINT signal to the corresponding hart in this core.	R/W	0

5.14.5.8 CPC Power Controls Deep Sleep (RAM_SLEEP_REG) Register (offset = see below)

This register controls deep sleep and shut down power state of RAMs. It is instantiated for each power domain.

Offset: 0x9050 from GCR_BASE for CM power domain.

0x9150 from GCR_BASE for DBU power domain.

0xa050 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.90 Power Controls Deep Sleep Register Bit Assignments

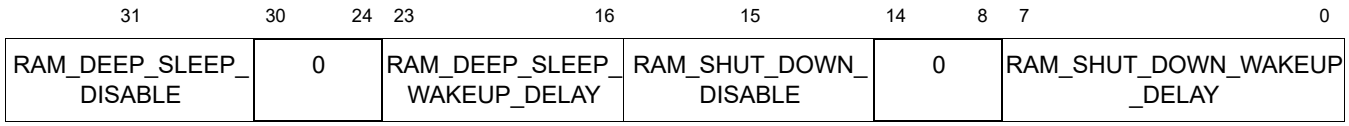


Table 92: Power Controls Deep Sleep Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
RAM_DEEP_SLEEP_DISABLE	31	When this bit is 0, RAMs on the local device will be signaled to enter a Deep Sleep low power state when the CPC power state for the device is DClkOff (D2). When this bit is 1, the CPC will not signal the RAMs on the local device to enter the Deep Sleep low power state.	R/W	From configuration
0	30:24	Reserved	R	0
RAM_DEEP_SLEEP_WAKEUP_DELAY	23:16	This value is used when the CPC is waking up a RAM that has been previously placed in the Deep Sleep state. The CPC ensures that the RAM is signaled to come out of Deep Sleep mode for the number of CPC clocks specified this register prior transitioning the local device into a running state.	R/W	From configuration
RAM_SHUT_DOWN_DISABLE	15	When this bit is 0, RAMs on the local device will be signaled to enter a Shut Down low power state when the CPC power state for the device is PwrDwn (D0), UClkOff (U2), or U0,U1,D1 transitional states. When this bit is 1, the CPC will not signal the RAMs on the local device to enter the Shut Down low power state.	R/W	From configuration
0	14:8	Reserved	R	0
RAM_SHUT_DOWN_WAKEUP_DELAY	7:0	This value is used when the CPC is waking up a RAM that has been previously placed in the Shut Down state. The CPC ensures that the RAM is signaled to come out of Shut Down mode for the number of CPC clocks specified this register prior transitioning the local device into a running state.	R/W	From configuration

5.14.5.9 CPC Power Fault Status (FAULT_STATUS) Register (offset = see below)

This register domain fault status, for FUSA CPUs only. It is instantiated for the CM and Core power domains, for FUSA CPUs only.

Offset: 0x9068 from GCR_BASE for CM power domain.

0xa068 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.91 Power Fault Status Register Bit Assignments

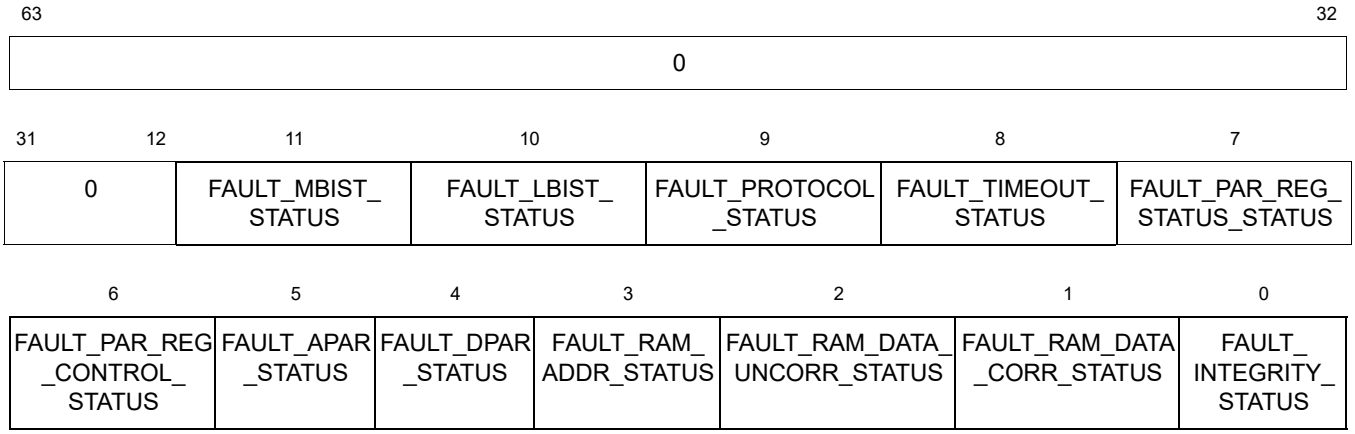


Table 93: Power Fault Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_ STATUS	11	An MBIST error was detected.	R	0
FAULT_LBIST_ STATUS	10	An LBIST error was detected.	R	0
FAULT_PROTOCOL_ STATUS	9	An interface protocol fault was detected.	R	0
FAULT_TIMEOUT_ STATUS	8	A transaction time-out error was detected.	R	0
FAULT_PAR_REG_ STATUS_STATUS	7	A parity error in a status register detected.	R	0
FAULT_PAR_REG_ CONTROL_STATUS	6	A parity error in a control register was detected.	R	0
FAULT_APAR_ STATUS	5	An address path parity error was detected.	R	0
FAULT_DPAR_ STATUS	4	A data path parity error was detected.	R	0
FAULT_RAM_ADDR_ STATUS	3	An SRAM address fault was detected.	R	0
FAULT_RAM_DATA_U NCORR_STATUS	2	An un-correctable data fault in SRAM was detected.	R	0
FAULT_RAM_DATA_C ORR_STATUS	1	A correctable data fault in SRAM detected.	R	0

Table 93: Power Fault Status Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
FAULT_INTEGRITY_STATUS	0	An integrity check fault was detected.	R	0

5.14.5.10 CPC Power Fault Set (FAULT_SET) Register (offset = see below)

This register any bit written to 1 will set the corresponding fault bit. Bits written to 0 have no effect. Read back is zero. It is instantiated for the CM and Core power domains, for FUSA CPUs only.

Offset: 0x9070 from GCR_BASE for CM power domain.

0xa070 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.92 Power Fault Set Register Bit Assignments

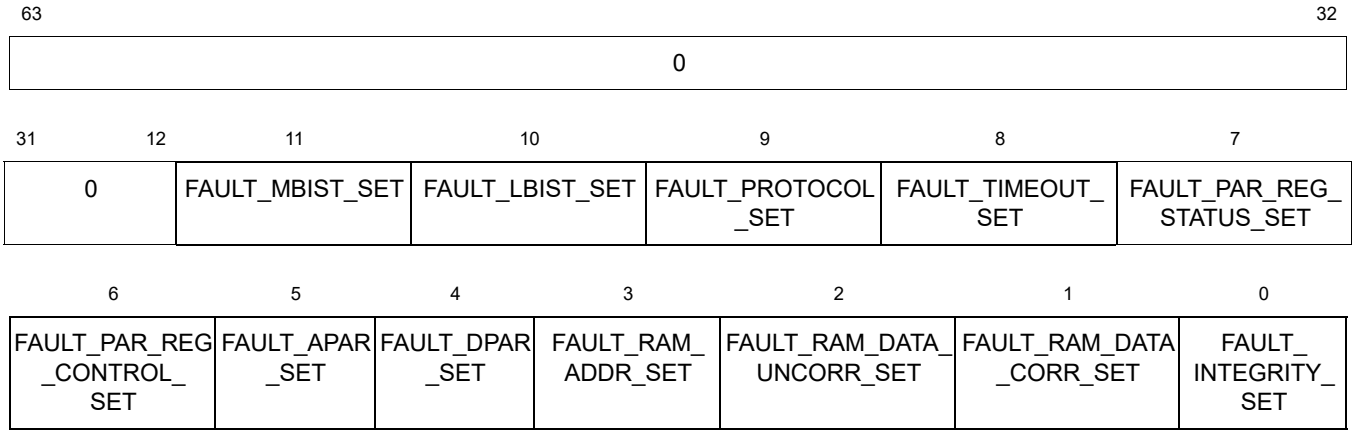


Table 94: Power Fault Set Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_SET	11	Write 1 to set the MBIST error bit.	W1S	0
FAULT_LBIST_SET	10	Write 1 to set the LBIST error bit.	W1S	0
FAULT_PROTOCOL_SET	9	Write 1 to set the interface protocol fault was detected.	W1S	0
FAULT_TIMEOUT_SET	8	Write 1 to set the transaction time-out error bit.	W1S	0
FAULT_PAR_REG_STATUS_SET	7	Write 1 to set the parity register error bit.	W1S	0
FAULT_PAR_REG_CONTROL_SET	6	Write 1 to set the parity error control bit.	W1S	0
FAULT_APAR_SET	5	Write 1 to set the address path parity error bit.	W1S	0
FAULT_DPAR_SET	4	Write 1 to set the data path parity error bit.	W1S	0
FAULT_RAM_ADDR_SET	3	Write 1 to set the the SRAM address fault bit.	W1S	0
FAULT_RAM_DATA_UNCORR_SET	2	Write 1 to set the un-correctable data fault bit.	W1S	0
FAULT_RAM_DATA_CORR_SET	1	Write 1 to set the correctable data fault bit.	W1S	0
FAULT_INTEGRITY_SET	0	Write 1 to set the integrity check fault bit.	W1S	0

5.14.5.11 CPC Power Fault Clear (FAULT_CLR) Register (offset = see below)

This register any bit written to 1 will clear the corresponding fault bit. Bits written to 0 have no effect. Read back is zero. It is instantiated for the CM and Core power domains, for FUSA CPUs only.

Offset: 0x9078 from GCR_BASE for CM power domain.

0xa078 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.93 Power Fault Clear Register Bit Assignments

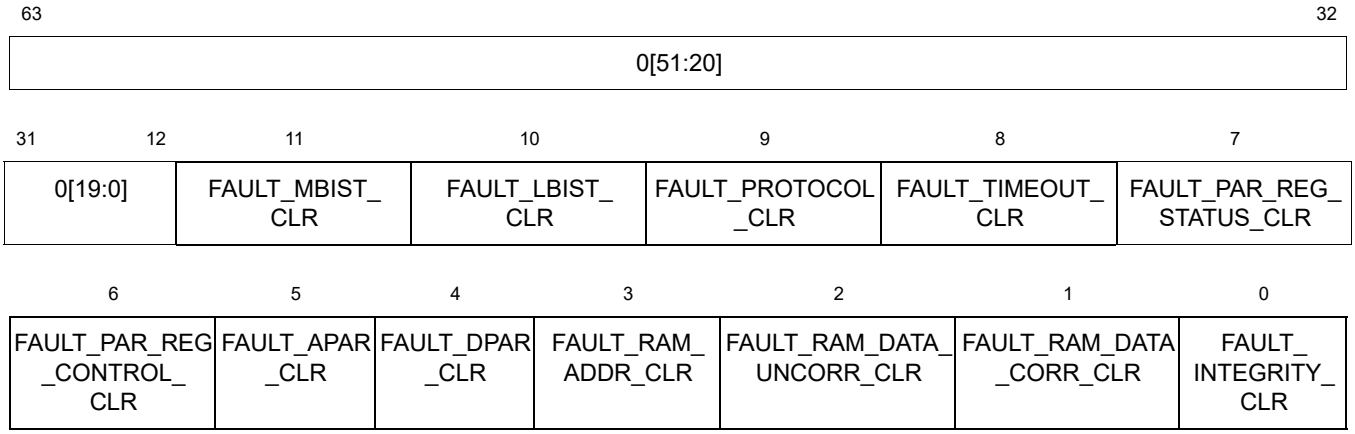


Table 95: Power Fault Clear Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	63:12	Reserved	R	0
FAULT_MBIST_CLR	11	Write 1 to clear the MBIST error bit.	W1C	0
FAULT_LBIST_CLR	10	Write 1 to clear the LBIST error bit.	W1C	0
FAULT_PROTOCOL_CLR	9	Write 1 to clear the interface protocol fault was detected.	W1C	0
FAULT_TIMEOUT_CLR	8	Write 1 to clear the transaction time-out error bit.	W1C	0
FAULT_PAR_REG_STATUS_CLR	7	Write 1 to clear the parity register error bit.	W1C	0
FAULT_PAR_REG_CONTROL_CLR	6	Write 1 to clear the parity error control bit.	W1C	0
FAULT_APAR_CLR	5	Write 1 to clear the address path parity error bit.	W1C	0
FAULT_DPAR_CLR	4	Write 1 to clear the data path parity error bit.	W1C	0
FAULT_RAM_ADDR_CLR	3	Write 1 to clear the the SRAM address fault bit.	W1C	0
FAULT_RAM_DATA_UNCORR_CLR	2	Write 1 to clear the un-correctable data fault bit.	W1C	0
FAULT_RAM_DATA_CORR_CLR	1	Write 1 to clear the correctable data fault bit.	W1C	0
FAULT_INTEGRITY_CLR	0	Write 1 to clear the integrity check fault bit.	W1C	0

5.14.5.12 CPC Power Configuration (CONFIG) Register (offset = see below)

This register contains configuration parameters for the domain.

It is instantiated for each power domain.

Offset: 0x9090 from GCR_BASE for CM power domain.

0x9190 from GCR_BASE for DBU power domain.

0xa090 + 0x100 * CORENUM from GCR_BASE for Core[0..63] power domain.

Figure 5.94 Power Configuration Register Bit Assignments

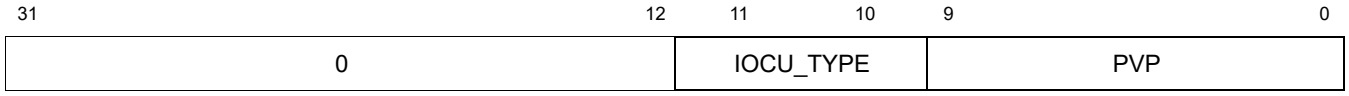


Table 96: Power Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:12	Reserved	R	0
IOCU_TYPE	11:10	Indicates type of Local Agent 0: The local agent is a CPU Core 1: The local agent is a non-caching IOCU.	R	From configuration
PVP	9:0	One less than the number of harts for CPUs, 0 for IOCUs.	R	From configuration

5.14.6 FDC.Global Registers

The FDC.Global region contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 97: FDC.Global Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x3F000	FDC.Global.FDACSR	FDC Access Control and Status Register
0x3F008	FDC.Global.FDCFG	FDC Configuration Register
0x3F010	FDC.Global.FDSTAT	FDC Configuration Register
0x3F018	FDC.Global.FDRX	FDC Receive Register
0x3F020 0x3F028 ..... 0x3F098	FDC.Global.FDTX[0-15]	FDC Transmit Registers

5.14.6.1 FDC Global Access Control and Status (FDACSR) Register (offset = 0x3F000)

This register is FDC access control and status register

Figure 5.95 Global Access Control and Status Register Bit Assignments

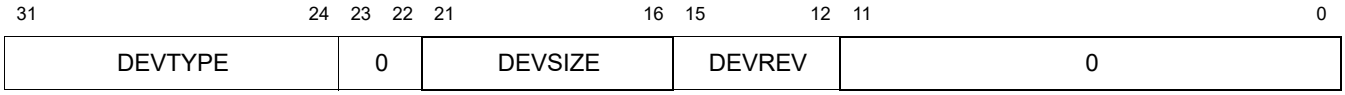


Table 98: Global Access Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
DEVTYPE	31:24	The type of device.	R	253
0	23:22	Reserved	R	0
DEVSIZE	21:16	The number of extra 64-byte blocks allocated to this device. The value 0x2 indicates that this device uses 2 extra, or 3 total blocks.	R	2
DEVREV	15:12	The revision number of the device. The value 0x1 indicates that this is the second version of FDC (with deeper FIFOs).	R	1
0	11:0	Reserved	R	0

5.14.6.2 FDC Global Configuration (FDCFG) Register (offset = 0x3F008

This register is FDC configuration register

Figure 5.96 Global Configuration Register Bit Assignments

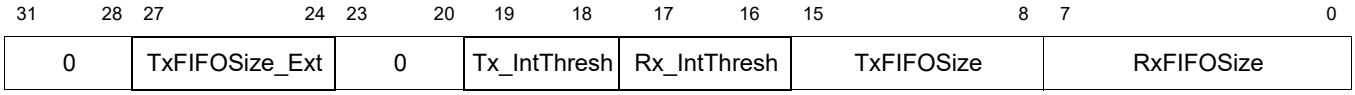


Table 99: Global Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:28	Reserved	R	0
TxFIFOSize_Ext	27:24	Upper bits of transmit FIFO size.	R	0
0	23:20	Reserved	R	0
Tx_IntThresh	19:18	Transmit Interrupt Threshold. 00 = no interrupt 01 = FIFO Full 10 = FIFO Not Empty 11 = FIFO Almost Full (contains 0 or 1 empty entry)	R/W	0
Rx_IntThresh	17:16	Receive Interrupt Threshold. 00 = no interrupt 01 = FIFO Full 10 = FIFO Not Empty 11 = FIFO Almost Full (contains 0 or 1 empty entry)	R/W	0
TxFIFOSize	15:8	Lower bits of the number of entries in the Transmit FIFO.	R	0
RxFIFOSize	7:0	Number of entries in the Receive FIFO. Set using TAP CONTROL register.	R	0

5.14.6.3 FDC Global Status (FDSTAT) Register (offset = 0x3F010)

This register is FDC status register.

Figure 5.97 Global Status Register Bit Assignments

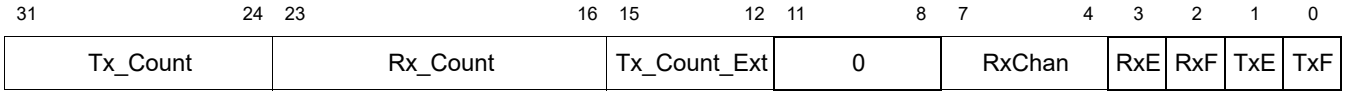


Table 100: Global Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
Tx_Count	31:24	Lower bits of number of occupied entries in the transmit FIFO.	R	0
Rx_Count	23:16	Number of occupied entries in the receive FIFO.	R	0
Tx_Count_Ext	15:12	Upper bits of number of occupied entries in the transmit FIFO.	R	0
0	11:8	Reserved	R	0
RxChan	7:4	Channel number of the top item in the receive FIFO. Only valid if FIFO not empty.	R	Undefined
RxE	3	Receive FIFO is empty.	R	1
RxF	2	Receive FIFO is full.	R	0
TxE	1	Transmit FIFO is empty.	R	1
TxF	0	Transmit FIFO is full.	R	0

5.14.6.4 FDC Global Receive (FDRX) Register (offset = 0x3F018)

This register is FDC receive register

Figure 5.98 Global Receive Register Bit Assignments

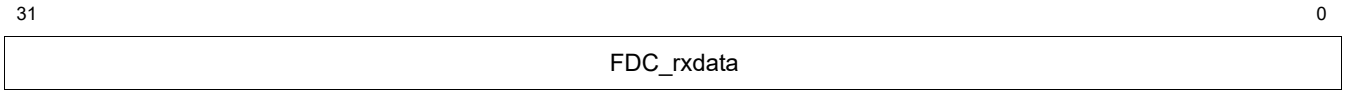


Table 101: Global Receive Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
FDC_rxdata	31:0	If FDC receive FIFO is not empty, reading this register will return the top item from the FIFO and advance the FIFO. If the FDC receive FIFO is empty, the effect of reading this register is undefined.	R	Undefined

5.14.6.5 FDC Global Transmit (FDTX[0-15]) Register (offset = 0x3F020)

This register is FDC transmit registers.

Figure 5.99 FDC Global Transmit Register Bit Assignments

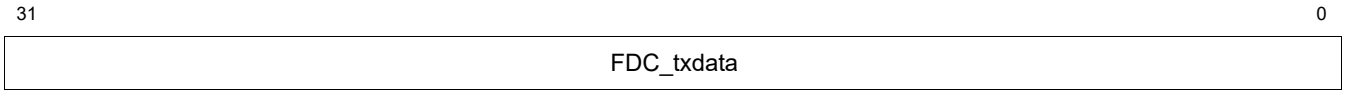


Table 102: FDC Global Transmit Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
FDC_txdata	31:0	IF FDC transmit FIFO is not full, writing this register adds one item to the transmit FIFO. The channel number (n) is derived from the low-order bits of the register address (offset 0x20 = channel 0, offset 0x28 = channel 1, etc). If FDC transmit FIFO is full, the effect of writing this register is undefined.	W	Undefined

5.14.7 Trace Funnel (TRF) Global Registers

The TRF.Global region contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 103: TRF.Global Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x3F100	TRF.Global.CONTROL	Trace Funnel Control
0x3F108	TRF.Global.CONFIG	Trace Funnel Configuration
0x3F110	TRF.Global.WRITEPTR	Trace Funnel Write Pointer
0x3F118	TRF.Global.READPTR	Trace Funnel Read Pointer
0x3F120 0x3F128 ..... 0x3F158	TRF.Global.DATA[0-7]	Trace Data [0-7]
0x3F160	TRF.Global.STUSER	System Trace User Control
0x3F168	TRF.Global.STENABLE	System Trace Enable

5.14.7.1 TRF Global Trace Funnel Control (CONTROL) Register (offset = 0x3F100)

This register is trace funnel control register

Figure 5.100 Global Trace Funnel Control Register Bit Assignments

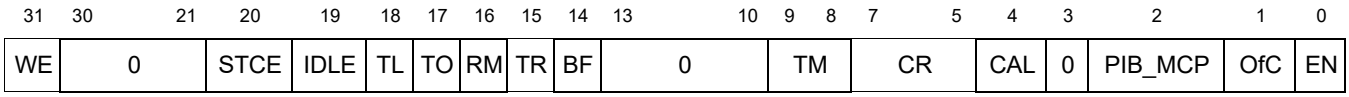


Table 104: Global Trace Funnel Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
WE	31	Write enable. Other fields of this register are written only when WE is written with 1 in the same write operation.	W	0
0	30:21	Reserved	R	0
STCE	20	System trace port enable. Write 1 to allow the funnel to capture system trace data.	R/W	0
IDLE	19	Funnel idle. Set when funnel has finished progressing pending TWs. In on-chip trace mode, idle is set when all TWs has been written to the on-chip memory. In off-chip trace mode, idle is set when all TWs has been sent to the PIB.	R	1
TL	18	Trace lock. Because either program or probe may use trace, before using trace they must first lock the trace resource by writing 1 to this field and its owner ID in field TO. Lock is successful when both TL and TO read back the written values. Only the owner can unlock by writing 0 to TL and its owner ID in field TO.	R	0
TO	17	Trace owner. 1 for probe and 0 for program.	R/W	0
RM	16	Read memory. Write 1 to initiate an SRAM read at TF_READPTR.	W	0
TR	15	Trace memory reset. When written to 1, the address pointers for on-chip trace memory are reset to zero. BF is also reset.	W	0
BF	14	Buffer full indicator. Set when the on-chip trace memory is full and writing has wrapped back to the beginning. This bit is cleared by writing 1 to TR.	R	0
0	13:10	Reserved	R	0

Table 104: Global Trace Funnel Control Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
TM	9:8	Trace Mode. This field determines how the trace memory is filled when using the simple-break control in the PDtrace IF to start or stop trace. In Trace-To mode, the on-chip trace memory is filled, continuously wrapping around, overwriting older Trace Words, as long as there is trace data coming from the core. In Trace-From mode, the on-chip trace memory is filled from the point that the core starts tracing until the on-chip trace memory is full (when the write pointer address is the same as the start pointer address). In both cases, de-asserting the EN bit in this register will also stop the fill to the trace memory. If a TCBTRIGx trigger control register is used to start/stop tracing, then this field should be set to Trace-To mode. Description Read / Write Reset State Compliance Name Bits TM Trace Mode 00 Trace-To 01 Trace-From 10 Reserved 11 Reserved	R	0
CR	7:5	Off-chip Clock Ratio. Writing this field sets the ratio of the core clock to the off-chip trace memory interface clock. Table 8.6 shows the encoding which only includes ratios where the processor clock is faster than trace clock. Remark: For example, a clock ratio of 1:2 implies a two times slowdown of the Probe interface clock to the core clock. However, one data packet is sent per core clock rising edge, while a data packet is sent on every edge of the Probe interface clock, since the Probe interface works in double data rate (DDR) mode. Please refer to MIPS PDTrace Specification document for the encoding.	R/W	0
CAL	4	Calibrate off-chip trace interface. If set, the off-chip trace pins will produce the trace pattern shown below in consecutive trace clock cycles. If more than 4 data pins exist, the pattern is replicated for each set of 4 pins. The pattern repeats from top to bottom until the Cal bit is deasserted. Note: The clock source of the TCB and PIB must be running. Please refer to MIPS PDTrace Specification document for the pattern.	R/W	0
0	3	Reserved	R	0
PIB_MCP	2	This bit is meaningful only if OfC bit is set to 1. If PIB_MCP is set to 1, trace is sent to off-chip memory via MCP port if set to 0, trace is sent to off-chip memory via PIB. This bit is not active if only on-chip mode is implemented. If MCP interface is not present this bit is read only and set to 0.	R/W	0
OfC	1	If set to 1, trace is sent to off-chip memory via PIB. If set to 0, trace is sent to on-chip memory. This bit is read-only if only on-chip or only off-chip mode is implemented.	R/W	Preset

Table 104: Global Trace Funnel Control Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
EN	0	Enables trace funnel to accept data from cores, CM and system trace and send to trace storage. Setting EN to 0 stops accepting data from trace sources but continues transferring trace words already accepted to SRAM. In on-chip trace-from mode, EN would be set to 0 when on-chip trace memory is full.	R/W	0

5.14.7.2 TRF Global Trace Funnel Configuration (CONFIG) Register (offset = 0x3F108)

This register is the trace funnel configuration register.

Figure 5.101 Global Trace Funnel Configuration Register Bit Assignments

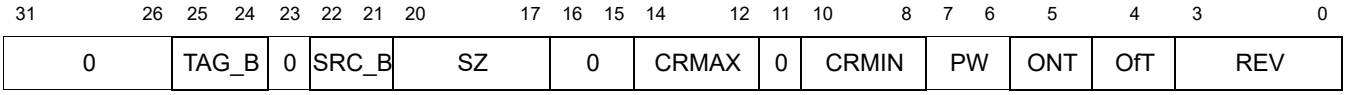


Table 105: Global Trace Funnel Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:26	Reserved	R	0
TAG_B	25:24	Size of TAG field in TRACE_WORD. Encoded as: 00 5 bits 01 6 bits 10 7 bits 11 8 bits	R	Preset
0	23	Reserved	R	0
SRC_B	22:21	Size of SRC filed in TRACE WORD. Encoded as: 00 1 bit 01 2 bits 10 3 bits 11 4 bits	R	Preset
SZ	20:17	On-chip trace memory size. This field holds the encoded size of the on-chip trace memory. The size in bytes is given by 2(SZ+8) and ranges from 256 bytes to 8 mega-bytes.	R	Preset
0	16:15	Reserved	R	0
CRMAX	14:12	Off-chip Maximum Clock Ratio. This field indicates the maximum ratio of the core clock to the off-chip trace memory interface clock.	R	Preset
0	11	Reserved	R	0
CRMIN	10:8	Off-chip Minimum Clock Ratio. This field indicates the minimum ratio of the core clock to the off-chip trace memory interface clock.	R	Preset
PW	7:6	Probe Width: Number of bits available on the off-chip trace interface TR_DATA pins. The number of TR_DATA pins is encoded, as shown in the table. PW Number of bits used on TR_DATA. Encoded as: 00 4 bits 01 8 bits 10 16 bits 11 reserved	R	Preset
ONT	5	Indicates on-chip trace is present.	R	Preset
OfT	4	Indicates off-chip trace is present.	R	Preset
REV	3:0	Indicates revision of Trace Funnel.	R	Preset

5.14.7.3 TRF Global Trace Funnel Write Pointer (WRITEPTR) Register (offset = 0x3F110)

This register is trace funnel write pointer.

Figure 5.102 Global Trace Funnel Write Pointer Register Bit Assignments

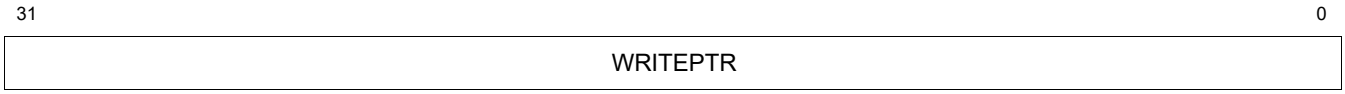


Table 106: Global Trace Funnel Write Pointer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
WRITEPTR	31:0	Contains the word address of the next SRAM location to be written when trace is being recorded. The format of this register is dependent upon the memory data width: 31:N 0 N-1:M WRITEPTR M-1:0 0 The lower M bits are always 0 since the pointer is always aligned on a TRF_MEM_DATA_WIDTH bits boundary. If this register is written to before enabling the funnel, the same value must be written to the read pointer.	R/W	0

5.14.7.4 TRF Global Trace Funnel Read Pointer (READPTR) Register (offset = 0x3F118)

This register is trace funnel read pointer

Figure 5.103 Global Trace Funnel Read Pointer Register Bit Assignments

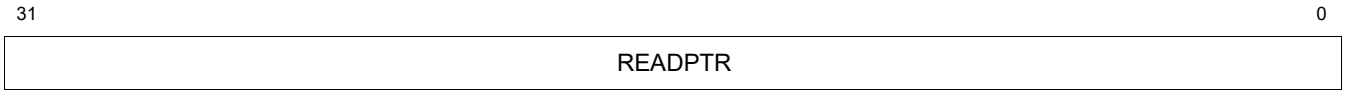


Table 107: Global Trace Funnel Read Pointer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
READPTR	31:0	Contains the word address of the next SRAM location to be read when trace is being read via the RRB slave. The format of this register is dependent upon the memory data width: 31:N 0 N-1:M READPTR M-1:0 0 The lower M bits are always 0 since the pointer is always aligned on a TRF_MEM_DATA_WIDTH bits boundary. READPTR is automatically incremented and an SRAM read operation is initiated when last TF_DATA register is read	R/W	0

5.14.7.5 TRF Global Trace Data (DATA[0-7]) Register (offset = see below)

This register is trace data 0 .. 7 register.

Offset: 0x3f120 + 0x8 * i from GCR_BASE for TRF.Global.DATA[i].

Figure 5.104 Global Trace Data Register Bit Assignments

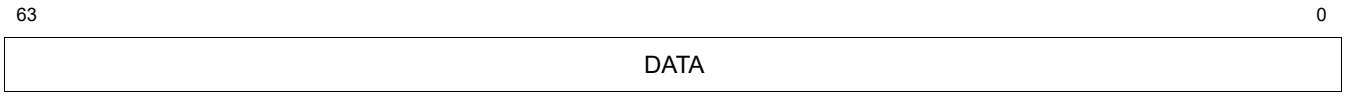


Table 108: Global Trace Data Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
DATA	63:0	Holding register for trace word data read from SRAM. A trace memory SRAM read is initiated when TRF.Global.CONTROL.RM is written with 1 or when TRF.Global.DATA[7] is read. To read a continuous block of trace, initialize TRF.GLOBAL.READPTR to the desired starting location, write 1 to TRF.Global.CONTROL.RM, then read TRF.Global.DATA[0] through TRF.Global.DATA[7] in order. Repeat TRF.Global.DATA[0] through TRF.Global.DATA[7] until all words have been read.	R/W	0

5.14.7.6 TRF Global System Trace User Control (STUSER) Register (offset = 0x3F160)

This register is system trace user control.

Figure 5.105 Global System Trace User Control Register Bit Assignments

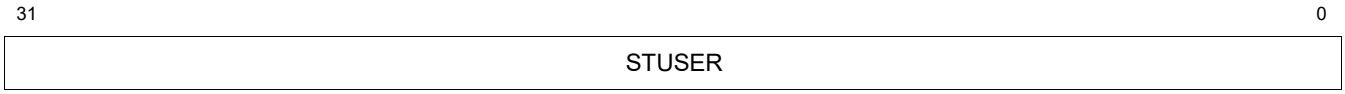


Table 109: Global System Trace User Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
STUSER	31:0	System trace user control, wired to the top level pin of co_tc_sys_user_ctl.	R/W	0

5.14.7.7 TRF Global System Trace Enable (STENABLE) Register (offset = 0x3F168)

This register system trace enable.

Figure 5.106 Global System Trace Enable Register Bit Assignments

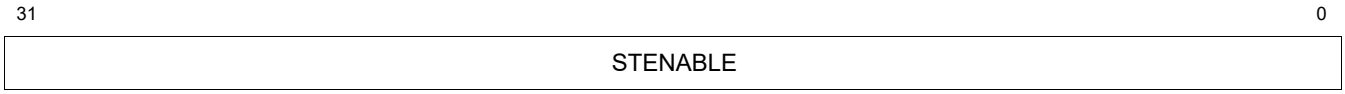


Table 110: Global System Trace Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
STENABLE	31:0	System trace enable, wired to the top level pin of co_tc_sys_enable.	R/W	0

5.14.8 GCR.U User Mode Registers

The GCR.U region contains shadow copies of specific GCRs which software can choose to make accessible from user mode. The GCR.U region occupies the last 4KB of the 512KB memory mapped register block, at offset 0x7F000 - 0x7FFFF from GCR_BASE.

The accessibility of the memory mapped registers from machine, supervisor and user modes is controlled by allowing or disallowing access to the corresponding physical addressees via programming of Physical Memory Protection (PMP) and page tables.

Table 111: GCR.U User Mode Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x7F050	GCR.U.MTIME_REG	U-mode shadow copy of mtime register
0x7F090	GCR.U.HRTIME_REG	U-mode shadow copy of high resolution timer register

5.14.8.1 GCR.U User Mode Timer (MTIME_REG) Register (offset = 0x7F050)

This register read-only shadow copy of the RISC-V mtime register.

Software can chose whether or not to make this register and the other registers in the 4KB GCR.U block accessible from U-mode by appropriate programming of PMP and page tables.

Figure 5.107 User Mode Timer Register Bit Assignments

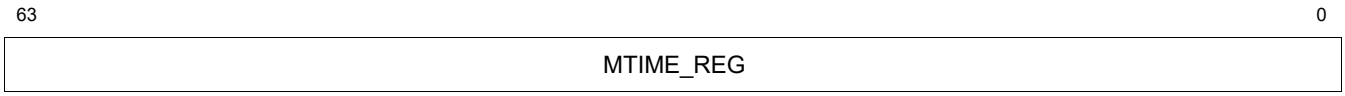


Table 112: User Mode Timer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MTIME_REG	63:0	Read-only shadow copy of the RISC-V mtime register.	R	0

5.14.8.2 GCR.U High Resolution Timer (L2_CONFIG) Register (offset = 0x7F090)

This register read-only shadow copy of the high resolution timer register.

Software can chose whether or not to make this register and the other registers in the 4KB GCR.U block accessible from U-mode by appropriate programming of PMP and page tables.

Figure 5.108 High Resolution Timer Register Bit Assignments

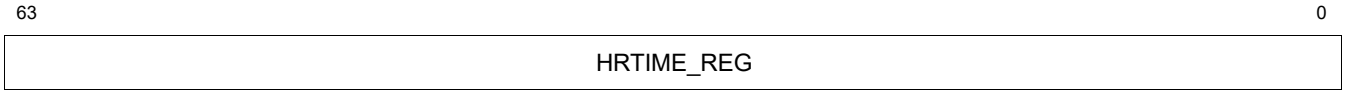


Table 113: High Resolution Timer Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
HRTIME_REG	63:0	High resolution timer register.	R	0

Chapter 6

Power Management

Power management in the P8700-F Multiprocessing System is handled by the Cluster Power Controller (CPC). The P8700-F CPC uses the concept of domains to manage both power and clocking throughout the device. Using registers, the programmer can enable or disable these domains in order to reduce overall power consumption.

The CPC implements two types of domains; power and clock. In each case, registers are instantiated on a per-domain basis so that the domain can be individually controlled by kernel software. This is true for each power domain and each clock domain.

- For the power domains, kernel software uses registers in the CPC to control the power to individual elements in the system such as cores, IOCU's, and the Coherence Manager (CM). The various power domains that can be individually controlled are defined in the section entitled Power Domains.
- For the clock domains, kernel software uses registers in the CPC to control the clock frequency to the individual elements in the system such as cores, IOCU's, Coherence Manager (CM), and memory. In addition to clock management for the various devices in the P8700-F Multiprocessing System, the CPC also provides the ability to change the clock ratios in memory, and put the caches into a low-power state. The various clock domains that can be individually controlled are defined in the section entitled Clock Domains.

This chapter provides an overview of how power is managed in the P8700-F Multiprocessing System and identifies the various power and clock domains the programmer can use to manage power consumption in the device. Other programming principles include setting the device to coherent or non-coherent mode, requestor access of CPC registers, system power-up policy, programming examples of a clock domain change and clock delay change, powering up the CPC in standalone mode (no cores enabled), reset detection, Hart run/suspend mechanism, local RAM shutdown and wakeup procedure, accessing registers in another power domain, and fine tuning internal and external signal delays to help the programmer easily integrate the device into a system environment.

6.1 Overview

This section provides an overview of the power and clock management schemes implemented in the P8700-F Multiprocessing System.

6.1.1 Power Domains

Figure 6.1 shows the various power domains in the P8700-F Multiprocessing System. Registers are instantiated for each power domain to allow for individual control. Note that in this figure, core 1 through core n are optional blocks depending on the system configuration.

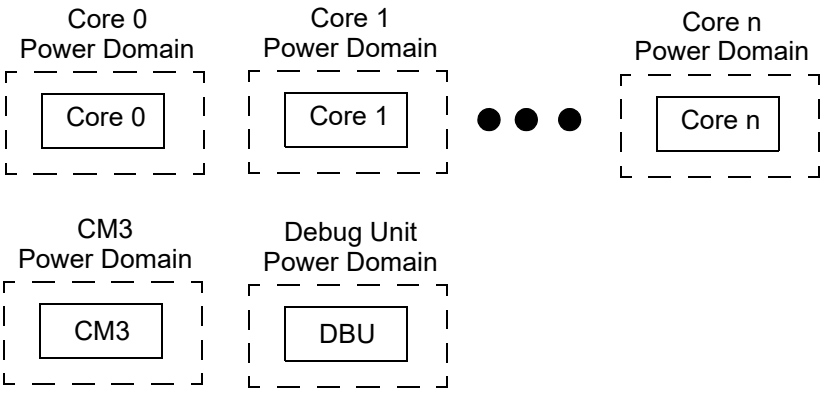


Figure 6.1 Power Domains in the P8700-F Multiprocessing System

6.1.2 Clock Domains

Figure 6.2 shows the various clock domains in the P8700-F Multiprocessing System. Each clock domain shown can be individually controlled using the CPC register interface.

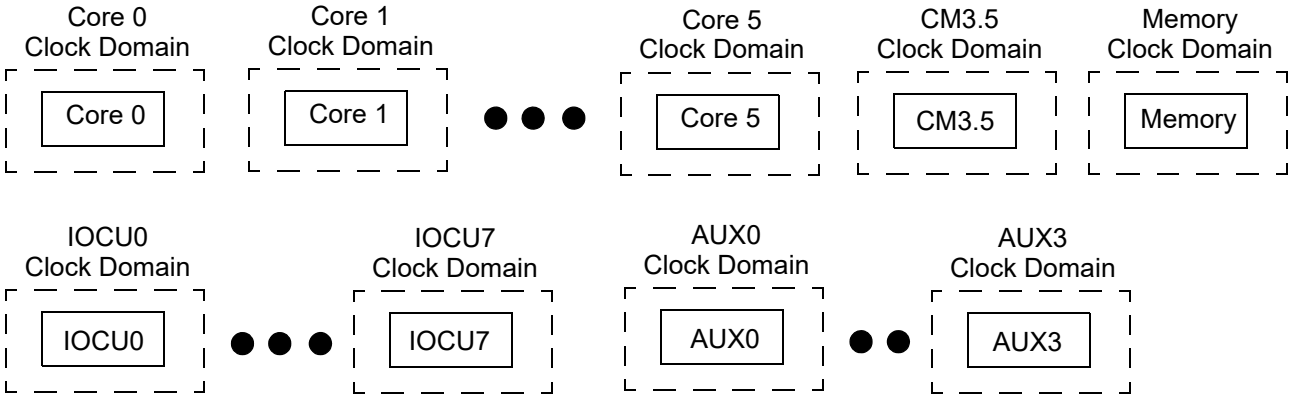


Figure 6.2 Clock Domains in the P8700-F Multiprocessing System

6.1.3 Core and IOCU Selection

Figure 6.2 shows the maximum possible number of cores and IOCUs that can be instantiated into the P8700-F MPS. However, the total number of cores and IOCUs cannot exceed eight. So for example, if there are two cores, there cannot be more than six IOCUs. If there are four cores, there cannot be more than four IOCUs, etc.

6.1.4 Overview of Power States

Each device in Figure 6.1, except the CM, contains its own set of Core-Local registers that can be used to independently place each device into one of the following four power states by programming the CMD field (bits 3:0) of the *CPC Local Command* Register.

Note that each command can only be executed in non-coherent mode. If a command is executed in coherent mode, the command is queued, but not processed by the CPC until the device has transitioned from coherent mode to non-coherent mode. For more information, refer to the section entitled Enabling Coherent Mode.

The states are as follows:

- **ClockOff** - a power domain is brought into *ClockOff* state when a value of 0x1 is programmed into the 4-bit CMD field of the *CPC.Pwr.CMD_REG* register. If the domain was powered down before, the power-on sequence is applied according to *CPC_Core_STAT_CONF_REG* settings. If the domain was active before and was in non-coherent operation, the power domain is brought into the *ClockOff* state. A domain in the *ClockOff* state can be sent into operation using the *PwrUp* command.

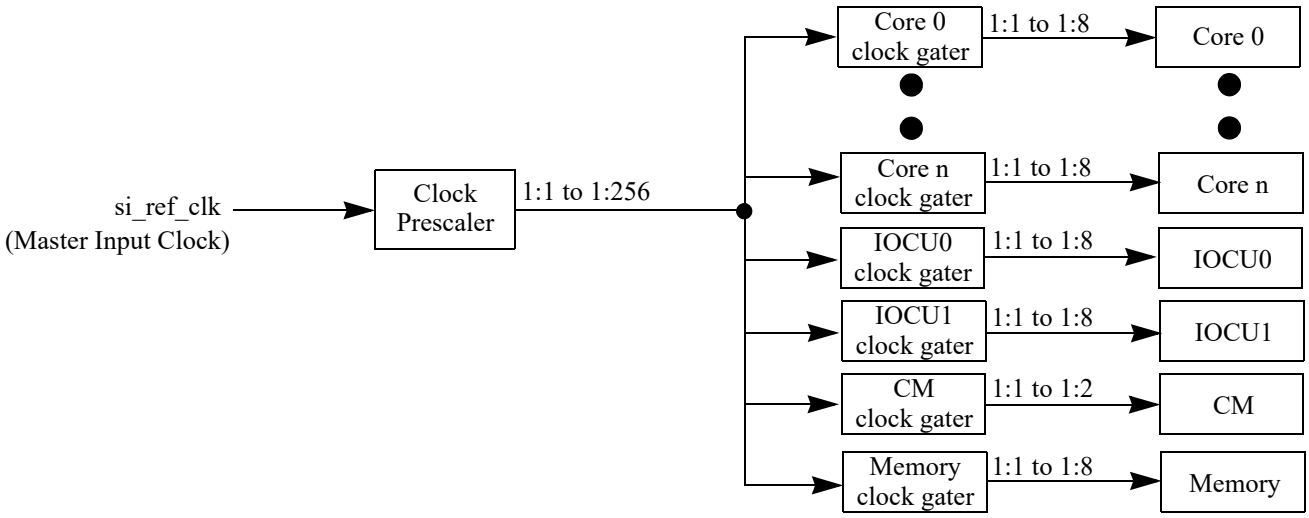
  A *ClockOff* command given to a domain in coherent operation remains inactive until the device has left the coherent mode of operation. Sending a *ClkOff* command to the CPC before a previous command has completed causes the CPC domain target to be redirected towards *ClockOff*. However, the previous steady state can be observed temporarily before the newly programmed state is reached. Refer to the section entitled Enabling Coherent Mode for more information on enabling and disabling coherence mode.
- **PwrDown**. A power domain is brought into *PwrDown* state when a value of 0x2 is programmed into the 4-bit CMD field of the *CPC.Pwr.CMD_REG* register. This command uses setup values in the *CPC_Core_STAT_CONF_REG* register.

  A *PwrDown* command given to a domain in coherent operation will remain inactive until the device has left the coherent mode of operation. Sending a *PwrDown* command to the CPC before a previous command has completed causes the CPC domain target to be redirected towards *PwrDown*.
- **PwrUp** - A power domain is brought into *PwrUp* state when a value of 0x3 is programmed into the 4-bit CMD field of the *CPC.Pwr.CMD_REG* register. This command uses setup values in the *CPC_Core_STAT_CONF_REG* register. The execution of this command depends on the previous domain power state. If the domain is in the powered-down state, a *PwrUp* command enables power for the domain, applies the clocks and reset, and brings the domain into an operational state.
- **Reset** - A power domain is brought into *Reset* state when a value of 0x4 is programmed into the 4-bit CMD field of the *CPC.Pwr.CMD_REG* register. This command allows a domain in the non-coherent operation to be reset. It also can be sent to a domain in power-down or clock-off mode. The domain will then become active, and a reset sequence is executed which leads to an operational steady state of the domain.

6.2 Individual Clock Gating

The P8700-F Multiprocessing System provides two levels of clock gating. In addition to the individual clock gating of each device, global clock gating to all devices simultaneously can be performed by adjusting the ratio of the clock prescaler as shown in Figure 6.3.

Figure 6.3 Individual and Global Clock Gating in the P8700-F Multiprocessing System



The clock prescaler can be programmed to reduce the master input clock by a frequency range of 1:1 to 1:256. The output of the prescaler becomes the master clock input to all other devices in the system.

6.3 Global Control Block Register Map

All registers in the Global Control Block are 64 bits wide and should only be accessed using aligned 64-bit uncached load/stores. Reads from unpopulated registers in the CPC address space return 0x0, and writes to those locations are silently dropped without generating any exceptions.

Table 6.1 Global Control Block Register Map (Relative to Global Control Block Offset)

Register Offset in Block	Name	Type	Description
0x0008	CPC Global Sequence Delay Counter (*CPC_SEQDEL_REG*)	R/W	Time between microsteps of a CPC domain sequencer in CPC clock cycles.
0x0010	CPC Global Rail Delay Counter Register (*CPC_RAIL_REG*)	R/W	Rail power-up timer to delay CPS sequencer progress until the gated rail has stabilized.
0x0018	CPC Global Reset Width Counter Register (*CPC_RESETLEN_REG*)	R/W	Duration of any domain reset sequence.
0x0020	CPC Global Revision Register (*CPC_REVISION_REG*)	R	RTL Revision of CPC
0x0028	CPC Global Clock Control Register (*CPC_CC_CTL_REG*)	R	CPC global clock change configuration, control and status. Enables clock change for all clock change enable domains of the cluster.

Table 6.1 Global Control Block Register Map (Relative to Global Control Block Offset) *(continued)*

Register Offset in Block	Name	Type	Description
0x0030	CPC Global CM Powerup Register (*CPC_PWRUP_CTL_REG*)	R	Controls Power of CM even independent of Cores' power states.
0x0038	CPC Reset Release Register (*CPC_RES_REL_REG*)	R	Control Reset release and Clock Enable timing.
0x0040	CPC Global Reset Occurred Register (*CPC_ROCC_CTL_REG*)	R	Register to indicate which cores have been reset.
0x0048	CPC Global Reset Occurred Register (*CPC_PRESCALE_CC_CTL_REG*)	R	Controls Precale Clock changes.
0x0050	MTIME Register (CPC_MTIME_REG)	R/W	RISCV timer. Register can be written to synchronize with other cluster's time.
0x0058	Counter Control for MTIME and HRTIME (CPC_TIMECTL_REG)	R/W	Support for Software-assisted multi-cluster time synchronization
0x0060	RESERVED		Reserved.
0x0068	CPC Global Fault Status Register (CPC_FAULT_STATUS)	R/W	Logical OR of all domain status registers.
0x0070	CPC Global Fault Supported Register (CPC_FAULT_SUPPORTED)	R/W	
0x0078	CPC Global Fault Enable Register (CPC_FAULT_ENABLE)	R/W	
0x0080, 0x0088		R/W	
0x0090	HRTIME Counter Register (CPC_HRTIME_REG)	R/W	
0x0098 - 0x0134	CPC GLOBAL RESERVED	R/W	
0x0138	CPC Global Config Register (CPC_CONFIG)	R/W	
0x0140	CPC System Configuration Register (CPC_SYS_CONFIG)	R/W	
0x0200 - 0x03FF	CPC_IOCU Clock Change Control Register (CPC_IOCUx_CC_CTL_REG) with x from 0 to 63)	R/W	
0x0400	CPC_MEM_CC_CTL_REG	R/W	
0x0404 - 0x0408	CPC_AUXn_CC_CTL_REG , n = 0 3	R/W	

6.4 Local Control Blocks

All registers in the CPC Local Control Block are 64 bits wide and should only be accessed using aligned 64-bit uncached load/stores. Reads from unpopulated registers in the CPC address space return 0x0, and writes to those locations are silently dropped without generating any exceptions. A set of these registers exists for each core in the P8700-F MPS.

The register offsets shown are relative to the offsets listed in Table 6.2.

Table 6.2 Core-Local Block Register Map

Register Offset in Block	Name	Type	Description
0x000	CPC Power Command Register (*CPC.Pwr.CMD_REG*)	R/W	Places a new CPC domain state command into this individual domain sequencer. This register is not available within the CM sequencer. Writes to the CM CMD register are ignored while reads will return zero.
0x008	CPC Core Status and Configuration register (*CPC_Core_STAT_CONF_REG*)	R/W	Individual domain power status and domain configuration register. Reflects domain micro-sequencer execution. Initiates micro-sequencer after status register programming. Reflects command execution status.
0x018	CPC Global Clock Change Control Register (*CPC.Global.CC_CTL_REG*)	R/W	Controls clock changes on corresponding device
0x020	CPC Power Hart Stop Register (*CPC.Pwr.VP_STOP_REG*)	R/W	Stops execution of the Hart.
0x028	CPC Power Hart Run Register (*CPC.Pwr.VP_RUN_REG*)	R/W	Starts execution of the Hart.
0x030	CPC Power Hart Running Register (*CPC.Pwr.VP_RUNNING_REG*)	R/W	Indicates which Harts are in the run state.
0x050	CPC Power RAM Sleep Register (*CPC.Pwr.RAM_SLEEP_REG*)	R/W	Controls the Deep Sleep and Shut Down power state of the RAMs.

6.5 CPC Register Programming

This section describes some of the programming functions that can be performed via the CPC registers.

6.5.1 Cluster Power Controller Register Address Map

The CPC uses memory locations within the global and core-local address space. All address locations in this document are relative to a base address of 0x0000_8000.

In Table 6.3, all registers are accessed using 32-bit aligned uncached load/stores. All address locations in this document are relative to the fixed offset CPC base address from GCR_BASE.

Table 6.3 CPC Address Map

Block Offset	Size (bytes)	Description
0x0000 - 0x01FF	512B	**Global Control Block**. Contains registers pertaining to the global system functionality. This address section contains a single set of registers that is visible to all CPUs.

Table 6.3 CPC Address Map

Block Offset	Size (bytes)	Description
0x0200 - 0x0408 0x04040 - 0x04078	KB	Clock Control Register for CPC_IOCU, CPC_MEM, CM MSTR, CPC_AUX, and CPC_IOMMU.
0x1000 - 0x5F90	8 KB	CPC Core Local registers For CM, DBU and Core local (Core0 to Core63)
0x5F94 - 0xDFFF		Reserved.

6.5.2 Global Control Block Register Map

All registers in the Global Control Block are 64 bits wide and should only be accessed using aligned 64-bit uncached load/stores. Reads from unpopulated registers in the CPC address space return 0x0, and writes to those locations are silently dropped without generating any exceptions.

6.5.3 Local Control Blocks

All registers in the *CPC Local Control Block* are 64 bits wide and should only be accessed using aligned 64-bit uncached load/stores. Reads from unpopulated registers in the CPC address space return 0x0, and writes to those locations are silently dropped without generating any exceptions.

A set of these registers exists for each core in the P8700-F MPS. In the case of some CPC registers, a set of registers exists per power domain or per clock domain.

6.5.4 Requestor Access to CPC Registers

6.5.4.1 Register Interface

The CPC allows up to eight requestor's in a system. A requestor can be either a core or an IOCU. The requestor may not have unrestricted access to the CPC registers. During boot time, the programmer determines which requestor's are provided access to the CPC registers by programming the *Global Access Privilege* register located at offset 0x120 in the CM register map. The 8-bit *ACCESS_EN* field (bits 7:0) of this register selects up to eight cores, and bits 23:16 enable access for IOCU7 through IOCU0 respectively.

The MIPS default for *ACCESS_EN* field is 0xFF, meaning that all cores in the system have access to the CPC register set. In addition, bits 23:16 are set to allow IOCU7 through IOCU0 access to the CPC register set. To disable access to the registers for a particular requestor, kernel software need only clear the bit corresponding to that core or IOCU, and all write requests to the CPC registers by that requestor will be ignored.

6.5.5 Enabling Coherent Mode

The P8700-F Multiprocessing System allows each power domain to be placed in either a coherent or non-coherent mode. Because the P8700-F implements a directory-based coherence protocol, MIPS recommends that each domain be placed in coherent mode during normal operation. The non-coherent mode should only be used during boot-up and power-down. Software should not execute any cacheable memory accesses (instruction fetch or load/store) while coherence is disabled.

Register Interface

Coherency is enabled when *gcr_cl_coh_en* in bit 11 (COH_EN) of the *Core-Local Status and Configuration* register equals 0x1. This register resides in the CM local register block at offset address (0x20F8 + 0x100 x CoreNum). There is one of these registers per power domain.

Note that if a power domain is in coherent mode and a change to the power state is initiated, the caches must be flushed prior to disabling coherence mode.

Coherent Mode Enable Code Example

The base address for the location of the CM GCR registers is programmed into the CSR CMGCRBase register. As a reference, a value of 0x0000_1FB8_0000 is used (MIPS default) to indicate the base location of the CM global control registers. In this case, the base value is read from the CSR register and an offset is added to it to derive the exact register address where the *Core Local Coherence Control* register is located.

By default, coherence is disabled in the P8700-F MPS.

6.5.6 Master Clock Prescaler

The clock prescaler is used to reduce the frequency of all devices in the system simultaneously.

The prescaler can be programmed as follows using the global *CPC Prescale Clock Change Control* register located at offset address 0x0048.

1. Verify that the *PRESCALE_CLK_RATIO_CHANGE_EN* bit of this register (bit 8) is set. This bit must be set before the *CLK_PRESCALE* field can be changed.

2. Optionally, the programmer can read the *PRESCALE_CLK_RATIO* field in bits 26:23 of this register to determine the current clock prescaler ratio.

3. Program the *CLK_PRESCALE* field (bits 7:0) to set the clock ratio. A value of 0x00 indicates a 1:1 clock ratio (no difference between input and output frequency of the prescaler). A value of 0xFF indicates a 1:256 ratio between the master input clock and the output of the prescaler.

The 8-bit CLK_PRESCALE field can be programmed as follows to select the prescaler ratio.

Table 6.4 Encoding of the CLK_PRESCALE Field

Encoding	Description
0x00	No prescaling
0x01	Divide input clock by 2
0x02	Divide input clock by 3
0x03	Divide input clock by 4
0x04	Divide input clock by 5
.....	.....
0xFD	Divide input clock by 254
0xFE	Divide input clock by 255
0xFF	Divide input clock by 256

For an example of how to program these fields, refer to step 1 of the procedure in Section 6.5.7.1, "Clock Domain Change Example — Register Programming Sequence".

By default, the clock prescaler is disabled in the P8700-F MPS. The clock prescaler is enabled and the clock divide ratio is set to divide by 4. Note that the PRESCALE_CLK_RATIO field in bits 23:16 of this register is a read-only field that is updated by hardware and allows kernel software to quickly read this register to determine the current clock ratio.

6.5.7 Individual Device Clock Ratio Modification

Based on the input clock frequency to each individual device supplied by the clock prescaler, each device can further reduce the clock by a frequency range of 1:1 to 1:8, except for the CM, which can be programmed with a frequency ratio of either 1:1 or 1:2 relative to its input clock as shown in the figure. This is accomplished by programming the CLK_RATIO field (bits 2:0) of each *CPC Local Clock Change Control* register located at offset address 0x0018. For an example of how to program this field, refer to step 2 of the procedure in the section entitled Clock Domain Change Example — Register Programming Sequence.

6.5.7.1 Clock Domain Change Example — Register Programming Sequence

The following example shows how to run core 0 at full speed, and core 2 at quarter-speed to save power. Assume the following:

- 2-core system
- 1 Hart per core
- si_ref_clk input frequency of 1 GHz
- Prescaler output of 1 GHz
- Core 0 input frequency of 1 GHz
- Core 1 input frequency of 250 MHz

In this example, the *si_ref_clk* input to the clock prescaler is 1 GHz. As shown above, the output frequency of the prescaler in this example is also 1 GHz. This ratio is accomplished by programming the global *CPC Prescale Clock Change Control* register located at offset address 0x0048 as follows. Note that this register is global and is seen by all cores and all individual devices (clock domains) in the system.

Register Interface

To program the clock prescaler for this example:

1. Write a value of 0x100 to the global *CPC Prescale Clock Change Control* register located at offset address 0x0048. This value sets the CLK_PRESCALE field to a value of 0x00, indicating a 1:1 relationship between the input clock and the output clock. This value also sets the PRESCALE_CLK_RATIO_CHANGE_EN bit to indicate that the value in the CLK_PRESCALE field is valid.

2. In this example the core 0 is running at full speed. Core 1 is running at 1/4 speed. To set the ratio of the clock generators for core 0 so it operates at 1 GHz, and core 1 so it operates at 250 MHz, program the individual *CPC Local Clock Change Control* registers. This register is instantiated as one per clock domain, so in this case each core has its own register since each core is in its own domain.

3. Set the SET_CLK_RATIO bit in the *CPC Global Clock Change Control* register located at offset 0x0028 to initiate a clock change for all clock domains participating in the clock change, which is cores 0 - 3 in this example. This bit is cleared by hardware once the clock change has completed.

Table 6.5 shows the programming of the CLK_RATIO field (bits 2:0) of the corresponding *CPC Local Clock Change Control* register located at offset address 0x0018.

Table 6.5 Programming the CLK_RATIO Field of the CPC Local Clock Change Registers

Core	CLK_RATIO Value	Clock Ratio	Core Clock Frequency
0	3'b000	1:1	1 GHz
1	3'b100	4:1	250 MHz

Poll the following registers to determine when the clock change has completed.

- Read the CPC_CC_CTL_REG register to determine when bit 8 (SET_CLK_RATIO) is 0. If SET_CLK_RATIO is 1, the change request is still pending.
- Read the CPC_CC_CTL_REG to determine when bit 10 (CLK_CHANGE_ACTIVE) is 0. If CLK_CHANGE_ACTIVE = 1, the clock change is in progress.
- When both of these bits are zero, the clock change has completed. At this point, another clock change could be requested.

6.5.7.2 Clock Change Delay

The *CPC_CC_CTL_REGCC_DELAY* field in bits 29:20 of the *CPC Global Clock Control* register is used to optimize the amount of delay during a clock change. This can be done if all clock domain ratios are low. For example, if all current clock ratios are less than 1:4 the value of the delay could be reduced. The intent is that clock domain changes do not happen very often, so setting the default of 80 clocks should not be a problem and leaving this value at its default delay is recommended. This register could also be used to extend the state delay period if desired.

6.5.8 CM Standalone Powerup

Normally, the CM is automatically powered-up if any core is powered-up. Conversely, the CM is automatically powered-down if all cores are powered-down. The P8700-F allows for the CM to be powered-up even if no core is powered-up. This is useful for system debug/setup via the DBU.

6.5.8.1 Register Interface

This functionality is controlled by the CPC Global Power Up register (CPC_PWRUP_CTL_REG) located at offset address 0x0030.

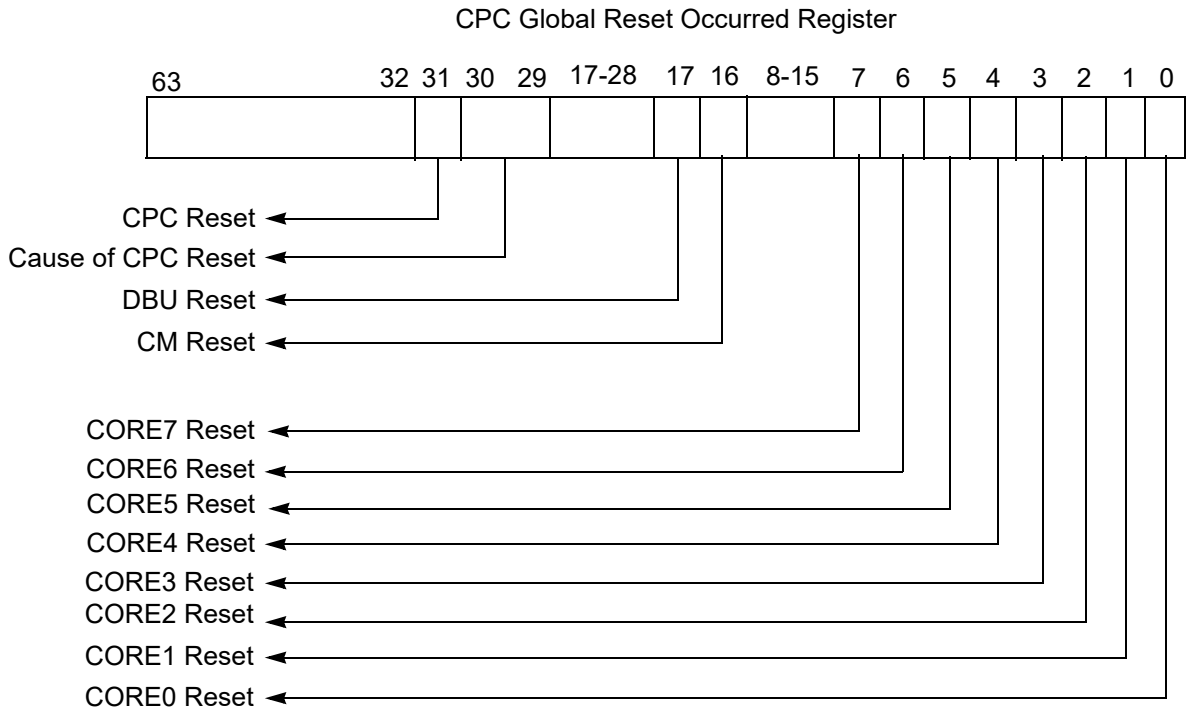
The DBU may execute a one-time power-up of the CM by writing a 1 to this register. If the CM is not operational at the time this bit is set by the DBU, it will transition from its current state to an operational state. If the CM is already operational, setting this bit has no meaning and the register write is ignored.

6.5.9 Reset Detection

The CM provides a series of read-only bits that allow the programmer to determine when a given device connected to the CM has been reset, including the CPC itself. Whenever a device is reset, the corresponding bit of the *CPC Global Reset Occurred* register (CPC_ROCC_CTL_REG) at offset 0x0040 is set.

In addition to the reset detection, this register also contains a 2-bit field (RESET_CAUSE) that indicates the type of reset for the CPC block. Reset options are cold reset, external warm reset, and watchdog timer reset. The functionality of this register is shown in Figure 6.4.

Figure 6.4 Reset Detection in the P8700-F Multiprocessing System



6.5.10 VP Run/Suspend

Three registers are used to control the power state of each Hart in the system. The P8700-F Multiprocessing system supports up to four Harts per core, and up to six cores per system. Each of these registers is instantiated per core.

Three registers are used to control this functionality:

- *VP Run* register (WO)
- *VP Stop* register (WO)
- *VP Running* register (RO)

Register Interface

The *VP Run* register is a Write-only register used to set each Hart to the run state. The *VP Run* register contains a 2-bit field, where each bit is dedicated to a particular Hart, up to two per core. Prior to setting one of these bits, kernel software must ensure that the Hart in question is not already running by reading the corresponding bit in the *VP Running* register. If a given bit in the *VP Running* register is cleared, setting the corresponding bit in the *Hart Run* register places the Hart in the run state. If a given bit in the *VP Running* register is already set, setting the corresponding bit in the *VP Run* register has no meaning. The value in this register is reset whenever the associated core is reset. The *VP Run* register can also be cleared by hardware, as well as the Debug unit.

The *VP Stop* register is a write-only register used to stop a Hart. If a given bit in the *Hart Running* register is set, setting the corresponding bit in the *VP Stop* register places the Hart in the suspend state. Writing a 0 to any of the bits in the *VP Stop* register has no effect.

The *VP Running* register is a read-only register that indicates the run state of each Hart in a given core. These bits are set and cleared by hardware based on the programming of the *VP Run* and *VP Stop* registers by kernel software as described above.

Note that for each of these registers, the two Harts correspond to the register bits as follows:

- Bit 0 = Hart0
- Bit 1 = Hart1

For example, to set Hart2 of a given core to the Run state, kernel software would do the following,

1. Read bit 2 of the *VP Running* register. If this bit is already set, Hart2 is already running and no action need be taken.

2. If bit 2 of the *VP Running* register is cleared, indicating that Hart2 is in the Suspend state, kernel software sets bit 2 of the *Hart Run* register to set Hart2 to the Run state.

To set Hart2 of a given core to the Suspend state, kernel software would do the following,

1. Read bit 2 of the *VP Running* register. If this bit is already cleared, Hart2 is already in the Suspend state and no action need be taken.

2. If bit 2 of the *VP Running* register is set, indicating that Hart2 is in the Run state, kernel software sets bit 2 of the *VP Stop* register to set Hart2 to the Suspend state.

6.5.11 Local RAM Deep Sleep / Shutdown and Wakeup Delay

The CM allows the local RAM's within a given power domain (cores, CM, IOCU, etc) to be placed into either Shutdown mode where the clocks are turned off, or Deep Sleep mode where the clocks are running at a fraction of their normal frequency. This functionality is controlled through the *CPC Local RAM Sleep Control* register (*CPC_CL_RAM_SLEEP*) located at offset 0x1050 + 0x100 * CM/DBU/Core_num.

This register is instantiated per power domain, so each domain has the ability to power cycle its own local RAM devices.

6.5.11.1 RAM Deep Sleep Mode

When bit 31 (RAM_DEEP_SLEEP_DISABLE) of the *CPC_CL_RAM_SLEEP* is cleared (logic '0'), the RAM's on the local device enter the Deep Sleep low power state when the CPC power state for the device reaches the ClockOff state. In this state the clocks to the local RAM's within that power domain are running at a fraction of their normal frequency.

The CPC also provides a way to delay the transition from the deep sleep state to the run state using bits 23:16 RAM_DEEP_SLEEP_WAKEUP_DELAY) of the *CPC_CL_RAM_SLEEP* register. Once awoken, the CPC delays the transition to the run state by the value programmed into this field in order to provide sufficient time for the RAMs to wake up from Deep Sleep. The delay can range from 1 to 255 (0xFF) clocks.

6.5.11.2 RAM Shut Down Mode

When bit 15 (RAM_SHUT_DOWN_DISABLE) of the *CPC_CL_RAM_SLEEP* is cleared (logic '0'), the RAM's on the local device enter the Shutdown low power state when the CPC power state for the device reaches the PwrDwn state. In this state the clocks to the local RAM's within that power domain are off. The RAM's remain in the Shutdown low power state even if the CPC power state changes to ClkOff without transitioning to the operational state.

The CPC also provides a way to delay the transition from the shutdown state to the run state using bits 7:0 RAM_SHUT_DOWN_WAKEUP_DELAY) of the *CPC_CL_RAM_SLEEP* register. Once

awoken, the CPC delays the transition to the run state by the value programmed into this field in order to provide sufficient time for the RAMs to wake up from the Shut Down state. The delay can range from 1 to 255 (0xFF) clocks.

6.5.12 Fine Tuning Internal and External Signal Delays

This section describes those register fields that can be used to delay the assertion of external signals relative to one another, as well as the internal domain sequencer state machine. These registers are used to help accommodate a wide variety of timing constraints in the system. Signals can be lengthened or shortened accordingly in order to meet system timing.

6.5.12.1 Global Sequence Delay Count

The Sequence Delay register (*CPC_SEQDEL_REG*) located at offset 0x0008 in the CPC Global Control Block, contains a 10-bit MICROSTEP field that describes the number of clock cycles each domain sequencer state machine will take to advance to the next state.

The 10-bit MICROSTEP field contains a default value of 0x002, indicating a 2-cycle delay. However, should additional delay be required based on the system implementation, this register provides the programmer with the ability to increase the sequence delay as necessary.

Domain sequencing begins once the RAILDELAY field has counted down to zero. Refer to the section entitled Rail Delay for more information.

The 10-bit MICROSTEP field is encoded as follows:

Table 6.6 Encoding of MICROSTEP Field

Encoding	Description
0x000	1-cycle delay
0x001	2-cycle delay
0x002	3-cycle delay
0x003	4-cycle delay
0x004	5-cycle delay
.....	.....
0x3FD	1022-cycle delay
0x3FE	1023-cycle delay
0x3FF	1024-cycle delay

6.5.12.2 Rail Delay

The Rail Delay register (*CPC_RAIL_REG*) located at offset 0x010 in the CPC Global Register Block contains a 10-bit counter field (*RAILDELAY*) used to schedule delayed start of power domain sequencing after the *RailEnable*[1] signal has been activated by the CPC. This allows the CPC to compensate for slew rates at the gated rail.

The 10-bit counter value (*RAILDELAY*) delays the power-up sequence per domain. The power-up sequence starts after *RAILDELAY* has been loaded into the internal counter and a count-down to zero has concluded. At IP configuration time, the contents of the *CPC_RAIL_REG* register are preset. However, for fine tuning, the register can be written at run time.

1. This signal is shown only for illustration purposes. Refer to the *P8700-F Integrator's Guide* for the exact name and usage of this signal.

The 10-bit RAILDELAY field is encoded as follows:

Table 6.7 Encoding of RAILDELAY Field

Encoding	Description
0x000	1-cycle delay
0x001	2-cycle delay
0x002	3-cycle delay
0x003	4-cycle delay
0x004	5-cycle delay
.....	.....
0x3FD	1022-cycle delay
0x3FE	1023-cycle delay
0x3FF	1024-cycle delay

The default value for this register has been determined by MIPS as the value that should work in the majority of system implementations. As such, this value should not need to be changed. However, should a problem arise where additional delay is required in order to meet system timing, this register provides the programmer with the ability to increase the delay as necessary.

For more information on how this counter is used, refer to the *Global Sequence Delay Count* section in the System Integration chapter of the *P8700-F Integrator's Guide* for more information.

6.5.12.3 Reset Delay

During the power-up sequence, reset is applied. Typically, reset is active until the domain responds by asserting the internal *Reset_Hold* signal. However, the *Global Reset Width Counter* register (*CPC_RESETLEN_REG)* at offset 0x0018 allows reset to be extended beyond the assertion of *Reset_Hold*. A series of down-counters are used to delay various reset pins used to boot the CM as described in the following subsections.

The default value for this register has been determined by MIPS as the value that should work in the majority of system implementations. As such, this value should not need to be changed. However, should a problem arise where additional delay is required in order to meet system timing, this register provides the programmer with the ability to increase the delay as necessary.

For more information on these counters and the corresponding hardware signals that can be delayed, refer to the *Reset Delay* section in the P8700-F Integrator's Guide for more information.

Programming the Global Reset Width Counter Register (RESETLEN)

The RESETLEN down counter is used to extend the various reset signals using bits 9:0 of the *CPC Global Reset Width Counter Register* (*CPC_RESETLEN_REG*) at offset 0x0018. This register

field is programmed with a delay value between 1 and 1024 clock cycles as shown in Table 6.8.

Table 6.8 Encoding of the RESETLEN Field

Encoding	Description
0x000	1-cycle delay
0x001	2-cycle delay
0x002	3-cycle delay
0x003	4-cycle delay
0x004	5-cycle delay
.....	.....
0x3FD	1022-cycle delay
0x3FE	1023-cycle delay
0x3FF	1024-cycle delay

Programming the Global Reset Release Register — Core Reset Release (RESREL1)

The output of the RESETLEN counter described above is used to load a secondary internal counter with the value programmed into the RES_REL_LEN field of the CPC Global Reset Release Register (*CPC_RES_REL_REG*) located at offset 0x0038. This register is used to determine the amount of delay between the time the configuration signals are stable at the respective core(s), and the time that the core reset is released.

Bits 9:0 of this register (RES_REL_LEN) are programmed with a delay value between 1 and 1024 clock cycles. The encoding of this field is identical to the RESETLEN field shown in Table 6.8. Once this counter reaches 0, the *Domain_Reset_n*[2] signal is deasserted to the core(s), allowing them to come out of reset.

Programming the Global Reset Release Register — Domain Ready (RESREL2)

The output of the RESREL1 counter is used to load a third internal counter (RESREL2) with the value programmed into the RES_REL_LEN field of the CPC Global Reset Release Register (*CPC_RES_REL_REG*) located at offset 0x0038. This register is used to determine the amount of delay between the time the *Domain_Reset_n* signal is deasserted, and the deassertion of the *Domain_Ready* signal, indicating that the core is ready to begin execution. Note that the same register field (RES_REL_LEN) of the *CPC_RES_REL_REG* register is used to load both the RESREL1 and RESREL2 counters.

The third internal counter (RESREL2) requires that the RESREL1 counter has reached zero before counting can begin. Once the RESREL2 counter reaches 0, the *Domain_Ready* signal is asserted to the core(s), allowing the core to begin execution.

For more information on how these counters are loaded and the signals affected once the counts reach zero, refer to the *Global Sequence Delay Count* section in the *System Integration* chapter of the *P8700-F Integrator's Guide*.

2. This signal is shown only for illustration purposes. Refer to the *Global Sequence Delay Count* section of the P8700-F Integrator's Guide for more information on the usage of this signal.

Chapter 7

Control and Status Registers (CSR)

This chapter defines the following types of Control and Status Registers, or CSR's. These include:

- Section 7.1, "Machine Mode Registers"
- Section 7.2, "User Mode Registers"
- Section 7.3, "MIPS Custom Registers"
- Section 7.4, "MIPS Hybrid Debug Registers"

7.1 Machine Mode Registers

Note that some registers are in the RV64 64-bit format, while others are in the RV32 32-bit format.

Table 7.1 Machine Mode Registers

Offset from GCR_BASE	Register Name	Description
0xF12	MarchID	Machine Architecture ID register.
0x342	MCause	Machine mode Cause register.
0xF14	MHartID	Machine Hart ID register.
0xF13	MimpID	Machine Implementation ID register.
0xF11	MVendorID	Machine Vendor ID register.

7.1.1 Machine Architecture ID Register (MarchID) — offset = 0xF12

The Machine Architecture ID register (MarchID) is an implementation dependent read-only register specifying the microarchitecture version of the core. For MIPS Technologies implementations, the microarchitecture version is broken down into "class" and "uarch" versions as described below.

Figure 7.1 Machine Architecture ID Register Bit Assignments

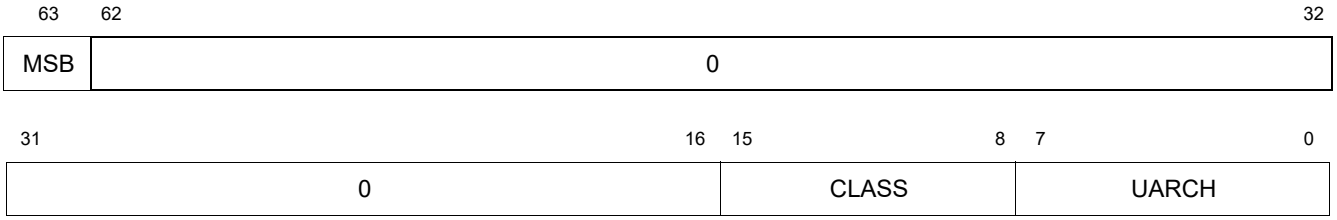


Table 7.2 Machine Architecture ID Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MSB	63	The most significant bit of the marchid register is set to one for commercial RISC-V cores, including MIPS Technologies implementations.	R	From configuration
0	62:16	Reserved	R	0
CLASS	15:8	A MIPS Technologies specific field encoding the core "class" as follows: 0x00: M-class core (alias = M) 0x01: I-class core (alias = I) 0x02: P-class core (alias = P) 0x03 - 0xFF: Reserved	R	From configuration
UARCH	7:0	A MIPS Technologies specific field encoding the core microarchitecture sub-version for the specified core class. See the core user manual for details.	R	From configuration

7.1.2 Machine Cause Register (mcause) — offset = 0x342

Machine mode Cause register. This register provides the exception code when an exception is taken.

Figure 7.2 Machine Cause Register Bit Assignments — RV64

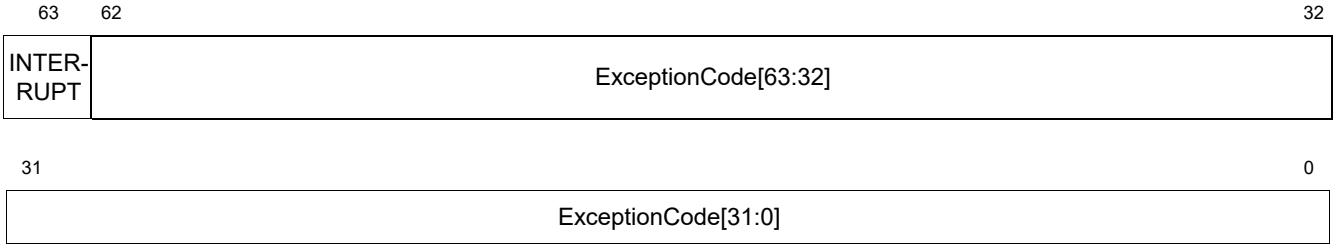


Figure 7.3 Machine Cause Register Bit Assignments — RV32

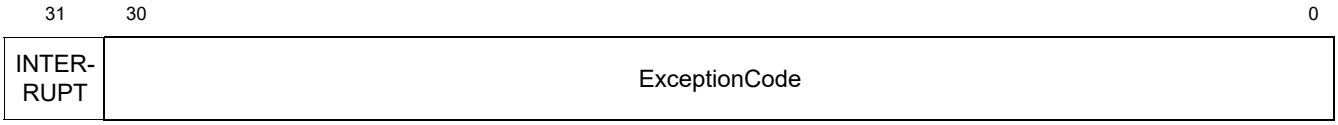


Table 7.3 Machine Cause Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
INTERRUPT	63 31	Indicates if the most resent trap to machine mode caused by an interrupt? 0: Recent trap not caused by machine mode. 1: Recent trap caused by machine mode.	R/W	Undefined
ExceptionCode	62:0 30:0	Code identifying most recent machine mode exception. The meaning of the encoding depends on type of interrupt as follows: Refer to Table 7.4 for an encoding of this field, Non-interrupt meaning. Refer to Table 7.5 for an encoding of this field, Interrupt meaning. Refer to Table 7.6 for an encoding of this field, NMI meaning.	WLRL	Undefined

Table 7.4 Exception Codes — Non-Interrupt Meaning

Encoding	Alias	Meaning
0	InstAddrMisaligned	Instruction address misaligned
1	InstAccessFault	Instruction access fault
2	IllegalInst	Illegal instruction
3	Breakpoint	Breakpoint
4	LoadAddrMisaligned	Load address misaligned
5	LoadAccessFault	Load access fault

Table 7.4 Exception Codes — Non-Interrupt Meaning *(continued)*

Encoding	Alias	Meaning
6	StoreAddrMisaligned	Store address misaligned
7	StoreAccessFault	Store access fault
8	ECallU	Environment call from U-mode
9	ECallS	Environment call from S-mode
10	ECallVS	Environment call from VS-mode
11	ECall]M	Environment call from M-mode
12	InstPageFault	Instruction page fault
13	LoadPageFault	Load page fault
15	StorePageFault	Store page fault
20	GuestInstPageFault	Guest Instruction Page Fault
21	GuestLoadPageFault	Guest Load Page Fault
22	VirtualInst	Virtual instruction Fault
23	GuestStorePageFault	Guest Store Page Fault
24	InstTLBMiss	Instruction TLB Miss Exception
25	LoadTLBMiss	Load TLB Miss Exception
26	ReadTime	Read Time Exception
27	StoreTLBMiss	Store TLB Miss Exception
28	GuestInstTLBMiss	Guest Instruction TLB Miss Exception
29	GuestLoadTLBMiss	Guest Load TLB Miss Exception
30	GuestReadTime	Guest Read Time Exception
31	GuestStoreTLBMiss	Guest Store TLB Miss Exception
48	CacheError	Cache Error exception
60	satpTLBMiss	satp TLB Miss Exception (when tlb_non_leaf implemented only).
61	vsatpTLBMiss	vsatp TLB Miss Exception (when tlb_non_leaf implemented only).
62	hgatpTLBMiss	hgatp TLB Miss Exception (when tlb_non_leaf implemented only).
63	hgatpTLBMissTW	Table Walk hgatp TLB Miss Exception (when tlb_non_leaf implemented only).

Table 7.5 Exception Codes — Interrupt Meaning

Encoding	Alias	Meaning
1	SupervisorSoftwareInt	Supervisor Software Interrupt
3	MachineSoftwareInt	Machine Software Interrupt

Table 7.5 Exception Codes — Interrupt Meaning *(continued)*

Encoding	Alias	Meaning
5	SupervisorTimerInt	Supervisor Timer Interrupt
7	MachineTimerInt	Machine Timer Interrupt
9	SupervisorExternalInt	Supervisor External Interrupt
11	MachineExternalInt	Machine External Interrupt
23	ImpreciseBusErrorInt	Imprecise Bus Error Interrupt

Table 7.6 Exception Codes — NMI Meaning

Encoding	Alias	Meaning
0	Unknown	”Unknown” NMI cause, used by MIPS Technologies RISC-V implementations.

7.1.3 Machine Hart ID Register (mhartID) — 0xF14

This read-only register contains a number uniquely identifying the hart within the system. For RISC-V systems in general, a hart with mhartid = 0 must be present, and other harts can be assigned any uniquely identifying number.

For MIPS Technologies implementations, the hartid is constructed from the number of the current clusters within the system, the number of the current cores within the current cluster, and the number of the current harts within the current core, as described below. This register is organized in the RV32 format.

Figure 7.4 Machine Hart ID Register Bit Assignments

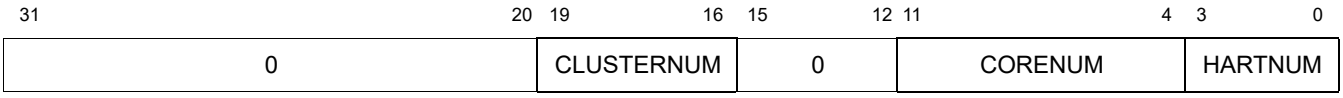


Table 7.7 Machine Hart ID Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:20	Reserved	R	
CLUSTERNUM	19:16	Cluster number. For MIPS Technologies implementations, a contiguous number starting at zero uniquely identifying the cluster in the system.	R	From configuration
0	15:12	Reserved.	R	
CORENUM	11:4	Core number. For MIPS Technologies implementations, a contiguous number starting at zero uniquely identifying the core in the cluster.	R	From configuration
HARTNUM	3:0	Hart number. For MIPS Technologies implementations, a contiguous number starting at zero uniquely identifying the hart in the core.	R	From configuration

7.1.4 Machine Implementation ID Register (mimpid) — offset = 0xF13

Machine IMPlementation ID register. mimpid is an implementation dependent read-only register specifying the implementation version of the core. For MIPS Technologies implementations, the implementation version is broken down into "major", "minor", "patch" and "config" versions as described below.

Figure 7.5 Machine Implementation ID Register Bit Assignments

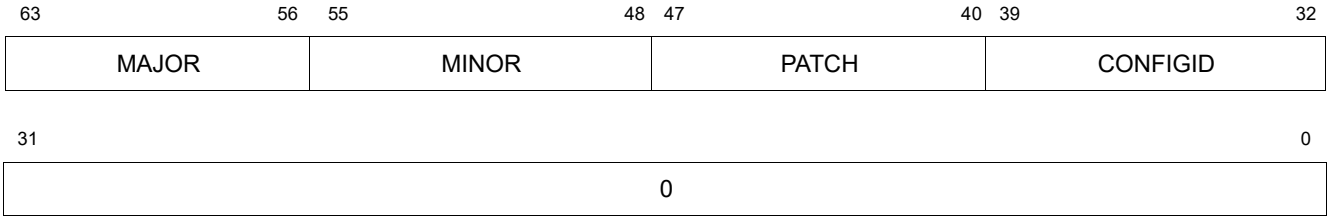


Table 7.8 Machine Implementation ID Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MAJOR	63:56	A MIPS Technologies specific field encoding the core major release version.	R	From configuration
MINOR	55:48	A MIPS Technologies specific field encoding the core minor release version.	R	From configuration
PATCH	47:40	A MIPS Technologies specific field encoding the core patch release version.	R	From configuration
CONFIGID	39:32	A MIPS Technologies specific field which identifies the core configuration. The encoding scheme for this field may vary by core type, see the core user manual for details.	R	From configuration
0	31:0	Reserved.	R	

7.1.5 Machine Vendor ID Register (mvendorid) — offset = 0xF11

This register is organized in the RV32 format.

Figure 7.6 Machine Vendor ID Register Bit Assignments

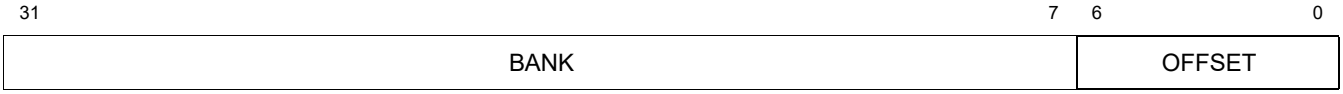


Table 7.9 Machine Vendor ID Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
BANK	31:7	Number of one-byte continuation codes in the JEDEC manufacturer ID. Equal to 0x2 for MIPS Technologies implementations.	R	From configuration
OFFSET	6:0	Final byte of JEDEC manufacturer ID, with parity bit discarded. Equal to 0x27 for MIPS Technologies implementations.	R	From configuration

7.2 User Mode Registers

The P8700 core does not implement the time register. However, this register is defined below.

7.2.1 Time Register (time) — offset = 0xC01

The time CSR provides a read-only copy of the mtime memory mapped register. In the P8700, accessing the time CSR will result in an Illegal Instruction exception where the functionality can be emulated in software.

Figure 7.7 Time Register Bit Assignments

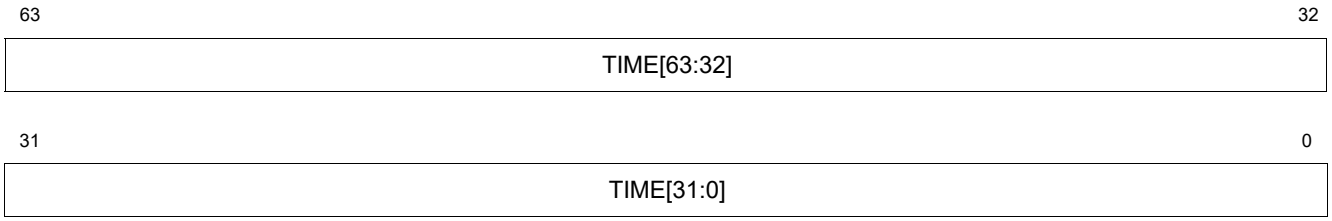


Table 7.10 Time Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
TIME	63:0	Stores a copy of the mtime memory mapped register.	R	From configuration

7.3 MIPS Custom Registers

MIPS Technologies implementations use the following custom CSRs, which are described in more detail in the following subsections. The address map for the custom CSR's is shown in Table 7.11.

Table 7.11 MIPS Custom Registers Map

Address Offset	Register Name
0x7C0	mipstvec
0x7C3	mipstval
0x7C4	mipsscratch
0x7C5	mipscacheerr
0x7C6	mipserrctl
0x7CB	mipsintctl
0x7CC	mipsdsprambase
0x7D0	mipsconfig0 [if required]
0x7D1	mipsconfig1
0x7D2	mipsconfig2 [if required]
0x7D3	mipsconfig3 [if required]
0x7D4	mipsconfig4 [if required]
0x7D5	mipsconfig5
0x7D6	mipsconfig6
0x7D7	mipsconfig7
0x7D8	mipsconfig8
0x7D9	mipsconfig9
0x7DA	mipsconfig10
0x7DB	mipsconfig11
0x7E0	pmacfg0
0x7E1	pmacfg1
0x7E2	pmacfg2
0x7E3	pmacfg3
0x7E4	pmacfg4
0x7E5	pmacfg5
0x7E6	pmacfg6
0x7E7	pmacfg7
0x7E8	pmacfg8
0x7E9	pmacfg9
0x7EA	pmacfg10

Table 7.11 MIPS Custom Registers Map *(continued)*

Address Offset	Register Name
0x7EB	pmacfg11
0x7EC	pmacfg12
0x7ED	pmacfg13
0x7EE	pmacfg14
0x7EF	pmacfg15

7.3.1 MIPS Trap Vector Base Address Register (mipstvec) — offset = 0x7C0

The MIPS Trap-VECtor base-address register is a programmable base address for custom machine mode exceptions for MIPS Technologies implementations of RISCV. An alignment constraint of HART.vectored_int_align bytes is imposed on writes to mipstvec when setting the register to vectored mode; that is, the corresponding number of lower bits of the BASE value are zeroed out by the hardware when bit zero of the written value equals 1.

This register is available in the 64-bit RV64 format and the 32-bit RV32 format.

Figure 7.8 MIPS Trap Vector Base Address Register Bit Assignments — RV64

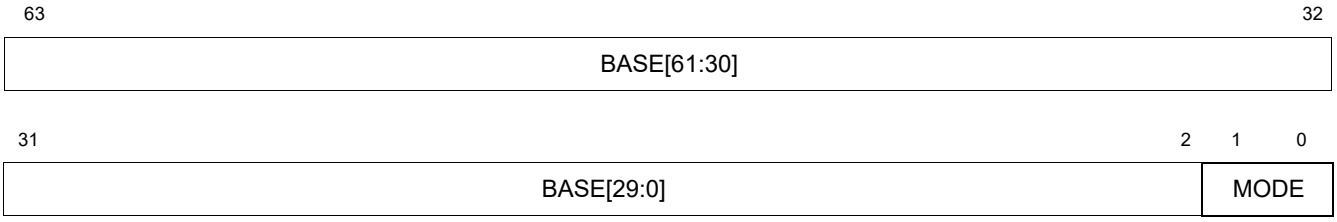


Figure 7.9 MIPS Trap Vector Base Address Register Bit Assignments — RV32

Table 7.12 MIPS Trap Vector Base Address Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
BASE	63:2 31:2	Base address for MIPS Technologies custom machine mode exceptions.	R/W	Undefined
MODE	1:0	The MODE field is encoded as follows: 0: All MIPS technologies custom machine mode exceptions set pc to CSR.mipstvec.BASE << 2. Alias = Direct. 1: MIPS technologies custom machine mode exceptions . set pc to (CSR.mipstvec.BASE << 2) + 4 * cause. Alias = vectored. 2 = 3: Reserved.	WARL	Undefined

7.3.2 MIPS Trap Value Register (mipstval) — offset = 0x7C3

MIPS Trap VALue. For cores which implement software table walk and non-leaf TLB entries, this register is written by hardware with the address of the missing PTE on a TLB miss. The register is also writable by software and can be used as a scratch register when handling mipstvec exceptions.

This register is available in the 64-bit RV64 format and the 32-bit RV32 format.

Figure 7.10 MIPS Trap Value Register Bit Assignments — RV64

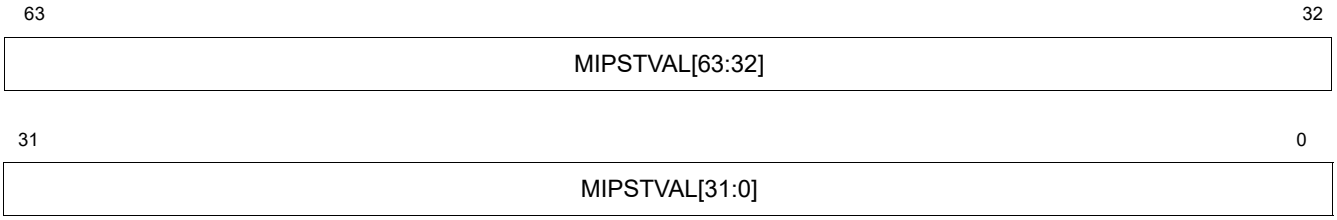


Figure 7.11 MIPS Trap Value Register Bit Assignments — RV32

Table 7.13 MIPS Trap Value Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MIPSTVAL	63:0 31:0	Trap value. This register is written by hardware with the address of the missing PTE on a TLB miss.	R/W	Undefined

7.3.3 MIPS Scratch Register (mipsscratch) — offset = 0x7C4

MIPS SCRATCH. For cores which implement software table walk and non-leaf TLB entries, this register provides scratch space for the TLB miss handler.

This register is available in the 64-bit RV64 format and the 32-bit RV32 format.

Figure 7.12 MIPS Scratch Register Bit Assignments — RV64

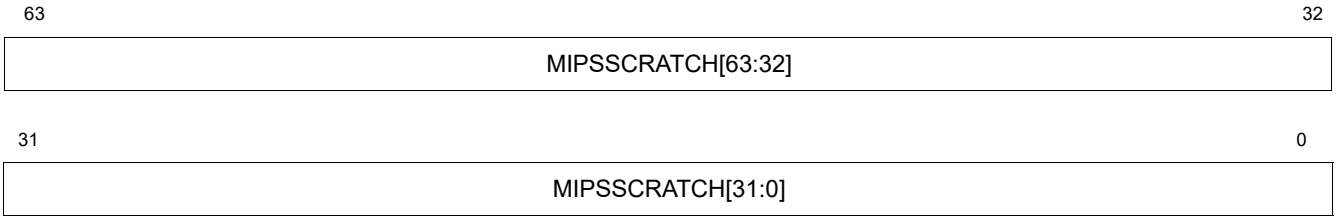


Figure 7.13 MIPS Scratch Register Bit Assignments — RV32

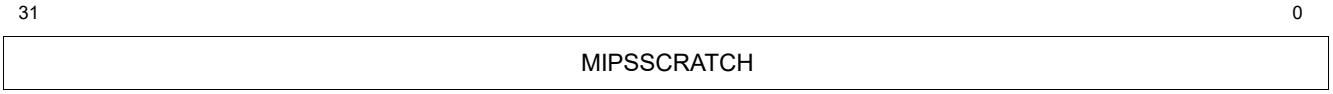


Table 7.14 MIPS Scratch Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MIPSSCRATCH	63:0 31:0	Provides scratch space for the TLB miss handler.	R/W	Undefined

7.3.4 MIPS Cache Error Register (mipscacheerr) — offset = 0x7C5

This register is implemented per-core register indicating the cause of cache errors.

Figure 7.14 MIPS Cache Error Register Bit Assignments

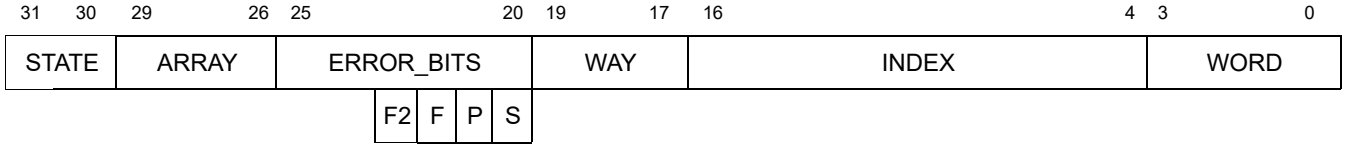


Table 7.15 MIPS Cache Error Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
STATE	31:30	Cache error state. This field is encoded as follows: 00: None. No Error 01: Corrected. Corrected Error (includes recovery by invalidating a clean line with uncorrectable error) 10: Uncorrectable error 11: Reserved	R/W	0
ARRAY	29:26	Identifies the part of the cache that encountered the error. This 4-bit field is encoded as follows: 0x0: L1 I-cache Tag. Alias = ICTag 0x1: L1 I-cache Data. Alias = ICData 0x2: L1 D-cache Tag. Alias = DCTag 0x3: L1 D-cache Data. Alias = DCData 0x4: FTLB tag. Alias = FTLB Tag. 0x5: FTLB data. Alias = FTLB Data. 0x6: L2Tag (also includes RRB bus parity). Alias = L2Tag 0x7: L2Data (also includes MCP bus parity). Alias = L2Data. 0x8 - 0xF. Reserved.	R/W	0

Table 7.15 MIPS Cache Error Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
ERROR_BITS	25:20	For correctable errors, this field encodes the bit position of the detected error within the RAM word. Encoding: 0x00 - 0x3E: Bit position of error within RAM word 0x3F: Bit position cannot be determined (when a double-bit error was "corrected" by invalidating a clean line) - corrected by invalidating the whole line.	R/W	0
	23	F2. For uncorrectable errors: Second fatal error detected while CacheErr still holds details of a previous uncorrectable/unrecoverable error (does not include cases where a double-bit error was "corrected" by invalidating a clean line).	R/W	0
	22	F. For uncorrectable errors: Fatal - Memory silently corrupted (ECC clean) (tag error on dirty replacement victim is currently the only Fatal case). Corrupted data may be present in the cache/memory subsystem with valid/clean ECC.	R/W	0
	21	P. For uncorrectable errors: Persistent error detected. A correctable (single-bit) error remained in the RAM after correction was attempted.	R/W	0
	20	S. For uncorrectable errors: Scapegoat error detected. Signaled if error was signaled on Scapegoat VP or if a second uncorrectable/unrecoverable error was detected. The error details recorded in the CacheErr register may not correspond to the instruction or thread that took the Cache Error exception. This can occur when a second uncorrectable error is detected while the CacheErr register still contains details of a previous uncorrectable error, or when an error is detected on a RAM access that cannot be attributed to a specific instruction (such as a capacity replacement).	R/W	0
WAY	19:17	Indicates the cache or FTLB way where error was detected.	R/W	0
INDEX	16:4	Indicates the cache or FTLB index where error was detected.	R/W	0
WORD	3:0	Indicates the word in the cache line (for D-cache data RAM error) where the error occurred.	R/W	0

7.3.5 MIPS Error Control Register (mipserrctrl) — offset = 0x7C6

MIPS Error Control register. This is a per-core CSR controlling bus and parity error handling.

Figure 7.15 MIPS Error Control Register Bit Assignments

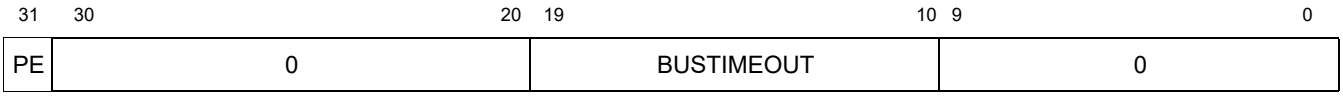


Table 7.16 MIPS Error Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PE	31	Parity enable. This bit enables or disables ECC protection for the L1 I-cache, L1 D-cache, and FTLB.	R / R/W	0
0	30:20	Reserved	R	0
BUSTIMEOUT	19:10	Timeout count. This timer can only be programmed in increments of 1024 cycles. Thus, the field available to software for programming is 19:10. If this field is written with 0, the timeout detection is disabled.	R/W	0
0	9:0	Reserved	R	0

7.3.6 MIPS Interrupt Control Register (mipsintctl) — offset = 0x7CB

MIPS Interrupt Control Register. Setting bits of this register causes the routing of selected interrupts.

Figure 7.16 MIPS Interrupt Control Register Bit Assignments

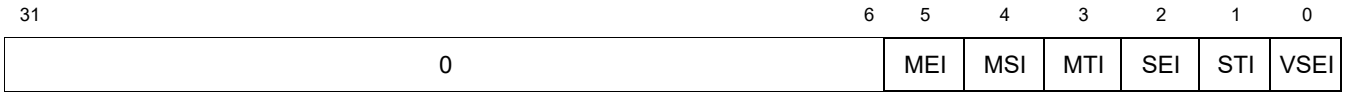


Table 7.17 MIPS Interrupt Control Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	30:6	Reserved.	R	0
MEI	5	When this bit is set, MIPS hardware interrupt #5 routes to mip.MEIP. Otherwise it routes to custom interrupt bit mip[20].	R/W	Undefined
MSI	4	When this bit is set, MIPS hardware interrupt #4 routes to mip.MSIP. Otherwise it routes to custom interrupt bit mip[19].	R/W	Undefined
MTI	3	When this bit is set, MIPS hardware interrupt #3 routes to mip.MTIP. Otherwise it routes to mip.VSTIP.	R/W	Undefined
SEI	2	When this bit is set, MIPS hardware interrupt #2 routes to mip.SEIP. Otherwise it routes to custom interrupt bit mip[18].	R/W	Undefined
STI	1	When this bit is set, MIPS hardware interrupt #1 routes to mip.STIP. Otherwise it routes to custom interrupt bit mip[17].	R/W	Undefined
VSEI	0	When this bit is set, MIPS hardware interrupt #0 routes to mip.VSEIP. Otherwise it routes to custom interrupt bit mip[16].	R/W	Undefined

7.3.7 MIPS DSPRAM Base Register (mipsdsprambase) — offset = 0x7CC

MIPS DSPRAM Base Register. Per-core register containing the base address of MIPS Technologies DSPRAM.

Figure 7.17 MIPS DSPRAM Base Register Bit Assignments

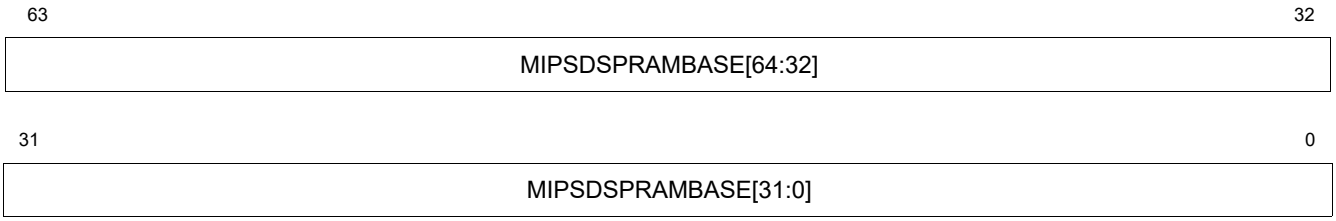


Table 7.18 MIPS DSPRAM Base Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MIPSDSPRAMBASE	63:0	Contains MIPS DSPRAM Base address in memory.	R/W	Undefined

7.3.8 MIPS Configuration 1 Register (mipsconfig1) — offset = 0x7D1

MIPS Configuration register 1. Per-core register containing collection of bitfields showing custom capabilities and status for the MIPS Technologies implementation of the RISCV standard.

Figure 7.18 MIPS Configuration 1 Register Bit Assignments

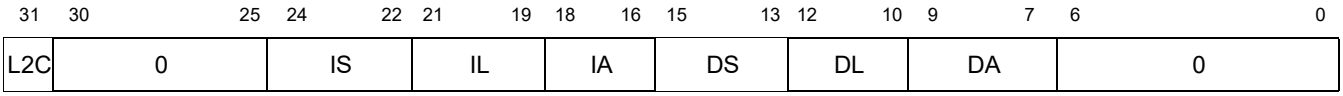


Table 7.19 MIPS Configuration 1 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
L2C	31	When this bit is set, the L2 cache exists and its size can be found via the L2_CONFIG GCR. An L3 cache may also exist and its size can be found via the L3_CONFIG GCR.	R	From configuration
0	30:25	Reserved.	R	From configuration
IS	24:22	Number of I-cache sets. Number of I-cache sets is 2**(IS+6) if IS != 7 else 32. 000: 2*6 = 12 001: 2*(6+1) = 14. etc.	R	From configuration
IL	21:19	I-cache line size. This field encodes the I-cache line size in bytes. is 0 if IL == 0 else 2**(IL+1) 000: 0 bytes 001: 2*2 = 4 bytes 010: 2*3 = 6 bytes 011: 2*4 = 8 bytes 100: 2*5 = 10 bytes	R	From configuration
IA	18:16	I-cache Associativity. Number of I-cache ways is IA + 1. 000: 1-way 001: 2-way 010: 3-way 011: 4-way 100: 5-way 101: 6=way 110: 7-way 111: 8-way	R	From configuration
DS	15:13	D-cache Sets. Number of D-cache sets is 2**(DS+6) if DS != 7 else 32.	R	From configuration
DL	12:10	D-cache Line size. D-cache line size in bytes is 0 if DL == 0 else 2**(DL+1)	R	From configuration
DA	9:7	D-cache Associativity. Number of D-cache ways is DA + 1.	R	From configuration
0	6:0	Reserved	R	From configuration

7.3.9 MIPS Configuration 5 Register (mipsconfig5) — offset = 0x7D5

MIPS Configuration register 5. Per-hart register containing collection of bit fields showing custom capabilities and status for the MIPS Technologies implementation of the RISCV standard.

Figure 7.19 MIPS Configuration 5 Register Bit Assignments

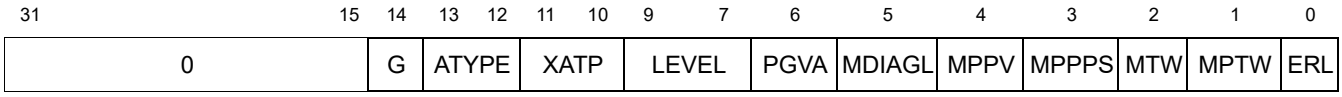


Table 7.20 MIPS Configuration 5 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:15	Reserved	R	
G	14	When set, this bit indicates that the most recent TLB miss was on a Global non-leaf page. This bit is R/W if the software table walker and non-leaf TLB entries are implemented.	R/W	Undefined
ATYPE	13:12	This field indicates the most recent type of TLB miss and is encoded as follows: 00: Instruction TLB miss. Alias is 'Inst'. 01: Load TLB miss. Alias is 'Load'. 10: Reserved. 11: Store TLB miss. Alias is 'Store'. This bit is R/W if the software table walker and non-leaf TLB entries are implemented.	R/W	Undefined
XATP	11:10	Which (X) Address Translation and Protection. Specifies which of SATP, VSATP or HGATP is to be targeted by MTLBWR instruction. Set by hardware on a TLB miss exception to type of translation which has missed. This field is implemented on cores with a software table walker and non-leaf TLB entries.This field is encoded as follows: 00: SATP CSR is targeted. Alias os 'SATP'. 01: VSATP CSR is targeted. Alias os 'VSATP'. 10: HGATP CSR is targeted. Alias os 'HGATP'. 11: Reserved.	R/W	Undefined
LEVEL	9:7	PTE Level. Specifies the page table level to be written by the MTLBWR instruction. This field is set by hardware on a TLB miss exception to the level of the page table walk has missed. Implemented on cores with software table walker and non-leaf TLB entries. Also writable by software, only levels used by the supported table walker modes are legal (e.g. 0..3 for Sv48). This field is WARL if software table walker and non-leaf TLB entries are implemented.	WARL	Undefined

Table 7.20 MIPS Configuration 5 Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
PGVA	6	Previous Guest Physical Address. This bit is copied from mstatus.GVA on M-mode exceptions using the mipstvec exception vector, or M-mode exceptions when mipsconfig5.MTW = 1. When PGVA is set, MRET behavior is modified to set mstatus.GVA to 1 instead of 0. Implemented on H-extension cores with software table walker only.	R/W	0
MDIAGL	5	MDIAG lock. Software can write this bit to 1 to permanently disable the MDIAGR/MDIAGW instructions. Can only be unlocked by a CPU reset.	W1Only	0
MPPV	4	Machine Previous-Previous Virtualization Mode - Set to 1 on mipstvec exceptions if mstatus.MPV is 1, or on other M-mode exceptions if mipsconfig5.MTW is 1 and mstatus.MPV is 1. When MPPV is set, MRET behavior is modified to set mstatus.MPV to 1 instead of 0. Implemented on H-extension cores with software table walker only.	R/W	0
MPPPS	3	Machine Previous-Previous Privilege Supervisor - Set to 1 on mipstvec exceptions if mstatus.MPP is 1 (supervisor), or on other M-mode exceptions when mipsconfig5.MTW=1 and mstatus.MPP is 1. When MPPPS = 1, MRET behavior is modified to set mstatus.MPP to 1 instead of 0. Implemented on cores with software table walker only.	R/W	0
MTW	2	Machine Table Walk. Setting this bit forces M-mode loads and stores to execute with table walker mapping and privilege. Cleared by M-mode traps and restored from MPTW by MRET. Implemented on cores with software table walker only.	R/W	0
MPTW	1	Machine Previous Table Walk. This bit contains the value of the mipsconfig5.MTW bit prior to the most recent M-mode trap, restored to MTW by MRET. Implemented on cores with software table walker only.	R/W	0
ERL	0	Error Level. This bit is set on NMI and Cache Error exceptions. Cleared by MRET and SRET instructions. Forces all memory accesses to be uncached and disables all interrupts except for Reset and NMI.	R/W	0

7.3.10 MIPS Configuration 6 Register (mipsconfig6) — offset = 0x7D6

MIPS Configuration register 6. Per-hart register containing collection of bit fields showing custom capabilities and status for the MIPS Technologies implementation of the RISCV standard.

Figure 7.20 MIPS Configuration 6 Register Bit Assignments

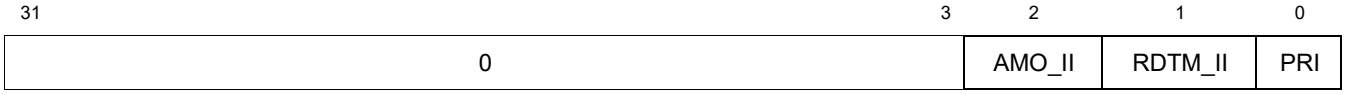


Table 7.21 MIPS Configuration 6 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:3	Reserved	R	
AMO_II	2	Atomic Memory Operation Illegal Instruction. When set, executing one of the AMO* instructions on a "no_amo" core gives an illegal instruction exception. Otherwise, cases where the AMO instruction does not generate any addressing related exceptions (page faults, TLB misses or access faults) give a custom mipstvec exception with mcause set to the Illegal Instruction value, allowing for fast emulation of the atomic memory operation. LR/SC instructions are not affected by this bit.	R/W	0
RDTM_II	1	Read Time Illegal Instruction. When set, reading from the time CSR always gives an illegal instruction exception. Otherwise, cases where reading from the time CSR is enabled at the current privilege level give a custom "Read Time" mipstvec exception with mcause set to 26, allowing for fast emulation of the rdtime operation.	R/W	0
PRI	0	Hart has priority for MCP accesses.	R/W	0

7.3.11 MIPS Configuration 7 Register (mipsconfig7) — offset = 0x7D7

MIPS Configuration register 7. Per-hart register containing collection of bit fields showing custom capabilities and status for the MIPS Technologies implementation of the RISCV standard.

Figure 7.21 MIPS Configuration 7 Register Bit Assignments

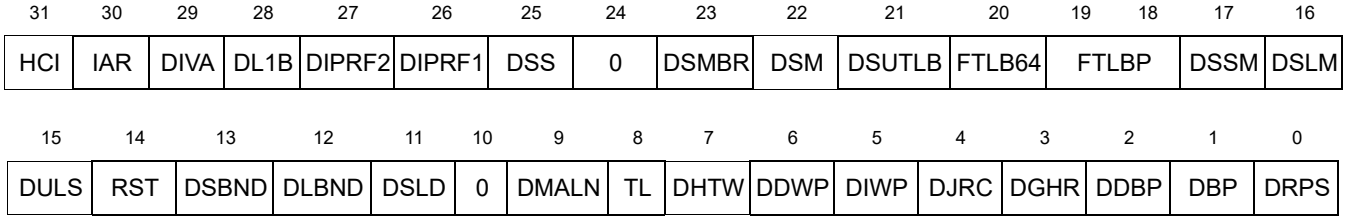


Table 7.22 MIPS Configuration 7 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
HCI	31	When set by hardware, Hardware Cache Initialization is present.	R	From configuration
IAR	30	Instruction Alias Removed.	R/W	0
DIVA	29	Disable Instruction Virtual Aliasing. Setting this bit disables the hardware alias removal on the instruction cache. If this bit is cleared, alias removal is not disabled.	R/W	0
DL1B	28	When this bit is set, the L1 Branch Target Buffer is disabled. 0: Enabled 1: Disabled	R/W	0
DIPRF2	27	When this bit is set, disable Icache PReFetch to line plus 2. 0: Enabled 1: Disabled	R/W	0
DIPRF1	26	When this bit is set, disable Icache PReFetch to line plus 1. 0: Enabled 1: Disabled	R/W	0
DSS	25	Disable Streaming Stores. When supported, setting this bit disables the performance enhancement which avoids filling the cache from memory for sequential stores which modify all bytes in cache line.	R/W	0
0	24	Reserved.	R/W	0
DSMBR	23	When this bit is set, disable Sleep Mode when a long bus transaction is pending. 0: Enabled 1: Disabled	R/W	0
DSM	22	When this bit is set, disable Sleep Mode. 0: Enabled 1: Disabled	R/W	0
DSUTLB	21	When this bit is set, disable speculative handling of uTLB misses. 0: Enabled 1: Disabled	R/W	0

Table 7.22 MIPS Configuration 7 Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
FTLB64	20	FTLB holds 64KB pages. In implementations where the FTLB cannot hold 4KB pages and 64KB pages simultaneously, software can set this bit to 1 to indicate that 64KB pages are expected to be more common and should be stored in the FTLB (with 4KB pages stored in the VTLB). In such implementations, if this bit is zero (the reset value) 4KB pages will be stored in the FTLB and 64KB pages will be stored in the VTLB. In implementations where the FTLB can hold 4KB and 64KB pages simultaneously, this bit is reserved. 0: 64KB pages stored to FTLB, 4 KB pages stored to VTLB. 1: 64KB pages stored to VTLB, 4 KB pages stored to FTLB.	R/W	0
FTLBP	19:18	FTLB Probability. Probability that a TLB write which can go to the FTLB is sent to the VTLB. This field is encoded as follows: 00: 1 out of 64 writes which could go to the FTLB goes to the VTLB, 01: 1 out of 32 FTLB writes goes to the VTLB. 10: 1 out of 16 FTLB writes goes to the VTLB. 11: No FTLB writes go to the VTLB.	R/W	0
DSSM	17	When this bit is set, disable speculative bus fetch requests for a store miss.	R/W	0
DSLM	16	When this bit is set, disable speculative bus fetch requests for a load miss.	R/W	0
DULS	15	When this bit is set, disable unaligned load/stores.	R/W	0
RST	14	Reset TAGE. A 0 -> 1 transition of this bit causes the TAGE branch prediction unit to be reset. To reset TAGE again, rewrite this bit to 0, then set it to 1 again.	R/W	0
DSBND	13	When this bit is set, store bonding is disabled. 0: Enabled 1: Disabled	R/W	0
DLBND	12	When this bit is set, load bonding is disabled. 0: Enabled 1: Disabled	R/W	0
DSLD	11	When this bit is set, disable the speculative issue of instructions that consume the result of a load.	R/W	0
0	10	Reserved.	R/W	0
DMALN	9	Disable misaligned load/store. When set, all misaligned accesses generate an address misaligned exception.	R/W	0
TL	8	When this bit is set, MIPS Trace Logic is implemented. 0: Not implemented 1: Implemented	R	From configuration
DHTW	7	When this bit is set, disable the hardware table walker.	R/W	0
DDWP	6	When this bit is set, disable data cache way prediction.	R/W	0
D1WP	5	When this bit is set, disable instruction cache way prediction.	R/W	0
DJRC	4	When this bit is set, disable the Jump Register Cache.	R/W	0
DGHR	3	When this bit is set, disable the branch history table.	R/W	0

Table 7.22 MIPS Configuration 7 Register Bit Descriptions *(continued)*

Name	Bits	Description	R/W	Reset State
DDBP	2	When this bit is set, disable dynamic branch prediction.	R/W	0
DBP	1	When this bit is set, disable branch prediction.	R/W	0
DRPS	0	When this bit is set, disable the return prediction stack.	R/W	0

7.3.12 MIPS Configuration 8 Register (mipsconfig8) — offset = 0x7D8

MIPS Configuration register 8. Per-core register containing collection of bitfields showing custom capabilities and status for MIPS Technologies implementations of RISCV. This register selects the Prefetch tracker based on GS, ES and PTRS bits.

This register provides the indexing for indirect access of the registers present in Prefetch tracker which can be accessed indirectly using mipsconfig9 and mipsconfig10 CSRs.

Figure 7.22 MIPS Configuration 8 Register Bit Assignments

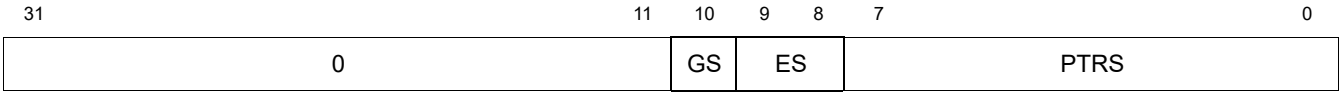


Table 7.23 MIPS Configuration 8 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:11	Reserved	R	
GS	10	Group Select	R/W	Undefined
ES	9:8	Entry Select.	R/W	Undefined
PTRS	7:0	Prefetch tracker register select. This field is encoded as follows: 0x00: Mismatch counter (Alias = Mismatch) 0x01: Mismatch/Confidence counter (Alias = MismatchConfidence) 0x02: Age counter (Alias = Age) 0x03: Learn stride counter (Alias = LearnStride) 0x04: Age prescaler counter (AgePrescaler) 0x05 - 0xFF: Reserved.	R/W	Undefined

7.3.13 MIPS Configuration 9 Register (mipsconfig9) — offset = 0x7D9

MIPS Configuration register 9. Per-core register containing collection of bitfields showing custom capabilities and status for MIPS Technologies implementations of RISCV. This register contains data to be indirectly written into the Prefetch tracker registers.

Figure 7.23 MIPS Configuration 9 Register Bit Assignments

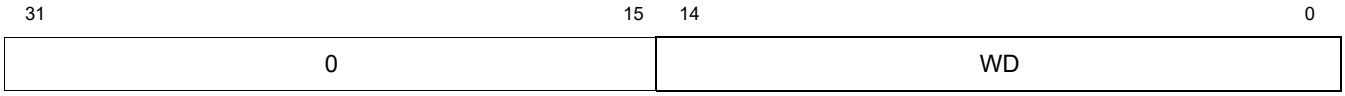


Table 7.24 MIPS Configuration 9 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:15	Reserved	R	
WD	14:0	Write Data. Based on the Register select bits, entry, and group select bits, the following bits are updated: Mismatch Counter 2:0 Mismatch / Confidence Counter 3:0 Age Counter 5:0 Learn Stride Counter 14:0 Age Prescaler Counter 5:0	R/W	Undefined

7.3.14 MIPS Configuration 10 Register (mipsconfig10) — offset = 0x7DA

MIPS Configuration register 10. Per-core register containing collection of bitfields showing custom capabilities and status for MIPS Technologies implementations of RISCV. This register contains data to be indirectly read from the Prefetch tracker registers.

Figure 7.24 MIPS Configuration 10 Register Bit Assignments

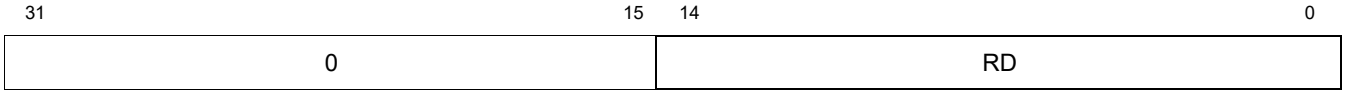


Table 7.25 MIPS Configuration 10 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:15	Reserved	R	
RD	14:0	Read Data. Based on the Register select bits, entry, and group select bits, the following bits are updated: Mismatch Counter 2:0 Mismatch / Confidence Counter 3:0 Age Counter 5:0 Learn Stride Counter 14:0 Age Prescaler Counter 5:0	R/W	Undefined

7.3.15 MIPS Configuration 11 Register (mipsconfig11) — offset = 0x7DB

MIPS Configuration register 11. Per-core register containing collection of bitfields showing custom capabilities and status for MIPS Technologies implementations of RISCV. This register contains data to be indirectly read from the Prefetch tracker registers.

Figure 7.25 MIPS Configuration 11 Register Bit Assignments

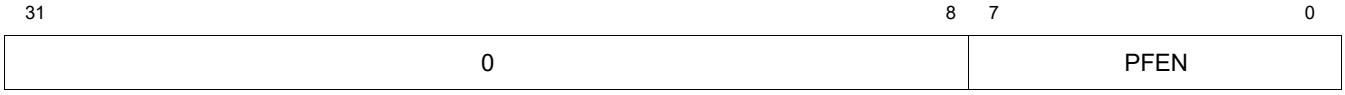


Table 7.26 MIPS Configuration 11 Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:8	Reserved	R	
PFEN	7:0	Prefetch Enable. When set, the prefetch tracker is enabled, else it is disabled. **What is the encoding of this field?	R/W	0

7.3.16 PMA Configuration Registers

This section contains 16 PMA configuration registers that store a total of 64 PMA configurations. Each PMAxCFG configuration listed below is represented by an 8-bit field. The encoding of each of these fields is identical and is described in Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

Table 7.27 shows the address mapping for each of these 16 registers.

For example, if the RV64 format is used, the PMACFG0 register is 64 bits and contains fields PMA7CFG - PMA0CFG. In this case PMA3CFG - PMA0CFG are in the lower 32 bits, and PMA7CFG - PMA4CFG are in the upper 32 bits.

If the RV32 format is used, the PMACFG0 register is 32 bits and contains fields PMA3CFG - PMA0CFG. In this case, the upper 32-bit value does not exist. As such, two registers at two contiguous addresses are required to make a 64-bit value as shown in the table below. For example, in the RV-32 format, the lower 32 bits are in the PMACFG0 register, and the upper 32 bits are in the PMACFG1 register.

Table 7.27 PMA Configuration Register Address Mapping

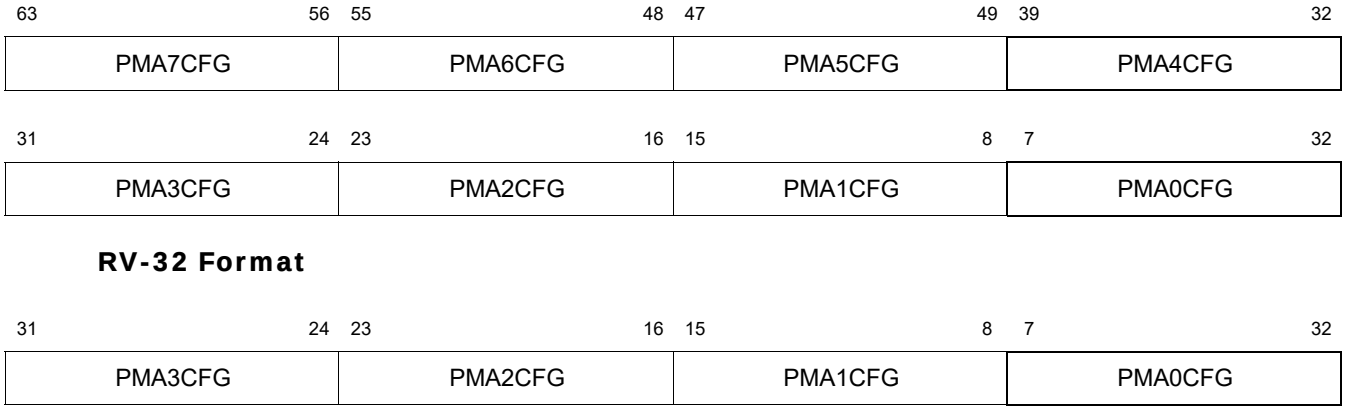
Address Offset	Register Name	Format	
		RV64	RV32
0x7E0	PMACFG0	PMA7CFG - PMA0CFG	PMA3CFG - PMA0CFG
0x7E1	PMACFG1	---------	PMA7CFG - PMA4CFG
0x7E2	PMACFG2	PMA15CFG - PMA8CFG	PMA11CFG - PMA8CFG
0x7E3	PMACFG3	---------	PMA15CFG - PMA12CFG
0x7E4	PMACFG4	PMA23CFG - PMA16CFG	PMA19CFG - PMA16CFG
0x7E5	PMACFG5	---------	PMA23CFG - PMA20CFG
0x7E6	PMACFG6	PMA31CFG - PMA24CFG	PMA27CFG - PMA24CFG
0x7E7	PMACFG7	---------	PMA31CFG - PMA28CFG
0x7E8	PMACFG8	PMA39CFG - PMA32CFG	PMA35CFG - PMA32CFG
0x7E9	PMACFG9	---------	PMA39CFG - PMA36CFG
0x7EA	PMACFG10	PMA47CFG - PMA40CFG	PMA43CFG - PMA40CFG
0x7EB	PMACFG11	---------	PMA47CFG - PMA44CFG
0x7EC	PMACFG12	PMA55CFG - PMA48CFG	PMA51CFG - PMA48CFG
0x7ED	PMACFG13	---------	PMA55CFG - PMA52CFG
0x7EE	PMACFG14	PMA63CFG - PMA56CFG	PMA59CFG - PMA56CFG
0x7EF	PMACFG15	---------	PMA63CFG - PMA60CFG

7.3.16.1 PMA Configuration 0 Control and Status Register (PMACFG0) — offset = 0x7E0

PMA Configuration register 0. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.26 PMA Configuration 0 Control and Status Register Bit Assignments



RV-32 Format

Table 7.28 PMA Configuration 0 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA7CFG	63:56	PMA7 configuration field in the RV-64 format.	R/W	Undefined
PMA6CFG	55:48	PMA6 configuration field in the RV-64 format.	R/W	Undefined
PMA5CFG	47:40	PMA5 configuration field in the RV-64 format.	R/W	Undefined
PMA4CFG	39:32	PMA4 configuration field in the RV-64 format.	R/W	Undefined
PMA3CFG	31:24	PMA3 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA2CFG	23:16	PMA2 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA1CFG	15:8	PMA1 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA0CFG	7:0	PMA0 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.2 PMA Configuration 1 Control and Status Register (PMACFG1) — offset = 0x7E1

PMA Configuration register 1. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.27 PMA Configuration 1 Control and Status Register Bit Assignments

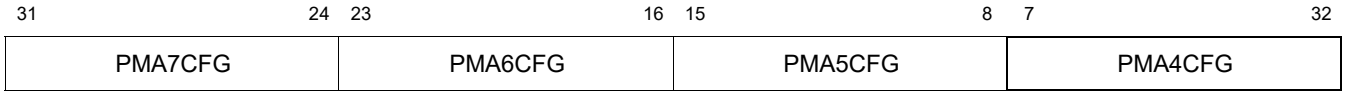


Table 7.29 PMA Configuration 1 Control and Status Register Bit Descriptions

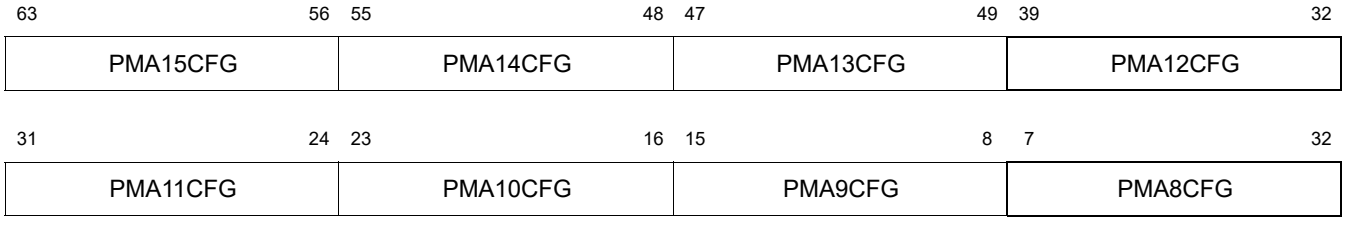
Name	Bits	Description	R/W	Reset State
PMA7CFG	31:24	PMA7 configuration field in the RV-32 format.	R/W	Undefined
PMA6CFG	23:16	PMA6 configuration field in the RV-32 format.	R/W	Undefined
PMA5CFG	15:8	PMA5 configuration field in the RV-32 format.	R/W	Undefined
PMA4CFG	7:0	PMA4 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.3 PMA Configuration 2 Control and Status Register (PMACFG2) — offset = 0x7E2

PMA Configuration register 2. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.28 PMA Configuration 2 Control and Status Register Bit Assignments



RV-32 Format

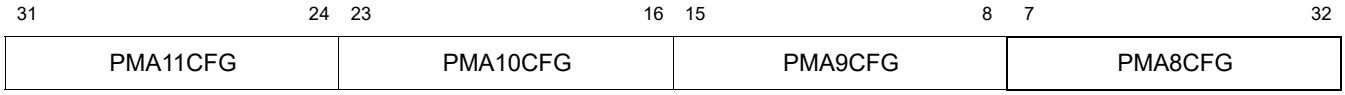


Table 7.30 PMA Configuration 2 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA15CFG	63:56	PMA15 configuration field in the RV-64 format.	R/W	Undefined
PMA14CFG	55:48	PMA14 configuration field in the RV-64 format.	R/W	Undefined
PMA13CFG	47:40	PMA13 configuration field in the RV-64 format.	R/W	Undefined
PMA12CFG	39:32	PMA12 configuration field in the RV-64 format.	R/W	Undefined
PMA11CFG	31:24	PMA11 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA10CFG	23:16	PMA10 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA9CFG	15:8	PMA9 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA8CFG	7:0	PMA8 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.4 PMA Configuration 3 Control and Status Register (PMACFG3) — offset = 0x7E3

PMA Configuration register 3. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.29 PMA Configuration 3 Control and Status Register Bit Assignments

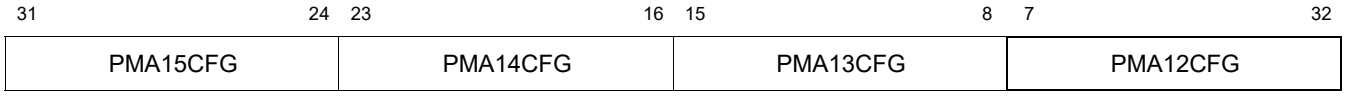


Table 7.31 PMA Configuration 3 Control and Status Register Bit Descriptions

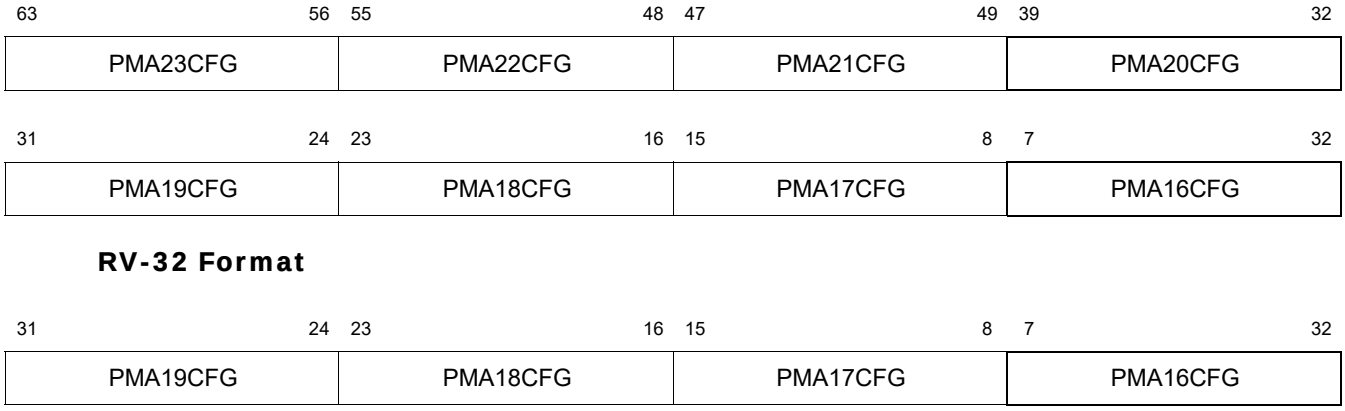
Name	Bits	Description	R/W	Reset State
PMA15CFG	31:24	PMA15 configuration field in the RV-32 format.	R/W	Undefined
PMA14CFG	23:16	PMA14 configuration field in the RV-32 format.	R/W	Undefined
PMA13CFG	15:8	PMA13 configuration field in the RV-32 format.	R/W	Undefined
PMA12CFG	7:0	PMA12 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.5 PMA Configuration 4 Control and Status Register (PMACFG4) — offset = 0x7E4

PMA Configuration register 4. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.30 PMA Configuration 4 Control and Status Register Bit Assignments



RV-32 Format

Table 7.32 PMA Configuration 4 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA23CFG	63:56	PMA23 configuration field in the RV-64 format.	R/W	Undefined
PMA22CFG	55:48	PMA22 configuration field in the RV-64 format.	R/W	Undefined
PMA21CFG	47:40	PMA21 configuration field in the RV-64 format.	R/W	Undefined
PMA20CFG	39:32	PMA20 configuration field in the RV-64 format.	R/W	Undefined
PMA19CFG	31:24	PMA19 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA18CFG	23:16	PMA18 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA17CFG	15:8	PMA17 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA16CFG	7:0	PMA16 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.6 PMA Configuration 5 Control and Status Register (PMACFG5) — offset = 0x7E5

PMA Configuration register 5. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.31 PMA Configuration 5 Control and Status Register Bit Assignments

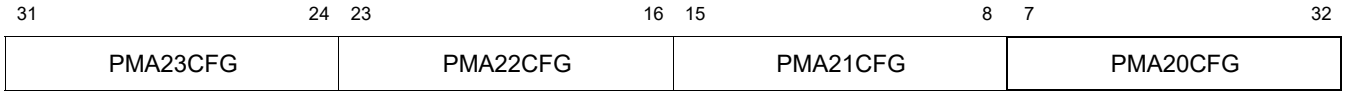


Table 7.33 PMA Configuration 5 Control and Status Register Bit Descriptions

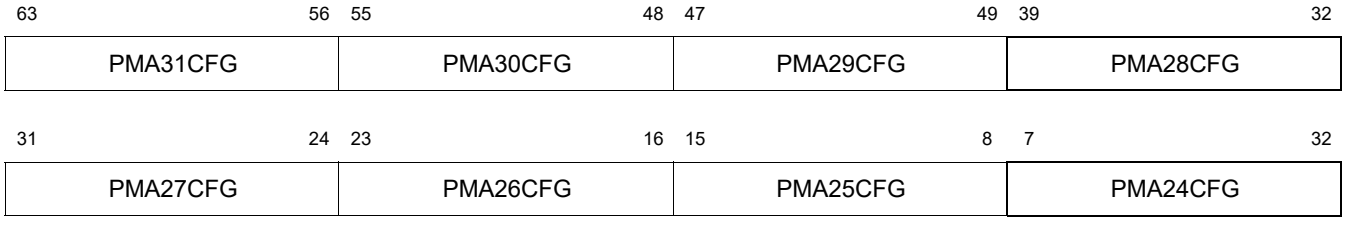
Name	Bits	Description	R/W	Reset State
PMA23CFG	31:24	PMA23 configuration field in the RV-32 format.	R/W	Undefined
PMA22CFG	23:16	PMA22 configuration field in the RV-32 format.	R/W	Undefined
PMA21CFG	15:8	PMA21 configuration field in the RV-32 format.	R/W	Undefined
PMA20CFG	7:0	PMA20 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.7 PMA Configuration 6 Control and Status Register (PMACFG6) — offset = 0x7E6

PMA Configuration register 6. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.32 PMA Configuration 6 Control and Status Register Bit Assignments



RV-32 Format

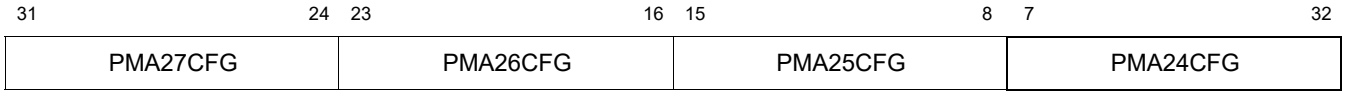


Table 7.34 PMA Configuration 6 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA31CFG	63:56	PMA31 configuration field in the RV-64 format.	R/W	Undefined
PMA30CFG	55:48	PMA30 configuration field in the RV-64 format.	R/W	Undefined
PMA29CFG	47:40	PMA29 configuration field in the RV-64 format.	R/W	Undefined
PMA28CFG	39:32	PMA28 configuration field in the RV-64 format.	R/W	Undefined
PMA27CFG	31:24	PMA27 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA26CFG	23:16	PMA26 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA25CFG	15:8	PMA25 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA24CFG	7:0	PMA24 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.8 PMA Configuration 7 Control and Status Register (PMACFG7) — offset = 0x7E7

PMA Configuration register 7. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.33 PMA Configuration 7 Control and Status Register Bit Assignments

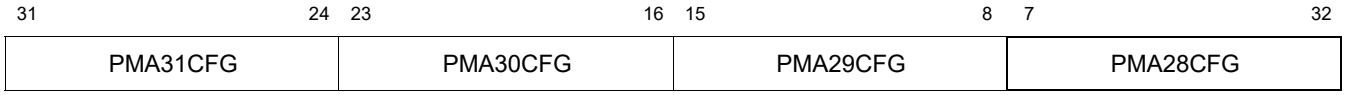


Table 7.35 PMA Configuration 7 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA31CFG	31:24	PMA31 configuration field in the RV-32 format.	R/W	Undefined
PMA30CFG	23:16	PMA30 configuration field in the RV-32 format.	R/W	Undefined
PMA29CFG	15:8	PMA29 configuration field in the RV-32 format.	R/W	Undefined
PMA28CFG	7:0	PMA28 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.9 PMA Configuration 8 Control and Status Register (PMACFG8) — offset = 0x7E8

PMA Configuration register 8. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.34 PMA Configuration 8 Control and Status Register Bit Assignments

RV-32 Format

Table 7.36 PMA Configuration 8 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA39CFG	63:56	PMA39 configuration field in the RV-64 format.	R/W	Undefined
PMA38CFG	55:48	PMA38 configuration field in the RV-64 format.	R/W	Undefined
PMA37CFG	47:40	PMA37 configuration field in the RV-64 format.	R/W	Undefined
PMA36CFG	39:32	PMA36 configuration field in the RV-64 format.	R/W	Undefined
PMA35CFG	31:24	PMA35 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA34CFG	23:16	PMA34 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA33CFG	15:8	PMA33 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA32CFG	7:0	PMA32 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.10 PMA Configuration 9 Control and Status Register (PMACFG9) — offset = 0x7E9

PMA Configuration register 9. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.35 PMA Configuration 9 Control and Status Register Bit Assignments

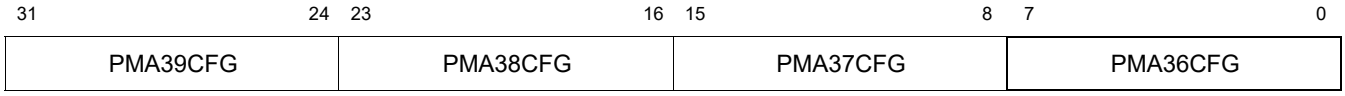


Table 7.37 PMA Configuration 9 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA39CFG	31:24	PMA39 configuration field in the RV-32 format.	R/W	Undefined
PMA38CFG	23:16	PMA38 configuration field in the RV-32 format.	R/W	Undefined
PMA37CFG	15:8	PMA37 configuration field in the RV-32 format.	R/W	Undefined
PMA36CFG	7:0	PMA36 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.11 PMA Configuration 10 Control and Status Register (PMACFG10) — offset = 0x7EA

PMA Configuration register 10. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.36 PMA Configuration 10 Control and Status Register Bit Assignments

RV-32 Format

Table 7.38 PMA Configuration 10 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA47CFG	63:56	PMA47 configuration field in the RV-64 format.	R/W	Undefined
PMA46CFG	55:48	PMA46 configuration field in the RV-64 format.	R/W	Undefined
PMA45CFG	47:40	PMA45 configuration field in the RV-64 format.	R/W	Undefined
PMA44CFG	39:32	PMA44 configuration field in the RV-64 format.	R/W	Undefined
PMA43CFG	31:24	PMA43 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA42CFG	23:16	PMA42 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA41CFG	15:8	PMA41 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA40CFG	7:0	PMA40 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.12 PMA Configuration 11 Control and Status Register (PMACFG11) — offset = 0x7EB

PMA Configuration register 11. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.37 PMA Configuration 11 Control and Status Register Bit Assignments

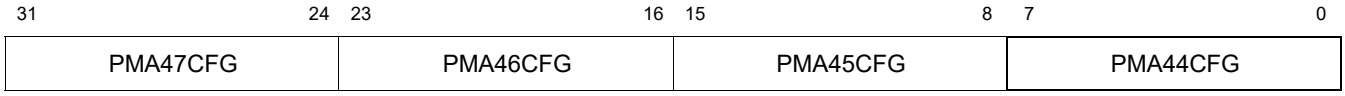


Table 7.39 PMA Configuration 11 Control and Status Register Bit Descriptions

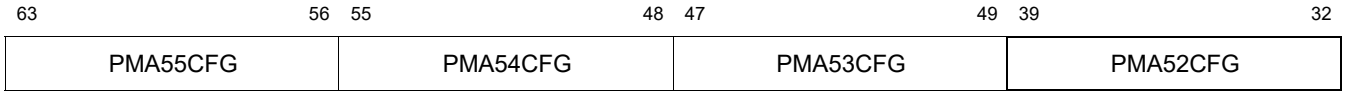
Name	Bits	Description	R/W	Reset State
PMA47CFG	31:24	PMA47 configuration field in the RV-32 format.	R/W	Undefined
PMA46CFG	23:16	PMA46 configuration field in the RV-32 format.	R/W	Undefined
PMA45CFG	15:8	PMA45 configuration field in the RV-32 format.	R/W	Undefined
PMA44CFG	7:0	PMA44 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.13 PMA Configuration 12 Control and Status Register (PMACFG12) — offset = 0x7EC

PMA Configuration register 12. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.38 PMA Configuration 12 Control and Status Register Bit Assignments



RV-32 Format

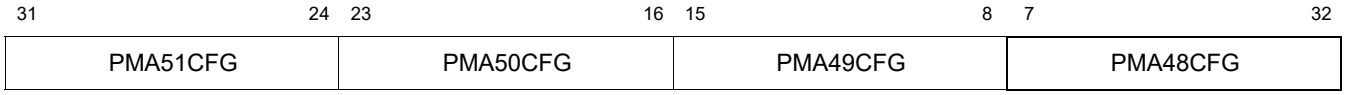


Table 7.40 PMA Configuration 12 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA55CFG	63:56	PMA55 configuration field in the RV-64 format.	R/W	Undefined
PMA54CFG	55:48	PMA54 configuration field in the RV-64 format.	R/W	Undefined
PMA53CFG	47:40	PMA53 configuration field in the RV-64 format.	R/W	Undefined
PMA52CFG	39:32	PMA52 configuration field in the RV-64 format.	R/W	Undefined
PMA51CFG	31:24	PMA51 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA50CFG	23:16	PMA50 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA49CFG	15:8	PMA49 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA48CFG	7:0	PMA48 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.14 PMA Configuration 13 Control and Status Register (PMACFG13) — offset = 0x7ED

PMA Configuration register 13. This register is only used in the RV32 format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.39 PMA Configuration 13 Control and Status Register Bit Assignments

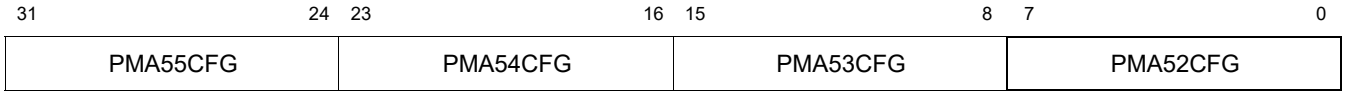


Table 7.41 PMA Configuration 13 Control and Status Register Bit Descriptions

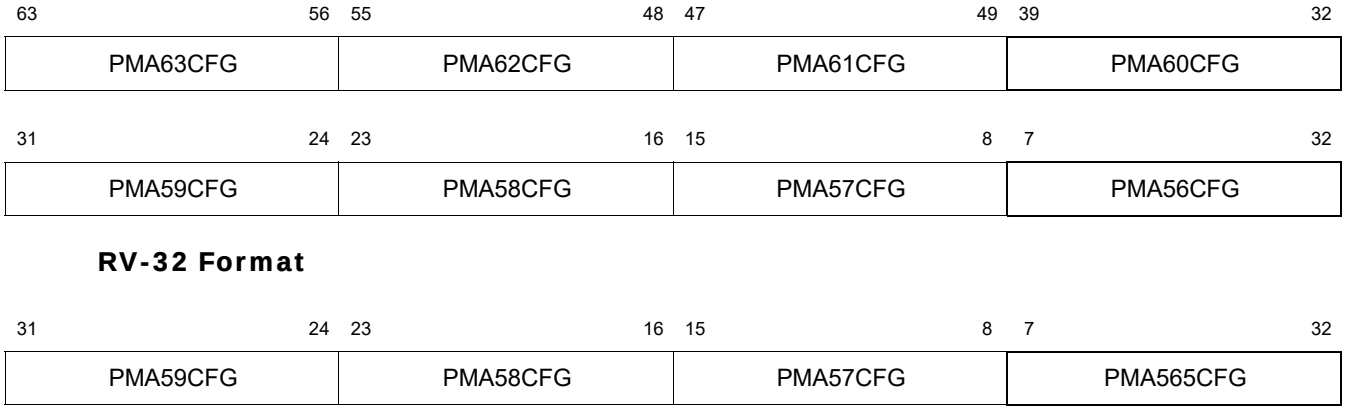
Name	Bits	Description	R/W	Reset State
PMA55CFG	31:24	PMA55 configuration field in the RV-32 format.	R/W	Undefined
PMA54CFG	23:16	PMA54 configuration field in the RV-32 format.	R/W	Undefined
PMA53CFG	15:8	PMA53 configuration field in the RV-32 format.	R/W	Undefined
PMA52CFG	7:0	PMA52 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.15 PMA Configuration 14 Control and Status Register (PMACFG14) — offset = 0x7EE

PMA Configuration register 14. This register is either 64 or 32 bits depending on the RV format. For the encoding of each field in this register, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-64 Format

Figure 7.40 PMA Configuration 14 Control and Status Register Bit Assignments



RV-32 Format

Table 7.42 PMA Configuration 14 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA63CFG	63:56	PMA63 configuration field in the RV-64 format.	R/W	Undefined
PMA62CFG	55:48	PMA62 configuration field in the RV-64 format.	R/W	Undefined
PMA61CFG	47:40	PMA61 configuration field in the RV-64 format.	R/W	Undefined
PMA60CFG	39:32	PMA60 configuration field in the RV-64 format.	R/W	Undefined
PMA58CFG	31:24	PMA59 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA58CFG	23:16	PMA58 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA57CFG	15:8	PMA57 configuration field in the RV-64 or RV-32 format.	R/W	Undefined
PMA56CFG	7:0	PMA56 configuration field in the RV-64 or RV-32 format.	R/W	Undefined

7.3.16.16 PMA Configuration 15 Control and Status Register (PMACFG15) — offset = 0x7EF

PMA Configuration register 15. This register is only used in the RV32 format. For the encoding of each field, refer to Section 7.3.16.17, "PMA0CFG - PMA63CFG Bit Assignments".

RV-32 Format

Figure 7.41 PMA Configuration 15 Control and Status Register Bit Assignments

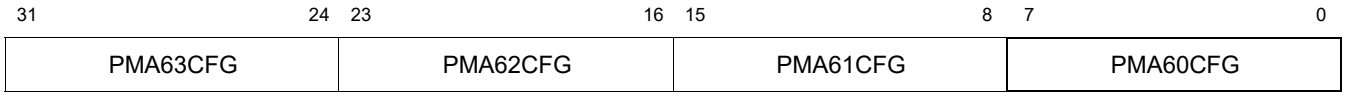


Table 7.43 PMA Configuration 15 Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
PMA63CFG	31:24	PMA63 configuration field in the RV-32 format.	R/W	Undefined
PMA62CFG	23:16	PMA62 configuration field in the RV-32 format.	R/W	Undefined
PMA61CFG	15:8	PMA61 configuration field in the RV-32 format.	R/W	Undefined
PMA60CFG	7:0	PMA60 configuration field in the RV-32 format.	R/W	Undefined

7.3.16.17 PMA0CFG - PMA63CFG Bit Assignments

The PMA Configuration 0 through PMA Configuration 15 registers listed in the previous sub-sections store the 8-bit configuration values for PMA0CFG through PMA63CFG. Refer to Table 7.27 for the mapping of PMA values to each specific register.

As mentioned above, all of the 8-bit PMAxCFG fields described above have the exact same encoding. The cacheability and speculation attributes are included for each PMA.

Figure 7.42 PMA0CFG - PMA63CFG Register Field Assignments

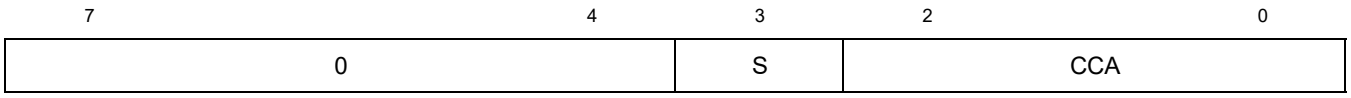


Table 7.44 PMA0CFG - PMA63CFG Register Field Descriptions

Name	Bits	Description	R/W	Reset State
0	7:4	Reserved.	R/W	Undefined
S	3	When this bit is set to 1, speculation is enabled for the specific region.	R/W	Undefined
CCA	2:0	Cacheability and coherency attributes, This field is encoded as follows: 000: Cacheable. Equivalent to MIPS CCA = 3 on CPUs without cache coherency and MIPS CCA = 5 on CPUs with cache coherency. Alias = C. 001: Reserved. 010: Uncacheable. Equivalent to MIPS CCA=2. Alias = UC. 011: Uncached Accelerated. Equivalent to MIPS CCA = 7. Alias = UCA. 100 - 111: Reserved.	R/W	Undefined

7.4 MIPS Hybrid Debug Registers

There is one MIPS hybrid debug register as described below.

Figure 7.43 MIPS Hybrid Debug Register Bit Assignments

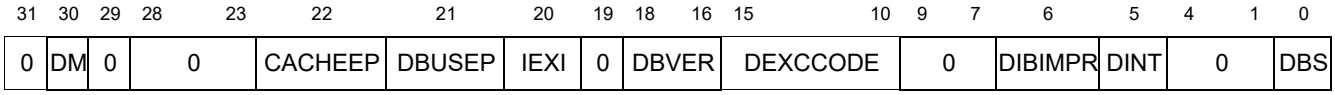


Table 7.45 MIPS Hybrid Debug Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31	Reserved.	R	0
DM	30	Debug Mode. When set, indicates that the processor is operating in debug mode.	R/W	Undefined
0	29	Reserved.	R	0
0	28:23	Reserved.	R	0
CACHEEP	22	When set, this bit indicates that a precise cache parity error is pending.	R/W1	0
DBUSEP	21	When set, this bit indicates that a Data Bus Error exception is pending.	R/W1	0
IEXI	20	Imprecise Error eXception Inhibit. When set, imprecise exceptions are deferred in debug mode. Set on entry to debug mode, cleared on exit.	R/W	0
0	19	Reserved.	R	0
DBGVER	18:16	Debug revision.	R	From configuration
DEXCCODE	15:10	Cause of latest exception in debug mode.	R	Undefined
0	9:7	Reserved.	R	0
DIBIMPR	6	When set, this bit indicates that instruction breakpoints are precise. This bit always reads as 0.	R	0
DINT	5	When set, this bit indicates that a debug interrupt occurred. This bit is cleared by hardware when an exception occurs in debug mode.	R	0
0	4:1	Reserved.	R	0
DBS	0	When set by hardware, this bit indicates that a single step exception has occurred. This bit is cleared by hardware.	R	0

Chapter 8

Interrupt Controller

The Interrupt Controller processes internal and external interrupts in the P8700-F Multiprocessing System. It supports up to 512 external interrupts (configurable in multiples of 8), which are prioritized and routed to the selected hart for servicing. The interrupt priority and routing are programmed via memory-mapped registers. The interrupt controller also implements per-hart timer and software interrupts, non-maskable interrupt routing and watchdog timers. The Interrupt Controller is compatible with the RISC-V Advanced Interrupt Architecture (AIA) specification.

8.1 Overview

The Interrupt Controller implements the following components defined by the RISC-V architecture:

- Advanced Platform-Level Interrupt Controller (APLIC)
- Advanced Core Local Interrupt (ACLINT) Machine-level Timer
- ACLINT Machine-level Software Interrupt (MSWI)
- ACLINT Supervisor-level Software Interrupt (SSWI)
- Watchdog Timer (WDT)

In addition to the standard components, the Interrupt Controller implements custom extensions to support Non-Maskable Interrupt (NMI) routing, timer synchronization, and Watchdog Timer (WDT) configuration.

Note that interrupt events defined as "local" by the RISC-V ISA (such as Local Count Overflow Interrupts and Bus Error Interrupts) are handled internally by the CPU core, and do not involve the Interrupt Controller.

The Interrupt Controller does not implement the RISC-V IMSIC component or the CPU/hart CSRs defined by the RISC-V AIA extension. Consequently, hardware virtualization of interrupts is not supported and delivery of interrupts to virtual guests requires software intervention.

8.1.1 Multi-cluster Support

Each cluster in the P8700-F Multiprocessing System instantiates an Interrupt Controller block, as shown in Figure 1. It is generally preferable that SoC designs connect the same set of interrupt sources to the APLIC interrupt inputs of all clusters in parallel to give software a uniform view of the hardware state across all clusters. However, it is also possible for the SoC design to statically partition the hardware interrupt sources between clusters to suit a specific application.

The memory-mapped registers in the Interrupt Controller are accessible to all clusters in the MPS. This enables software to program a software interrupt as an inter-processor interrupt (IPI) on a remote cluster by writing to the ACLINT registers of the target cluster.

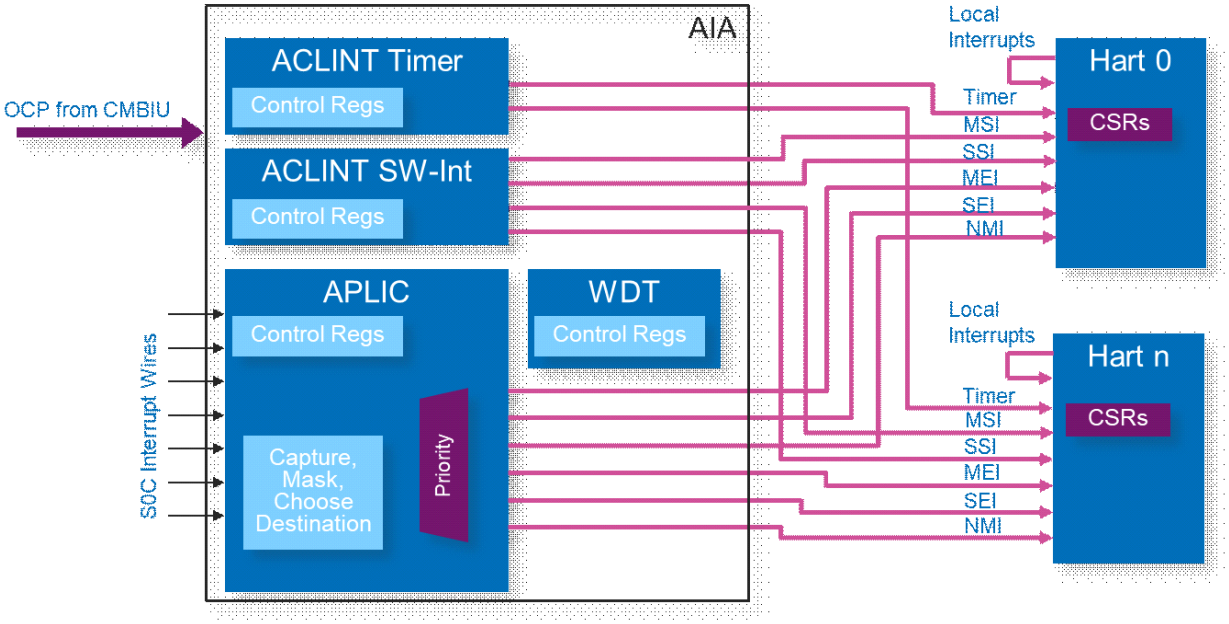


Figure 8.1 Interrupt Controller Block Diagram

This chapter describes how to program the various elements of the interrupt controller using both register examples and code examples. Some of these elements include GIC register layout and distribution, determining the number of external interrupts, configuring individual interrupt sources, scheduling timer interrupts and signaling inter-processor interrupts.

8.1.2 APLIC

The APLIC is responsible for detecting hardware interrupt events from the SoC, prioritizing them and routing them to the assigned hart for servicing. It supports up to 512 interrupt inputs (configurable in increments of 8), although 3 interrupt inputs are reserved for interrupt events originating within the cluster and are not available for external use. Interrupt inputs can be individually programmed to support rising-edge-sensitive, falling-edge-sensitive, high-level-sensitive or low-level-sensitive interrupt signaling.

The APLIC is automatically configured to the number of harts in the cluster, and the APLIC hart index is given by the concatenation of the CORENUM and HARTNUM fields of the mhartid CSR.

The APLIC supports two interrupt domains; a Machine-level domain and a Supervisor-level domain. Both domains are associated with all harts in the cluster, allowing interrupts to be signaled as either Machine External Interrupts (MEI) or Supervisor External Interrupts (SEI). Interrupts are signaled to a hart in "direct delivery mode".

In addition to the standard APLIC functionality, the P8700-F APLIC provides the ability to route any of the external interrupt inputs to a hart non-maskable interrupt (NMI) input. This is accomplished by use of the following custom memory-mapped registers: snmie, setnmienum, clrnmie and clrnmienum (analogous to the standard setie, setienum, clrie and clrienum registers, respectively). If nmie[k] is 1 and interrupt enable bit k (from the setie/clrie registers) is zero, interrupt source k will be treated as an NMI.

In this case, when interrupt source k is asserted, an NMI will be signaled to the hart selected by target[k].HartIndex, and target[k].IPRIO will be ignored. This is only applicable to interrupts at the root (M-Mode) APLIC domain; interrupts delegated to a child (S-Mode) APLIC domain are not available for use as NMI.

8.1.3 ACLINT

The ACLINT is divided into three component devices: the Machine-level Timer (MTIMER), Machine-level Software Interrupter (MSWI), and the Supervisor-level Software Interrupter (SSWI) that provide timer interrupts and software/inter-processor interrupts to the harts in the cluster.

8.1.3.1 MTIMER

The MTIMER device implements the mtime and mtimecmp memory-mapped registers and associated Machine Timer Interrupt (MTI) functionality defined by the RISC-V Privileged Architecture. A single mtime register serves the entire cluster, while each hart has its own dedicated mtimecmp register.

Although it is conceptually part of the ACLINT, the mtime register is physically located in the the always-on power domain of the CPC block to avoid the need to resynchronize the timer with other clusters when a cluster is powered up while the overall system is running. The mtime register is located in the CPC section of the cluster register map.

The mtime register is driven by a dedicated reference clock (si_mtime_clk); the recommended frequency is 100MHz.

In addition to the standard MTIMER functionality, the P8700-F APLIC implements a custom control register (MTIMECTL) that allows the timer to be stopped and synchronizing the mtime counters in multiple clusters more precisely than a pure software synchronization algorithm.

The mtime synchronization procedure is as follows:

1. Software should write 1 to the MTIMECTL.STOP register bit in all clusters to be synchronized.

2. Software should write the desired starting count value (e.g. zero) to the mtime register of all clusters to be synchronized.

3. SoC logic outside of the MPS (presumably under software control) should simultaneously assert the cpc_mtime_start signal to all clusters to be synchronized. This will clear the STOP bit and restart all the counters at the same time.

8.1.3.2 MSWI and SSWI

The MSWI and SSWI devices are implemented as defined by the ACLINT specification. The hart index is given by the concatenation of the CORENUM and HARTNUM fields of the mhartid CSR.

8.1.4 Watchdog Timer

The Interrupt Controller provides a watchdog timer for each hart. The WDT consists of two registers; the WDCSR register (as defined in the RISC-V watchdog timer specification) and a custom configuration register (WDTCFG). The WDTCFG register controls the count-down frequency and selects the event to be triggered on Stage-1 and Stage-2 timeouts. The available timeout events are:

1. Signal an interrupt to the associated hart
2. Signal an NMI to the associated hart
3. Assert a per-cluster signal to the external SoC logic. This signal could be routed to SoC-level monitoring logic or to an interrupt input of another cluster.
4. Reset the cluster

8.2 ACLINT Memory Mapped Registers

8.2.1 ACLINT Machine Mode Memory Map

The ACLINT machine mode memory mapped registers start at offset 0x50000 from GCR_BASE, and use the register definitions specified in the RISC-V Advanced Core Local Interruptor Specification. Registers for the RISC-V Watchdog Timer Specification are also included in the ACLINT machine mode region.

The ACLINT machine mode region contains the following registers, which are described in detail in the subsequent per-register description pages:

Table 1: ACLINT Machine Mode Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x50000 0x50004 ..... 0x53FF8	ACLINT.MSIP[0-4094]	Per-hart machine software interrupt pending
0x54000 0x54008 ..... 0x5BFF0	ACLINT.MTIMECMP[0-4094]	Per-hart mtime compare
0x5C000 0x5C004 ..... 0x5CFFC	ACLINT.WDCFG[0-1023]	MIPS Technologies custom per-hart watchdog configuration
0x5D000 0x5D004 ..... 0x5DFFC	ACLINT.WDCSR[0-1023]	Per-hart watchdog configuration and status

8.2.1.1 ACLINT Machine Software Interrupt Pending (MSIP[0-4094]) Register (offset = see below)

This register is a machine software interrupt pending register. A machine software interrupt is asserted on hart mhartid when MSIP[mhartid[11:0]] is set to 1.

The MSIP register for hart mhartid is accessed at GCR_BASE + 0x50000 + 4 * mhartid[11:0].

Each MSIP register resets to 0, and the upper 31 bits are readonly, zero. MSIP registers for which there is no corresponding hart in the cluster are readonly, zero.

Offset: GCR_BASE + 0x50000, 0x50004, ... 0x53FF8

Figure 8.2 Machine Software Interrupt Pending Register Bit Assignments

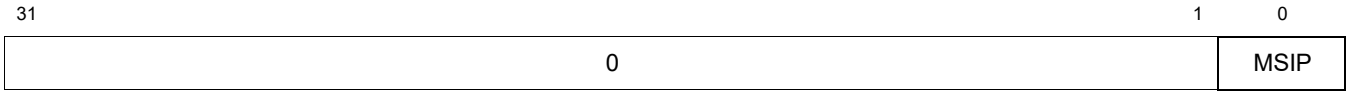


Table 2: Machine Software Interrupt Pending Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved.	R	0
MSIP	0	Machine software interrupt pending register.	R	0

8.2.1.2 ACLINT Machine Time Compare (MTIMECMP[0-4094]) Register (offset = see below)

This register is a machine time compare register. A machine timer interrupt is asserted on hart mhartid when CPC.Global.MTIME_REG > = ACLINT.MTIMECMP[mhartid[11:0]].

The MTIMECMP register for hart mhartid is accessed at GCR_BASE + 0x54000 + 4 * mhartid[11:0].

The architectural reset value of MTIMECMP is undefined. On MIPS Technologies implementations we reset it to all 1's.

Offset: GCR_BASE + 0x54000, 0x54008, ... 0x5BFF0

Figure 8.3 Machine Time Compare Register Bit Assignments

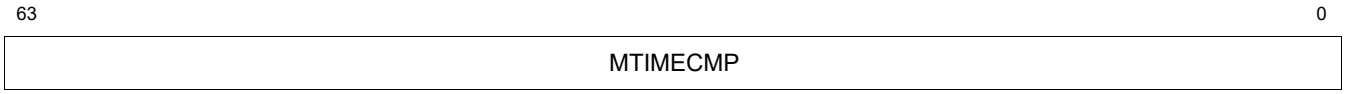


Table 3: Machine Time Compare Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
MTIMECMP	63:0	Machine time compare register.	R	0

8.2.1.3 ACLINT WatchDog ConFiG (WDCFG[0-1023]) Register (offset = see below)

The WDCFG register for hart mhartid is accessed at GCR_BASE + 0x5c000 + 4 * mhartid[11:0]. WDCFG registers for which there is no corresponding hart in the cluster are readonly, zero.

When the watchdog timer is configured to signal an interrupt, it will be signaled to the hart on bit 25 of the mip CSR.

Offset: GCR_BASE + 0x5c000, 0x5c004, ... 0x5CFFC

Figure 8.4 WatchDog ConFiG Register Bit Assignments

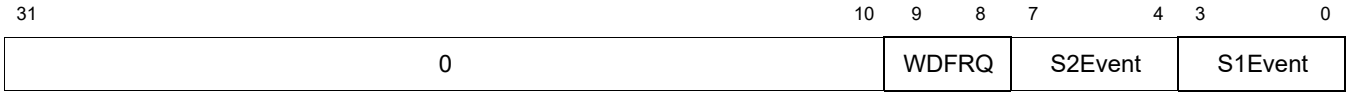


Table 4: WatchDog ConFiG Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:10	Reserved	R	0
WDFRQ	9:8	WDT count-down frequency: counter decrements when bit 8 * (WDFRQ + 1) of CPC.Global.MTIME_REG transitions from 0 to 1.	R/W	0
S2Event	7:4	Event to trigger on Stage-2 timeout Encoding 0: Alias = Interrupt, Meaning assert interrupt via APLIC Encoding 1: Alias = NMI, Meaning assert NMI Encoding 2: Alias = Reset, Meaning assert Reset Encoding 3: Alias = TopLevel, Meaning assert top-level pin to SoC logic	R/W	0
S1Event	3:0	Event to trigger on Stage-1 timeout. When S1Event is set to 4, the WDT will behave as an interval timer. When the counter reaches zero, an interrupt will be signaled and the counter will be reinitialized to WDCSR.WTOCNT but the WDCSR.S1WTO bit will not be set. In this mode, the WDT will periodically signal the stage-1 interrupt at a fixed interval, and never signal the stage-2 event. Encoding 0: Alias = Interrupt, Meaning assert interrupt via APLIC Encoding 1: Alias = NMI , Meaning assert NMI Encoding 2: Alias = Reset , Meaning reset Encoding 3: Alias = TopLevel, Meaning top-level pin to SoC logic Encoding 4: Alias = IntervalTimer, Meaning Interrupt Interval Timer	R/W	0

8.2.1.4 ACLINT WatchDog Control and Status (WDCSR[0-1023]) Register (offset = see below)

The WDCSR register for hart mhartid is accessed at GCR_BASE + 0x5d000 + 4 * mhartid[11:0]. WDCSR registers for which there is no corresponding hart in the cluster are readonly, zero.

Offset: GCR_BASE + 0x5D000, 0x5D004, ... 0x5DFFC

Figure 8.5 WatchDog Control and Status Register Bit Assignments

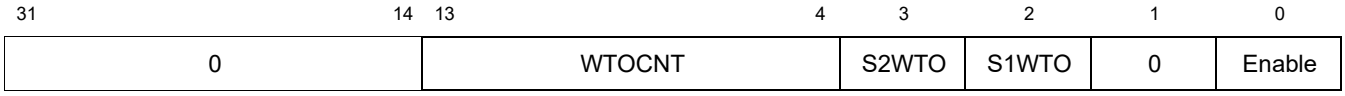


Table 5: WatchDog Control and Status Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:14	Reserved	R	0
WTOCNT	13:4	Watchdog timer out count. Writes to WDCSR and stage-1 timeouts cause a timeout counter to be initialized to WTOCNT.	R/W	Undefined
S2WTO	3	Stage-2 watchdog timeout has occurred. Set when timeout counter is zero and S1WTO=1 (unless WDCFG.S1Event=4)	R/W	Undefined
S1WTO	2	Stage-1 watchdog timeout has occurred. Set when timeout counter is zero.	R/W	Undefined
0	1	Reserved	R	0
Enable	0	Enable watchdog timer.	R/W	0

8.2.2 Aclint Supervisor Mode Memory Map

The ACLINT supervisor mode memory mapped registers start at offset 0x6C000 from GCR_BASE, and use the register definitions specified in the RISC-V Advanced Core Local Interruptor Specification.

The ACLINT supervisor mode region contains the following registers, which are described in detail in the subsequent per-register description pages:

Table 6: ACLINT Supervisor Mode Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x6C000 0x6C004 ..... 0x6FFF8	ACLINT.SETSSIP[0-4094]	Per-hart set supervisor software interrupt pending

8.2.2.1 ACLINT SET Supervisor Software Interrupt Pending (SETSSIP[0-4094]) Register (offset = see below)

This register set supervisor software interrupt pending register. A supervisor software interrupt is asserted on hart mhartid when SETSSIP[mhartid[11:0]] is written to 1. The SETSSIP register ignores writes of zero and always reads as zero.

The SETSSIP register for hart mhartid is accessed at GCR_BASE + 0x6c000 + 4 * mhartid[11:0]. SETSSIP registers for which there is no corresponding hart in the cluster are readonly, zero.

Offset: GCR_BASE + 0x6C000, 0x6C004, ... 0x6FFF8

Figure 8.6 SET Supervisor Software Interrupt Pending Register Bit Assignments

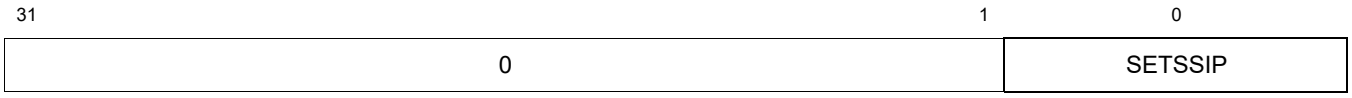


Table 7: SET Supervisor Software Interrupt Pending Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
SETSSIP	0	Set supervisor software interrupt pending register	R	0

8.3 APLIC Memory Mapped Registers

8.3.1 APLIC Machine Domain Memory Map

The APLIC machine domain starts at offset 0x40000 from GCR_BASE, and uses the register definitions and address offsets for an APLIC domain as specified in the RISC-V Advanced Interrupt Architecture. The offsets and registers within the domain are identical to those for the APLIC supervisor domain.

The APLIC machine domain contains the following registers, which are described in detail in the subsequent per-register descriptions:

Table 8: APLIC Machine Domain Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x40000	APLIC.M.domaincfg	Machine domain configuration
0x40004 0x40008 ..... 0x40FFC	APLIC.M.sourcecfg[1-1023]	Machine source configuration
0x41C00 0x41C04 ..... 0x41C7C	APLIC.M.setip[0-31]	Set machine interrupt pending by mask
0x41CDC	APLIC.M.setipnum	Set machine interrupt pending by number
0x41D00 0x41D04 ..... 0x41D7C	APLIC.M.in_clrip[0-31]	Read machine source input or clear machine interrupt pending by mask
0x41DDC	APLIC.M.clripnum	Clear machine interrupt pending by number
0x41E00 0x41E04 ..... 0x41E7C	APLIC.M.setie[0-31]	Set machine interrupt enable by mask
0x41EDC	APLIC.M.setienum	Set machine interrupt enable by number
0x41F00 0x41F04 ..... 0x41F7C	APLIC.M.clrie[0-31]	Clear machine interrupt enable by mask
0x41FDC	APLIC.M.clrienum	Clear machine interrupt enable by number
0x42000	APLIC.M.setipnum_le	Set supervisor interrupt pending by number, Little-endian
0x43004 0x43008 ..... 0x43FFC	APLIC.M.target[1-1023]	Specify target hart and priority for machine interrupt source

Table 8: APLIC Machine Domain Memory Mapped Registers*(continued)*

Offset from GCR_BASE	Register Block Name	Description
0x44000 0x44020 ..... 0x4BFE0	APLIC.M.Hart[0-1023].idelivery	Enable machine interrupt delivery for hart
0x44004 0x44024 ..... 0x4BFE4	APLIC.M.Hart[0-1023].iforce	Force machine interrupt for hart
0x44008 0x44028 ..... 0x4BFE8	APLIC.M.Hart[0-1023].ithreshold	Specify machine interrupt priority threshold for hart
0x44018 0x44038 ..... 0x4BFF8	APLIC.M.Hart[0-1023].topi	Read top priority pending machine interrupt for hart
0x4401C 0x4403C 0x4BFFC	APLIC.M.Hart[0-1023].claimi	Claim top priority pending machine interrupt for hart

8.3.2 APLIC Supervisor Domain Memory Map

The APLIC supervisor domain starts at offset 0x60000 from GCR_BASE, and uses the register definitions and address offsets for an APLIC domain as specified in the RISC-V Advanced Interrupt Architecture. The offsets and registers within the domain are identical to those for the APLIC machine domain.

The APLIC supervisor domain contains the following registers, which are described in detail in the subsequent per-register descriptions

Table 9: APLIC Supervisor Domain Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x60000	APLIC.S.domaincfg	Supervisor domain configuration
0x60004 0x60008 ..... 0x60FFC	APLIC.S.sourcecfg[1-1023]	Supervisor source configuration
0x61C00 0x61C04 ..... 0x61C7C	APLIC.S.setip[0-31]	Set supervisor interrupt pending by mask
0x61CDC	APLIC.S.setipnum	Set supervisor interrupt pending by number
0x61D00 0x61D04 ..... 0x61D7C	APLIC.S.in_clrip[0-31]	Read supervisor source input or clear supervisor interrupt pending by mask
0x61DDC	APLIC.S.clripnum	Clear supervisor interrupt pending by number
0x61E00 0x61E04 ..... 0x61E7C	APLIC.S.setie[0-31]	Set supervisor interrupt enable by mask
0x61EDC	APLIC.S.setienum	Set supervisor interrupt enable by number
0x61F00 0x61F04 ..... 0x61F7C	APLIC.S.clrie[0-31]	Clear supervisor interrupt enable by mask
0x61FDC	APLIC.S.clrienum	Clear supervisor interrupt enable by number
0x62000	APLIC.S.setipnum_le	Set supervisor interrupt pending by number, Little-endian
0x63004 0x63008 ..... 0x63FFC	APLIC.S.target[1-1023]	Specify target hart and priority for supervisor interrupt source

Table 9: APLIC Supervisor Domain Memory Mapped Registers *(continued)*

Offset from GCR_BASE	Register Block Name	Description
0x64000 0x64020 ..... 0x6BFE0	APLIC.S.Hart[0-1023].idelivery	Enable supervisor interrupt delivery for hart
0x64004 0x64024 ..... 0x6BFE4	APLIC.S.Hart[0-1023].iforce	Force supervisor interrupt for hart
0x64008 0x64028 ..... 0x6BFE8	APLIC.S.Hart[0-1023].ithreshold	Specify supervisor interrupt priority threshold for hart
0x64018 0x64038 ..... 0x6BFF8	APLIC.S.Hart[0-1023].topi	Read top priority pending supervisor interrupt for hart
0x6401C 0x6403C 0x6BFFC	APLIC.S.Hart[0-1023].claimi	Claim top priority pending supervisor interrupt for hart

8.3.3 APLIC Custom Memory Map

The APLIC custom region starts at offset 0x4c000 from GCR base, and contains the following registers, which are described in more detail in the subsequent per-register descriptions

Table 10: APLIC Custom Memory Mapped Registers

Offset from GCR_BASE	Register Block Name	Description
0x4C000 0x4C004 ..... 0x4C07C	APLIC.setnmie[0-31]	Set NMI enabled bit by mask
0x4C0DC	APLIC.setnmienum	Set NMI enabled bit by number
0x4C100 0x4C104 ..... 0x4C17C	APLIC.clrnmie[0-31]	Clear NMI enabled bit by mask
0x4C1DC	APLIC.clrnmienum	Clear NMI enabled bit by number

8.3.3.1 APLIC Domain Configuration (DOMAINCFG) Register (offset = see below)

This register domain configuration register. per-domain register containing the APLIC domain's

configuration status.

Offset: APLIC + 0x00000

GCR_BASE + 0x40000 # APLIC.M

GCR_BASE + 0x60000 # APLIC.S

Figure 8.7 Domain Configuration Register Bit Assignments

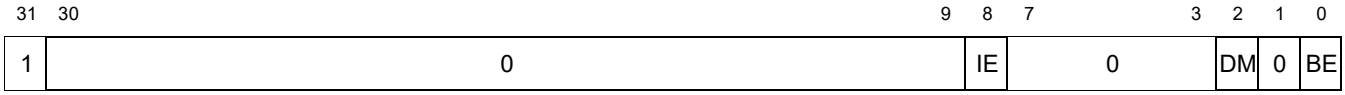


Table 11: Domain Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
1	31	Allows current endianness to be identified by reading domaincfg. The P8700 supports Little Endian mode.	R	1
0	30:9	Reserved	R	0
IE	8	Interrupts Enabled for this domain?	R/W	0
0	7:3	Reserved	R	0
DM	2	Read only-0 when IMSIC not supported. Delivery Mode Encoding 0: Alias = Direct, Meaning Direct delivery mode Encoding 1: Alias = MSI, Meaning MSI delivery mode	R	0
0	1	Reserved	R	0
LE	0	This bit is always 0 to indicate Little Endian addressing mode. Note that Big Endian mode is not supported in the P8700.	R	0 (LE mode)

8.3.3.2 APLIC Source Configuration (SOURCECFG[1-1023]) Register (offset = see below)

This register source configuration register. Per domain, per-interrupt source read/write registers containing configuration status for each interrupt source in the APLIC domain.

The sourcecfg[i] register for source i is accessed at the APLIC domain base address + 4 * i.

Offset: APLIC + 0x00004, 0x00008, ... 0x00ffc

GCR_BASE + 0x40004, 0x40008, ... 0x40ffc # APLIC.M

GCR_BASE + 0x60004, 0x60008, ... 0x60ffc # APLIC.S

Figure 8.8 Source Configuration Register Bit Assignments

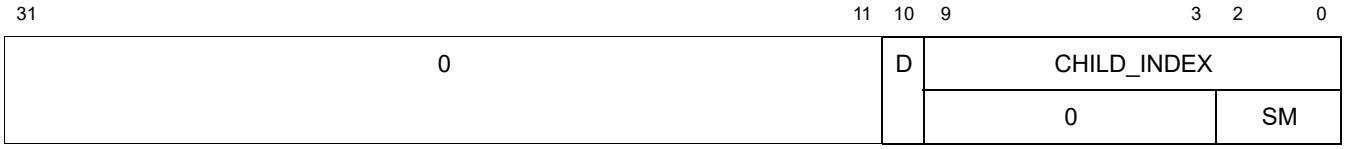


Table 12: Source Configuration Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:11	Reserved	R	0
D	10	Read/write in Machine domain, readonly 0 in Supervisor domain. Is the Machine domain interrupt source delegated to the Supervisor domain?	R/W	0
CHILD_INDEX	9:0	Target domain for delegated interrupts. Only one target domain (Supervisor, CHILD_INDEX = 0) is currently supported by MIPS Technologies implementations. These bits are only used as CHILD_INDEX when sourcecfg.D = 1. When sourcecfg.D = 0, bits 2:0 are used as the SM (source mode) bitfield.	R	0
SM	2:0	Source Mode. These bits are only used as SM when sourcecfg.D=0. When sourcecfg.D = 1, bits 9:0 are used as the CHILD_INDEX bitfield. Encoding 0: Alias = Inactive, Meaning Inactive in this domain (and not delegated) Encoding 1: Alias = Detached, Meaning Active, detached from the source wire Encoding 4: Alias = Edge1, Meaning Active, edge-sensitive, asserted on rising edge Encoding 5: Alias = Edge0, Meaning Active, edge-sensitive, asserted on falling edge Encoding 6: Alias = Level1, Meaning Active, level-sensitive, asserted when high Encoding 7: Alias = Level0, Meaning Active, level-sensitive, asserted when low	R/W	0

8.3.3.3 APLIC SET Interrupt Pending (SETIP[0-31]) Register (offset = see below)

This register set interrupt pending register. A write to the per-domain setip[i] register sets the interrupt pending bit 32 * i + j for every bit position j which is 1 in the written value. A read of setip[i] register returns a bitmask of those interrupt sources in the range [32i + 31:32i] for which the interrupt is pending.

Only interrupt sources which are active in the targeted APLIC domain can be read or written.

When the sourcecfg.SM field for the interrupt source is configured to be in Level0 or Level1 mode, the interrupt source is tied directly to the external interrupt input signal, and writes to setip are ignored, while reads of setip return the rectified value of the external interrupt signal.

Offset: APLIC + 0x01c00, 0x01c04, ... 0x01c7c

GCR_BASE + 0x41c00, 0x41c04, ... 0x41c7c # APLIC.M

GCR_BASE + 0x61c00, 0x61c04, ... 0x61c7c # APLIC.S

Figure 8.9 SET Interrupt Pending Register Bit Assignments

Table 13: SET Interrupt Pending Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETIP	31:0	Set interrupt pending register.	R/W	0

8.3.3.4 APLIC Input/Clear Interrupt Pending (IN_CLRIP[0-31]) Register (offset = see below)

This register input/clear interrupt pending register. A write to the per-domain in_crlip[i] register clears the interrupt pending bit 32 * i + j for every bit position j which is 1 in the written value.

Only interrupt sources which are active in the targeted APLIC domain can be written. When the sourcecfg.SM field for the interrupt source is configured to be in Level0 or Level1 mode, the interrupt source is tied directly to the external interrupt input signal and writes to in_clrip are ignored.

A read of in_clrip[i] register returns a bitmask of the rectified input value for interrupt sources in the range [32i + 31:32i], where the rectified input value is the input source value if the interrupt is in Edge1 or Level1 mode, the inverted input source value if the interrupt is in Edge0 or Level0 mode, or zero otherwise.

Offset: APLIC + 0x01d00, 0x01d04, ... 0x01d7c

GCR_BASE + 0x41d00, 0x41d04, ... 0x41d7c # APLIC.M

GCR_BASE + 0x61d00, 0x61d04, ... 0x61d7c # APLIC.S

Figure 8.10 Input/Clear Interrupt Pending Register Bit Assignments

Table 14: Input/Clear Interrupt Pending Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
IN_CLRIP	31:0	INput/CLeaR Interrupt Pending Register.	R/W	0

8.3.3.5 APLIC Set Interrupt-Pending Number (SETIPNUM) Register (offset = see below)

This register set interrupt-pending number register. On writes, set interrupt pending bit for the num-bered interrupt source to 1. Only interrupt sources which are active in the targeted APLIC domain and not configured as level sensitive can be written. Reads return zero.

Offset: APLIC + 0x01cdc

GCR_BASE + 0x41cdc # APLIC.M

GCR_BASE + 0x61cdc # APLIC.S

Figure 8.11 Set Interrupt-Pending Number Register Bit Assignments

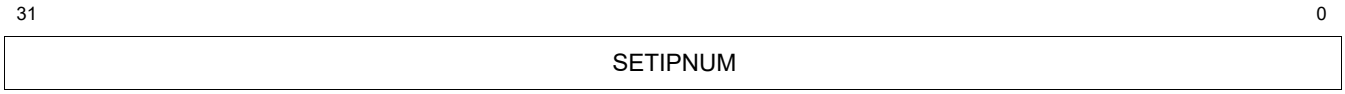


Table 15: Set Interrupt-Pending Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETIPNUM	31:0	Set interrupt-pending number register.	R	0

8.3.3.6 APLIC Clear IP Number (CLRIPNUM) Register (offset = see below)

This register clear IP number register. On writes, clear interrupt pending bit for the numbered interrupt source. Only interrupt sources which are active in the targeted APLIC domain and not configured as level sensitive can be written. Reads return zero.

Offset: APLIC + 0x01ddc

GCR_BASE + 0x41ddc # APLIC.M

GCR_BASE + 0x61ddc # APLIC.S

Figure 8.12 Clear IP Number Register Bit Assignments

Table 16: Clear IP Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLRIPNUM	31:0	Clear IP number register.	R	0

8.3.3.7 APLIC Set Interrupt Enable (SETIE[0-31]) Register (offset = see below)

This register set interrupt enable register. A write to the per-domain setie[i] register sets the interrupt enable bit 32 * i + j for every bit position j which is one in the written value. Only interrupt sources which are active in the targeted APLIC domain can be written.

A read of the SETIE[i] register returns a bit-mask of those interrupt sources in the range [32i + 31:32i] for which the interrupt is enabled.

Offset: APLIC + 0x01e00, 0x01e04, ... 0x01e7c

GCR_BASE + 0x41e00, 0x41e04, ... 0x41e7c # APLIC.M

GCR_BASE + 0x61e00, 0x61e04, ... 0x61e7c # APLIC.S

Figure 8.13 Set Interrupt Enable Register Bit Assignments

Table 17: Set Interrupt Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETIE	31:0	Set interrupt enable register.	R/W	0

8.3.3.8 APLIC Clear Interrupt Enable (CLRIE[0-31]) Register (offset = see below)

This register clear interrupt enable register. A write to the per-domain CLRIE[i] register clears the interrupt enable bit 32 * i + j for every bit position j which is one in the written value. Only interrupt sources which are active in the targeted APLIC domain can be written.

Offset: APLIC + 0x01f00, 0x01f04, ... 0x01f7c

GCR_BASE + 0x41f00, 0x41f04, ... 0x41f7c # APLIC.M

GCR_BASE + 0x61f00, 0x61f04, ... 0x61f7c # APLIC.S

Figure 8.14 Clear Interrupt Enable Register Bit Assignments

Table 18: Clear Interrupt Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLRIE	31:0	Clear interrupt enable register.	R	0

8.3.3.9 APLIC Set Interrupt Enable Number (SETIENUM) Register (offset = see below)

This register set interrupt enable number register. On writes, set interrupt enable bit for the numbered interrupt source to 1. Only interrupt sources which are active in the targeted APLIC domain can be written.

Offset: APLIC + 0x01edc

GCR_BASE + 0x41edc # APLIC.M

GCR_BASE + 0x61edc # APLIC.S

Figure 8.15 Set Interrupt Enable Number Register Bit Assignments

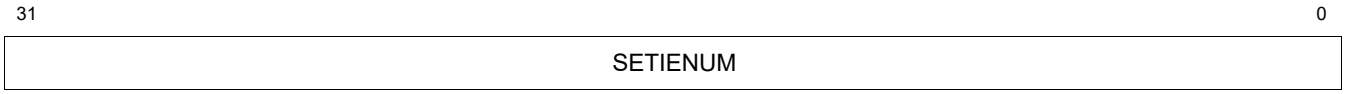


Table 19: Set Interrupt Enable Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETIENUM	31:0	Set interrupt enable number register.	R	0

8.3.3.10 APLIC Clear Interrupt Enable Number (CLRIENUM) Register (offset = see below)

This register clear interrupt enable number register. On writes, clear interrupt enable bit for the numbered interrupt source. Only interrupt sources which are active in the targeted APLIC domain and not configured as level sensitive can be written.

Offset: APLIC + 0x01fdc

GCR_BASE + 0x41fdc # APLIC.M

GCR_BASE + 0x61fdc # APLIC.S

Figure 8.16 Clear Interrupt Enable Number Register Bit Assignments

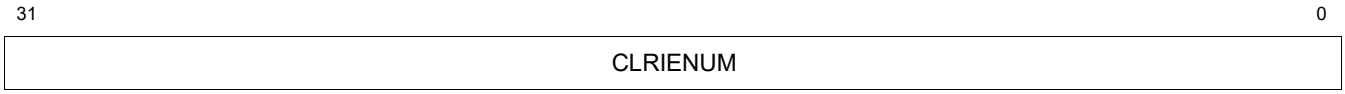


Table 20: Clear Interrupt Enable Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLRIENUM	31:0	Clear interrupt enable number register.	R	0

8.3.3.11 APLIC Set Interrupt-Pending Number (SETIPNUM_LE) Register (offset = see below)

This register set interrupt-pending number (Little Endian) register. On writes, set interrupt pending bit for the numbered interrupt source to 1. Only interrupt sources which are active in the targeted APLIC domain and not configured as level sensitive can be written.

Offset: APLIC + 0x02000

GCR_BASE + 0x42000 # APLIC.M

GCR_BASE + 0x62000 # APLIC.S

Figure 8.17 Set Interrupt-Pending Number Register Bit Assignments

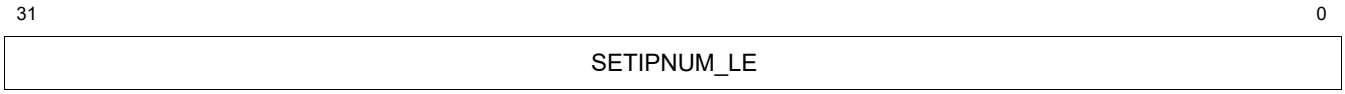


Table 21: Set Interrupt-Pending Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETIPNUM_LE	31:0	Set interrupt-pending number (Little Endian) register.	R	0

8.3.3.12 APLIC Target (TARGET[1-1023]) Register (offset = see below)

This register is target register. Per domain, per-interrupt source registers for configuring the target hart number and priority for each interrupt source in the APLIC domain.

The target[i] register for source i is accessed at the APLIC domain base address + 0x3004 + 4 * i.

Offset: APLIC + 0x03004, 0x03008, ... 0x03ffc

GCR_BASE + 0x43004, 0x43008, ... 0x43ffc # APLIC.M

GCR_BASE + 0x63004, 0x63008, ... 0x63ffc # APLIC.S

Figure 8.18 Target Register Bit Assignments

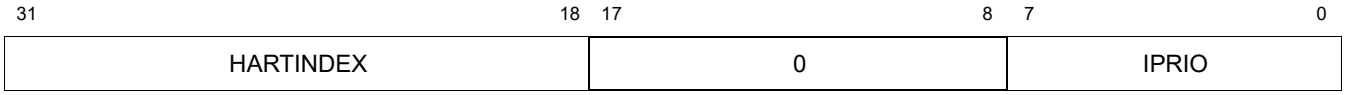


Table 22: Target Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
HARTINDEX	31:18	Index of hart to be targeted by this interrupt source. For MIPS Technologies implementations, the index is mhartid[11:0].	R/W	0
0	17:8	Reserved.	R	0
IPRIO	7:0	Priority of this interrupt source. Values in the range (1<<APLIC.ipriolen) - 1:1 are supported, with 1 being the highest priority.	R/W	1

8.3.3.13 APLIC Interrupt Delivery (HART[0-1023].IDELIVERY) Register (offset = see below)

This register interrupt delivery register. Per-domain, per-hart registers for configuring whether deliv-ery of each interrupt source is enabled.

The idelivery register for hart mhartid is accessed at the APLIC domain base address + 0x4000 + 0x20 * mhartid[11:0].

Offset: APLIC + 0x04000, 0x04020, ... 0x0bfe0

GCR_BASE + 0x44000, 0x44020, ... 0x4bfe0 # APLIC.M

GCR_BASE + 0x64000, 0x64020, ... 0x6bfe0 # APLIC.S

Figure 8.19 Interrupt Delivery Register Bit Assignments

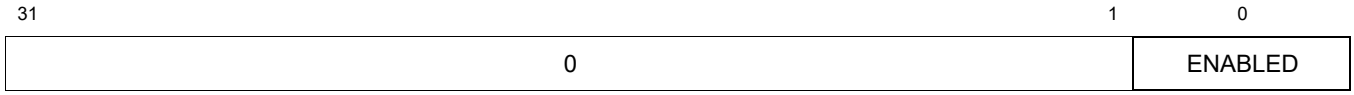


Table 23: Interrupt Delivery Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
ENABLED	0	Interrupt delivery register.	R	0

8.3.3.14 APLIC Interrupt Force (HART[0-1023].IFORCE) Register (offset = see below)

This register interrupt force register. Per-domain, per-hart registers for specifying whether a interrupt in the APLIC domain is forced for each hart in the domain.

The iforce register for hart mhartid is accessed at the APLIC domain base address + 0x4004 + 0x20 * mhartid[11:0].

Offset: APLIC + 0x04004, 0x04024, ... 0x0bfe4

GCR_BASE + 0x44004, 0x44024, ... 0x4bfe4 # APLIC.M

GCR_BASE + 0x64004, 0x64024, ... 0x6bfe4 # APLIC.S

Figure 8.20 Interrupt Force Register Bit Assignments

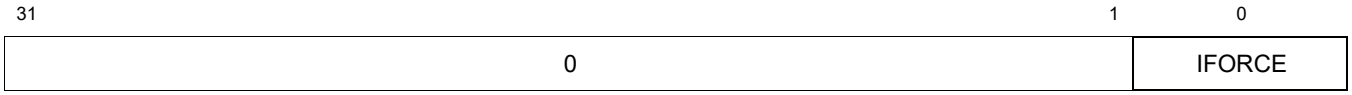


Table 24: Interrupt Force Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:1	Reserved	R	0
IFORCE	0	Interrupt force register.	R	0

8.3.3.15 APLIC Interrupt Threshold (HART[0-1023].ITHRESHOLD) Register (offset = see below)

This register interrupt threshold register. Per-domain, per-hart registers specifying the interrupt priority threshold for each hart in the domain. A value of zero means no threshold is applied. A non zero value means that interrupts with priority value greater than or equal to the threshold will be ignored.

The ithreshold register for hart mhartid is accessed at the domain APLIC base address + 0x4008 + 0x20 * mhartid[11:0].

Offset: APLIC + 0x04008, 0x04028, ... 0x0bfe8

GCR_BASE + 0x44008, 0x44028, ... 0x4bfe8 # APLIC.M

GCR_BASE + 0x64008, 0x64028, ... 0x64fe8 # APLIC.S

Figure 8.21 Interrupt Threshold Register Bit Assignments

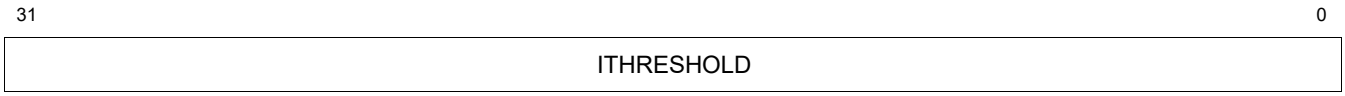


Table 25: Interrupt Threshold Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
ITHRESHOLD	31:0	Interrupt threshold register.	R	0

8.3.3.16 APLIC Top Interrupt (HART[0-1023].TOPI) Register (offset = see below)

This register top interrupt register. Registers specifying the top priority pending interrupt for each hart in the domain.

The topi register for hart mhartid is accessed at the domain APLIC base address + 0x4018 + 0x20 * mhartid[11:0].

Offset: APLIC + 0x04018, 0x04038, ... 0x0bff8

GCR_BASE + 0x44018, 0x44038, ... 0x4bff8 # APLIC.M

GCR_BASE + 0x64018, 0x64038, ... 0x6bff8 # APLIC.S

Figure 8.22 Top Interrupt Register Bit Assignments

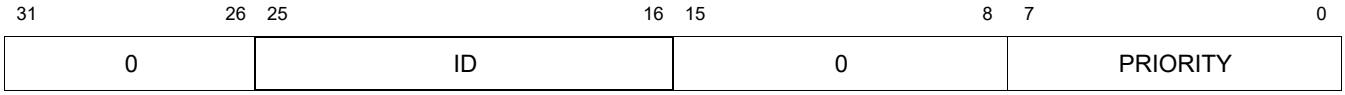


Table 26: Top Interrupt Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:26	Reserved	R	0
ID	25:16	ID register.	R	0
0	15:8	Reserved	R	0
PRIORITY	7:0	Priority pending interrupt for each hart in the domain.	R	0

8.3.3.17 APLIC Claim Interrupt (HART[0-1023].CLAIMI) Register (offset = see below)

This register claim interrupt register. Per-domain, per-hart register for claiming and deasserting the Harts top priority interrupt in the domain.

Reading the claimi register returns the current value of the topi register for this hart, i.e. the highest priority pending interrupt source number and the corresponding IPRIO value. In addition, the interrupt pending signal for that interrupt source is cleared, unless the interrupt is in level-sensitive mode, in which case the interrupt pending signal is directly tied to the external interrupt signal and can only be cleared by change in the external interrupt signal value.

If no interrupt is currently pending for the hart, i.e. topi equals 0, then the forcei register for the hart is cleared by a read of claimi.

Writes to the claimi register are ignored.

The claimi register for hart mhartid is accessed at the domain APLIC base address + 0x401c + 0x20 * mhartid[11:0].

Offset: APLIC + 0x0401c, 0x0403c, ... 0x0bffc

GCR_BASE + 0x4401c, 0x4403c, ... 0x4bffc # APLIC.M

GCR_BASE + 0x6401c, 0x6403c, ... 0x6bffc # APLIC.S

Figure 8.23 Claim Interrupt Register Bit Assignments

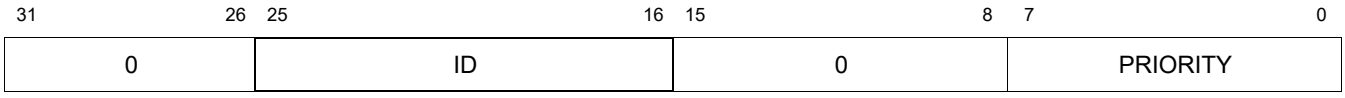


Table 27: Claim Interrupt Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
0	31:26	Reserved	R	0
ID	25:16	ID register.	R	0
0	15:8	Reserved	R	0
PRIORITY	7:0	Per-hart register for claiming and deasserting the Harts top priority interrupt in the domain.	R	0

8.3.3.18 APLIC Set NMI Enable (SETNMIE[0-31]) Register (offset = see below)

This register set NMI enable register. A write to setnmie[i] register sets the NMI enable bit 32 * i + j for every bit position j which is 1 in the written value. A read of setnmie[i] register returns a bitmask of those interrupt sources in the range [32i + 31:32i] for which the NMI enabled bit is currently set.

When interrupt source i is pending in the machine domain and not enabled (i.e. APLIC.sourcecfg.D=0, APLIC.ip[i] is set and APLIC.ie[i] is clear) and NMIs are enabled for the source (i.e. APLIC.nmie[i] is set) then the interrupt is delivered to the target hart as an NMI.

Offset: GCR_BASE + 0x4c000, 0x4c004, ... 0x4c078

Figure 8.24 Set NMI Enable Register Bit Assignments

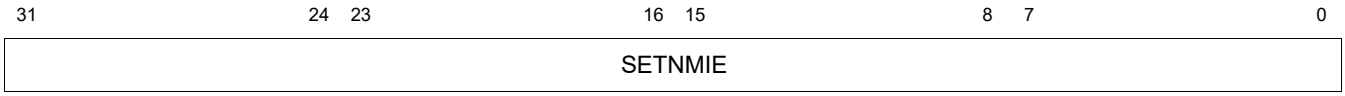


Table 28: Set NMI Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETNMIE	31:0	Set NMI enable register.	R	0

8.3.3.19 APLIC Set NMI Number (SETNMIENUM) Register (offset = 0x4C0DC)

This register set NMI number register. On writes, set the NMI enable bit for the numbered interrupt source to 1. Reads return zero.

When interrupt source i is pending in the machine domain and not enabled (i.e. APLIC.sourcecfg.D=0, APLIC.ip[i] is set and APLIC.ie[i] is clear) and NMIs are enabled for the source (i.e. APLIC.nmie[i] is set) then the interrupt is delivered to the target hart as an NMI.

Figure 8.25 Set NMI Number Register Bit Assignments

Table 29: Set NMI Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
SETNMIENUM	31:0	Set NMI number register.	R	0

8.3.3.20 APLIC Clear NMI Enable (CLRNMIE[0-31]) Register (offset = see below)

This register clear NMI enable register. A write to clrnmie[i] register clears the NMI enable bit 32 * i + j for every bit position j which is 1 in the written value.

When interrupt source i is pending in the machine domain and not enabled (i.e. APLIC.sourcecfg.D=0, APLIC.ip[i] is set and APLIC.ie[i] is clear) and NMIs are enabled for the source (i.e. APLIC.nmie[i] is set) then the interrupt is delivered to the target hart as an NMI.

Offset: GCR_BASE + 0x4c100, 0x4c104, ... 0x4c178

Figure 8.26 Clear NMI Enable Register Bit Assignments

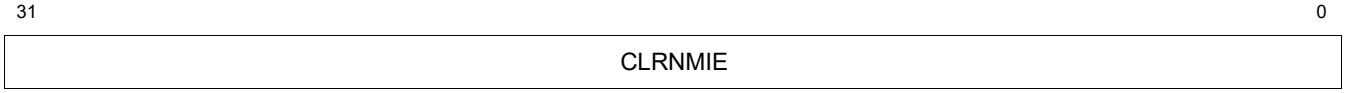


Table 30: Clear NMI Enable Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLRNMIE	31:0	Clear NMI enable register.	R	0

8.3.3.21 APLIC Clear NMI Number (CLRNMIENUM) Register (offset = 0x4C1DC)

This register clear NMI number register. On writes, clear the NMI enable bit for the numbered interrupt source. Reads return zero.

When interrupt source i is pending in the machine domain and not enabled (i.e. APLIC.sourcecfg.D=0, APLIC.ip[i] is set and APLIC.ie[i] is clear) and NMIs are enabled for the source (i.e. APLIC.nmie[i] is set) then the interrupt is delivered to the target hart as an NMI.

Figure 8.27 Clear NMI Number Register Bit Assignments

Table 31: Clear NMI Number Register Bit Descriptions

Name	Bits	Description	R/W	Reset State
CLRNMIENUM	31:0	Clear NMI number register.	R	0

Chapter 9

Floating-Point Unit (FPU)

This chapter describes the optional MIPS RV64 Floating-Point Unit (FPU).

9.1 Features Overview

The P8700-F core features an optional IEEE 754 compliant 3rd generation Floating Point Unit (FPU3).

The FPU contains thirty-two, 64-bit registers used by FPU instructions. Single precision floating point instructions use the lower 32 bits of the 64 bit register. Double precision floating point instructions use the the entire 64 bits of the register.

The FPU is fully synthesizable and operates at the same clock speed as the CPU. The P8700-F core can issue up to two instructions per cycle to the FPU.

The FPU contains two execution pipelines. These pipelines operate in parallel with the integer core and do not stall when the integer pipeline stalls. This allows long-running FPU operations such as divide or square root, to be partially masked by system stall and/or other integer unit instructions.

A scheduler in the ISU block issues instructions to the two FPU functional units. The exception model is ‘precise’ at all times.

The FPU supports fused multiply-adds as defined by the IEEE Standard for Floating-Point Arithmetic 754TM-2008. All floating point denormalized input operands and results are fully supported in hardware.

The FPU supports scalar FPU instructions.

9.2 FPU Execution Units

The P8700-F FPU contains two execution units, one for short operations (EXS) and one for long operations (EXL).

9.2.1 Short Operations

The short data path contains an integer add unit, logical unit, and div unit. The integer add unit and the logical unit each have 2-cycle latency outputs. One divide instruction can be issued to the div unit at a time. That divide will be worked on iteratively. Until the divide is done no other divide instructions can be issued.

The short execution unit (EXES) executes the following instructions:

- All instructions that are sent back to the integer unit, including stores, move-from, and branches

- Most 2-source logical operands.
- Floating point compares
  - fmin/fmax
  - fclass
  - Sign injection: FSGNJ.S, FSGNJN.S, FSGNJX.S
  - feq/fle/flt

Results are written to the Working Register File (WRF).

9.2.2 Long Operations

The long execution unit (EXEL) implements the following operations:

- FP adds, converts, multiplies, and divide-square roots
- Logical operations with 3 sources

Results are written to the Working Register File (WRF).

9.3 Data Formats

The FPU provides both floating-point and fixed-point data types, which are described below:

- The single- and double-precision floating-point data types are those specified by IEEE Standard 754.
- The signed integers provided by the CPU architecture.

9.3.1 Floating-Point Formats

The FPU provides the following two floating-point formats:

- A 32-bit single-precision floating point (type S)
- A 64-bit double-precision floating point (type D)

The floating-point data types represent numeric values as well as the following special entities:

- Two infinities, $+\infty$ and $-\infty$
- Signaling non-numbers (SNaNs)
- Quiet non-numbers (QNaNs)
- Numbers of the form: $(-1)^s\ 2^E\ b_0.b_1\ b_2..b_{p-1}$, where:
  - s = 0 or 1
  - E = any integer between E_min and E_max, inclusive
  - b_i = 0 or 1 (the high bit, b_0, is to the left of the binary point)
  - p is the signed-magnitude precision

The single and double floating-point data types are composed of three fields—sign, exponent, fraction—whose sizes are listed in Table 9.1.

Table 9.1 Parameters of Floating-Point Data Types

Parameter	Single	Double
Bits of mantissa precision, p	24	53
Maximum exponent, E_max	+127	+1023
Minimum exponent, E_min	-126	-1022
Exponent *bias*	+127	+1023
Bits in exponent field, *e*	8	11
Representation of b_0 integer bit	hidden	hidden
Bits in fraction field, *f*	23	52
Total format width in bits	32	64
Magnitude of largest representable number	3.4028234664e+38	1.7976931349e+308
Magnitude of smallest normalized representable number	1.1754943508e-38	2.2250738585e-308

Layouts of these three fields are shown in Figures 9.1 and 9.2 below. The fields are:

- 1-bit sign, *s*
- Biased exponent, $e = E + bias$
- Binary fraction, $f = .b_1\ b_2..b_{p-1}$ (the *b*0 bit is *hidden*; it is not recorded)

Figure 9.1 Single-Precision Floating-Point Format (S)

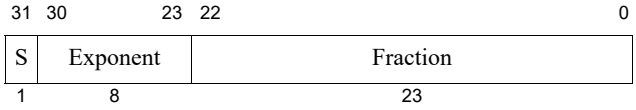
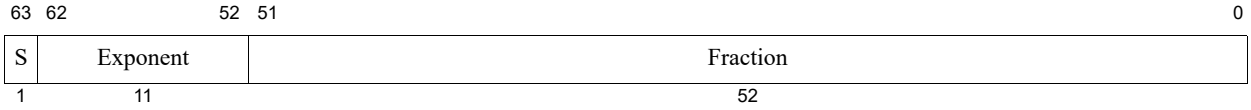


Figure 9.2 Double-Precision Floating-Point Format (D)



Values are encoded in the specified format using the unbiased exponent, fraction, and sign values listed in Table 9.2. The high-order bit of the Fraction field, identified as b_1, is also important for NaNs.

Table 9.2 Value of Single or Double Floating-Point Data Type Encoding

Unbiased E	f	s	b_1	Value V	Type of Value	Typical Single Bit Pattern[1]	Typical Double Bit Pattern[1]
E_max + 1	≠ 0		1	SNaN	Signaling NaN (*FCSR* = 0)	`0x7fffffff`	`0x7fffffff ffffffff`
			0	QNaN	Quiet NaN (*FCSR* = 0)	`0x7fbfffff`	`0x7ff7ffff ffffffff`

Table 9.2 Value of Single or Double Floating-Point Data Type Encoding *(continued)*

Unbiased E	f	s	b_1	Value V	Type of Value	Typical Single Bit Pattern[1]	Typical Double Bit Pattern[1]
E_max + 1	≠ 0		0	SNaN	Signaling NaN (*FCSR* = 1)	0x7fbfffff	0x7ff7ffff ffffffff
			1	QNaN	Quiet NaN (*FCSR* = 1)	0x7fffffff	0x7fffffff ffffffff
E_max +1	0	1		− ∞	Minus infinity	0xff800000	0xfff00000 00000000
		0		+ ∞	Plus infinity	0x7f800000	0x7ff00000 00000000
E_max to *E_min*		1		- $(2^E)(1.f)$	Negative normalized number	0x80800000 through 0xff7fffff	0x80100000 00000000 through 0xffefffff ffffffff
		0		+ $(2^E)(1.f)$	Positive normalized number	0x00800000 through 0x7f7fffff	0x00100000 00000000 through 0x7fefffff ffffffff
E_min -1	≠ 0	1		- $(2^{E\_min})(0.f)$	Negative denormalized number	0x807fffff	0x800fffff ffffffff
		0		+ $(2^{E\_min})(0.f)$	Positive denormalized number	0x007fffff	0x000fffff ffffffff
E_min -1	0	1		- 0	Negative zero	0x80000000	0x80000000 00000000
		0		+ 0	Positive zero	0x00000000	0x00000000 00000000

1. The "Typical" nature of the bit patterns for the NaN and denormalized values reflects the fact that the sign might have either value (NaN) and that the fraction field might have any non-zero value (both). As such, the bit patterns shown are one value in a class of potential values that represent these special values.

9.3.1.1 Normalized and Denormalized Numbers

For single and double data types, each representable nonzero numerical value has just one encoding; numbers are kept in normalized form. The high-order bit of the p-bit mantissa, which lies to the left of the binary point, is "hidden," and not recorded in the *Fraction* field. The encoding rules permit the value of this bit to be determined by looking at the value of the exponent. When the unbiased exponent is in the range *E_min* to *E_max*, inclusive, the number is normalized and the hidden bit must be 1. If the numeric value cannot be normalized because the exponent would be less than *E_min*, then the representation is denormalized, the encoded number has an exponent of *E_min* – 1, and the hidden bit has the value 0. Plus and minus zero are special cases that are not regarded as denormalized values.

9.3.1.2 Reserved Operand Values—Infinity and NaN

A floating-point operation can signal IEEE exception conditions, such as those caused by uninitialized variables, violations of mathematical rules, or results that cannot be represented. If a program does not trap IEEE exception conditions, a computation that encounters any of these conditions proceeds without trapping but generates a result indicating that an exceptional condition arose during the computation. To permit this case, each floating-point format defines representations (listed in the table above) for plus infinity (+∞), minus infinity (–∞), quiet non-numbers (QNaN), and signaling non-numbers (SNaN).

9.3.1.3 Infinity and Beyond

Infinity represents a number with magnitude too large to be represented in the given format; it represents a magnitude overflow during a computation. A correctly signed ∞ is generated as the default result in division by zero operations and some cases of overflow.

Once created as a default result, ∞ can become an operand in a subsequent operation. The infinities are interpreted such that -∞ < (every finite number) < +∞. Arithmetic with ∞ is the limiting case of real arithmetic with operands of arbitrarily large magnitude, when such limits exist. In these cases, arithmetic on ∞ is regarded as exact, and exception conditions do not arise. The out-of-range indication represented by ∞ is propagated through subsequent computations. For some cases, there is no meaningful limiting case in real arithmetic for operands of ∞.

9.3.1.4 Signalling Non-Number (SNaN)

SNaN operands cause an Invalid Operation exception for arithmetic operations. SNaNs are useful values to put in uninitialized variables. An SNaN is never produced as a result value.

IEEE Standard 754 states that "Whether copying a signaling NaN without a change of format signals the Invalid Operation exception is the implementor's option." The RISC-V sign injection instructions are non-arithmetic; they do not signal IEEE 754 exceptions.

9.3.1.5 Quiet Non-Number (QNaN)

QNaNs provide retrospective diagnostic information inherited from invalid or unavailable data and results.

QNaN operands do not cause arithmetic operations to signal an exception. When a floating-point result is to be delivered, a QNaN operand causes an arithmetic operation to supply a QNaN result. QNaNs do have effects similar to SNaNs on operations that do not deliver a floating-point result—specifically, comparisons. For more information, see the detailed description of the floating-point compare instruction, fcmp.

When certain invalid operations not involving QNaN operands are performed but do not trap (because the trap is not enabled), a new QNaN value is created. Table 9.3 shows the QNaN value generated. The values listed for the fixed-point formats are the values supplied to satisfy IEEE Standard 754 when a QNaN or infinite floating-point value is converted to fixed point. There is no other feature of the architecture that detects or makes use of these "integer QNaN" values.

Table 9.3 Value Supplied When a New Quiet NaN is Created

Format	QNaN value (*FCSR* = 1)
Single floating point	0x7FC0_0000
Double floating point	0x7FF8_0000_0000_0000
Word fixed point	0x7FFF_FFFF (value when converting any FP number too big to represent as a 32-bit positive integer) 0x0000_0000 (value when converting any FP NaN) 0x8000_0000 (value when converting any FP number too small to represent as a 32-bit negative integer)
Longword fixed point	0x7FFF_FFFF_FFFF_FFFF (value when converting any FP number too big to represent as a 64-bit positive integer) 0x0000_0000 (value when converting any FP NaN) 0x8000_0000 (value when converting any FP number too small to represent as a 64-bit negative integer)

9.3.2 Signed Integer Formats

The FPU instruction set provides the following signed integer data types:

- A 32-bit Word fixed point (type W), shown in Figure 9.3.
- A 64-bit Longword fixed point (type L), shown in Figure 9.4.

The fixed-point values are held in 2's complement format, which is used for signed integers in the CPU. Unsigned fixed-point data types are not provided by the architecture; application software can synthesize computations for unsigned integers from the existing instructions and data types.

Figure 9.3 Word Fixed-Point Format (W)

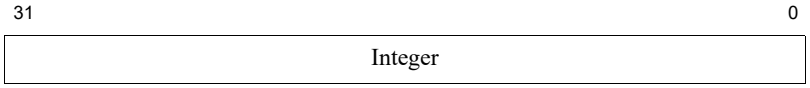
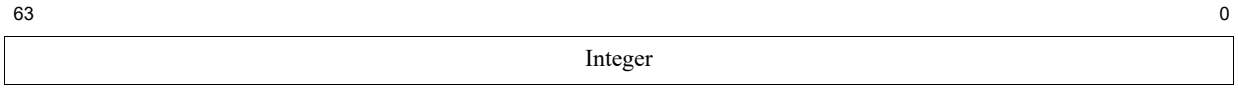


Figure 9.4 Longword Fixed-Point Format (L)



9.4 Floating-Point General Registers

This section describes the organization and use of the Floating-Point general Registers (FPRs). There are thirty-two 64-bit FPU registers.

9.4.1 FPRs and Formatted Operand Layout

FPU instructions that operate on formatted operand values specify the Floating-Point Register (FPR) that holds the value. Operands that are only 32 bits wide (*W* and *S* formats) use only half the space in an FPR.

Figures 9.5 and 9.6 show the FPR organization and the way that operand data is stored in them.

Figure 9.5 Single Floating-Point or Word Fixed-Point Operand in an FPR

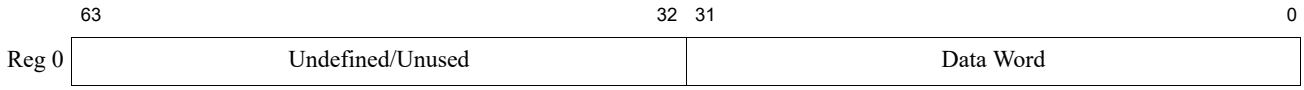
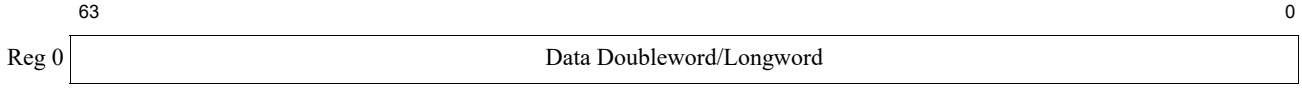


Figure 9.6 Double Floating-Point or Longword Fixed-Point Operand in an FPR



Chapter 10

Multithreading

The P8700-F Multiprocessing System (MPS) incorporates hardware multithreading that executes multiple threads in such a way that the threads appear to be run in parallel. This functionality is performed entirely in hardware and does not require any software control. Hence this chapter is only intended to provide an overview of multithreading and how it is implemented in the P8700-F MPS.

In the P8700-F, each thread is referred to as a Hart. Each Hart contains a complete system state (General, CSR, and FP registers, TLB mappings, interrupt and exception model). In addition, each thread has its own system debug, reset and various boot and exception vectors, and memory coherency.

There are multiple types of multithreading implementations. The P8700-F implements Simultaneous Multithreading, where the core can execute multiple threads in parallel every cycle. In addition, instructions from different threads can execute at the same time in the same pipeline stage. This allows for maximum throughput and minimization of idle hardware during execution. The P8700-F is a quad-issue machine, allowing up to two threads to execute in a single pipeline stage. In the P8700-F, all threads (up to 2) can be fetched, decoded, issued, executed, and graduated in parallel.

10.1 Instruction Flow

The P8700-F Instruction Fetch Unit (IFU) fetches instructions from a shared Instruction Cache (IC) for all two threads. It fetches four instructions (for a single thread) in a cycle, using a program counter (PC) for that thread. This quad of instructions are sent to the Instruction Decode Unit (IDU) and then on to the Execution Unit (EXU). The IFU fetches instructions in a round-robin manner.

The IFU also manages a shared Instruction TLB (ITLB) structure. The ITLB performs instruction address translation, allowing complete independence amongst threads. This ITLB is backed up by the larger Variable TLB (VTLB) and Fixed TLB (FTLB). The number of shared ITLB entries depends on the number of Harts implemented.

- 1 Hart = 6 entries
- 2 Harts = 12 entries

For example, if there is only one Hart, all entries of the ITLB are used by the Hart. Conversely, if there are two Harts, there are 12 ITLB entries that are shared between all of the Harts.

The thread's Instruction Virtual Address (IVA) is translated to Root Physical address (RPA). The ITLBs are used to store the double translation to minimize the number of entries and more importantly to improve performance by doing the translation in a single cycle.

The translated instructions are passed to the Execution Unit (EXU), which is responsible for decoding, issuing, executing and graduating the instructions. In addition, the EXU resolves all data and resource conflicts and manages precise exceptions. In the P8700-F, the instructions are issued out-of-order and graduated in-order.

Every cycle, the EXU decodes the top four instructions from each of the (up to) two threads.

Decoded instructions are inserted into an issue queue and are issues out-of-order based on resource availability and data dependencies.

It is capable of issuing instructions from any of the two Harts, hence the term Simultaneous Multithreading (SMT). If multiple instructions (>2) are available to issue, the EXU uses a fair issue policy to make sure all threads get equal representation.

Once the instructions are issued, they are executed in one of the functional units. During its execution, each instruction is appropriately tagged for thread identification and instruction order. This allows the proper instruction order to be maintained at graduation (completion) time. If an instruction completes, but an earlier instruction from the same thread has not graduated, the completed instruction remains in the graduation queue to maintain in-order completion.

10.2 Data Flow

Like the IFU mentioned above, the Load-Store Unit (LSU) manages a shared data cache to perform loads and stores for all threads. The LSU also performs a load and a store for either or both threads in a cycle. Multiple loads and stores from differing threads can be queued up to access the data cache.

The LSU processes loads and stores out-of-order and maintains cache coherency between threads. The data cache is organized as 4-way set associative cache, which eliminates most of the cache conflicts.

The LSU also manages a shared Data TLB (DTLB) structure. The DTLB performs data address translation, allowing complete independence amongst threads. The shared DTLB is backed up by the larger Variable TLB (VTLB) and Fixed TLB (FTLB).

The number of ITLB uTLB entries is determined by an ifdef in the logic and can be allocated in groups of 8 entries. The P8700 defines the number of groups as 2, for a total of 16 uTLB entries

In addition, the 16, 32, or 64 dual-entry Variable TLB (VTLB) is instantiated on a per-Hart basis. The 512 dual-entry Fixed TLB (FTLB) is shared between all Harts.

The thread's Data Virtual Address (DVA) is translated to a Root Physical address (RPA). The DTLBs operate much like the ITLBs to perform a translation in a single cycle.

Data stored by one thread does not become visible to other threads until the store instruction has graduated and the core has obtained ownership of the associated cache line (for cache-able accesses). In other words, data stored by one thread becomes visible to other threads in the same core at exactly the same point that it becomes visible to other cores in the system.

The P8700-F manages allocation of shared resources (such as data buffers) between threads to prevent starvation and ensure that all threads can make forward progress.

10.3 Independent Exception Model

Since each thread has a completely independent exception model, one thread cannot block another thread. This independent exception model includes: Synchronous Exceptions (Over-flow, TLB Miss, etc.), Asynchronous Interrupts (Int, NMI, etc.), Debug Exceptions (DInt),

and Reset. A thread can be reset to reboot, while the other threads are completely unaffected.

Chapter 11

Performance Counters

This section describes the performance counters for the core and CM3 blocks in the P8700-F Multiprocessing System.

- Section 11.1 “Core Performance Counters”
- Section 11.2 “CM3 Performance Counters”

11.1 Core Performance Counters

The P8700-F core contains four performance counters. Each counter has a Control register (mhpmevent) and an associated Count (mhpmcounter) register. Therefore, there are four Control registers an four Count registers per Hart. These registers are located at the following CSR locations.

Table 11.1 Core Performance Counter Registers

Register Name	Register Acronym	CSR Register Index
Performance Counter Control 3	mhpmevent3	0x323
Performance Counter Control 4	mhpmevent4	0x324
Performance Counter Control 5	mhpmevent5	0x325
Performance Counter Control 6	mhpmevent6	0x326
Performance Counter Count 3	mhpmcounter3	0xB03
Performance Counter Count 4	mhpmcounter4	0xB04
Performance Counter Count 5	mhpmcounter5	0xB05
Performance Counter Count 6	mhpmcounter6	0xB06

Each register is instantiated per-Hart. Therefore in a 2-Hart core, there are eight total *mhpmevent* registers and eight total *mhpmcounter* registers.

11.1.1 Performance Event Masking

The four mhpmevent registers allows for the masking of event counting for the following modes:

- M-mode (Machine)
- S-mode (Supervisor)
- U-mode (User)

When the corresponding bit is set as defined in Table 11.2, that mode is prohibited from counting events.

11.1.2 Core Performance Event Control Register (mhpmevent[6:3])

The four performance counter control registers (instantiated per Hart) at the locations shown in Table 11.1 above each have identical bit assignments. Therefore, only one register is shown below.

Figure 11.1 Performance Counter Control Register Format

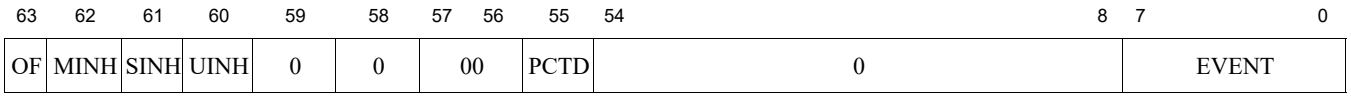


Table 11.2 Performance Counter Control Register Bit Descriptions

Bits	Name	Reset Val	Read/ Write	Description
63	OF	Undefined	R/W	OverFlow. Set when counter overflows. When overflow occurs with OF set, interrupt generation is disabled.
62	MINH	Undefined	R/W	Machine Inhibit. Inhibit counting of events in M-mode.
61	SINH	Undefined	R/W	Supervisor Inhibit. Inhibit counting of events in S-mode.
60	UINH	Undefined	R/W	User Inhibit. Inhibit counting of events in U-mode.
59:56	0	Undefined	R/W	Write as zero.
55	PCTD	Undefined	R/W	Performance Counter Trace Disable.
54:8	0	Undefined	R/W	Write as zero
7:0	EVENT	Undefined	WARL	Encoding of event to be monitored by the specified hardware performance monitor, with 0 meaning no event. The encoding for this field is shown in Table 11.5.

11.1.3 Core Performance Counter Count Register (mhpmcounter[6:3])

Each Performance Counter Control register described above has an associated Count register that counts the number of events as indicated by the EVENT field of the Control register. Refer to Table 11.1 for a listing and location of these registers. The Performance Counter Count registers are instantiated per-Hart.

Figure 11.2 Performance Counter Count Register Format

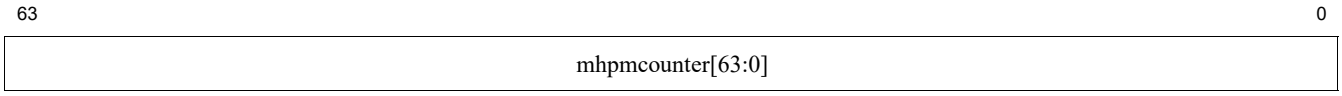


Table 11.3 Performance Counter Count Register Bit Descriptions

Name	Bits	Reset Val	Read/ Write	Description
mhpmcounter	63:0	Undefined	RW	Increments once for each event that is enabled by the corresponding Control Register. For example, if bit 62 (MINH) of the mhpmevent[3] register is cleared, then the value in the mhpmcounter[3] register will increment each time there is an M-mode event in Control register 3.

11.1.4 Core Performance Counter Events

The table below shows the encoding of the EVENT field in bits 7:0 of each Performance Counter Control register.

In the following table:

- All events are local to the Hart running except #128.
- All events are available to all performance counters.
- Event counting is edge counting; that is, an event occurs when the signal goes from not TRUE to TRUE.

Table 11.4 Core Performance Counter Events

Event ID	Event Name	Description
Execution Units		
1	num_grad	Number of graduated instructions
2	one_grad	Number of cycles in which one instruction graduated
3	two_grad	Number of cycles in which two instruction graduated
4	three_grad	Number of cycles in which three instruction graduated
5	four_grad	Number of cycles in which four instruction graduated
6	no_grad	Number of cycles in which no instruction graduated
7	alu_grad	Number of ALU instructions graduated
8	lsu_grad	Number of LSU instructions graduated
9	cti_grad	Number of CTI instructions graduated
10	mdu_grad	Number of MDU instructions graduated
11	fpu_grad	Number of FPU instructions graduated
12	load_grad	Number of LOAD instructions graduated
13	store_grad	Number of STORE instructions graduated

Table 11.4 Core Performance Counter Events*(continued)*

Event ID	Event Name	Description
14	no_isu	Number of cycle in which no instructions issued
15	one_isu	Number of cycle in which one instructions issued
16	two_isu	Number of cycle in which two instructions issued
17	three_isu	Number of cycle in which three instructions issued
18	four_isu	Number of cycle in which four instructions issued
19	isu_block	Number of times the Issue unit got stalled
20	dec_stall	Number of times the Decoder unit got stalled
21	dmap_stall	Number of times the Dependency Mapper stalled
22	ibfr_empty	Cycles in which instruction buffer is empty
23	itrkr_num_replay	Replays initiated by the scoreboard
24	br_grad	Conditional branches graduated
25	br_miss_grad	Mispredicted conditional branches graduated
26	jr_ret_grad	Returns (JR $31) graduated
27	jr_ret_miss_grad	Mispredicted Returns (JR $31) graduated
28	jr_grad	JR graduated
29	jr_miss_grad	Mispredicted JR graduated
30	br_t_grad	Taken conditional branches graduated
31	br_nt_grad	Not taken conditional branches graduated
32	redirect	Total redirects
33	num_exceptions	Total number of exceptions
34	exe_redir	EXE-redirect for a "Ret".
35	load_blocked	Number of cycles graduation was blocked of a load waiting to complete
36	sync_blocked	Number of cycles graduation was blocked of sync waiting to complete
37	rftch_inbnd_oel	OverEager loads (OEL) Misprediction update
38	rftch_inbnd_misalgn	Misaligned misprediction update
Load/Store Units		
64	dtlb_lookup	DTLB Lookups
65	dtlb_miss_new	DTLB Misses (new)
66	dtlb_miss_merge	DTLB Misses (merged with existing)
67	bond_load	Bonded Load
68	bond_store	Bonded Store
69	total_dcache_lookups	Total number of cache lookups
70	loads_dcache_lookup	Number of Load-type instns
71	stores_dcache_lookup	Number of Store-type instns
72	total_dcache_misses	Misses cache lookup
73	load_dcache_misses	Loads miss cache lookup
74	store_dcache_misses	Stores miss cache lookup
75	smb_full	Number of cycles SDB graduation was blocked due to SMB full

Table 11.4 Core Performance Counter Events*(continued)*

Event ID	Event Name	Description
76	DCACHE load Retries	Number of DCACHE Load Retries
77	L1 Prefetch from Prefetc tracker 0	L1 Prefetch Issued from Prefetch Tracker0
78	L1 Prefetch from Prefetc tracker 1	L1 Prefetch Issued from Prefetch Tracker1
79	L1 Prefetch from Prefetc tracker 2	L1 Prefetch Issued from Prefetch Tracker2
80	L1 Prefetch from Prefetc tracker 3	L1 Prefetch Issued from Prefetch Tracker3
81	L1 Prefetch from Prefetc tracker 4	L1 Prefetch Issued from Prefetch Tracker4
82	L1 Prefetch from Prefetc tracker 5	L1 Prefetch Issued from Prefetch Tracker5
83	L1 Prefetch from Prefetc tracker 6	L1 Prefetch Issued from Prefetch Tracker6
84	L1 Prefetch from Prefetc tracker 7	L1 Prefetch Issued from Prefetch Tracker7
85	rftch_inbnd_prd	Refetch due to Incorrect Bonding Prediction
Instruction Fetch Unit		
128		(Global) All Harts currently stalled
129	utb_access	Micro-tlb accesses
130		Number of times the Hart stalled waiting for MMU response to uTLB
131	utb_miss	Micro-tlb misses
132	ica_access	ICache accesses
133	ica_miss	ICache misses
134	ibuff_cred_stall	Instruction fetch stalled due to lack of IBUF credit
135		Number of times the Hart stalled waiting for PCBuffer credit.
136		Number of times the Hart stalled
137	redirect_stall	Redirect stall cycles performance count
138	ica_miss_stall	ICache miss stall cycles
139	uncached_stall	Uncached stall cycles
140	prb_full	PRB full, No JRC, RPC predictions
141	l1btb_rps_hit	True L1BTB RPS hits
142	l1btb_prs_mispred	L1BTB RPS mispredicts
143	rps_hit_l1btb_miss	RPS hits without L1BTB hit
144	ica_way_mispred	NFW or L1BTB based I$ way-hit mispredict
145	jtlb_miss	TLB Misses
146	no_spec_fetch_maar	Speculative Fetch restricted due to MAAR. No Allocation to RFB
147	fetch_kill_cti	Killed slots in fetch due to target/cti stall
148	no_ifu_2_idu_inst	IDU gate open, no IFU inst
149	l1btb_non_rps_hit	True L1BTB non-RPS hits
150	l1btb_non_rps_mispred	L1BTB Non RPS mispredicts
151	jal_b_hit_l1btb_miss	JAL/ Taken BRANCH from IB without L1BTB Hit
152	l1btb_hit_masked	L1BTB hit masked due to lack of credit to fetch Target
Trace Unit		
160		PDTrace backpressure stall

Table 11.4 Core Performance Counter Events*(continued)*

Event ID	Event Name	Description
161		PDTrace backpressure stall cycles
162		PDTrace overflow
163		PDTRace LSU backpressure stall

11.2 CM3 Performance Counters

11.2.1 CM3 Performance Counter Functionality

Performance characteristics of the CM3 can be measured via the CM3 performance counters. Two sets of identical programmable 32-bit performance counters in addition to a 32-bit cycle counter are implemented. The counters are controlled and accessed via GCR registers described in Chapter 8, "Coherency Manager". This section describes the operation of those registers.

The counters are started by writing a 1 to the *P0_CountOn*, *P1_CountOn* and *Cycl_Cnt_CountOn* bits in the *CM3 Performance Counter Control* Register (GCR_DB_PC_CTL Offset 0x0100). Each counter can be reset to 0, and the corresponding overflow bit (P0_OF, P1_OF, Cyc_Cnt_OF) is reset to 0 prior to the start of counting by writing a 1 to the *P0_Reset*, *P1_Reset* and *Cycl_Cnt_Reset* bits in the same access that sets the corresponding start bits. This functionality allows all three counters to be reset and started with a single GCR write.

The *CM3 Performance Counter Control* Register also controls how a counter overflow is handled. If the *Perf_Ovf_Stop* bit is set to 1, then all CM Performance counters will stop when one of the counters (including the Cycle Counter) reaches its maximum value of 0xFFFFFFFF. If instead the *Perf_Ovf_Stop* bit is set to 0, when a counter overflows, it rolls over and continues counting from 0.

If the *Perf_Int_En* bit is set to 1, an interrupt is generated when one of the counters (including the cycle counter) reaches its maximum value of 0xFFFFFFFF. The CM3 asserts the *so_cm_perf_cnt_int* signal which generates an interrupt only if the System Integrator has connected the *so_cm_perf_cnt_int* signal to one bit of *si_cm_int*.

When a performance counter overflows, the corresponding bit is automatically set in the *CM3 Performance Counter Overflow Status Register (GCR_DB_PC_OV).* A status bit is cleared by writing a 1 to it.

The event to be counted by each performance counter is designated by the event number set in the *P0_Event* and *P1_Event* fields of the *CM3 Performance Counter Event Select Register (GCR_DB_PC_EVENT)*. The events corresponding to the event numbers are listed and described in Table 11.5, "CM3 Performance Counter Event Types," on page 335.

Each event is further specified by the *CM3 Performance Counter Qualifier Register (GCR_DB_PC_QUALn)*. The meaning of this register is different for each event. The column labeled "Qualifier" in Table 11.5 shows the qualifiers that can be specified for each event. For example, the qualifiers for the Coherence Manager Request Event (event 1) are the request port, thread, cmd, CCA, size, etc.

The qualifiers for some events are composed of several groups. A performance counter will increment if the specified event occurs and the qualifier criteria is matched in all groups. For example, assume the *P0_Event* field in the *CM3 Performance Counter Event Select Register* is set to 1 (Coherence Manager Request). This event occurs when the CM3 serializes a request. However, the performance counter for this event will only count if the request meets the criteria programmed in all 12 groups in the Request Qualifier (see Table 11.5):

```
The port that issued the request has the corresponding Port qualifier bit set to 1.
AND
The thread that issued the request has the corresponding Thread qualifier bit set
to 1.
AND
The target of the request has the corresponding bit of the Target qualifier set to
1.
AND
The request command type has the corresponding Request Command qualifier bit set to
1.
AND
The Cachebility attribute (CCA) for the request has the corresponding CCA qualifier
bit set to 1.
AND
The size of the request has the corresponding Size qualifier bit set to 1.
AND
The L1 State of the request has the corresponding L1 State qualifier bit set to 1.
AND
The L2 state of the request has the corresponding L2 State qualifier bit set to 1.
AND
The L2 Locked state of the request has the corresponding L2 Locked qualifier bit
set to 1.
AND
The resulting eviction due to the request has the Eviction qualifier bit set to 1.
AND
The bank of the request has the Bank qualifier bit set to 1.
AND
The scheduler used for the request has the Scheduler qualifier bit set to 1.
```

Multiple bits within a qualification group may be set. In this case, the OR of all bits set within the group. For example, by setting the request port qualifier for Port 0 and Port 1, then a request will be counted if it originated from Port 0 or Port 1.

A qualifier group can be set to "don't care" by setting all bits within the group to 1. For example, to have performance counter 0 count all requests from port 1, program the *CM Performance Counter Event Select Register* and *CM Performance Counter Qualifier 0 Register* as follows:

```
Set P0_Event to 1(Coherence Manager Request)
Set Request Port Qualifer bit to 1 for Port 1
Set Request Port Qualifier bits to 0 for all other Ports
Set all other qualifer bits to 1 (causing the Thread, Target, Command, CCA, etc to
be ignored)
```

The two counters can be programmed to count a different event or the same event with different qualifiers. For example, to measure the ratio of requests from Port 1 vs. all Ports, set program Counter 0 to count requests from Port 1 (see previous example) and program Counter 1 to count all request from all Ports by setting *P1_Event* to 1 (Coherence Manager Request) and set *all* bits in the *CM Performance Counter Qualifier 1* Register to 1.

The cycle counter can be used to calculate the average rates of specified events. Continuing the above example, assuming the cycle counter is reset, started, and stopped simultaneously with the two performance counters, then the rate of requests from port 1 and all ports can be easily computed (value of each performance counter / value in cycle counter).

11.2.2 CM3 Performance Counter Usage Models

There are several models for using the CM3 performance counters. This sections discusses 3 possible models:

- Periodic Sampling - take many measurement samples of specific duration
- Stop and Interrupt when counter overflows - counters run until one overflows, then interrupt CPU
- Large count capability - enables unrestricted sample periods

One model for making performance measurements is for the software to set up and gather samples for a set period of time. The code sequence could follow the following steps:

```
start:
Write CM Event and Qualifier Registers for particular event of interest
Write CM Performance Counter Control Register to reset and start counters
Perf_Int_En = 0 (no interrupt on overflow)
Perf_Ovf_Stop = 0(no stop on overflow).
P1_Reset = 1, P1_CountOn = 1
P0_Reset = 1, P0_CountOn = 1
Cycl_Cnt_Reset = 1, Cycl_Cnt_CountOn = 1
Wait for some relatively small period of time (i.e., 2 seconds)
Write CM Performance Counter Control Register to stop counters
P1_Counton = 0, P0_CountOn=0, Cycl_Cnt_CountOn = 0
Read CM Performance Counter 0, Counter 1, and Cycle Counter Registers
If more events, go to start (or if measuring same counter go to step 2 instead)
```

A second CM3 performance counter usage model involves setting up the counters to stop and interrupt on overflow. This runs the counters until one of the counters (usually the cycle counter) reaches the 32-bit limit. An example of such a code sequence is:

```
start:
Write CM Event and Qualifier Registers for particular event of interest
Write CM Performance Counter Control Register to reset and start counters
Perf_Int_En = 1 (interrupt on overflow)
Perf_Ovf_Stop = 1(stop on overflow).
P1_Reset = 1, P1_CountOn = 1
P0_Reset = 1, P0_CountOn = 1
Cycl_Cnt_Reset = 1, Cycl_Cnt_CountOn = 1
When interrupt occurs:
Read CM Performance Counter Status Register
Read CM Performance Counter 0, Counter 1, and Cycle Counter Registers
Write CM Performance Counter Control Register to reset counters
(clears status register and interrupt)
P0_Reset = 1, P1_Reset = 1, Cycl_Cnt_Reset = 1
If more events, go to start (or if measuring same counter go to step 2 instead)
```

If larger counts than can fit into the 32-bit counters are required, the counters can be set up to interrupt, but not stop, on overflow. Memory variables can then count the number of overflows, as shown below:

```
start:
Write CM Event and Qualifier Registers for particular event of interest
Write CM Performance Counter Control Register to reset and start counters
Perf_Int_En = 1 (interrupt on overflow)
Perf_Ovf_Stop = 0 (do not stop on overflow).
P1_Reset = 1, P1_CountOn = 1
P0_Reset = 1, P0_CountOn = 1
Cycl_Cnt_Reset = 1, Cycl_Cnt_CountOn = 1
When interrupt occurs:
<status>=Read CM Performance Counter Status Register
Increment <overflow_count>[counter] for each counter with <status> = 1
Write <status> to CM Performance Counter Status Register to clear interrupt

When run limit is reached then :
```

```
Write CM Performance Counter Control Register to stop counters
P1_Counton = 0, P0_CountOn=0, Cycl_Cnt_CountOn = 0
Read CM Performance Counter 0, Counter 1, and Cycle Counter Registers
Write CM Performance Counter Control Register to reset counters
(clears status register and interrupt)
P0_Reset = 1, P1_Reset = 1, Cycl_Cnt_Reset = 1
If more events, go to start (or if measuring same counter go to step 2 instead)
```

In the above model, the final counts are calculated for each counter by multiplying `<overflow_count>[counter]` by 4G and adding the final values in the performance counter register.

Table 11.5 CM3 Performance Counter Event Types

Event #	Related Events	Qualifiers	Description/Comments
0	None	No events are enabled for counting. This is the lowest power mode.	
1	Coherence Manager Requests	Port Thread Target Cmd Prefetch CCA Size L1 State L2 State L2 Locked Eviction Bank Scheduler	Can be used in conjunction with a cycle count to determine the number of requests received in a given period of time. Refer to Table 11.6 for more information.
2	I/O Traffic Requests	Which IOCU Direction/Cacheability Size Length Prefetch Device ID Transaction ID	Counts the requests received by the IOCU. Refer to Table 11.7 for more information.
3	Memory Interface Requests	Direction Size Length Cacheability Source Thread Code/data Prefetch	Counts the number of Memory requests issued. Refer to Table 11.8 for more information.
7	MEM AXI Bus Utilization	channel ready valid	Measure Utilization of main memory AXI/ACE bus Refer to Table 11.9 for more information.

Table 11.5 CM3 Performance Counter Event Types *(continued)*

Event #	Related Events	Qualifiers	Description/Comments
8	IOCU0 AXI Bus Utilization	channel ready valid	Measure Utilization of corresponding IOCU bus Refer to Table 11.9 for more information.
9	IOCU1 AXI Bus Utilization		
10	IOCU2 AXI Bus Utilization		
11	IOCU3 AXI Bus Utilization		
12	IOCU4 AXI Bus Utilization		
13	IOCU5 AXI Bus Utilization		
14	IOCU6 AXI Bus Utilization		
15	IOCU7 AXI Bus Utilization		
17	CM PDTrace Dropped Messages	responses requests port enables	Count number of messages dropped by CM Trace due to overflow. Refer to Table 11.10for more information.
18	CM PDTrace overflow cycle length	None	Counts the number of clock cycles for which CM PDTrace overflow took to finish,

Table 11.6 Coherence Manager Request Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
63	2	Reserved	unused	Reserved for future use. Set all bits to 1.
62	1			
61	0			
60	1	Scheduler	Scheduler 1	Request processed by CM scheduler 1.
59	0		Scheduler 0	Request processed by CM scheduler 0.
58	1	Bank	Bank 1	Request sent to L2 bank 1.
57	0		Bank 0	Request sent to L2 bank 0.
56	2	Eviction	L2 eviction no L1 eviction	Request causes an L2 eviction but not and L1 eviction.
55	1		L2 eviction with L1 eviction	Request causes both an L2 and L1 eviction.
54	0		no L2 eviction	Request does not cause an eviction.
53	1	L2 Locked	Locked	L2 line is valid and locked.
52	0		Not locked	L2 line is not locked (or the line is invalid).
51	3	L2 State	Modified	L2 line is in state modified.
50	2		Exclusive	L2 line is in state exclusive.
49	1		Shared	L2 line is in state shared.
48	0		Invalid	L2 line is invalid.
47	2	L1 State	Exclusive/Modified	Line is Exclusive or Modifed in one of the cores.
46	1		Shared	Line is Shared in at least one of the cores.
45	0		Invalid	Line is not valid in any of the core L1s.
44	1	Size	line	Request for 1 cache line of data. Note: This counts the burst length as seen by the Coherent Manager. Requests form the I/O Sub-system may be longer, but the IOCU may break these into multiple smaller requests.
43	0		Less than a line	Request for less than a cache line.
42	2	CCA	Other	Request hasa cacheability attribute other than UC/ UCA.
41	1		UCA	Request has an accelerated un-cached cacheabil-ity attribute.
40	0		UC	Request has an un-cached cacheability attribute.

Table 11.6 Coherence Manager Request Qualification *(continued)*

Bit		Qualifier Group	Qualifier Value	Description/Comments
39	23	Request command	Other command	
38	22		L3 Cache	all L3 cacheop including FetchNLock.
37	21		L2 Cache	
36	20		L1D Cache	
35	19		L1I Cache	
34	18		Sync	
33	17		RegWrite	
32	16		RegRead	
31	15		Tag_Err	
30	14		GetToOwn	
29	13		Prefetch Write Invalidate	
28	12		Prefetch Share	
27	11		Prefetch Own	
26	10		CohReadDiscardAllocate	
25	9		CohWriteInvalidate	
24	8		CohWriteBack	
23	7		CohUpgradeSC	
22	6		CohUpgrade	
21	5		CohEvict	
20	4		CohReadDiscard	
19	3		CohReadShare	
18	2		CohReadOwn	
17	1		Legacy Write	Request is a legacy write command. This is used for all non-coherent writes. Note: When a processor is in coherent mode, L1 cache writebacks are always considered coherent, so the result is a CohWriteBack command, not a Legacy Write command.
16	0		Legacy Read	Request is a legacy read command. This is used for all non-coherent reads, including code fetches.
15	1	Target	Register bus target	Request targets a device on the register bus such as GCR, GIC, CPC, DBU, etc.
14	0		Memory	Request targets memory (coherent or non-coherent).
13				Reserved.

Table 11.6 Coherence Manager Request Qualification *(continued)*

Bit		Qualifier Group	Qualifier Value	Description/Comments
12				Reserved.
11	1	Thread	Thread 1	Request originated from thread 1.
10	0	Thread	Thread 0	Request originated from thread 0.
9	9	Port	Intervention	Request originated from an intervention.
8	8		Prefetch	Request originated from the prefetcher.
7	7		Port 7	Request originated from Input Port x, x is assigned Cores before IOCU. For example, for a 4 core, 2 IOCU configuration, the ports are assigned as follows: Port 5 : IOCU 1 Port 4: IOCU 0 Port 3: Core 3 Port 2: Core 2 Port 1: Core 1 Port 0: Core 0
6	6		Port 6	
5	5		Port 5	
4	4		Port 4	
3	3		Port 3	
2	2		Port 2	
1	1		Port 1	
0	0		Port 0	

Table 11.7 I/O Traffic Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
41:37	4:0	transaction ID	Specific transaction ID	Match specific transaction ID. This field is used only when All transaction ID is 0.
36	0		All transaction ID	If set, any transaction ID matches, transaction ID group ignored.
35:30	5:0	device ID	Specific device ID	Match specific device ID. This field is used only when All device ID is 0.
29	0		All device ID	If set, any device ID matches, device ID group ignored
28	1	Prefetch	prefetch	IOCU request is a prefetch
27	0		not prefetch	IOCU request is not a prefetch
26	1	Aligned	Misaligned	IOCU request address is not aligned
25	0		Aligned address	IOCU request address is aligned

Table 11.7 I/O Traffic Qualification *(continued)*

Bit		Qualifier Group	Qualifier Value	Description/Comments
24	8	Length	129-256	Number of transfers in a burst.
23	7		65-128	
22	6		33-64	
21	5		17-32	
20	4		9-16	
19	3		5-8	
18	2		3-4	
17	1		2	
16	0		1	
15	7	Size	128	Indicates the number of bytes in each transfer in the burst.
14	6		64	
13	5		32	
12	4		16	
11	3		8	
10	2		4	
9	1		2	
8	0		1	
7	2	Direction/ cacheability	Write - coherent	Coherent write request.
6	1		Write - UC	Uncached write request.
5	0		Read - coherent no allocate	Coherent read request without allocate.
4	1		Read - coherent with allocate	Coherent read request with allocate.
3	0		Read - UC	Uncached read request.
2:0	2:0	IOCU Number	0-7	Encoded value of which IOCU requests to count

Table 11.8 Memory Interface Request Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
45:41	4:0	Guest ID	Specific Guest ID	Match specific guest ID. This field is only used when All Guest ID is 0.
40	0	All Guest ID	All Guest ID	If set, any guest ID matches, Guest ID group ignored.
39	1	Prefetch	Prefetch	Prefetch memory request.
38	0		Not prefetch	Not a prefetch memory request.

Table 11.8 Memory Interface Request Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
37	1	Code/data	Code	Request indicated it was accessing code.
36	0		Data	Request indicated it was accessing data.
35				Reserved.
34				Reserved.
33	1	Thread	Thread 1	Request originated from thread 1.
32	0	Thread	Thread 0	Request originated from thread 0.
31		Reserved		
30				
29	7	Source	Input Port 7	Request originated from Input Port x, x is assigned Cores before IOCU. For example, for a 4 core, 2 IOCU configuration, the ports are assigned as follows: Port 5 : IOCU 1 Port 4: IOCU 0 Port 3: Core 3 Port 2: Core 2 Port 1: Core 1 Port 0: Core 0
28	6		Input Port 6	
27	5		Input Port 5	
26	4		Input Port 4	
25	3		Input Port 3	
24	2		Input Port 2	
23	1		Input Port 1	
22	0		Input Port 0	
21	3	Cacheability	Cacheable not read discard	Any coherent access that is not a read discard.
20	2		Cacheable read discard	Coherent read discard.
19	1		UCA	Uncached Accelerate access.
18	0		UC	Uncached access.
17	7	Length	7	Number of transfers in a burst.
16	6		6	
15	5		5	
14	4		4	
13	3		3	
12	2		2	
11	1		1	
10	0		0	

Table 11.8 Memory Interface Request Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
9	7	Size	128	Indicates the number of bytes in each transfer in the burst.
8	6		64	
7	5		32	
6	4		16	
5	3		8	
4	2		4	
3	1		2	
2	0		1	
1	1	Direction	Write	Write
0	0		Read	Read

Table 11.9 AXI Bus Utilization Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
6:4	2:0	channel	0: AR 1: AW 2: W 3: R 4: B	count transactions on the specified channel
3	1	ready	ready	count when xREADY signal is asserted
2	0		not_ready	count when xREADY signal is not asserted
1	1	valid	valid	count when xVALID is asserted
0	0		not_valid	count when xVALID is not asserted

Table 11.10 CM PDTrace Dropped Message Qualification

Bit		Qualifier Group	Qualifier Value	Description/Comments
7	0	trace_type	response	trace responses. only used for tmh1_mulp, tmh1, tmh0_mulp, tmh0, and ubrh
6	0		request	trace requests. only used for tmh1, tmh0

Table 11.10 CM PDTrace Dropped Message Qualification *(continued)*

Bit		Qualifier Group	Qualifier Value	Description/Comments
5	5	trace_port	tmh1_mulp	Count dropped messages due to multiple-responses for main pipeline #1
4	4		tmh1	Count dropped from main pipeline #1
3	3		tmh0_mulp	Count dropped messages due to multiple-responses for main pipeline #0
2	2		tmh0	Count dropped from main pipeline #0
1	1		prsh	Count messages dropped at perf counter tracing port
0	0		ubrh	Count messages dropped from uncached responses

Chapter 12

Instruction Latencies and Repeat Rates

This chapter provides the instructions latency and repeat rates for the following instruction types.

12.1 Definition of Terms

The terms *latency* and *repeat rate* are defined as follows:

Latency is defined as the minimum time between when an instruction issues, and the time that a subsequent dependent instruction may issue. For example, and ADD instruction has a latency of 1 cycle. Consider the following code sequence:

```
ADD x3, x1, x2
ADD x5, x4, x3
```

In this example the second ADD instruction is dependent on the value placed into r3 by the first ADD instruction. It may issue one cycle after the first ADD instruction issues.

Repeat rate is measured as the minimum issue interval time between independent instructions. For example, a MUL instruction has a latency of 4 cycles and a repeat rate of 1 cycle. Consider the following code sequence:

```
MUL x4, x1, x2
MUH x5, x1, x2
```

The MUL instruction multiplies the r1 and r2 values and places the lower half of the result into r4. The MUH instruction multiples the r1 and r2 values and places the upper half of the result into r5. In this case the MUH can issue one cycle after the MUL instruction issues.

Table 12.1 shows the latency and repeat rates for integer instructions.

Units is number of functional units in the CPU that can execute the instruction.

Unit Type is name of the functional unit that can execute the instruction.

Table 12.1 Instruction Latencies and Repeat Rates

Instruction	Definition	Latency	Rate	Unit Type	Units
ADD	Add	1	1	ALU	2
ADD.UW	Add - unsigned word only	1	1	ALU	2
ADDI	Add immediate	1	1	ALU	2
ADDIW	Add immediate unsigned word	1	1	ALU	2
ADDW	Add unsigned word	1	1	ALU	2
AND	Bitwise logical AND	1	1	ALU	2

Table 12.1 Instruction Latencies and Repeat Rates*(continued)*

Instruction	Definition	Latency	Rate	Unit Type	Units
ANDI	Bitwise logical AND immediate with a constant	1	1	ALU	2
ANDN	And operation with second operand inverted	1	1	ALU	2
AUIPC	Add upper immediate to PC	1	1	AL2	1
BEQ	Branch if GPR values are equal	1	1	CTI	1
BGE	Branch if GPR rs1 is greater than or equal to rs2	1	1	CTI	1
BGEU	Branch if GPR rs1 is greater than or equal to rs2, unsigned	1	1	CTI	1
BLT	Branch if GPR rs1 is less than GPR rs2	1	1	CTI	1
BLTU	Branch if GPR rs1 is less than GPR rs2, unsigned	1	1	CTI	1
BNE	Branch on not equal	1	1	CTI	1
CCMOV	Conditional move - MIPS custom	2	1	AL2	1
CLZ	Count leading zero	2	1	AL2	1
CLZW	Count leading zero word	2	1	AL2	1
CPOP	Count bits set	2	1	AL2	1
CPOPW	Count bits set word	2	1	AL2	1
CSRRC	Move from and clear CSR	4	1	MDU	1
CSRRCI	Move from and clear immediate CSR	4	1	MDU	1
CSRRS	Move from and set CSR	4	1	MDU	1
CSRRSI	Move from and set immediate CSR	4	1	MDU	1
CSRRW	Move from and to CSR	4	1	MDU	1
CSRRWI	Move from and write immediate CSR	4	1	MDU	1
CTZ	Count trailing zero	2	1	AL2	1
CTZW	Count trailing zero word	2	1	AL2	1
DIV	Divide integer signed	7 to 22		DIV	1
DIVU	Divide integer unsigned	7 to 22		DIV	1
DIVUW	Divide word integer unsigned	7 to 22		DIV	1
DIVW	Divide word integer signed	7 to 22		DIV	1
EBREAK	Software debug break point exception	3		IDU/GRU	1
ECALL	Cause a system call exception	3		IDU/GRU	1
EHB	Execution hazard barrier			IDU/GRU	1
FADD.D	Floating point add - double precision	4	1	FPU(EXL)	1
FADD.S	Floating point add	4	1	FPU(EXL)	1
FCLASS.D	floating point classify - double precision	1	1	FPU(EXS)	1
FCLASS.S	floating point classify	1	1	FPU(EXS)	1
FCVT.D.L	Convert integer doubleword to double precision float	4	1	FPU(EXL)	1
FCVT.D.LU	Convert unsigned integer doubleword to double precision float	4	1	FPU(EXL)	1
FCVT.D.W	Convert integer word to double precision float	4	1	FPU(EXL)	1
FCVT.D.WU	Convert unsigned integer word to double precision float	4	1	FPU(EXL)	1

Table 12.1 Instruction Latencies and Repeat Rates*(continued)*

Instruction	Definition	Latency	Rate	Unit Type	Units
FCVT.L.D	Convert double precision to integer doubleword	4	1	FPU(EXL)	1
FCVT.L.S	Convert single precision to integer doubleword	4	1	FPU(EXL)	1
FCVT.LU.D	Convert double precision to unsigned integer doubleword	4	1	FPU(EXL)	1
FCVT.LU.S	Convert single precision to unsigned integer doubleword	4	1	FPU(EXL)	1
FCVT.S.L	Convert integer doubleword to single precision float	4	1	FPU(EXL)	1
FCVT.S.LU	Convert unsigned integer doubleword to single precision float	4	1	FPU(EXL)	1
FCVT.S.W	Convert integer word to single precision float	4	1	FPU(EXL)	1
FCVT.S.WU	Convert unsigned integer word to single precision float	4	1	FPU(EXL)	1
FCVT.W.D	Convert double precision to integer word - double precision	4	1	FPU(EXL)	1
FCVT.W.S	Convert single precision to integer word	4	1	FPU(EXL)	1
FCVT.WU.D	Convert double precision to unsigned integer word - double precision	4	1	FPU(EXL)	1
FCVT.WU.S	Convert single precision to unsigned integer word	4	1	FPU(EXL)	1
FDIV.D	Floating point divide - double precision	17 - 37		FPU(EXL)	1
FDIV.S	Floating point divide	11 - 21		FPU(EXL)	1
FENCE	Order loads and stores	n/a	1	LSU	1
FENCE.I	Synchronize caches for instruction writes	n/a	1	LSU	1
FEQ.D	Floating point equal comparison - double precision	2	1	FPU(EXS)	1
FEQ.S	Floating point equal comparison	2	1	FPU(EXS)	1
FLD	Load doubleword from memory to an FPR	4	1	FPU	1
FLE.D	Floating point less than or equal comparison - double precision	2	1	FPU(EXS)	1
FLE.S	Floating point less than or equal comparison	2	1	FPU(EXS)	1
FLT.D	Floating point less than comparison - double precision	2	1	FPU(EXS)	1
FLT.S	Floating point less than comparison	2	1	FPU(EXS)	1
FLW	Load word from memory to an FPR	4	1	FPU	1
FMADD.D	Floating point fused multiply add - double precision	8	1	FPU(EXL)	1
FMADD.S	Floating point fused multiply add	8	1	FPU(EXL)	1
FMAX.D	Floating point maximum value - double precision	2	1	FPU(EXS)	1
FMAX.S	Floating point maximum value	2	1	FPU(EXS)	1
FMIN.D	Floating point minimum value - double precision	2	1	FPU(EXS)	1
FMIN.S	Floating point minimum value	2	1	FPU(EXS)	1
FMSUB.D	Floating point fused multiply subtract - double precision	8	1	FPU(EXL)	1
FMSUB.S	Floating point fused multiply subtract	8	1	FPU(EXL)	1
FMUL.D	Floating point multiply - double precision	5	1	FPU(EXL)	1
FMUL.S	Floating point multiply	5	1	FPU(EXL)	1

Table 12.1 Instruction Latencies and Repeat Rates*(continued)*

Instruction	Definition	Latency	Rate	Unit Type	Units
FMV.D.X	Move doubleword from GPR to float register	1	1	FPU(EXL)	1
FMV.W.X	Move word from GPR to float register	1	1	FPU	1
FMV.X.D	Move doubleword from floating register to GPR		1	FPU	1
FMV.X.W	Move word from floating register to GPR		1	FPU	1
FNMADD.S	Floating point fused multiply, negate and add	8	1	FPU(EXL)	1
FNMADD.S	Floating point fused multiply, negate and add - double precision	8	1	FPU(EXL)	1
FNMSUB.D	Floating point fused multiply, negate and subtract - double precision	8	1	FPU(EXL)	1
FNMSUB.S	Floating point fused multiply, negate and subtract	8	1	FPU(EXL)	1
FSD	Store doubleword from FPR to memory	n/a	1	FPU	1
FSGNJ.D	Floating point sign injection - double precision	1	1	FPU(EXS)	1
FSGNJ.S	Floating point sign injection	1	1	FPU(EXS)	1
FSGNJN.D	Floating point negated sign injection - double precision	1	1	FPU(EXS)	1
FSGNJN.S	Floating point negated sign injection	1	1	FPU(EXS)	1
FSGNJX.D	Floating point ex-ORed sign injection - double precision	1	1	FPU(EXS)	1
FSGNJX.S	Floating point ex-ORed sign injection	1	1	FPU(EXS)	1
FSQRT.D	Floating point square root - double precision	23 - 24		FPU(EXL)	1
FSQRT.S	Floating point square root	14 - 22		FPU(EXL)	1
FSUB.D	Floating point subtract - double precision	4	1	FPU(EXL)	1
FSUB.S	Floating point subtract	4	1	FPU(EXL)	1
FSW	Store word from FPR to memory	n/a	1	FPU	1
IHB	Instruction hazard barrier			IDU/GRU	1
JAL	Jump and Link	1	1	CTI	1
JALR	Jump and Link Register	1	1	CTI	1
LB	Load byte from memory as a signed value	4	1	LSU	1
LBU	Load byte from memory as an unsigned value	4	1	LSU	1
LD	Load doubleword from memory	4	1	LSU	1
LD	Load doubleword from memory	4	1	LSU	1
LDP	Load double pair	4	1	LSU	1
LH	Load halfword from memory as a signed value	4	1	LSU	1
LHU	Load halfword from memory as an unsigned value	4	1	LSU	1
LR.D	Load linked doubleword	4	1	LSU	1
LR.W	Load linked word	4	1	LSU	1
LUI	Load upper immediate	1	1	ALU	2
LW	Load word from memory as a signed value	4	1	LSU	1
LWP	Load word pair	4	1	LSU	1
LWU	Load word from memory as an unsigned value	4	1	LSU	1
MAX	Select maximum value - integer	1	1	ALU	2

Table 12.1 Instruction Latencies and Repeat Rates*(continued)*

Instruction	Definition	Latency	Rate	Unit Type	Units
MAXU	Select maximum value - unsigned integer	1	1	ALU	2
MIN	Select minimum value - integer	1	1	ALU	2
MINU	Select minimum value - unsigned integer	1	1	ALU	2
MRET	Return from Machine mode	3		IDU/GRU	1
MTLBWR	TLB Write	n/a	1	LSU	1
MTVECJ	Jump to address in MTVEC register			LSU	
MUL	Multiply integer signed	4	1	MUL	1
MULH	Multiply integer signed, return high doubleword	4	1	MUL	1
MULHSU	Multiply integer signed by unsigned, return high double-word	4	1	MUL	1
MULHU	Multiply integer unsigned, return high doubleword	4	1	MUL	1
MULW	Multiply word integer signed	3	1	MUL	1
OR	Bitwise logical OR	1	1	ALU	2
ORC.B	Set all bits of a byte to 1 if any bit is 1	1	1	ALU	2
ORI	OR immediate. Bitwise logical or with a constant.	1	1	ALU	2
ORN	Or operation with second operand inverted	1	1	ALU	2
PAUSE	Pause hart temporarily	n/a	1	LSU	1
PREF	Move data from memory into cache		1	LSU	1
REM	Modulo integer signed	7 to 22		DIV	1
REMU	Modulo integer unsigned	7 to 22		DIV	1
REMUW	Modulo word integer unsigned	7 to 22		DIV	1
REMW	Modulo word integer signed	7 to 22		DIV	1
REV8	Reverse bytes	1	1	ALU	2
ROL	Rotate left	1	1	ALU	2
ROLW	Rotate left word	1	1	ALU	2
ROR	Rotate right	1	1	ALU	2
RORI	Rotate right by immediate value	1	1	ALU	2
RORIW	Rotate right word by immediate value	1	1	ALU	2
RORW	Rotate right word	1	1	ALU	2
SB	Store a byte to memory	n/a	1	LSU	1
SC.D	Store conditional doubleword			LSU	
SC.W	Store conditional word			LSU	
SD	Store a doubleword to memory	n/a	1	LSU	1
SDP	Store double pair	n/a	1	LSU	1
SEXT.B	Sign extension - byte	1	1	ALU	2
SEXT.H	Sign extension - halfword	1	1	ALU	2
SFENCE.VMA	Supervisor Synchronize virtual-memory management	n/a		LSU	
SH	Store halfword to memory	n/a	1	LSU	1
SH1ADD	Shift 1 bit and add	1	1	ALU	2

Table 12.1 Instruction Latencies and Repeat Rates*(continued)*

Instruction	Definition	Latency	Rate	Unit Type	Units
SH1ADD.UW	Shift 1 bit and add - unsigned word only	1	1	ALU	2
SH2ADD	Shift 2 bit and add	1	1	ALU	2
SH2ADD.UW	Shift 2 bit and add - unsigned word only	1	1	ALU	2
SH3ADD	Shift 3 bit and add	1	1	ALU	2
SH3ADD.UW	Shift 3 bit and add - unsigned word only	1	1	ALU	2
SLL	shift left logical by a variable number of bits	1	1	ALU	2
SLLI	Shift left logical by a fixed number of bits	1	1	ALU	2
SLLI	Doubleword shift left logical	1	1	ALU	2
SLLI.UW	Shift left logical by imm value - unsigned word only	1	1	ALU	2
SLLIW	Shift word left logical by a fixed number of bits	1	1	ALU	2
SLLW	Shift word left logical by a variable number of bits	1	1	ALU	2
SLT	Set on less than	1	1	ALU	2
SLTI	Set on less than immediate	1	1	ALU	2
SLTIU	Set on less than immediate unsigned	1	1	ALU	2
SLTU	Set on less than, unsigned	1	1	ALU	2
SRA	Shift right arithmetic	1	1	ALU	2
SRAI	Shift right arithmetic by a fixed number of bits	1	1	ALU	2
SRAI	Shift word right arithmetic	1	1	ALU	2
SRAIW	Shift word right arithmetic	1	1	ALU	2
SRAW	Shift word right arithmetic variable number of bits	1	1	ALU	2
SRET	Return from Supervisor mode	3		IDU/GRU	1
SRL	shift right logical	1	1	ALU	2
SRLI	Shift right logical by a fixed number of bits	1	1	ALU	2
SRLI	Doubleword shift right logical	1	1	ALU	2
SRLIW	Shift word right logical	1	1	ALU	2
SRLW	Shift word right logical by a variable number of bits	1	1	ALU	2
SUB	Subtract	1	1	ALU	2
SUBW	Subtract unsigned word	1	1	ALU	2
SW	Store word to memory	n/a	1	LSU	1
SWP	Store word pair	n/a	1	LSU	1
WFI	Wait for interrupt	3		IDU/GRU	1
XOR	Exclusive OR	1	1	ALU	2
XORI	Exclusive OR immediate	1	1	ALU	2
XORN	XOr operation with second operand inverted	1	1	ALU	2
ZEXT.H	Zero extend halfword	1	1	ALU	2

Chapter 13

MIPS On-Chip Instrumentation

This chapter provides a brief overview of the interface and external debugging environment required to debug MIPS processors that incorporate the MIPS On-Chip Instrumentation (OCI) debug system for multi-core designs.

The MIPS OCI debug system has been developed to provide comprehensive debugging and performance-monitoring capabilities for multi-core processor designs where there can be one or two Harts per core and multiple cores per cluster.

For more information on OCI, refer to https://www.mips.com/develop/tools

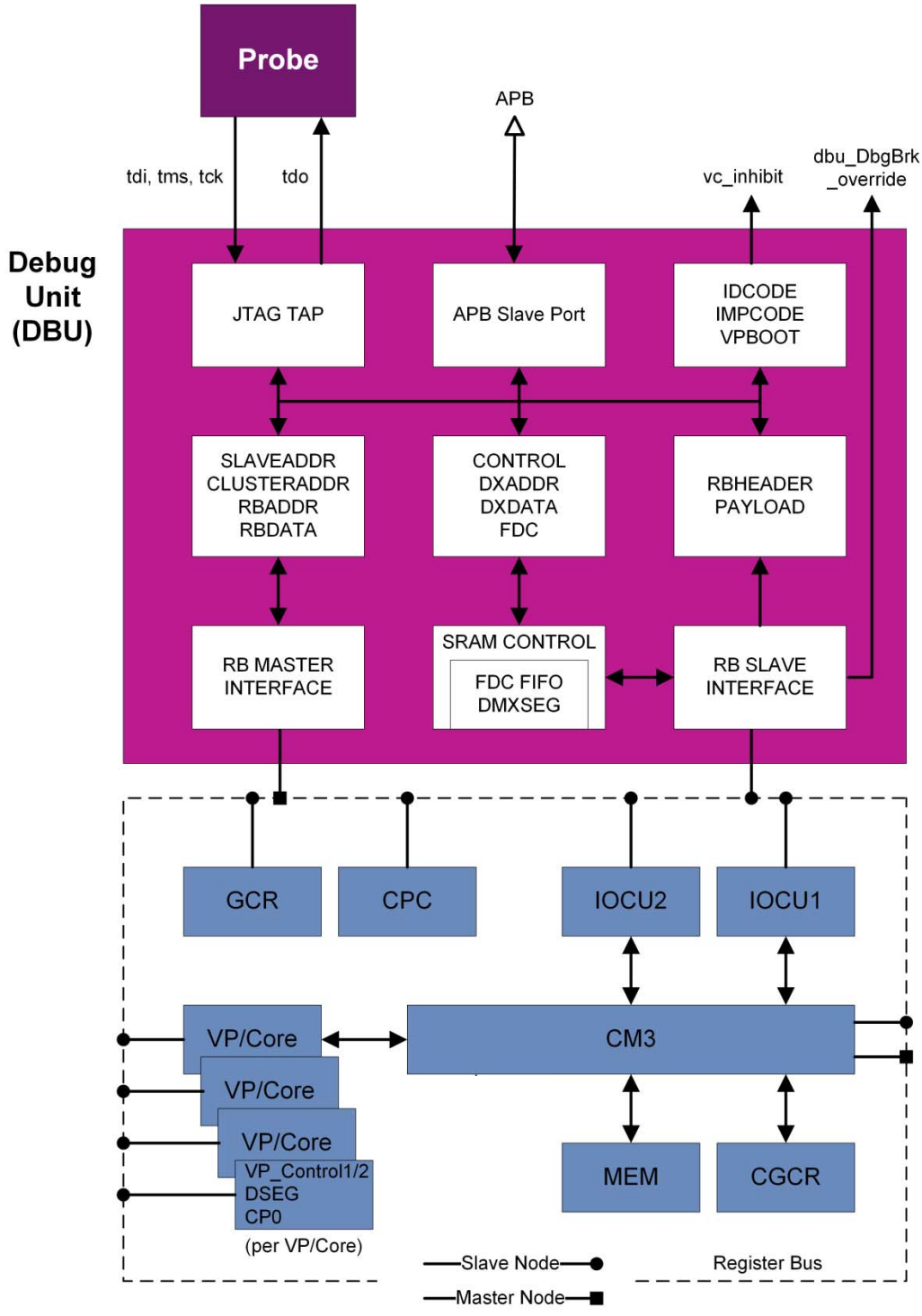
13.1 OCI Debug System Overview

The MIPS OCI Debug System comprises a dedicated on-chip module called the Debug Unit and various on-chip components that have dedicated debug resources from which debug data is gathered. These are connected by a Register Ring Bus (RRB).

13.1.1 Debug Unit (DBU)

There is one DBU per cluster of cores or Harts in a system. The DBU provides several functions to assist debugging. Figure 13.1 shows the OCI system as implemented in a typical single-cluster core.

Figure 13.1 OCI System Block Diagram



13.1.1.1 APB Slave Port

An APB Slave Port in the DBU provides connection to an APB enabled on-chip debug controller or emulator transactor interface.

13.1.1.2 JTAG TAP

A serial JTAG TAP allows connection to a JTAG debug probe. The JTAG TAP data registers reside in the DBU and allow read/write requests to the Harts being debugged via debug monitor code.

13.1.1.3 Debug Monitor

Debug monitor code is loaded into RAM in the DBU and schedules debug commands to the Harts via the Register Bus.

13.1.1.4 RAM

A dedicated block of RAM in the DBU that hosts the debug monitor code and contains the memory mapped area, dmxseg. Dmxseg is a mapped to the Harts debug memory segment, dmseg, and is accessed by the Hart when running in debug mode and when a debug probe is attached. This RAM also contains the FIFOs for Fast Debug Channels.

13.1.2 Register Bus

The DBU connects to Harts and other coherent devices on the Register Bus (RRB) using a packet-based protocol.

13.1.3 Number of Breakpoints

The P8700-F MPS implements 8 instruction breakpoint triggers and 8 data breakpoint triggers. Breakpoints are shared between all Harts.

13.1.4 Per Core/Hart Resources

13.1.4.1 Breakpoint Controller

Each Hart has its own independent breakpoint control logic and configuration registers.

13.1.4.2 Dseg

A memory mapped area of main memory, accessible from the processor in debug mode only. It contains the combined dmseg and drseg areas.

13.1.4.3 Dmseg

The debug memory segment of dseg that is accessed by the core when running in debug mode when a debug probe is attached. This area is mirrored by dmxseg in DBU RAM.

13.1.4.4 Drseg

A region of dseg that includes registers that are mapped to debug resources such as breakpoint configuration registers and sampling registers. The Hart and the DBU can read and write these registers indirectly via the coherence manager (CM). Drseg registers can be accessed from the DBU during normal and debug mode execution.

13.1.4.5 CSR Registers

CSR contains specific registers that facilitate and configure various aspects of a Hart's debug features.

13.1.5 Coherence Devices

The P8700-F Multiprocessing System contains the following coherent devices.

13.1.5.1 CPC (Cluster Power Controller)

Provides stop/run signals for Harts; reset occurred signals for the DBU, CM and Harts; registers for determining the state of each Harts power and clock rate; and power up and clock gating of the CM.

13.1.5.2 GCR (Global Configuration Registers)

A set of memory mapped registers that are used to configure and control various aspects of the CM, the coherence scheme and CM performance counters.

13.1.5.3 CGCR - (Custom Global Configuration Registers)

An optional block of custom registers that can be used to control system level functions.

13.1.5.4 CM - (Coherence Manager) (v3)

Controls the global ordering of requests and responses across core devices.

13.1.5.5 IOCU (I/O Coherence Unit)

Connects coherent devices to the Coherence Manager.

13.2 More Information

For more information on the MIPS OCI debug system, refer to the document entitled;

- MIPS Hybrid Debug Specification
- MIPS On-Chip Instrumentation; Debug Technical Reference Manual
- MIPS On-Chip Instrumentation PDtrace Specification
- MIPS On-Chip Instrumentation 64-Bit Debug Specification

Appendix A

Revision History

Change bars (vertical lines) in the margins of this document indicate significant changes in the document since its last release. Change bars are removed for changes that are more than one revision old.

Table A.1 Revision History

Revision	Date	Description
1.00	January 31, 2022	Initial release of P8700-F Programmers Guide
1.10	September 5, 2022	Updated Chapter 1, Architecture Overview Added Chapter 2, MMU Programming Updated Chapter 3, Caches Added Chapter 4, Exceptions Updated Chapter 5, CM3 Updated Chapter 6, CPC Added Chapter 8, FPU Added Chapter 9, SIMD Added Chapter 10, Virtualization Added Chapter 11, Multithreading Added Chapter 12, Performance Counters Updated Chapter 13, OCI Debug Added Chapter 14, Implementation Specific Instructions Added Chapter 15, Latency and Repeat Rate
1.20	March 31, 2023	Remove SIMD chapter Remove Virtualization chapter Updated Section 2.1.1, MMU Types Updated Section 2.1.2, Instruction TLB Updated Section 2.1.3, Data TLB Updated Section 2.1.4, Variable TLB Added Section 2.8, Hardware Page Table Walker Updated Section 3.2, Cache Subsystem Overview Updated Section 3.2.1.7, FENCE.I Instruction Usage Updated Section 3.2.1.8, MGINVI Instruction Usage Updated Section 3.5.1, L1 Instruction Cache Control Registers Updated Section 3.5.2, L1 Data Cache Control Registers Updated Section 3.5.3, L2 Cache CM GCR Control Registers Updated Section 5.5.1, Programming Another Hart in the Same Core Updated Section 5.5.7, Accessing the Core-Local and Core-Other Registers in the AIA Controller Updated Section 6.5.11, Hart Run/Suspend Updated Section 8.1, FPU Features Overview Updated Section 8.1.1, Short Operations Updated Section 8.1.2, Long Operations Updated Chapter 4, Exceptions and Interrupts

Table A.1 Revision History

Revision	Date	Description
1.30	May 14, 2023	Updated Chapter 10, Latency and Repeat Rate tables. Updated Figure 1-1, Block Diagram Updated Section 2.1.4, Variable Page Size, and related subsections Updated Section 2.1.5, Fixed Page Size Updated Figure 3-1, System Caches Updated Section 4.2, Selecting the Exception Address Removed selected exceptions from Section 4.5, Exception Descriptions Additional updates throughout Chapter 4, Exceptions Removed all Debug related exceptions from Chapter 4. Moved to MIPS Hybrid Debug Specification
1.40	June 29, 2023	Updated pseudocode example in Section 6.5.7.1, Clock Domain Change Example. Updated text and addressing in Section 6.5.5, Enabling Coherent Mode.
1.50	August 30, 2023	Removed Section 5.1.2, CM GCR Registers Removed Section 5.1.3, Core-Local GCR's Removed Section 5.1.3, Core-Other GCR's Removed Section 5.1.3, Core-Local and Core-Other Register Usage Updated Figure 1.1, Block Diagram Updated Figure 1.2, Cluster to Cluster Accesses Updated Figure 1.3, Core-Level Block Diagram Removed Section 1.7, CSR Register to Assembler Mapping Updated Section 1.7.2, MIPS RISC-V SDK Updated Section 2.1.4.1, VTLB Organization Removed Section 2.3, Shared TLB Removed references to MMID Updated Figure 3.1, System Caches Removed Section 3.2.1.8, MGINV.I Instruction Usage Updated Section 3.3, Cache Coherency Removed Section 3.4, Self Modified Code Removed Section 3.5, Register Interface Removed Section 3.6, L2 Cache Initialization Options Removed Section 3.9, Flushing the L1 Data Cache Removed Section 3.10, Setting the Memory Space Cache Coherency Removed Section 9.3.1, Disable Virtual Processor Instruction Removed Section 9.3.2, Enable Virtual Processor Instruction Updated Section 2.1.5, Fixed Page Size TLB Updated Section 2.7, Hardware Table Walker Updated Section 3.2.1.6, MCACHE Instruction Usage Updated Table 3.5, L2 Cache-ops Updated Section 3.2.11, Cache Instructions Updated Table 3.6, Encoding of Bits [24:22] of the MCACHE Instruction Updated Section 3.3, Cache Coherency Updated Section 4.2, Selecting the Exception Address Updated Section 5.3.1, CM GCR Register Interface Updated Chapter 7, Floating Point Unit

Table A.1 Revision History

Revision	Date	Description
1.60	April 11, 2024	Added new template changes to all pages. Added Section 3.2, LR and SC Instruction Considerations. Updated Section 12.1.3, Number of Breakpoints. Removed Section 9.3, Thread Management. Updated Section 3.3.6, Load/Store Bonding. Added Table 10.5, Core Performance Counter Events. Updated Section 10.1, Core Performance Counters. Added Section 10.1.1, Performance Event Masking. Reworked Section 10.1.2, Core Performance Event Control Register (mhpmevent[6:3]). Reworked Section 10.1.3, Core Performance Counter Count Register (mhpmcounter[3:0]). Updated Section 10.2, CM3 Performance Counters, to remove references to Root and Guest.
1.70	June 28, 2024	Updated Section 3.3.1.4, L1 Instruction Cache Replacement Policy. Updated Section 3.3.4, L1 Data Cache Replacement Policy. Updated Section 3.3.6, Load/Store Bonding. Updated Figure 6.4, Reset Detection in the P8700-F Multiprocessing System. Updated Section 9.1, Instruction Flow. Updated Section 9.2, Data Flow. Updated Section 12.1.2, Register Bus. Updated Section 12.1.3, Number of Breakpoints.
1.71	September 11, 2024	Remove references the Hypervisor. Remove references to Virtualization. Updated Chapter 2, MMU
1.80	November 13, 2024	Added Chapter 8, CSR Registers. Added Chapter 9, Interrupt Controller. Added Appendix B, Custom Instructions. Edited entire document to address references to select instructions. Added Zba_Zbb to product name throughout document.
1.81	March 3, 2025	Incorporate changes from internal review. Removed references to Big Endian mode. Added new instruction extensions and naming convention to all MIPS MDI instructions. Updated CCMOV instruction.
1.82	March 19, 2025	Added PAUSE instruction.
1.83	April 9, 2025	Added version numbering entry to each individual instruction. Updated Table B.1 in Appendix B to include family name and version number for each instruction.

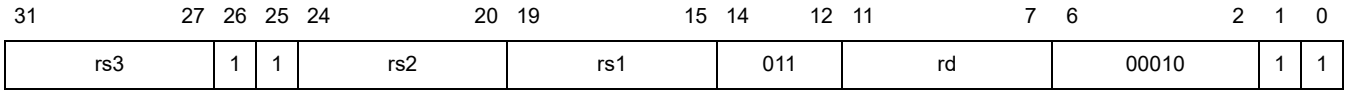
Appendix B

MIPS Defined Instructions

The following pages define the MIPS Technologies defined instructions. Table B.1 shows a listing of these instructions.

Table B.1 MIPS Custom Instructions

Instruction	Description	Extension	Version
CCMOV	Conditional move instruction.	xmipscmov	1.0
EHB	Execution Hazard Barrier. For MIPS Technologies implementations of RISCV cores, the EHB instruction may be used to clear speculation or state-change hazards in implementation dependent cases, and may be used in the future to explicitly synchronize custom state-changing instructions.	xmipsexectl	1.0
IHB	Instruction Hazard Barrier. The IHB instruction creates an instruction hazard barrier, meaning that it ensures that all subsequent instruction fetches (including those that are carried out speculatively) will be aware of state changes caused by prior instructions.	xmipsexectl	1.0
LDP	Load Doubleword Pair.	xmipslsp	1.0
LWP	Load Word Pair.	xmipslsp	1.0
MTLBWR	Machine TLB Write Random.	xmipsstw	1.0
PAUSE	Pause Hart temporarily.	xmipsexectl	1.0
PREF	Cache PREFetch operation.	xmipscbop	1.0
SDP	Store Doubleword Pair.	xmipslsp	1.0
SWP	Store Word Pair.	xmipslsp	1.0



Extension Name: xmipscmov

Extension Version: 1.0

Format: mips.ccmov $rd, $rs2, $rs1, $rs3

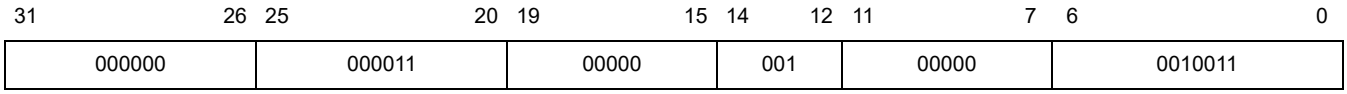
Description: Custom Conditional MOVe. Set $rd to $rs1 if $rs2 is not zero, else set $rd to $rs3.

Operation:

```
XR[rd] = XR[rs1] if XR[rs2]!= 0 else XR[rs3]
```

Exceptions:

Restrictions:



Extension Name: xmipsexectl

Extension Version: 1.0

Format: mips.ehb

Description: Execution Hazard Barrier. Clear all execution hazards before allowing any subsequent instructions to execute. EHB uses a ‘hint’ encoding of the SLLI instruction, with rd = 0, rs1 = 0 and imm = 3. The MIPS architecture uses the EHB instruction to insert an explicit execution hazard barrier after an architecture-state changing instruction that affects the execution of subsequent instructions. An EHB is not required in the RISC-V Architecture as such hazards are cleared implicitly by design.

For MIPS Technologies implementations of RISCV cores, the EHB instruction may be used to clear speculation or state-change hazards in implementation dependent cases, and may be used in the future to explicitly synchronize custom state-changing instructions.

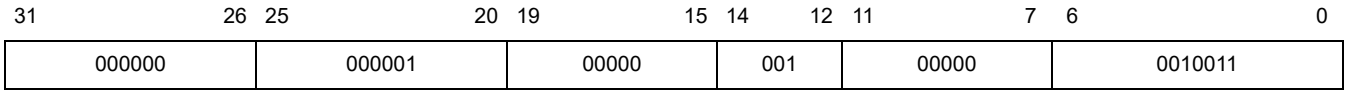
In some rare situations use of an EHB may improve performance by preventing a subsequent instruction from initiating an action which consumes a resource which is not ultimately needed.

Operation:

```
clear_execution_hazards()
```

Exceptions:

Restrictions:



Extension Name: xmipsexectl

Extension Version: 1.0

Format: mips.ihb

Description: Instruction Hazard Barrier. Clear all instruction hazards before allowing any subsequent instructions to fetch. IHB uses a ‘hint’ encoding of the SLLI instruction, with rd = 0, rs1 = 0 and imm = 1. The IHB instruction creates an instruction hazard barrier, meaning that it ensures that all subsequent instruction fetches (including those that are carried out speculatively) will be aware of state changes caused by prior instructions.

An IHB is required after an MCACHE instruction if subsequent instruction fetch depends on the cache operations performed by the MCACHE instruction.

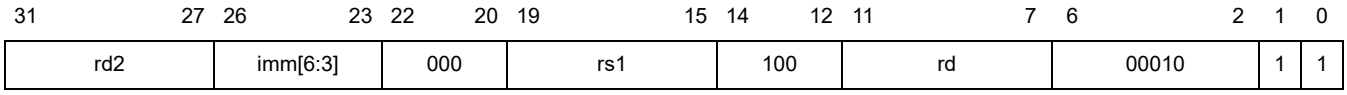
Other than the use of IHB in conjunction with the MCACHE instruction, software never needs to use IHB to obtain architecturally correct behavior. However in some rare situations use of an IHB may improve performance by preventing a subsequent instruction from initiating an action which is architecturally legal but which consumes a resource which is not ultimately needed.

Operation:

```
clear_instruction_hazards()
```

Exceptions:

Restrictions:



Extension Name: xmipslsp

Extension Version: 1.0

Format: mips.ldp $rd1, $rd2, offset($rs1)

Description: An LDP instruction is guaranteed to atomically read 16-byte data if its address is 16-byte aligned. If the address is 8-byte aligned but not 16-byte aligned, then the data is only guaranteed to be 8-byte atomic. For any other alignment, there is no atomicity guarantee.

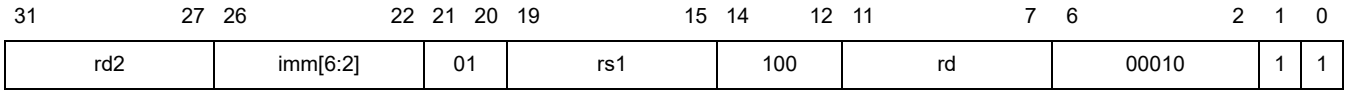
A Load Address Misaligned exception may occur if the virtual address targeted by the LDP instruction is not aligned to a 16-byte boundary. Performance optimized implementations will provide native hardware support for 8-byte aligned cases, including cases that cross a page boundary. When $rd1 and $rd2 are the same register, the value written to the output register is unknown.

Operation:

```
if not HART.udi:
     raise illegal_inst_exception("MIPS Technologies user defined instruction")

va = (XR[rs1] + offset) & XLEN_MASK

if CONFIG.pair_aligned and va & 15:
     raise address_misaligned_exception(va, 'Load', 'LDP not 16-byte aligned')

data = read_memory_at_va(va, nbytes=16)

if rd1 == rd2:
     XR[rd1] = UNKNOWN
else:
     XR[rd1] = data[63:0]
     XR[rd2] = data[127:64]
```

Exceptions:

Restrictions:



Extension Name: xmipslsp

Extension Version: 1.0

Format: mips.lwp $rd1, $rd2, offset($rs1)

rd1 = rd
11 offset = imm

Description: Load Word Pair. Load the signed word data value from address $rs1 + offset (register plus unsigned immediate) and write the result to integer register $rd1. Load the signed word data value from address $rs1 + offset + 4 and write the result to integer register $rd2.

An LWP instruction is guaranteed to atomically read 8-byte data if its address is 8-byte aligned. If the address is 4-byte aligned but not 8-byte aligned, then the data is only guaranteed to be 4-byte atomic. For any other alignment, there is no atomicity guarantee.

A Load Address Misaligned exception may occur if the virtual address targeted by the LWP instruction is not aligned to an 8-byte boundary. Performance optimized implementations will provide native hardware support for 4-byte aligned cases, including cases that cross a page boundary.

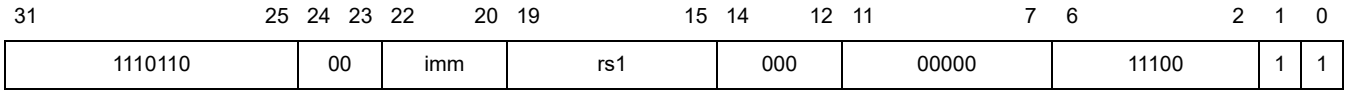
When $rd1 and $rd2 are the same register, the value written to the output register is unknown.

Operation:

```
if not HART.udi:
     raise illegal_inst_exception("MIPS Technologies user defined instruction")

va = (XR[rs1] + offset) & XLEN_MASK

if CONFIG.pair_aligned and va & 7:
     raise address_misaligned_exception(va, 'Load', 'LDP not 8-byte aligned')

data = read_memory_at_va(va, nbytes=8)

if rd1 == rd2:
     XR[rd1] = UNKNOWN
else:
     XR[rd1] = sign_extend(data[31:0], from_nbits=32) & XLEN_MASK
     XR[rd2] = sign_extend(data[63:32], from_nbits=32) & XLEN_MASK
```

Exceptions:

Restrictions:



Extension Name: xmipsstw

Extension Version: 1.0

Format: mips.mtlbwr $rs1, level

Description: Machine TLB Write Random. Update a random entry in the implementation dependent TLB. Create the mapping using the virtual address in mtval and the leaf PTE value stored in integer register $rs1. Use the mipsconfig5.LEVEL field to determine the level of the PTE in the page table, the mipsconfig5.XATP field to determine the translation type, and OR the mipsconfig5.G with the G field in the PTE value.

The level immediate field is deprecated and should be set to 0, except in the legacy definition (cores with HART.tlb_non_leaf false).

```
level = imm
```

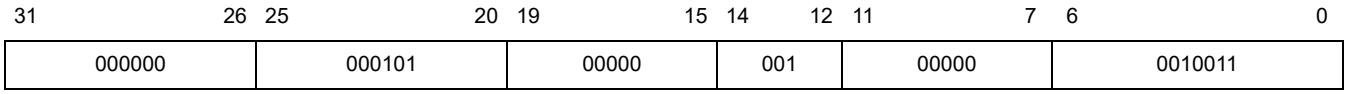
Operation:

```
if not HART.got_stw:
     raise illegal_inst_exception("MIPS software table walk not implemented")

if HART.priv < 3:
     raise illegal_inst_exception("MTLBWR without machine privilege")

if HART.tlb_non_leaf:
     if level != 0:
          raise illegal_inst_exception("MTLBWR level field deprecated for
          tlb_non_leaf")
     pte_level = CSR.mipsconfig5.LEVEL
     trace(f"Using mipsconfig5.LEVEL={pte_level}")
else:
     pte_level = level

if HART.tlb_non_leaf:
     if CSR.mipsconfig5.XATP == 0:
          xatp = 'satp'
     elif CSR.mipsconfig5.XATP == 1:
          xatp = 'vsatp'
     elif CSR.mipsconfig5.XATP == 2:
          xatp = 'hgatp'
     else:
          FatalError(f"Unsupported mipsconfig5.XATP value {CSR.mipsconfig5.XATP}")
else:
```

```
    xatp = 'vsatp' if CSR.misa.H and CSR.mstatus.GVA else 'satp'

va = CSR.mtval2 << 2 if xatp == 'hgatp' else CSR.mtval
pte = XR[rs1]

mtlbwr(va, pte, pte_level, xatp)
```

Exceptions:

Restrictions:



Extension Name: xmipsexectl

Extension Version: 1.0

Format: mips.pause

Description: This MIPS.PAUSE is an alternative custom opcode which is implemented to have the same behavior as PAUSE on some MIPS RISCV cores. It is a ‘hint’ encoding of the SLLI instruction, with rd = 0, rs1 = 0 and imm = 5. It will behave as a NOP instruction if no additional behavior beyond that of SLLI is implemented.

The purpose of the PAUSE instruction is to temporarily halt a hart to allow other harts to make forward progress more efficiently. This is particularly useful on multi-threaded processors, since the waiting hart may be using the same instruction pipeline as the harts which have active work to do.

One specific use case is when a hart is waiting to acquire an LR/SC lock. Entering a spin loop may delay the hart which owns the lock from completing its task and freeing the lock. The pseudocode and following description discuss a PAUSE implementation using a per hart ‘lr_bit’ which tracks whether there is an active LR/SC, but other implementations are possible.

When a hart is in the paused state, it should not issue any instructions. The paused state will be cleared either if the lr_bit for the hart gets cleared, if the hart takes an interrupt, or if a timer representing a maximum number of pause cycles expires. If an interrupt occurs, it is implementation dependent whether the relevant mepc/sepc CSR points to the PAUSE instruction or the instruction after the PAUSE.

In LR/SC lock software, the lr_bit of the waiting hart will always be cleared when the hart which owns the lock does a store instruction to the lock address in order to clear the lock. Thus the paused hart will always be woken when it has another opportunity to acquire the lock. After the PAUSE instruction completes, software is expected to attempt to acquire the lock again by re-executing the LR/SC sequence.

It is legal to implement PAUSE as a NOP instruction. In this case, the behavior of LR/SC lock software will be equivalent to executing a spin loop to acquire the lock. Software using PAUSE will still work, but the benefit of having the waiting hart not consume instruction issue slots will be lost.

The following assembly code example shows how the PAUSE instruction can be used to halt a hart while it is waiting to acquire an LR/SC lock.

```
acquire_lock:
   lr.w     t0, (a0)       /* Read software lock, set lr_bit. */
   bnez     t0, acquire_lock_retry /* Branch if software lock is taken. */
   addi     t0, t0, 1      /* Set the software lock. */
   sc.w     t0, t0, (a0)   /* Try to store the software lock. */
   beqz     t0, 10f        /* Branch if lock acquired successfully. */


acquire_lock_retry:
    pause                  /* Wait for lr_bit to clear before retrying. */
    j    acquire_lock      /* Now retry the operation. */
10:
    /* Critical Region Code */
    ...
```

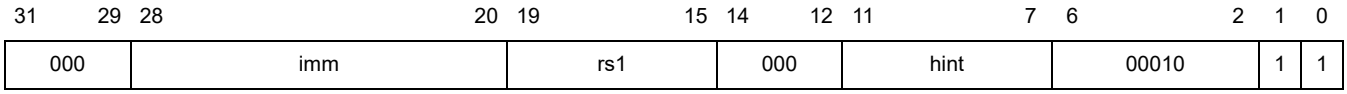
```
release_lock:
    fence
    sw zero,    (a0)            /* Release software lock, clearing lr_bit for any PAUSEd waiters */
```

Operation:

```
if CONFIG.zihintpause:
     pause()
```

Exceptions:

Restrictions:



Extension Name: xmipscbop

Extension Version: 1.0

Format: mips.pref

Description: PREFetch. Perform a prefetch operation of type hint at address $rs1 + imm. The PREF instruction requests that the processor take some action to improve program performance in accordance with the intended data usage specified by the hint argument. This is typically done by moving data to or from the cache at the specified address.

Bits [4:3] of the ‘hint’ argument specify the target cache as follows:

hint[4:3]	Target Cache
00	L1 instruction (I) cache
01	L1 data (D) cache
10	L2 (S) cache
11	L3 (T) cache

Bits[2:0] of the ‘hint’ argument specify the prefetch type, as follows:

hint[2:0]	Target Cache
000	Load
001	Store (or reserved for ICache)
010 - 110	Reserved
111	Prepare for store (or reserved for ICache)

The behavior for the prefetch types is as follows:

- hint[2:0] = 0: Load. Prefetched data is expected to be read (not modified), and should be fetched a if for a load.
- hint[2:0] = 1: Store Prefetched data is expected to be stored or modified and should be fetched as if for a store.
- hint[2:0] = 7: Prepare-for-store. Prepare a cache line for store, assuming that the current data for the range of addresses in the line can be discarded. If the line is not present in cache, evict or invalidate a line then zero-fill its data and update it to store the target address in a modified state. If the line is present in the cache, then upgrade it to a modified state.

The action taken for a specific PREF instruction is both system and context dependent. For all operations, any action, including doing nothing, is permitted as long as it does not change architecturally visible state or alter the

meaning of a program. In addition, for prepare-for-store operations only, modifying architectural state by replacing the contents of the targeted cache line with zeros is also permitted.

PREF does not cause addressing-related exceptions. If the address specified would cause an addressing exception, the exception condition is ignored and no data movement occurs.

For cached addresses, the expected and useful action is for the processor to prefetch a block of data that includes the effective address. The size of the block and the level of the memory hierarchy it is fetched into are implementation specific.

PREF neither generates a memory operation nor modifies the state of a cache line for addresses with an uncached memory access attribute.

Prefetch operations have no effect on cache lines that were previously locked with the MCACHE instruction.

In coherent multiprocessor implementations, if the effective address uses a coherent CCA, then the instruction causes a coherent memory transaction to occur. This means a prefetch issued on one processor can cause data to be evicted from the cache in another processor.

The memory transactions which occur as a result of a PREF instruction, such as cache refill or cache writeback, obey the same ordering and completion rules as other load or store instructions.

It is implementation dependent whether a Bus Error or Cache Error exception is reported if such an error is detected as a byproduct of the action taken by the PREF instruction. Implementations are encouraged to report such errors only if there is a specific requirement for high-reliability. Note that suppressing a bus or cache error in this case may require that the processor communicate to the system that the reference is speculative.

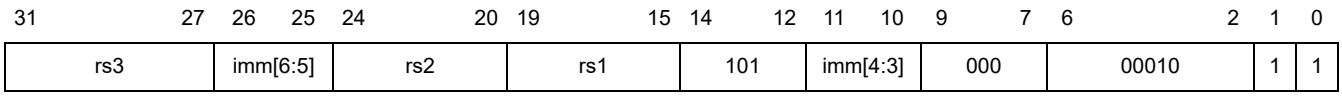
Operation:

```
if not HART.udi:
     raise illegal_inst_exception("MIPS Technologies user defined instruction")

va = (XR[rs1] + imm) & XLEN_MASK

cache_type = (
     'Icache' if hint[4:3] == 0 else
     'Dcache' if hint[4:3] == 1 else
     'Scache' if hint[4:3] == 2 else
     'Tcache' if hint[4:3] == 3 else FatalError("unexpected pref"))

op_type = (
     'Load' if hint[2:0] == 0 else
     'Store' if hint[2:0] == 1 and cache_type != 'Icache' else
     'PrepareForStore' if hint[2:0] == 7 and cache_type != 'Icache' else
     'Reserved')

cache = HART[cache_type]

if op_type == 'Reserved':
     trace(f"Reserved pref hint ({hint}) is a nop")
     return

if not cache.exists:
```

```
     trace("Pref to non-existent cache is a nop")
     return

if not pref_implemented(hint, cache_type, op_type):
     # It is legal to implement any pref operation as a nop
     trace("Unimplemented PREF hint")
     return

try:
     access_type = 'Store' if op_type in ('Store', 'PrepareForStore') else 'Load'
     pa, pma = translate_va(va, 1, access_type)
except EXCEPTION:
     return # Address exception on pref acts as a nop
if not is_cacheable(pma):
     trace("Pref to uncached address is a NOP")
     return

# Find index which holds this address in the cache (if any).
index = find_cache_index(cache, va, pa)

if index is None:
     if op_type == 'PrepareForStore':
          index = fill_cache(cache, va, pa, pma, prepare_for_store=True, excl
          sive=True)
     else:
          index = fill_cache(cache, va, pa, pma)
```

Exceptions:

Restrictions:



Extension Name: xmipslsp

Extension Version: 1.0

Format: mips.sdp $rs2, $rs3, offset($rs1)

Description:

Store Double Pair. Store the double word data value in integer register $rs2 to memory address $rs1 + offset (register plus unsigned immediate), and store the double word data value in integer register $rs3 to memory address $rs1 + offset + 8.

An SDP instruction is guaranteed to atomically write 16-byte data if its address is 16-byte aligned. If the address is 8-byte aligned but not 16-byte aligned, then the data is only guaranteed to be 8-byte atomic, and whether the 8-byte parts are ordered is implementation dependent. For any other alignment, there is no atomicity guarantee.

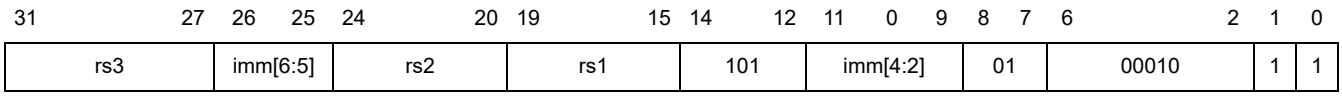
A Store Address Misaligned exception may occur if the virtual address targeted by the SDP instruction is not aligned to a 16-byte boundary. Performance optimized implementations will provide native hardware support for 8-byte aligned cases, including cases that cross a page boundary.

Operation:

```
if not HART.udi:
     raise illegal_inst_exception("MIPS Technologies user defined instruction")

va = (XR[rs1] + offset) & XLEN_MASK

if CONFIG.pair_aligned and va & 15:
     raise address_misaligned_exception(va, 'Store', 'SDP not 16-byte aligned')

if is_littleendian():
     data = XR[rs3] << 64 | XR[rs2]

write_memory_at_va(data, va, nbytes=16)
```

Exceptions:

Restrictions:



Extension Name: xmipslsp

Extension Version: 1.0

Format: mips.swp $rs2, $rs3, offset($rs1)

Description:

Store Word Pair. Store the word data value in integer register $rs2 to memory address $rs1 + offset (register plus unsigned immediate), and store the word data value in integer register $rs3 to memory address $rs1 + offset + 4.

An SWP instruction is guaranteed to atomically write 8-byte data if its address is 8-byte aligned. If the address is 4-byte aligned but not 8-byte aligned, then the data is only guaranteed to be 4-byte atomic, and whether the 4-byte parts are ordered is implementation dependent. For any other alignment, there is no atomicity guarantee.

A Store Address Misaligned exception may occur if the virtual address targeted by the SWP instruction is not aligned to an 8-byte boundary. Performance optimized implementations will provide native hardware support for 4-byte aligned cases, including cases that cross a page boundary.

Operation:

```
if not HART.udi:
     raise illegal_inst_exception("MIPS Technologies user defined instruction")

va = (XR[rs1] + offset) & XLEN_MASK

if CONFIG.pair_aligned and va & 7:
     raise address_misaligned_exception(va, 'Store', 'SDP not 8-byte aligned')

if is_littleendian():
     data = XR[rs3] << 32 | (XR[rs2] & 0xFFFFFFFF)

write_memory_at_va(data, va, nbytes=8)
```

Exceptions:

Restrictions: