ESWIN

EIC7700X SoC Technical Reference Manual

Public Release / Rev 1.1

2024/01/31

Notice

Copyright © 2024 by Beijing ESWIN Computing Technology Co., Ltd., and its ("ESWIN Computing"). All rights reserved.
ESWIN Computing retain all intellectual property and proprietary rights in and to this product. Without as expressly authorized, no part of the software may be released, copied, distributed, reproduced, modified, adapted, translated, or created derivative work of, in whole or in part.

INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF ESWIN. UNLESS OTHERWISE SPECIFIED IN THE CONTRACT, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT ARE PROVIDED "AS IS" WITHOUTH WARRANTIES, GUARANTEES, OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

"ESWIN" logo, and other ESWIN icons, are trademarks of Beijing ESWIN Computing Technology Co., Ltd. and its parent company.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

If applicable, Open-Source Software and/or free software licensing notices are available in the product installation.

Change History

Version No	Date	Descriptions
V1.1	2024-1-31	

Content

1 INTRODUCTION 1
1.1 OVERVIEW 1
1.2 APPLICATION 1
1.3 ARCHITECTURE 1
1.4 SYSTEM BOOT 4
2 HARDWARE 7
2.1 PACKAGE AND PINOUT 7
2.2 PIN NUMBER LIST 10
2.3 SIGNAL DESCRIPTION 21
2.4 PIN MULTIPLEXING TABLE 33
2.5 ESD RATINGS 38
2.6 SOLDERING PROCESS RECOMMENDATIONS 38
2.7 MOISTURE-SENSITIVE SPECIFICATIONS 41
2.8 ELECTRICAL SPECIFICATIONS 44
2.9 INTERFACE TIMING 51
3 SYSTEM 55
3.1 RESET 55
3.2 CLOCK 55
3.3 SYSTEM CONTROL REGISTER 142
3.4 PROCESSOR 173
3.5 INTERRUPT SYSTEM 364
3.6 SYSTEM ADDRESS MAPPING 379
3.7 INTERCONNECT 380
3.8 DMA CONTROLLER 391
3.9 SMMU 557
3.10 MAILBOX 574
3.11 WDT 578
3.12 RTC 591
3.13 PVT 599
3.14 LOW POWER SCHEME 600

Content of Figures

Figure 1-1 EIC700X block diagram 2
Figure 1-2 System boot procedure flow 5
Figure 1-3 Bootrom workflow 6
Figure 2-1 Top view 7
Figure 2-2 Bottom view 8
Figure 2-3 Side view 8
Figure 2-4 Enlarged view of detail "A" 9
Figure 2-5 Package dimensions 9
Figure 2-6 Curve of the lead-free reflow soldering process 39
Figure 2-7 Vacuum packaging materials 42
Figure 2-8 power-on sequences, and the power-off sequences 47
Figure 2-9 1000 Mbit/s RX timing of the RGMII 51
Figure 2-10 1000 Mbit/s TX timing of the RGMII 52
Figure 2-11 Read timing of the MDIO interface 52
Figure 2-12 Write timing of the MDIO interface 52
Figure 2-13 Timing parameters of the MDIO interface 52
Figure 2-14 SPICK timing 53
Figure 2-15 SPI timing in master mode (sph = 1) 53
Figure 2-16 SPI timing in master mode (sph = 0) 54
Figure 3-1 Clock and Reset block diagram 56
Figure 3-2 Clock structure 56
Figure 3-3 Timing diagram for programming SSMOD 65
Figure 3-4 MCPU 173
Figure 3-6 Linux User Application Memory Map Example 182
Figure 3-7 Hardware Table Walk Example 183
Figure 3-8 PMP Example Block Diagram 189
Figure 3-9 EIC7700X MCPU 203
Figure 3-10 EIC7700X MCPU Memory Interleaving Scheme 210
Figure 3-11 EIC7700X MCPU Interrupt Architecture Block Diagram 215
Figure 3-12 CLINT Block Diagram 235
Figure 3-13 CLINT Interrupts and Vector Table 235
Figure 3-14 CLINT Vector Table Example 236

Figure 3-15 PLIC Multi-Core Block Diagram 239
Figure 3-16 Cache Structure in the EIC7700X MCPU 249
Figure 3-17 Event Selector Fields 251
Figure 3-18 Bus-Error Unit Block Diagram 260
Figure 3-19 Organization of the L3 Cache Controller 274
Figure 3-20 Mapping of L3 Cache Ways to L3 LIM Addresses 275
Figure 3-21 Difference between L3 LIM and L3 Zero Device 276
Figure 3-22 Event Selector Fields 278
Figure 3-23 Debug External Triggers 313
Figure 3-24 Debug External Trigger Handshake 314
Figure 3-25 Trace Encoder Block Diagram 316
Figure 3-26 Trace Packer 320
Figure 3-27 Trace Encoder External Trigger Handshake 328
Figure 3-28 Trace Encoder External Triggers 328
Figure 3-29 Trace Funnel Block Diagram 337
Figure 3-30 Address mapping 380
Figure 3-31 NoC interconnect diagram 381
Figure 3-32 Bandwidth Limiter Behavior Example 386
Figure 3-33 Bandwidth Regulator Behavior Example 387
Figure 3-34 Flow of bandwith count 390
Figure 3-35 DMA Diagram 394
Figure 3-36 block diagram 557
Figure 3-37 Mailbox block diagram 575
Figure 3-38 Interrupt interconnect 575
Figure 3-39 Flow of Mailbox Initial 576
Figure 3-40 WDT block diagram 579
Figure 3-41 RTC block diagram 591
Figure 3-42 Timing Diagram for RTC Pre-scaler Counter with the RTC Counter 591
Figure 3-43 Voltage Domain and Power Domain 601
Figure 3-44 voltage domain and main power supplies 602
Figure 3-45 power domain and isolation/levelshift cells 603
Figure 3-46 TOP only power domain 604
Figure 3-47 AON power domain 605
Figure 3-48 SNOC only power domain 605
Figure 3-49 GPU power domain 606

Figure 3-50 HSP power domain 607
Figure 3-51 NPU power domain 607
Figure 3-52 LPDDR0 power domain 608
Figure 3-53 LPDDR1 power domain 609
Figure 3-54 MCPU power domain 609
Figure 3-55 D2D power domain 610
Figure 3-56 DSP power domain 611
Figure 3-57 VI power domain 612
Figure 3-58 VOpower domain 613
Figure 3-59 VC power domain 614
Figure 3-60 PCIE power domain 615
Figure 3-61 Low power Control signals timing diagram 616
Figure 3-62 PCU Block Diagram 616
Figure 3-63 Dual power control mode switch 617
Figure 3-64 HardWare Control Mode State Machine 618

Content of Tables

Table 1-1 Boot mode and medium selection table 4
Table 2-1 Pin quantit 10
Table 2-2 Pin Number Order Information 10
Table 2-3 Signal Description 21
Table 2-4 GPIOZ_x Multi-Function Pin 34
Table 2-5 Table 2-5 ESD Ratings 38
Table 2-6 Parameters of the lead-free reflow soldering process 39
Table 2-7 Temperature resistance standard for the lead-free package according to IPC/JEDEC 020D 40
Table 2-9 Mixing reflow soldering parameters 40
Table 2-11 Thermal resistance standard for the lead package 41
Table 2-12 Floor life 43
Table 2-13 Rebaking reference 43
Table 2-14 Operating environment parameters and requirements 44
Table 2-15 Absolute maximum rated voltage 45
Table 2-16 Operating conditions for the power supply under normal voltage 46
Table 2-17 GPIO DC Electrical Specifications 47
Table 2-18 voltages of EIC7700X 49
Table 2-19 DC electrical parameters for the signals which powered by VDDQ 49
Table 2-20 DC electrical parameters for the signals which powered by VDD2H 49
Table 2-21 SDIO electrical parameters (3.3 V) 49
Table 2-22 SDIO electrical parameters (1.8 V) 49
Table 2-23 RGMII electrical parameters (3.3 V) 50
Table 2-24 GRMII electrical parameters (1.8 V) 50
Table 2-25 Oscillator electrical parameters (1.8 V) 51
Table 2-26 The timing parameters of the MDIO interface 52
Table 3-1 description of the sources of reset 55
Table 3-2 PLL reference clock 64
Table 3-3 System Register Description 142
Table 3-4 MCPU feature summary 173
Table 3-5 RISC-V Specification Compliance 175
Table 3-6 Executable Memory Regions for the EIC7700X MCPU 175
Table 3-7 Hardware Prefetcher 0 CSR 177
Table 3-8 Hardware Prefetcher 1 CSR 177

Table 3-9 Physical Memory Attributes for External Region 178
Table 3-10 Physical Memory Attributes for Internal Regions 178
Table 3-11 Supervisor Address Translation and Protection Register 180
Table 3-12 Virtual Supervisor Address Translation and Protection Register 181
Table 3-13 Hypervisor Guest Address Translation and Protection Register 181
Table 3-14 PTE Configuration Bits 183
Table 3-15 PTE Encoding fields 184
Table 3-16 pmpXcfg Bitfield Description 186
Table 3-17 pmpaddrX Encoding Examples for A=NAPOT 187
Table 3-18 User Mode CSRs 190
Table 3-19 Supervisor Mode CSRs 191
Table 3-20 Machine Mode CSRs 192
Table 3-21 Debug Mode Registers 193
Table 3-22 User Mode CSRs 194
Table 3-23 Supervisor Mode CSRs 194
Table 3-24 Machine Mode CSRs 195
Table 3-25 Machine Mode CSRs 197
Table 3-26 Debug Mode Registers 197
Table 3-27 Machine Status Register (partial) 197
Table 3-28 Supervisor Status Register (partial) 198
Table 3-29 Privilege Modes with the Hypervisor Extension 199
Table 3-30 Hypervisor Extension CSRs 199
Table 3-31 Hypervisor Status Register (partial) 201
Table 3-32 Virtual Supervisor Status Register (partial) 201
Table 3-33 Timer and Counter CSRs 202
Table 3-34 Accrued Exception Flags 202
Table 3-35 Floating-Point Rounding Modes 203
Table 3-36 Port Properties 204
Table 3-37 Physical Memory Attributes for External Regions 205
Table 3-38 Physical Memory Attributes for Internal Regions 205
Table 3-39 EIC7700X MCPU Memory Map 206
Table 3-40 EIC7700X MCPU Core Complex Memory Map. Physical Memory Attributes: R–Read, W–Write, X–Execute, I–Instruction Cacheable, D–Data Cacheable, A–Atomics 207
Table 3-41 Exception Priority 211
Table 3-42 mcause Exception Codes 212

Table 3-43 summary of Exception and Interrupt CSRs 213
Table 3-44 Machine Status Register (partial) 216
Table 3-45 Encoding of RISC-V Privilege Levels 217
Table 3-46 Machine Trap Vector Register 217
Table 3-47 Encoding of mtvec.MODE 218
Table 3-48 Machine Interrupt Enable Register 218
Table 3-49 Machine Interrupt Pending Register 219
Table 3-50 Machine Cause Register 220
Table 3-51 Machine Interrupt Delegation Register 221
Table 3-52 Machine Exception Delegation Register 221
Table 3-53 Supervisor Status Register (partial) 222
Table 3-54 Supervisor Status Register (partial) 223
Table 3-55 Supervisor Interrupt Pending Register 223
Table 3-56 Supervisor Cause Register 224
Table 3-57 scause Exception Codes 224
Table 3-58 Supervisor Trap Vector Register 226
Table 3-59 Encoding of stvec.MODE 226
Table 3-60 Hypervisor Interrupt Delegation Register 228
Table 3-61 Hypervisor Exception Delegation Register 228
Table 3-62 Hypervisor Status Register (partial) 229
Table 3-63 Virtual Supervisor Status Register (partial) 229
Table 3-64 Hypervisor Interrupt Enable Register 230
Table 3-65 Hypervisor Interrupt Pending Register 230
Table 3-66 Hypervisor Virtual Interrupt Pending Register 231
Table 3-67 Virtual Supervisor Interrupt Enable Register 232
Table 3-68 Virtual Supervisor Interrupt Pending Register 232
Table 3-69 RNMI CSRs 233
Table 3-70 mncause.exception_code Fields 233
Table 3-71 mnstatus CSR Fields 234
Table 3-72 EIC7700X MCPU Interrupt IDs 237
Table 3-73 CLINT Memory Map 237
Table 3-74 PLIC Memory Map 239
Table 3-75 PLIC Interrupt Source Mapping 243
Table 3-76 Mapping of global_interrupts Signal Bits to PLIC Interrupt ID 243
Table 3-77 PLIC Interrupt Priority Register 244

Table 3-78 PLIC Interrupt Pending Register 1 244
Table 3-79 PLIC Interrupt Pending Register 17 244
Table 3-80 PLIC Interrupt Enable Register 1 for Hart 0 M-Mode 245
Table 3-81 PLIC Interrupt Enable Register 17 for Hart 0 M-Mode 245
Table 3-82 PLIC Interrupt Enable Register 1 for Hart 0 S-Mode 246
Table 3-83 PLIC Interrupt Enable Register 17 for Hart 0 S-Mode 246
Table 3-84 PLIC Clock Gate Disable Register 246
Table 3-85 PLIC Interrupt Priority Threshold Register for Hart 0 M-Mode 247
Table 3-86 PLIC Interrupt Priority Threshold Register for Hart 0 S-Mode 247
Table 3-87 PLIC Claim/Complete Register for Hart 0 M-Mode 248
Table 3-88 PLIC Claim/Complete Register for Hart 0 S-Mode 248
Table 3-89 Private L2 Cache Instances 249
Table 3-90 Private L2 Cache Feature Set 249
Table 3-91 Private L2 Cache Memory Map 250
Table 3-92 PL2 Performance Monitor Event Selectors 251
Table 3-93 PL2 Performance Monitor Counter Client Filter 251
Table 3-94 Master IDs in the Private L2 Cache 251
Table 3-95 PL2 Performance Monitor Counters 252
Table 3-96 PL2PM pmEventSelect Register 252
Table 3-97 PL2 Cache Configuration Register 256
Table 3-98 ECC Error Injection Register 257
Table 3-99 ECC Error Injection Done Register 257
Table 3-100 PL2 Cache Flush Register 258
Table 3-101 Cache Flush Commands 258
Table 3-102 PL2 Flush Count Register 259
Table 3-103 PL2 Configuration 0 Register 259
Table 3-104 Bus-Error Unit Memory Map 261
Table 3-105 Bus-Error Unit Error Events 261
Table 3-106 TL-UH to TL-C Adapter Memory Map 263
Table 3-107 TL-UH to TL-C Adapter Bridge Control Register 0 264
Table 3-108 TL-UH to TL-C Adapter Bridge Control Register 1 265
Table 3-109 TL-UH to TL-C Adapter Bridge Status Register 265
Table 3-110 TL-UH to TL-C Adapter Error Containment Enable Register 266
Table 3-111 TL-UH to TL-C Adapter Error Signal Enable Register 266
Table 3-112 TL-UH to TL-C Adapter Error Signal Pending Register 266

Table 3-113 TL-UH to TL-C Adapter Error Signal Pending Register 267
Table 3-114 TL-UH to TL-C Adapter Error Claim Register 267
Table 3-115 TL-UH to TL-C Adapter Error Lost Register 268
Table 3-116 TL-UH to TL-C Adapter Error Log Enable Register 268
Table 3-117 TL-UH to TL-C Adapter Error Log Status Register 269
Table 3-118 TL-UH to TL-C Adapter Error Log Address Register 269
Table 3-119 TL-UH to TL-C Adapter Performance Monitor Global Control Register 270
Table 3-120 TL-UH to TL-C Adapter Performance Monitor Control Register 1 270
Table 3-121 TL-UH to TL-C Adapter Performance Monitor Data Register 272
Table 3-122 TL-UH to TL-C Adapter Performance Monitor Data Register 272
Table 3-123 Master IDs in the TL-UH to TL-C Adapter 273
Table 3-124 L3 Features Access Summary 277
Table 3-125 L3 Performance Monitor Event Selectors 277
Table 3-126 L3 Performance Monitor Counter Client Filter 278
Table 3-127 L3 Performance Monitor Counters 278
Table 3-128 L3PM pmEventSelect Register 279
Table 3-129 L3 Cache Controller Memory Map 282
Table 3-130 L3 Cache Controller Memory Map 283
Table 3-131 L3 Cache Configuration Register 284
Table 3-132 Way Enable Register 284
Table 3-133 ECC Error Injection Register 285
Table 3-134 Way Mask 0 Register 287
Table 3-135 Master IDs in the L3 Cache Controller 287
Table 3-136 L3 Cache Feature Disable Register 288
Table 3-137 Power Dial CSR 290
Table 3-138 Debug Module Memory Map Seen from the Debug Module Interface 294
Table 3-139 Debug Module Memory Map from the Perspective of the Core 295
Table 3-140 Debug Module Memory Map from the Perspective of the Core 296
Table 3-141 Debug and Trigger Registers 297
Table 3-142 Debug Control and Status Register 298
Table 3-143 Trigger Select Register 299
Table 3-144 Trigger Data Register 1 299
Table 3-145 Trigger Data Registers 2 and 3 300
Table 3-146 Trigger CSRs When Used as Breakpoints 300
Table 3-147 Breakpoint Match Control Register 300

Table 3-148 NAPOT Size Encoding 302
Table 3-149 Encoding of priv 305
Table 3-150 Instruction Trace Interface Fields 305
Table 3-151 Debug Module Status Register 306
Table 3-152 Debug Module Control Register 308
Table 3-153 Abstract Control and Status Register 309
Table 3-154 Abstract Command Register 310
Table 3-155 Abstract Command Autoexec Register 310
Table 3-156 Debug Module Control and Status 2 Register 311
Table 3-157 Debug Abstract Commands 311
Table 3-158 System Bus vs. Program Buffer Comparison 315
Table 3-159 Trace Encoder Memory Map 317
Table 3-160 Core to Trace Encoder Interface 318
Table 3-161 CKSRC Description 319
Table 3-162 Trace Encoder Control Register 320
Table 3-163 Trace Encoder Implementation Register 323
Table 3-164 Instrumentation Trace Enable Register 324
Table 3-165 Instrumentation Trigger Enable Register 325
Table 3-166 Instrumentation Trace Stimulus Register 325
Table 3-167 Trace Encoder Timestamp Control Register 326
Table 3-168 Trace Encoder Timestamp Control Register 327
Table 3-169 Trace Encoder Timestamp Lower Bits Register 327
Table 3-170 Trace Encoder Timestamp Upper Bits Register 328
Table 3-171 External Trigger Input Control Register 329
Table 3-172 External Trigger Output Register 330
Table 3-173 Watchpoint Action Identifiers 331
Table 3-174 Core Watchpoint Control Register 331
Table 3-175 PC Sample Control Register 333
Table 3-176 PC Capture and Sample Implementation Register 333
Table 3-177 PC Capture Register 334
Table 3-178 PC Capture High Register 335
Table 3-179 PC Sample Register 335
Table 3-180 PC Sample High Register 336
Table 3-181 Trace Funnel Control Register 337
Table 3-182 Trace Funnel Control Register 338

Table 3-183 Trace Funnel Implementation Register 338
Table 3-184 Trace Funnel Sink Base Register 339
Table 3-185 Trace Funnel Sink Base High Bits Register 339
Table 3-186 Trace Funnel Sink Limit Register 339
Table 3-187 Trace Funnel Sink Write Pointer Register 340
Table 3-188 RAS Optimization Example 341
Table 3-189 Trace Funnel Timestamp Lower Bits Register 341
Table 3-190 Trace Funnel Timestamp Upper Bits Register 342
Table 3-191 Direct Branch Message Description 342
Table 3-192 Indirect Branch Message Description 342
Table 3-193 Direct Branch with Sync Message Description 343
Table 3-194 Indirect Branch with Sync Message Description 343
Table 3-195 Program Trace Synchronization Message Description 343
Table 3-196 Error Message Description 344
Table 3-197 Program Trace Correlation Message Description 344
Table 3-198 In-Circuit Trace Message Description (Reports PC) 344
Table 3-199 Data Acquisition Message (from ITC) Description 345
Table 3-200 Ownership Trace Message (from ITC) Description 345
Table 3-201 Address Compression Example 345
Table 3-202 Timestamp Compression Description 346
Table 3-203 Nexus Transmission Protocol Description 347
Table 3-204 RAS Optimization Example 348
Table 3-3-205 Repeat Branch Optimization Example 1 349
Table 3-206 Repeat Branch Optimization Example 2 349
Table 3-207 Sync Messages 349
Table 3-208 Indirect Branch History Message 350
Table 3-209 Indirect Branch History with Sync Message 350
Table 3-210 Resource Full Message 351
Table 3-211 Program Trace Correlation Message (HTM format) 351
Table 3-212 Event Trace Control Register 352
Table 3-213 Event Trace Message Fields 353
Table 3-214 In-Circuit Trace Message 354
Table 3-215 Branch Prediction Mode CSR 356
Table 3-216 EIC7700X MCPU Feature Disable 0 CSR 357
Table 3-217 EIC7700X MCPU Feature Disable CSR 0 Usage 359

Table 3-218 EIC7700X MCPU Feature Disable 1 CSR 360
Table 3-219 RISC-V mcpu interrupt list 364
Table 3-220 SCPU interrupt list 374
Table 3-221 LPCPU interrupt list 376
Table 3-222 DSP0 interrupt list 378
Table 3-223 DSP1 interrupt list 378
Table 3-224 Configure registers of limiter mode 385
Table 3-225 Configure registers of regulator mode 386
Table 3-226 qos generator of NoCs 387
Table 3-227 transaction probe point 388
Table 3-228 error probe point information 390
Table 3-229 error transaction information 390
Table 3-230 DMA0 Req 391
Table 3-231 DMA1 req 392
Table 3-232 Interrupt Mapping 575
Table 3-233 Mailbox address mapping 576
Table 3-234 Registers of Mailbox 577
Table 3-235 combination of voltages 603
Table 3-236 TOP only power domain description 604
Table 3-237 AON only power domain description 605
Table 3-238 SNOC only power domain description 605
Table 3-239 GPU sys power domain description 606
Table 3-240 HSP sys power domain description 607
Table 3-241 NPU sys power domain description 607
Table 3-242 LPDDR0 sys power domain description 608
Table 3-243 LPDDR1 sys power domain description 609
Table 3-244 MCPU sys power domain description 609
Table 3-245 D2D sys power domain description 610
Table 3-246 DSP power domain description 611
Table 3-247 DSP power state description 611
Table 3-248 VI power domain description 612
Table 3-249 VI power state description 612
Table 3-250 VO power domain description 613
Table 3-251 VO power state description 613
Table 3-252 VI power domain description 614

Table 3-253 PCIE power domain description 615
Table 3-254 PCIE power state description 615
Table 3-255 PCU registers description 618
Table 3-256 PCU registers address description 620

1 Introduction

1.1 Overview

EIC7700X is a low power and high performance SoC chip for edge AI application base on TSMC 12nm FFC, embedded CPU which support RISC-V instruction set. EIC7700X Integrates independent NPU with computing power up to 9.3TOPS INT8, and provides a complete set of peripheral interface include audio and video.

1.2 Application

1.2.1 AI Box

The data delay and network bandwidth pressure handled by the traditional cloud infrastructure can no longer meet all applications and scenarios. Edge computing can provide solutions for this part of the market, and the front and rear devices can work together to achieve smooth interconnection. Through edge AI Box, not only can many intelligent products efficiently collaborate, but it can also connect with enterprise business and seamlessly integrate with multiple industry businesses.

1.2.2 Industrial Vision

The characteristic of industrial machine vision systems is to provide a degree of production automation, mainly in dangerous working environments that are not suitable for manual work or situations where artificial vision cannot meet the requirements. Machine vision is commonly used to replace artificial vision, and machine vision detection can greatly improve production efficiency and automation, which is the foundation for achieving intelligent manufacturing.

1.2.3 Educational Scenarios

Education Smart Station is an edge intelligent computing device for educational scenarios, with super strong audio and video decoding processing capabilities, high computing power, fast deployment, and easy management. The product is widely used in teaching interactive live streaming, online teaching and research, remote course supervision, intelligent learning analysis and other scenarios, providing efficient and high-quality intelligent processing and analysis capabilities for various types of audio and video data.

1.3 Architecture

1.3.1 Block diagram

EIC7700X SoC chip block diagram as bellow.

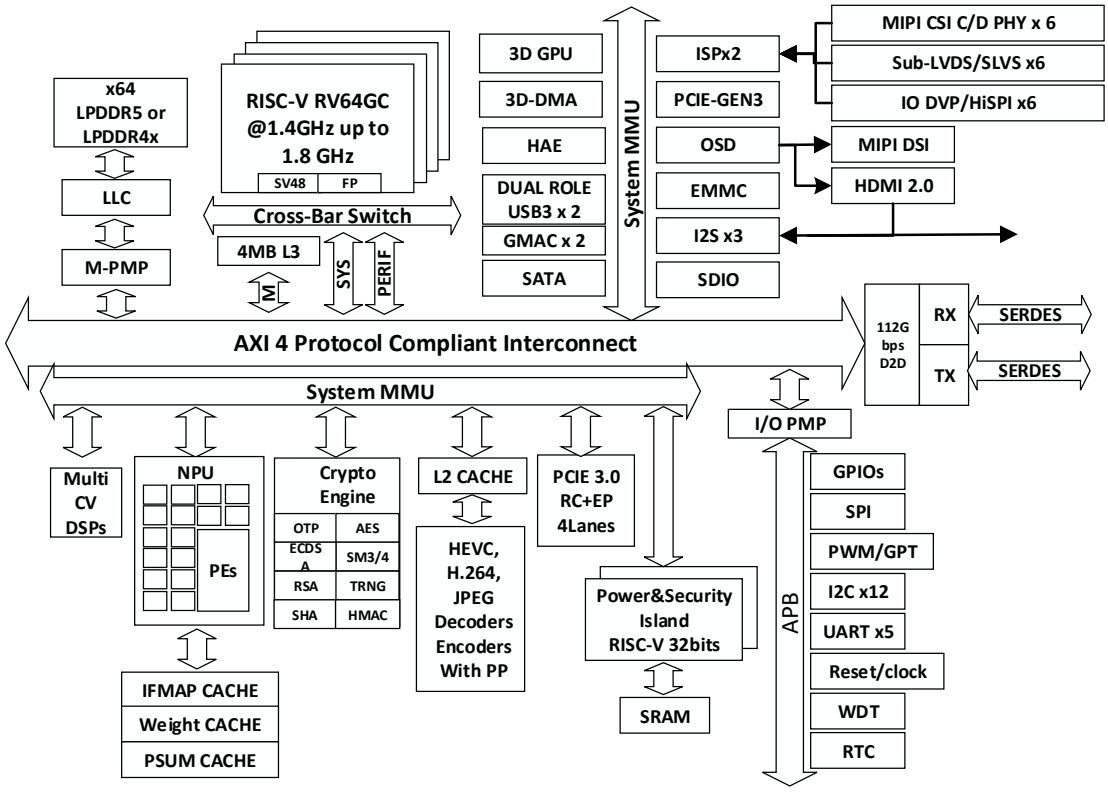


Figure 1-1 EIC700X block diagram

1.3.2 Processor

- RISC-V RV64GC 4core@1.4GHz, up to 1.8GHz
- Private L1 Cache 32K(I)+32KB(D)，private L2Cache 256KB，shared L3 Cache 4MB

1.3.3 Co-processor

- RV321 single core L1Cahe 64KB(I) + 64KB(D)

1.3.4 Intelligent accelerator

- NPU accelerator，up to 19.95 TOPS(INT8)，9.975TOPS(FP16 or INT16)
- Multiple DSPs @1040MHz for computer vision, support 512bit SIMD

1.3.5 Memory interface

- 8GB/16GB 64 bits LPDDR4x @4266MHz
- 8GB/16GB 64 bits LPDDR5 @6400MHz
- eMMC 5.1
- 2x SDIO
- SATA3@6GBps

1.3.6 Video encoder

- H.265(HEVC) 4K@60fps，13x1080@30fps
- H.264(AVC) 4K@60fps，10x1080p@30fps

1.3.7 JPEG encoder

- JPEG ISO/IEC 10918-1，ITU-T T.81

- Baseline process (support Huffman coding Interleaved YUV420, YUV422，Monochrome)
- Lossless process (support 8-bit with Huffman coding Interleaved YUV420，Monochrome)
- MJPEG format (T.81 Annex H) in AVI container

1.3.8 Video decoder

- H.265(H.264） 8K@50fps，4x4K@30fps，32x1080p@30fps
- H.264(AVC) 8K@30fps，4x4K@30fps，16x1080p@30fps

1.3.9 Video image proccess

- HAE(2D GPU)
- 3D GPU (support OpenGL-ES 3.2、EGL 1.4、OpenCL 1.2/2.1 EP2、Vulkan 1.2、Android NN HAL)
- ISP
- Distortion correction

1.3.10 Video input

- Support MIPI DPHY v2.1, CPHY v1.2 and Sub LVDS/SLVS/HiSPi interface
- Support up to 6 channels camera input, with 2lane(trio) per channel
- Support maximum 3.5Gbps/lane MIPI-DPHY interface
- Support maximum 3.5Gbps/trio MIPI-CPHY interface
- Single maximum 1.0Gbps/lane LVDS/Sub-LVDS/HiSPi interface
- Channel connection mode settings:

  Mode0: Reserved
  Mode1: Reserved
  Mode2: Channel 0,1 for 4lanes/3trios sensor
  Channel 2,3 for 4lanes/3trios sensor
  Channel 4,5 for 4lanes/3trios sensor
  Mode3: Channel 0,1 for 4lanes/3trios senor
  Channel 2,3 for 4lanes/3trios sensor
  Channel 4 for 2lanes/2trios sensor
  Channel 5 for 2lanes/2trios sensor
  Mode4: Channel 0,1 for 4lanes/3trios sensor
  Channel 2 for 2lanes/2trios sensor
  Channel 3 for 2lanes/2trios sensor
  Channel 4 for 2lanes/2trios sensor
  Channel 5 for 2lanes/2trios sensor
  Mode5: Channel 0 for 2lanes/2trios sensor
  Channel 1 for 2lanes/2trios sensor
  Channel 2 for 2lanes/2trios sensor
  Channel 3 for 2lanes/2trios sensor
  Channel 4 for 2lanes/2trios sensor
  Channel 5 for 2lanes/2trios sensor

1.3.11 Video output

- OSD(three layer which are video ,mouse,background)
- HDMI2.0 support HDCP1.4/2.2
- MIPI-DSI TX 2.5GHz 4lanes

1.3.12 High-Speed interface

- Dual mode (RC + EP) PCI-Express Gen3.0 4xLanes
- 2 x 10/100/1000 BASE-T GMAC
- DRD (Host + Device) 2 x USB 3.0

1.3.13 Security

- TEE (Trusted Execution Environment)，TRNG， RSA4096，AES，SM4，DES，HMAC(MD5,SHA-1,SHA-224.SHA256,SM-3,SHA512/224,SHA-512/256,SHA-384),CRC32,dual core hardware accelerate
- 16KB OTP

1.3.14 Other module

- 12 x I2C @ 1Mbps
- 5 x UARTs
- 2 x SPI@50MHz
- PWM
- 16xMaibox
- 112xGPIOs
- 4 x WDT
- RTC
- 3xJTAG@20MHz
- 3 x I2S (slave + master)@192kbps
- FAN_CTRL
- PVT

1.4 System Boot

1.4.1 Boot Mode

Two modes of startup are supported:

- Secure boot mode
  Booting from internal ROM is only allowed and verification signatures for bootrom and uboot are required.
- Non-secure mode
  According to boot_sel system can be booted from external storage, it can be configured whether the bootrom and uboot need to verify signatures.

The following boot medium are supported:

- UART
- NOR Flash
- eMMC
- USB

1.4.2 Boot Medium

The selection of boot mode and medium is decided by the security control bit and boot_sel, whose combination is shown in Table 1-1.

Table 1-1 Boot mode and medium selection table

security mode	boot_sel[3]	boot_sel[2]	boot_sel[1:0]	bootrom src	boot CPU	uboot src
1'b1	X	X	2'b00	ROM	SCPU	UART
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB

security mode	boot_sel[3]	boot_sel[2]	boot_sel[1:0]	bootrom src	boot CPU	uboot src
1'b0	1'b0	1'b0	2'b00	ROM	SCPU	UART
			2'b10			Boot-SPI
			2'b11			USB
		1'b1	2'b00	Boot-SPI		UART
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB
	1'b1	X	2'b00	Boot-SPI	MCPU	UART
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB

The security bit is automatically read from the OTP after the chip is powered on, and boot_sel is controlled by the external pins (BOOT_SEL0, BOOT_SEL1, BOOT_SEL2, BOOT_SEL3) of the chip.

In secure mode, bootrom is only allowed to be loaded from internal ROM, and bootrom code can only be executed by SCPU, opensbi and uboot code can be loaded from the corresponding source according to boot_sel[1:0] selection.

In non-secure mode, bootrom can be loaded from internal ROM or external NOR Flash according to boot_sel[2]. boot_sel[3] selects booting from SCPU or MCPU. The code of opensbi and uboot can be loaded from the corresponding source according to the boot_sel[1:0] selection. In non-secure mode, bootrom can only be loaded from external NOR Flash when booting from MCPU (boot_sel[3]= 1'b1).

1.4.3 Boot Flow

The system boot procedure flow is shown in Figure 1-2.

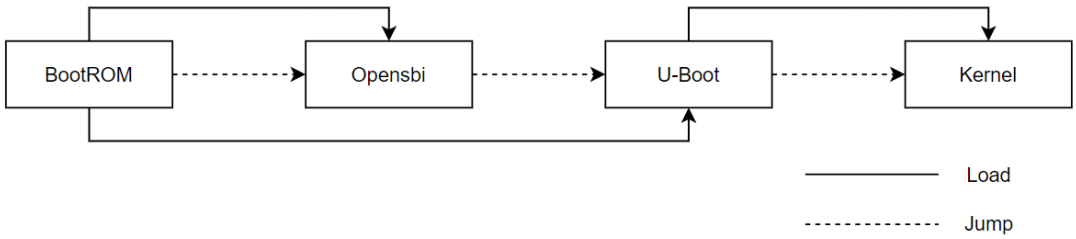


Figure 1-2 System boot procedure flow

BootROM runs on SCPU/MCPU (non-secure mode only), opensbi, uboot and kernel run on MCPU. The workflow of bootrom is mainly divided into three stages: system initialization, image loading and execution, and security services, as shown in Figure 1-3.

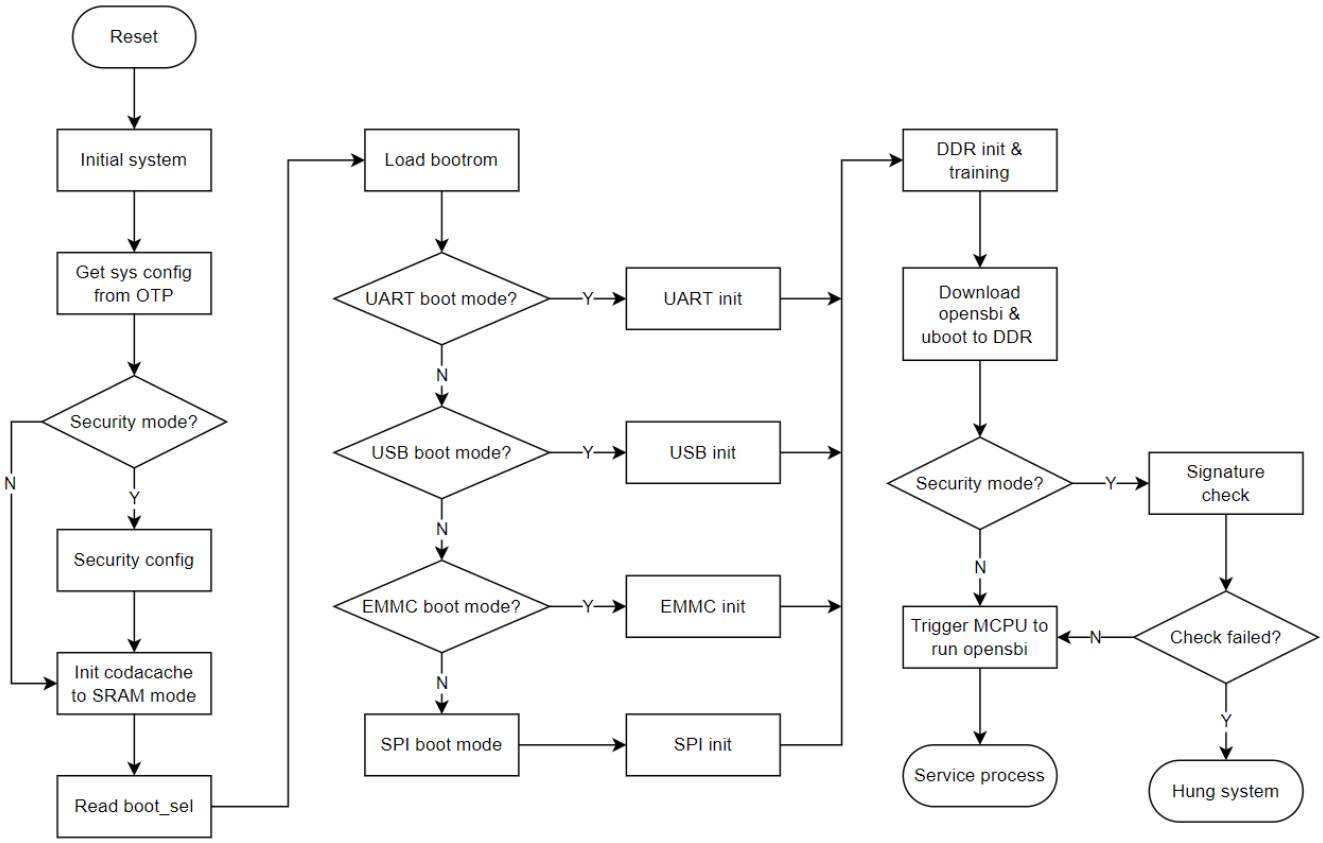


Figure 1-3 Bootrom workflow

- System initialization
  In the system initialization stage, the basic configuration of each module is completed, including:
  - Switch the main frequency of SCPU;
  - Configure the scu and set the relevant registers in sec_csr;
  - Initialize or configure uart, crypto engine, interrupt, malloc in sequence;
  - Read security mode from OTP control register;
  - In security mode, PMP initialization, security configuration and other preparations are needed;
  - In non-security mode, signature verification is not required when loading images, and MEM-PMP protection is not required for each system resource.
  - Initialize codacache to SRAM mode, codacache is 4Mbyte, which can temporarily store the code required for DDR phy initialization.
- Image loading and execution
  During booting, multiple images can be loaded consecutively, and the specific number can be obtained from the bootchain. The system chooses to boot from different media according to the boot_sel, which is divided into the following four situations:
  - UART

    When booting from UART, the image format accepted by the bootrom must be ihex, and the load address of the image must be configured in the boot file.
  - USB

    When booting from USB, the device need to be emulated as a disk.
  - eMMC

    When booting from eMMC, the eMMC driver needs to be initialized.

- NOR Flash

  When booting from NOR Flash, SPI-Flash needs to be initialized.

First, it receives the DDR init code, then receives the bootloader image package, loads it into the DDR, and finally triggers the MCPU to execute the bootloader. After receiving the complete image, if it is in security mode, bootrom needs to verify the signature of the image, and only after the signature verification is passed, the image can be processed and executed.

- Security service

  After all images are loaded and processed, the bootrom enters the security service processing state.

2 Hardware

2.1 Package and Pinout

2.1.1 Package

EIC7700X uses the flip-chip ball grid array (FCBGA) package. It has 1067 pins, its body size is 23 mm x 23 mm (0.91 in. x 0.91 in.), and its ball pitch is 0.65 mm (0.03 in.).

Figure 2-1 to Figure 2-4 show the package of EIC7700X . Figure 2-5 shows the package dimensions.

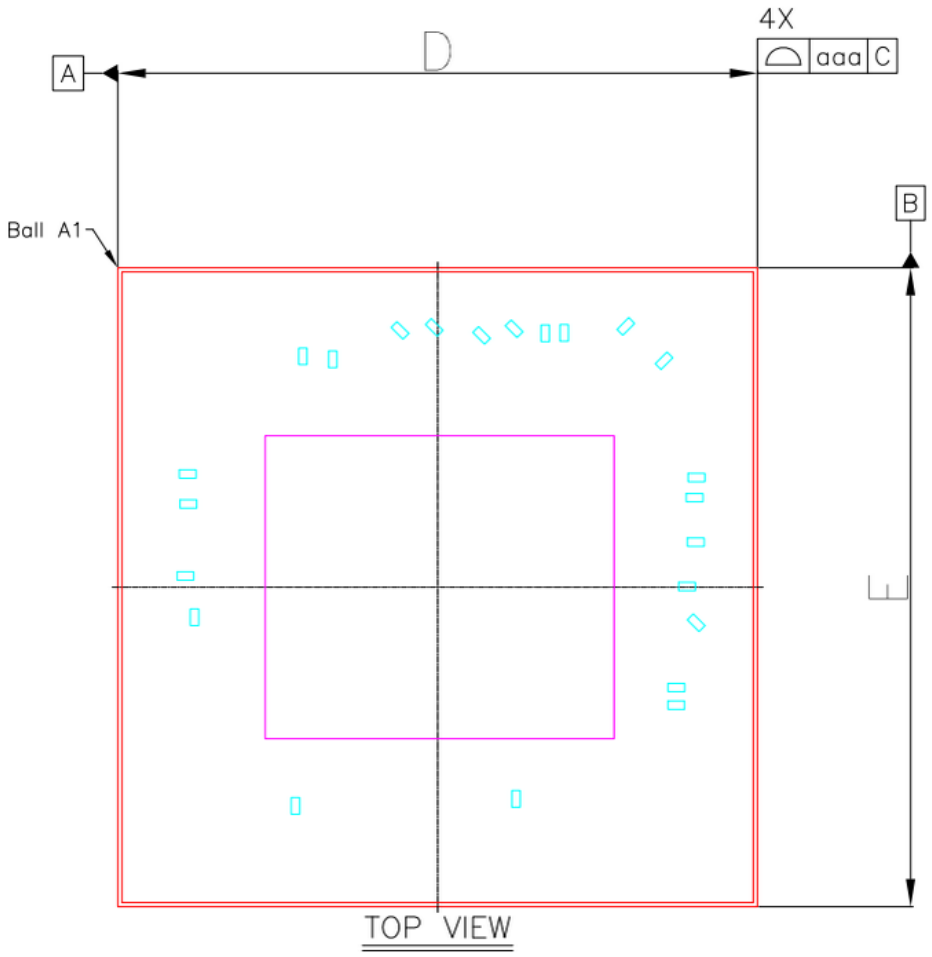


Figure 2-1 Top view

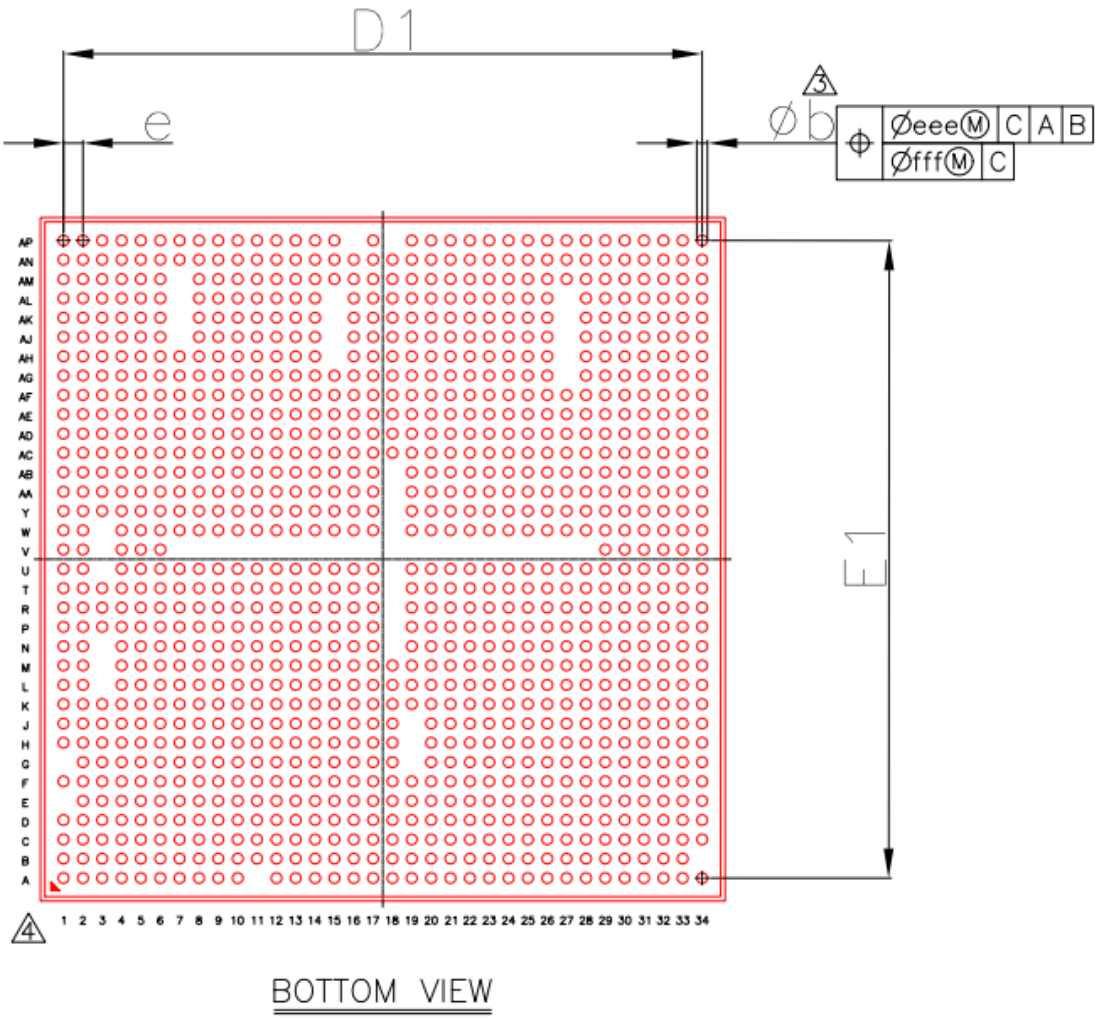


Figure 2-2 Bottom view

Figure 2-3 Side view

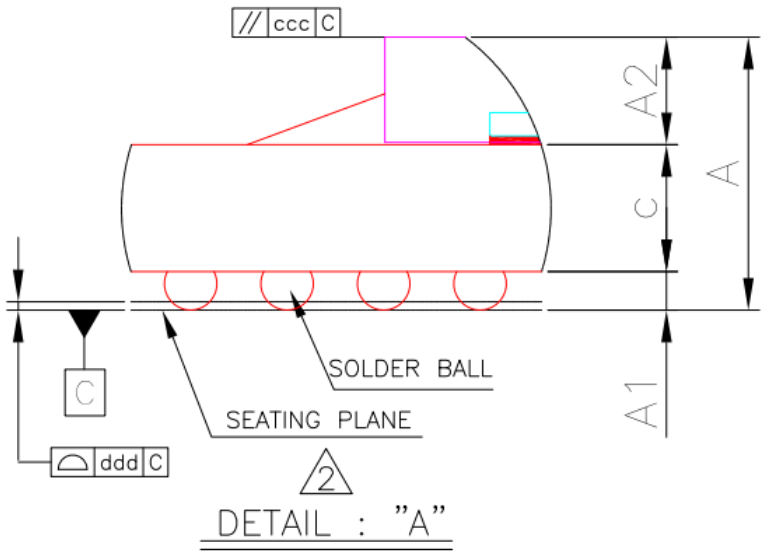


Figure 2-4 Enlarged view of detail "A"

Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.690	1.778	1.866	0.067	0.070	0.073
A1	0.200	0.250	0.300	0.008	0.010	0.012
A2	0.840	0.860	0.880	0.033	0.034	0.035
c	0.598	0.668	0.738	0.024	0.026	0.029
D	22.900	23.000	23.100	0.902	0.906	0.909
E	22.900	23.000	23.100	0.902	0.906	0.909
D1	---	21.450	---	---	0.844	---
E1	---	21.450	---	---	0.844	---
e	---	0.650	---	---	0.026	---
b	0.310	0.360	0.410	0.012	0.014	0.016
aaa	0.200			0.008		
ccc	0.200			0.008		
ddd	0.150			0.006		
eee	0.150			0.006		
fff	0.080			0.003		
MD/ME	34/34					

NOTE :

1. CONTROLLING DIMENSION : MILLIMETER.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
4. THE PATTERN OF PIN 1 FIDUCIAL IS FOR REFERENCE ONLY.
5. BALL PLACEMENT USE 0.35 mm SOLDER BALL. BGA PAD SOLDER MASK OPENING = 0.30 mm.

Figure 2-5 Package dimensions

2.1.2 Pinout

Table 2-1 lists the pin quantity of the EIC7700X by type.

Table 2-1 Pin quantit

Pin Type	Quantity
I/O	112
Digital power	177
Digital GND	443
Total	1097

2.2 Pin Number List

Table 2-2 Pin Number Order Information

Signal Name	Ball Number	Signal Name	Ball Number
C0_LPDDR_ATO	G22	C1_LPDDR_ATO	L29
C0_LPDDR_DTO	G23	C1_LPDDR_DTO	R29
C0_LPDDR_RESET_N	F24	C1_LPDDR_RESET_N	R28
C0_LPDDR_ZN	F22	C1_LPDDR_ZN	N28
C0_LPDDRA_CA0	A17	C1_LPDDRA_CA0	J34
C0_LPDDRA_CA1	B17	C1_LPDDRA_CA1	J33
C0_LPDDRA_CA2	A15	C1_LPDDRA_CA2	G34
C0_LPDDRA_CA3	F16	C1_LPDDRA_CA3	G29
C0_LPDDRA_CA4	B15	C1_LPDDRA_CA4	G33
C0_LPDDRA_CA5	D16	C1_LPDDRA_CA5	G31
C0_LPDDRA_CLK_C	E17	C1_LPDDRA_CLK_C	H30
C0_LPDDRA_CLK_T	D17	C1_LPDDRA_CLK_T	H31
C0_LPDDRA_DMI0	D20	C1_LPDDRA_DMI0	L31
C0_LPDDRA_DMI1	E14	C1_LPDDRA_DMI1	E32
C0_LPDDRA_DQ0	A18	C1_LPDDRA_DQ0	K34
C0_LPDDRA_DQ1	B18	C1_LPDDRA_DQ1	K33
C0_LPDDRA_DQ2	A19	C1_LPDDRA_DQ2	L34
C0_LPDDRA_DQ3	B19	C1_LPDDRA_DQ3	L33
C0_LPDDRA_DQ4	A20	C1_LPDDRA_DQ4	M34
C0_LPDDRA_DQ5	B20	C1_LPDDRA_DQ5	M33
C0_LPDDRA_DQ6	A21	C1_LPDDRA_DQ6	N34
C0_LPDDRA_DQ7	B21	C1_LPDDRA_DQ7	N33
C0_LPDDRA_DQ8	C11	C1_LPDDRA_DQ8	B32
C0_LPDDRA_DQ9	B12	C1_LPDDRA_DQ9	B33
C0_LPDDRA_DQ10	B11	C1_LPDDRA_DQ10	C34
C0_LPDDRA_DQ11	B13	C1_LPDDRA_DQ11	C33
C0_LPDDRA_DQ12	A13	C1_LPDDRA_DQ12	D34
C0_LPDDRA_DQ13	A12	C1_LPDDRA_DQ13	E34
C0_LPDDRA_DQ14	B14	C1_LPDDRA_DQ14	F34
C0_LPDDRA_DQ15	A14	C1_LPDDRA_DQ15	F33
C0_LPDDRA_DQS0_C	E21	C1_LPDDRA_DQS0_C	M30
C0_LPDDRA_DQS0_T	D21	C1_LPDDRA_DQS0_T	M31
C0_LPDDRA_DQS1_C	E13	C1_LPDDRA_DQS1_C	D32
C0_LPDDRA_DQS1_T	D13	C1_LPDDRA_DQS1_T	D33
C0_LPDDRA_LP4CKE0_LP5CS0	A16	C1_LPDDRA_LP4CKE0_LP5CS0	H34
C0_LPDDRA_LP4CKE1_LP5CS1	B16	C1_LPDDRA_LP4CKE1_LP5CS1	H33

Signal Name	Ball Number	Signal Name	Ball Number
C0_LPDDRA_LP4CS0_LP5CA6	D18	C1_LPDDRA_LP4CS0_LP5CA6	J31
C0_LPDDRA_LP4CS1	F18	C1_LPDDRA_LP4CS1	J29
C0_LPDDRA_WCK0_C	E19	C1_LPDDRA_WCK0_C	K30
C0_LPDDRA_WCK0_T	D19	C1_LPDDRA_WCK0_T	K31
C0_LPDDRA_WCK1_C	E15	C1_LPDDRA_WCK1_C	F31
C0_LPDDRA_WCK1_T	D15	C1_LPDDRA_WCK1_T	F32
C0_LPDDRB_CA0	A26	C1_LPDDRB_CA0	V34
C0_LPDDRB_CA1	B26	C1_LPDDRB_CA1	V33
C0_LPDDRB_CA2	A28	C1_LPDDRB_CA2	Y34
C0_LPDDRB_CA3	F28	C1_LPDDRB_CA3	W29
C0_LPDDRB_CA4	B28	C1_LPDDRB_CA4	Y33
C0_LPDDRB_CA5	D28	C1_LPDDRB_CA5	W31
C0_LPDDRB_CLK_C	E27	C1_LPDDRB_CLK_C	V30
C0_LPDDRB_CLK_T	D27	C1_LPDDRB_CLK_T	V31
C0_LPDDRB_DMI0	D30	C1_LPDDRB_DMI0	AA31
C0_LPDDRB_DMI1	D24	C1_LPDDRB_DMI1	R31
C0_LPDDRB_DQ0	A29	C1_LPDDRB_DQ0	AA34
C0_LPDDRB_DQ1	B29	C1_LPDDRB_DQ1	AA33
C0_LPDDRB_DQ2	A30	C1_LPDDRB_DQ2	AB34
C0_LPDDRB_DQ3	B30	C1_LPDDRB_DQ3	AB33
C0_LPDDRB_DQ4	A31	C1_LPDDRB_DQ4	AC34
C0_LPDDRB_DQ5	A32	C1_LPDDRB_DQ5	AC33
C0_LPDDRB_DQ6	B31	C1_LPDDRB_DQ6	AD34
C0_LPDDRB_DQ7	A33	C1_LPDDRB_DQ7	AD33
C0_LPDDRB_DQ8	A22	C1_LPDDRB_DQ8	P34
C0_LPDDRB_DQ9	B22	C1_LPDDRB_DQ9	P33
C0_LPDDRB_DQ10	A23	C1_LPDDRB_DQ10	R34
C0_LPDDRB_DQ11	B23	C1_LPDDRB_DQ11	R33
C0_LPDDRB_DQ12	A24	C1_LPDDRB_DQ12	T34
C0_LPDDRB_DQ13	B24	C1_LPDDRB_DQ13	T33
C0_LPDDRB_DQ14	A25	C1_LPDDRB_DQ14	U34
C0_LPDDRB_DQ15	B25	C1_LPDDRB_DQ15	U33
C0_LPDDRB_DQS0_C	E29	C1_LPDDRB_DQS0_C	AB30
C0_LPDDRB_DQS0_T	D29	C1_LPDDRB_DQS0_T	AB31
C0_LPDDRB_DQS1_C	E23	C1_LPDDRB_DQS1_C	P30
C0_LPDDRB_DQS1_T	D23	C1_LPDDRB_DQS1_T	P31
C0_LPDDRB_LP4CKE0_LP5CS0	B27	C1_LPDDRB_LP4CKE0_LP5CS0	W33
C0_LPDDRB_LP4CKE1_LP5CS1	A27	C1_LPDDRB_LP4CKE1_LP5CS1	W34
C0_LPDDRB_LP4CS0_LP5CA6	D26	C1_LPDDRB_LP4CS0_LP5CA6	U31
C0_LPDDRB_LP4CS1	F26	C1_LPDDRB_LP4CS1	U29
C0_LPDDRB_WCK0_C	D31	C1_LPDDRB_WCK0_C	Y30
C0_LPDDRB_WCK0_T	C31	C1_LPDDRB_WCK0_T	Y31
C0_LPDDRB_WCK1_C	E25	C1_LPDDRB_WCK1_C	T30
C0_LPDDRB_WCK1_T	D25	C1_LPDDRB_WCK1_T	T31
EMMC_CLK	A8	SDIO0_CLK	B3
EMMC_CMD	B9	SDIO0_CMD	A3
EMMC_D0	A9	SDIO0_D0	A4
EMMC_D1	B10	SDIO0_D1	B4
EMMC_D2	A10	SDIO0_D2	A2
EMMC_D3	B8	SDIO0_D3	B2

Signal Name	Ball Number	Signal Name	Ball Number
EMMC_D4	B7	SDIO1_CLK	E8
EMMC_D5	A7	SDIO1_CMD	D8
EMMC_D6	A6	SDIO1_D0	D9
EMMC_D7	A5	SDIO1_D1	E9
EMMC_DS	B6	SDIO1_D2	D7
EMMC_RSTN_OUT	B5	SDIO1_D3	E7
MIPI_CSI0_ATB	AK28	MIPI_CSI3_ATB	AG28
MIPI_CSI0_CKN	AN27	MIPI_CSI3_CKN	AK33
MIPI_CSI0_CKP	AP27	MIPI_CSI3_CKP	AK34
MIPI_CSI0_D0N	AN26	MIPI_CSI3_D0N	AJ33
MIPI_CSI0_D0P	AP26	MIPI_CSI3_D0P	AJ34
MIPI_CSI0_D1N	AN28	MIPI_CSI3_D1N	AL33
MIPI_CSI0_D1P	AP28	MIPI_CSI3_D1P	AL34
MIPI_CSI0_MCLK	AJ12	MIPI_CSI3_MCLK	AM13
MIPI_CSI0_REXT	AL30	MIPI_CSI3_REXT	AK31
MIPI_CSI0_XHS	AJ11	MIPI_CSI3_XHS	AL13
MIPI_CSI0_XVS	AK10	MIPI_CSI3_XVS	AL12
MIPI_CSI1_ATB	AJ28	MIPI_CSI4_ATB	AH30
MIPI_CSI1_CKN	AP30	MIPI_CSI4_CKN	AG33
MIPI_CSI1_CKP	AN30	MIPI_CSI4_CKP	AG34
MIPI_CSI1_D0N	AN31	MIPI_CSI4_D0N	AH33
MIPI_CSI1_D0P	AP31	MIPI_CSI4_D0P	AH34
MIPI_CSI1_D1N	AN29	MIPI_CSI4_D1N	AF33
MIPI_CSI1_D1P	AP29	MIPI_CSI4_D1P	AF34
MIPI_CSI1_MCLK	AM11	MIPI_CSI4_MCLK	AM14
MIPI_CSI1_REXT	AL31	MIPI_CSI4_REXT	AJ30
MIPI_CSI1_XHS	AL11	MIPI_CSI4_XHS	AN13
MIPI_CSI1_XVS	AK11	MIPI_CSI4_XVS	AP13
MIPI_CSI2_ATB	AH28	MIPI_CSI5_ATB	AH31
MIPI_CSI2_CKN	AN33	MIPI_CSI5_CKN	AF30
MIPI_CSI2_CKP	AP33	MIPI_CSI5_CKP	AF31
MIPI_CSI2_D0N	AN32	MIPI_CSI5_D0N	AE30
MIPI_CSI2_D0P	AP32	MIPI_CSI5_D0P	AE31
MIPI_CSI2_D1N	AM34	MIPI_CSI5_D1N	AG30
MIPI_CSI2_D1P	AM33	MIPI_CSI5_D1P	AG31
MIPI_CSI2_MCLK	AM12	MIPI_CSI5_MCLK	AL14
MIPI_CSI2_REXT	AK30	MIPI_CSI5_REXT	AJ31
MIPI_CSI2_XHS	AP12	MIPI_CSI5_XHS	AK13
MIPI_CSI2_XVS	AN12	MIPI_CSI5_XVS	AK14
USB0_DM	F1	USB1_DM	C1
USB0_DP	F2	USB1_DP	C2
USB0_ID	E5	USB1_ID	C4
USB0_PWREN	H6	USB1_PWREN	H5
USB0_REXT	D6	USB1_REXT	D5
USB0_SSRX_M	G5	USB1_SSRX_M	E2
USB0_SSRX_P	G6	USB1_SSRX_P	E3
USB0_SSTX_M	F4	USB1_SSTX_M	D1
USB0_SSTX_P	F5	USB1_SSTX_P	D2
USB0_VBUS	E6	USB1_VBUS	D4
DVDD_USB0	K10	DVDD_USB1	K11

Signal Name	Ball Number	Signal Name	Ball Number
RGMII0_CLK_125	H4	RGMII1_CLK_125	P1
RGMII0_INTB	J2	RGMII1_INTB	P2
RGMII0_MDC	N2	RGMII1_MDC	T4
RGMII0_MDIO	N1	RGMII1_MDIO	T5
RGMII0_RXCLK	J5	RGMII1_RXCLK	P3
RGMII0_RXD0	J3	RGMII1_RXD0	P5
RGMII0_RXD1	K3	RGMII1_RXD1	R5
RGMII0_RXD2	K4	RGMII1_RXD2	R4
RGMII0_RXD3	K5	RGMII1_RXD3	R3
RGMII0_RXDV	J4	RGMII1_RXDV	P4
RGMII0_TXCLK	K1	RGMII1_TXCLK	R1
RGMII0_TXD0	L1	RGMII1_TXD0	T1
RGMII0_TXD1	L2	RGMII1_TXD1	T2
RGMII0_TXD2	M1	RGMII1_TXD2	U1
RGMII0_TXD3	M2	RGMII1_TXD3	U2
RGMII0_TXEN	K2	RGMII1_TXEN	R2
HDMI_CEC	AJ21	PCIE_CLKN	AN15
HDMI_HPD	AN25	PCIE_CLKP	AP15
HDMI_REXT	AP25	PCIE_CLKREQ_N	AH14
HDMI_SCL	AL21	PCIE_PERST_N	AJ14
HDMI_SDA	AK21	PCIE_REXT	AH19
HDMI_TX0N	AP22	PCIE_RX0N	AK16
HDMI_TX0P	AN22	PCIE_RX0P	AJ16
HDMI_TX1N	AP23	PCIE_RX1N	AL17
HDMI_TX1P	AN23	PCIE_RX1P	AK17
HDMI_TX2N	AP24	PCIE_RX2N	AK18
HDMI_TX2P	AN24	PCIE_RX2P	AJ18
HDMI_TXCKN	AP21	PCIE_RX3N	AL19
HDMI_TXCKP	AN21	PCIE_RX3P	AK19
MIPI_DSI_ATB	AL28	PCIE_TX0N	AN16
MIPI_DSI_CKN	AK24	PCIE_TX0P	AM16
MIPI_DSI_CKP	AL24	PCIE_TX1N	AP17
MIPI_DSI_D0N	AL22	PCIE_TX1P	AN17
MIPI_DSI_D0P	AK22	PCIE_TX2N	AN18
MIPI_DSI_D1N	AK23	PCIE_TX2P	AM18
MIPI_DSI_D1P	AL23	PCIE_TX3N	AP19
MIPI_DSI_D2N	AL25	PCIE_TX3P	AN19
MIPI_DSI_D2P	AK25	PCIE_WAKE_N	AH17
MIPI_DSI_D3N	AK26	SATA_REXT	F7
MIPI_DSI_D3P	AL26	SATA_RXM	H1
MIPI_DSI_REXT	AL29	SATA_RXP	H2
KEY_RESET_N	AP3	SATA_TXM	G3
CHIP_MODE	G11	SATA_TXP	G2
MODE_SET0	F14	BOOT_SEL0	D11
MODE_SET1	G12	BOOT_SEL1	E11
MODE_SET2	G13	BOOT_SEL2	F11
MODE_SET3	G14	BOOT_SEL3	F12
S_MODE	G15	LPDDR_REF_CLK	H27
POR_SEL	AN3	RST_OUT_N	AN2
I2S_MCLK	AL5	JTAG0_TCK	AF1

Signal Name	Ball Number	Signal Name	Ball Number
I2S0_BCLK	AN6	JTAG0_TDI	AF2
I2S0_SDI	AP7	JTAG0_TDO	AG2
I2S0_SDO	AP6	JTAG0_TMS	AH1
I2S0_WCLK	AN7	JTAG1_TCK	AG4
I2S1_BCLK	AP4	JTAG1_TDI	AG5
I2S1_SDI	AP5	JTAG1_TDO	AH5
I2S1_SDO	AN5	JTAG1_TMS	AH4
I2S1_WCLK	AN4	JTAG2_TCK	AJ2
I2S2_BCLK	AK5	JTAG2_TDI	AJ3
I2S2_SDI	AM5	JTAG2_TDO	AJ1
I2S2_SDO	AM6	JTAG2_TMS	AJ4
I2S2_WCLK	AL6	JTAG2_TRST	AH2
SPI0_CLK	V1	I2C0_SCL	AE5
SPI0_CS_N	W1	I2C0_SDA	AF5
SPI0_D0	V2	I2C1_SCL	AE3
SPI0_D1	Y1	I2C1_SDA	AE4
SPI0_D2	W2	I2C10_SCL	AE1
SPI0_D3	Y2	I2C10_SDA	AE2
SPI1_CLK	AA1	I2C11_SCL	AF4
SPI1_CS0_N	AC1	I2C11_SDA	AF3
SPI1_CS1_N	AB2	I2C2_SCL	AK8
SPI1_D0	AA2	I2C2_SDA	AL8
SPI1_D1	AC3	I2C3_SCL	AM8
SPI1_D2	AC2	I2C3_SDA	AL9
SPI1_D3	AB1	I2C4_SCL	AN8
SPI2_CS0_N	AJ5	I2C4_SDA	AP8
SPI2_CS1_N	AJ6	I2C5_SCL	AM9
SPI3_CLK	AC4	I2C5_SDA	AL10
SPI3_CS_N	AC5	GPIO0	AM4
SPI3_DI	AD4	GPIO106	AL2
SPI3_DO	AD5	GPIO11	AH3
UART0_RX	Y4	GPIO111	AL4
UART0_TX	Y3	GPIO27	AL1
UART1_CTS	AA4	GPIO28	AM1
UART1_RTS	AB5	GPIO29	AM2
UART1_RX	Y5	GPIO34	AL3
UART1_TX	AA5	GPIO5	AG1
UART2_RX	AB4	GPIO92	AK4
UART2_TX	AB3	GPIO93	AK1
AVDDHV_PLL	L28	GPIO95	AK2
AVDDHV_PLL	AB17	FAN_PWM	AD2
AVDDHV_PLL	AC17	FAN_TACH	AD1
VDD_LPDDR	L20	VDDQ_LPDDR	J21
VDD_LPDDR	L21	VDDQ_LPDDR	J22
VDD_LPDDR	L22	VDDQ_LPDDR	J24
VDD_LPDDR	L23	VDDQ_LPDDR	K22
VDD_LPDDR	L24	VDDQ_LPDDR	K24
VDD_LPDDR	L25	VDDQ_LPDDR	K25
VDD_LPDDR	M25	VDDQ_LPDDR	K26
VDD_LPDDR	N25	VDDQ_LPDDR	M26

Signal Name	Ball Number	Signal Name	Ball Number
VDD_LPDDR	P25	VDDQ_LPDDR	M27
VDD_LPDDR	R25	VDDQ_LPDDR	P26
VDD_LPDDR	T25	VDDQ_LPDDR	P27
VDD_LPDDR	U25	VDDQ_LPDDR	T26
VDD2H_LPDDR	H22	VDDQ_LPDDR	T27
VDD2H_LPDDR	H23	VDDQ_LPDDR	U26
VDD2H_LPDDR	H25	VDD_NPU	L4
VDD2H_LPDDR	M28	VDD_NPU	L5
VDD2H_LPDDR	P28	VDD_NPU	L7
VDD2H_LPDDR	T28	VDD_NPU	L8
VAA_VDD2H_LPDDR0	G25	VDD_NPU	L11
VAA_VDD2H_LPDDR1	N29	VDD_NPU	L12
VDD_SOC	L18	VDD_NPU	L15
VDD_SOC	M18	VDD_NPU	L16
VDD_SOC	N17	VDD_NPU	M7
VDD_SOC	P17	VDD_NPU	M8
VDD_SOC	P19	VDD_NPU	M11
VDD_SOC	P20	VDD_NPU	M12
VDD_SOC	P22	VDD_NPU	M15
VDD_SOC	P23	VDD_NPU	M16
VDD_SOC	R19	VDD_NPU	N4
VDD_SOC	R20	VDD_NPU	N5
VDD_SOC	R22	VDD_NPU	N9
VDD_SOC	R23	VDD_NPU	N10
VDD_SOC	U17	VDD_NPU	N13
VDD_SOC	U20	VDD_NPU	N14
VDD_SOC	U21	VDD_NPU	P10
VDD_SOC	U22	VDD_NPU	P13
VDD_SOC	W20	VDD_NPU	P14
VDD_SOC	W23	VDD_NPU	R11
VDD_SOC	W24	VDD_NPU	R12
VDD_SOC	W27	VDD_NPU	R15
VDD_SOC	Y17	VDD_NPU	R16
VDD_SOC	Y20	VDD_NPU	T11
VDD_SOC	Y23	VDD_NPU	T12
VDD_SOC	Y24	VDD_NPU	T15
VDD_SOC	Y27	VDD_NPU	T16
VDD_SOC	AA21	VDD_NPU	U4
VDD_SOC	AA22	VDD_NPU	U5
VDD_SOC	AA25	VDD_NPU	U9
VDD_SOC	AA26	VDD_NPU	U10
VDD_SOC	AB19	VDD_NPU	U13
VDD_SOC	AB21	VDD_NPU	U14
VDD_SOC	AB22	VDD_NPU	W4
VDD_SOC	AB25	VDD_NPU	W5
VDD_SOC	AB26	VDD_NPU	W7
VDD_SOC	AC19	VDD_NPU	W9
VDD_SOC	AC21	VDD_NPU	W10
VDD_SOC	AC22	VDD_NPU	W13
VDD18_POR	AB16	VDD_NPU	W14

Signal Name	Ball Number	Signal Name	Ball Number
VDD18_PVT_DDR	J28	VDD_NPU	Y9
VDD18_PVT_SOC	W16	VDD_NPU	Y10
VDDIO_EMMCSD	J13	VDD_NPU	Y13
VDDIO_EMMCSD	K13	VDD_NPU	Y14
VDDIO33_EMMCSD	J15	VDD_CPU	AA11
VDDIO33_EMMCSD	K15	VDD_CPU	AA12
VDDIO_OTP	AF15	VDD_CPU	AB11
VDDIO18	J17	VDD_CPU	AB12
VDDIO18	K17	VDD_CPU	AB14
VDDIO18	AA7	VDD_CPU	AC13
VDDIO18	AA8	VDD_CPU	AC14
VDDIO18	AB7	VDD_CPU	AD13
VDDIO18	AB8	VDD_CPU	AD14
VDDIO18	AC9	VDD_CPU	AE11
VDDIO18	AC10	VDD_CPU	AE12
VDDIO18	AD9	VDD_CPU	AE14
VDDIO18	AD10	VDD_CPU	AF11
VDDIO18	AE7	VDD_CPU	AF12
VDDIO18	AE8	VDD_CPU	AG11
VDDIO18	AF7	VDD_CPU	AG12
VDDIO18	AF8	VDD_CPU	AH11
VDDIO18	AG7	VDD_CPU	AH12
VDDIO18	AG8	VP_CSI01	AE23
VDDIO18	AH7	VP_CSI01	AF23
VDDIO18	AH8	VP_CSI23	AE25
VDDIO18_RGMII	P8	VP_CSI23	AF25
VDDIO18_RGMII	R8	VP_CSI45	AE27
VDDIO18_RGMII	T8	VP_CSI45	AF27
VDDIO33_RGMII0	P6	VPH_CSI01	AD24
VDDIO33_RGMII0	P7	VPH_CSI23	AD25
VDDIO33_RGMII1	T6	VPH_CSI45	AD26
VDDIO33_RGMII1	T7	VP_DSI	AF21
VDDIO33_USB	H10	VPH_DSI	AE21
VDDIO33_USB	H11	VP_HDMI	AF19
VP_USB0	J9	VPH_HDMI	AE19
VP_USB1	J11	VP_SATA	J7
VPTX_USB0	J10	VPH_SATA	G8
VPTX_USB1	J12	VPTX_SATA	J8
VP_PCIE	AE16	VPH_PCIE	AD16
VP_PCIE	AE17	VPH_PCIE	AD17
XIN_24M	AP10	VSSA_PVT_DDR	H28
XOUT_24M	AN10	VSSA_PVT_SOC	W15
VSS	A1	VSS	V6
VSS	A34	VSS	V29
VSS	B1	VSS	V32
VSS	C3	VSS	W6
VSS	C5	VSS	W8
VSS	C6	VSS	W11
VSS	C7	VSS	W12
VSS	C8	VSS	W17

Signal Name	Ball Number	Signal Name	Ball Number
VSS	C9	VSS	W19
VSS	C10	VSS	W21
VSS	C12	VSS	W22
VSS	C13	VSS	W25
VSS	C14	VSS	W26
VSS	C15	VSS	W28
VSS	C16	VSS	W30
VSS	C17	VSS	W32
VSS	C18	VSS	Y6
VSS	C19	VSS	Y7
VSS	C20	VSS	Y8
VSS	C21	VSS	Y11
VSS	C22	VSS	Y12
VSS	C23	VSS	Y15
VSS	C24	VSS	Y16
VSS	C25	VSS	Y19
VSS	C26	VSS	Y21
VSS	C27	VSS	Y22
VSS	C28	VSS	Y25
VSS	C29	VSS	Y26
VSS	C30	VSS	Y28
VSS	C32	VSS	Y29
VSS	D3	VSS	Y32
VSS	D10	VSS	AA3
VSS	D12	VSS	AA6
VSS	D14	VSS	AA9
VSS	D22	VSS	AA10
VSS	E4	VSS	AA13
VSS	E10	VSS	AA14
VSS	E12	VSS	AA15
VSS	E16	VSS	AA16
VSS	E18	VSS	AA17
VSS	E20	VSS	AA19
VSS	E22	VSS	AA20
VSS	E24	VSS	AA23
VSS	E26	VSS	AA24
VSS	E28	VSS	AA27
VSS	E30	VSS	AA28
VSS	E31	VSS	AA29
VSS	E33	VSS	AA30
VSS	F3	VSS	AA32
VSS	F6	VSS	AB6
VSS	F8	VSS	AB9
VSS	F9	VSS	AB10
VSS	F10	VSS	AB13
VSS	F13	VSS	AB15
VSS	F15	VSS	AB20
VSS	F17	VSS	AB23
VSS	F19	VSS	AB24
VSS	F20	VSS	AB27

Signal Name	Ball Number	Signal Name	Ball Number
VSS	F21	VSS	AB28
VSS	F23	VSS	AB29
VSS	F25	VSS	AB32
VSS	F27	VSS	AC6
VSS	F29	VSS	AC7
VSS	F30	VSS	AC8
VSS	G4	VSS	AC11
VSS	G7	VSS	AC12
VSS	G9	VSS	AC15
VSS	G10	VSS	AC16
VSS	G16	VSS	AC18
VSS	G17	VSS	AC20
VSS	G18	VSS	AC23
VSS	G20	VSS	AC24
VSS	G21	VSS	AC25
VSS	G24	VSS	AC26
VSS	G26	VSS	AC27
VSS	G27	VSS	AC28
VSS	G28	VSS	AC29
VSS	G30	VSS	AC30
VSS	G32	VSS	AC31
VSS	H3	VSS	AC32
VSS	H7	VSS	AD3
VSS	H8	VSS	AD6
VSS	H9	VSS	AD7
VSS	H12	VSS	AD8
VSS	H13	VSS	AD11
VSS	H14	VSS	AD12
VSS	H15	VSS	AD15
VSS	H16	VSS	AD18
VSS	H17	VSS	AD19
VSS	H18	VSS	AD20
VSS	H20	VSS	AD21
VSS	H21	VSS	AD22
VSS	H24	VSS	AD23
VSS	H26	VSS	AD27
VSS	H29	VSS	AD28
VSS	H32	VSS	AD29
VSS	J1	VSS	AD30
VSS	J6	VSS	AD31
VSS	J14	VSS	AD32
VSS	J16	VSS	AE6
VSS	J18	VSS	AE9
VSS	J20	VSS	AE10
VSS	J23	VSS	AE13
VSS	J25	VSS	AE15
VSS	J26	VSS	AE18
VSS	J27	VSS	AE20
VSS	J30	VSS	AE22
VSS	J32	VSS	AE24

Signal Name	Ball Number	Signal Name	Ball Number
VSS	K6	VSS	AE26
VSS	K7	VSS	AE28
VSS	K8	VSS	AE29
VSS	K9	VSS	AE32
VSS	K12	VSS	AE33
VSS	K14	VSS	AE34
VSS	K16	VSS	AF6
VSS	K18	VSS	AF9
VSS	K19	VSS	AF10
VSS	K20	VSS	AF13
VSS	K21	VSS	AF14
VSS	K23	VSS	AF16
VSS	K27	VSS	AF17
VSS	K28	VSS	AF18
VSS	K29	VSS	AF20
VSS	K32	VSS	AF22
VSS	L6	VSS	AF24
VSS	L9	VSS	AF26
VSS	L10	VSS	AF28
VSS	L13	VSS	AF29
VSS	L14	VSS	AF32
VSS	L17	VSS	AG3
VSS	L19	VSS	AG6
VSS	L26	VSS	AG9
VSS	L27	VSS	AG10
VSS	L30	VSS	AG13
VSS	L32	VSS	AG14
VSS	M4	VSS	AG15
VSS	M5	VSS	AG16
VSS	M6	VSS	AG17
VSS	M9	VSS	AG18
VSS	M10	VSS	AG19
VSS	M13	VSS	AG20
VSS	M14	VSS	AG21
VSS	M17	VSS	AG22
VSS	M19	VSS	AG23
VSS	M20	VSS	AG24
VSS	M21	VSS	AG25
VSS	M22	VSS	AG26
VSS	M23	VSS	AG29
VSS	M24	VSS	AG32
VSS	M29	VSS	AH6
VSS	M32	VSS	AH9
VSS	N6	VSS	AH10
VSS	N7	VSS	AH13
VSS	N8	VSS	AH16
VSS	N11	VSS	AH18
VSS	N12	VSS	AH20
VSS	N15	VSS	AH21
VSS	N16	VSS	AH22

Signal Name	Ball Number	Signal Name	Ball Number
VSS	N19	VSS	AH23
VSS	N20	VSS	AH24
VSS	N21	VSS	AH25
VSS	N22	VSS	AH26
VSS	N23	VSS	AH29
VSS	N24	VSS	AH32
VSS	N26	VSS	AJ8
VSS	N27	VSS	AJ9
VSS	N30	VSS	AJ10
VSS	N31	VSS	AJ13
VSS	N32	VSS	AJ17
VSS	P9	VSS	AJ19
VSS	P11	VSS	AJ20
VSS	P12	VSS	AJ22
VSS	P15	VSS	AJ23
VSS	P16	VSS	AJ24
VSS	P21	VSS	AJ25
VSS	P24	VSS	AJ26
VSS	P29	VSS	AJ29
VSS	P32	VSS	AJ32
VSS	R6	VSS	AK3
VSS	R7	VSS	AK6
VSS	R9	VSS	AK9
VSS	R10	VSS	AK12
VSS	R13	VSS	AK20
VSS	R14	VSS	AK29
VSS	R17	VSS	AK32
VSS	R21	VSS	AL16
VSS	R24	VSS	AL18
VSS	R26	VSS	AL20
VSS	R27	VSS	AL32
VSS	R30	VSS	AM3
VSS	R32	VSS	AM10
VSS	T3	VSS	AM15
VSS	T9	VSS	AM17
VSS	T10	VSS	AM19
VSS	T13	VSS	AM20
VSS	T14	VSS	AM21
VSS	T17	VSS	AM22
VSS	T19	VSS	AM23
VSS	T20	VSS	AM24
VSS	T21	VSS	AM25
VSS	T22	VSS	AM26
VSS	T23	VSS	AM27
VSS	T24	VSS	AM28
VSS	T29	VSS	AM29
VSS	T32	VSS	AM30
VSS	U6	VSS	AM31
VSS	U7	VSS	AM32
VSS	U8	VSS	AN1

Signal Name	Ball Number	Signal Name	Ball Number
VSS	U11	VSS	AN9
VSS	U12	VSS	AN11
VSS	U15	VSS	AN14
VSS	U16	VSS	AN20
VSS	U19	VSS	AN34
VSS	U23	VSS	AP1
VSS	U24	VSS	AP2
VSS	U27	VSS	AP9
VSS	U28	VSS	AP11
VSS	U30	VSS	AP14
VSS	U32	VSS	AP20
VSS	V4	VSS	AP34
VSS	V5		

2.3 Signal Description

The EIC7700X application processor pin assignment is described in the following table

Table 2-3 Signal Description

Signal Name	Description	Type	Power Supply
LPDDR Chanel0			
C0_LPDDR_ATO	C0 Analog test output	O	VDDQ_LPDDR
C0_LPDDR_DTO	C0 Digital test output	I/O	VDD2H_LPDDR
C0_LPDDR_RESET_N	C0 DRAM reset signal output, active low.	O	VDD2H_LPDDR
C0_LPDDR_ZN	C0 Calibration reference, connected to VDDQ through a 240Ω ±1% resistor.	I/O	VDDQ_LPDDR
C0_LPDDRA_CA0	C0 Rank A CA0	O	VDDQ_LPDDR
C0_LPDDRA_CA1	C0 Rank A CA1	O	VDDQ_LPDDR
C0_LPDDRA_CA2	C0 Rank A CA2	O	VDDQ_LPDDR
C0_LPDDRA_CA3	C0 Rank A CA3	O	VDDQ_LPDDR
C0_LPDDRA_CA4	C0 Rank A CA4	O	VDDQ_LPDDR
C0_LPDDRA_CA5	C0 Rank A CA5	O	VDDQ_LPDDR
C0_LPDDRA_CLK_C	C0 Rank A differential clock output negative.	O	VDDQ_LPDDR
C0_LPDDRA_CLK_T	C0 Rank A differential clock output positive.	O	VDDQ_LPDDR
C0_LPDDRA_DMI0	C0 Rank A DM0 and DBI0	O	VDDQ_LPDDR
C0_LPDDRA_DMI1	C0 Rank A DM1 and DBI1	O	VDDQ_LPDDR
C0_LPDDRA_DQ0	C0 Rank A Data0	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ1	C0 Rank A Data1	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ2	C0 Rank A Data2	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ3	C0 Rank A Data3	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ4	C0 Rank A Data4	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ5	C0 Rank A Data5	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ6	C0 Rank A Data6	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ7	C0 Rank A Data7	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ8	C0 Rank A Data8	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ9	C0 Rank A Data9	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ10	C0 Rank A Data10	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ11	C0 Rank A Data11	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ12	C0 Rank A Data12	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ13	C0 Rank A Data13	I/O	VDDQ_LPDDR

Signal Name	Description	Type	Power Supply
C0_LPDDRA_DQ14	C0 Rank A Data14	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ15	C0 Rank A Data15	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS0_C	C0 Rank A Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS0_T	C0 Rank A Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS1_C	C0 Rank A Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS1_T	C0 Rank A Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRA_LP4CKE0_LP5CS0	LPDDR5: C0 Rank A CS0 LPDDR4/x:C0 Rank A CKE0	O	VDD2H_LPDDR
C0_LPDDRA_LP4CKE1_LP5CS1	LPDDR5: C0 Rank A CS1 LPDDR4/x:C0 Rank A CKE1	O	VDD2H_LPDDR
C0_LPDDRA_LP4CS0_LP5CA6	LPDDR5: C0 Rank A CA6 LPDDR4/x:C0 Rank A CS0	O	VDDQ_LPDDR
C0_LPDDRA_LP4CS1	LPDDR5: Not used LPDDR4/x:C0 Rank A CS1	O	VDDQ_LPDDR
C0_LPDDRA_WCK0_C	C0 Rank A Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK0_T	C0 Rank A Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK1_C	C0 Rank A Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK1_T	C0 Rank A Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRB_CA0	C0 Rank B CA0	I/O	VDDQ_LPDDR
C0_LPDDRB_CA1	C0 Rank B CA1	I/O	VDDQ_LPDDR
C0_LPDDRB_CA2	C0 Rank B CA2	I/O	VDDQ_LPDDR
C0_LPDDRB_CA3	C0 Rank B CA3	I/O	VDDQ_LPDDR
C0_LPDDRB_CA4	C0 Rank B CA4	I/O	VDDQ_LPDDR
C0_LPDDRB_CA5	C0 Rank B CA5	I/O	VDDQ_LPDDR
C0_LPDDRB_CLK_C	C0 Rank B differential clock output negative.	I/O	VDDQ_LPDDR
C0_LPDDRB_CLK_T	C0 Rank B differential clock output positive.	I/O	VDDQ_LPDDR
C0_LPDDRB_DMI0	C0 Rank B DM0 and DBI0	I/O	VDDQ_LPDDR
C0_LPDDRB_DMI1	C0 Rank B DM1 and DBI1	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ0	C0 Rank B Data0	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ1	C0 Rank B Data1	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ2	C0 Rank B Data2	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ3	C0 Rank B Data3	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ4	C0 Rank B Data4	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ5	C0 Rank B Data5	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ6	C0 Rank B Data6	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ7	C0 Rank B Data7	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ8	C0 Rank B Data8	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ9	C0 Rank B Data9	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ10	C0 Rank B Data10	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ11	C0 Rank B Data11	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ12	C0 Rank B Data12	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ13	C0 Rank B Data13	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ14	C0 Rank B Data14	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ15	C0 Rank B Data15	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS0_C	C0 Rank B Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS0_T	C0 Rank B Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS1_C	C0 Rank B Byte1 Data strobe negative	I/O	VDDQ_LPDDR

Signal Name	Description	Type	Power Supply
C0_LPDDRB_DQS1_T	C0 Rank B Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRB_LP4CKE0_LP5CS0	LPDDR5: C0 Rank B CS0 LPDDR4/x:C0 Rank B CKE0	I/O	VDD2H_LPDDR
C0_LPDDRB_LP4CKE1_LP5CS1	LPDDR5: C0 Rank B CS1 LPDDR4/x:C0 Rank B CKE1	I/O	VDD2H_LPDDR
C0_LPDDRB_LP4CS0_LP5CA6	LPDDR5: C0 Rank B CA6 LPDDR4/x:C0 Rank B CS0	I/O	VDDQ_LPDDR
C0_LPDDRB_LP4CS1	LPDDR5: Not used LPDDR4/x:C0 Rank B CS1	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK0_C	C0 Rank B Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK0_T	C0 Rank B Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK1_C	C0 Rank B Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK1_T	C0 Rank B Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
LPDDR Chanel1			
C1_LPDDR_ATO	C1 Analog test output	O	VDDQ_LPDDR
C1_LPDDR_DTO	C1 Digital test output	I/O	VDD2H_LPDDR
C1_LPDDR_RESET_N	C1 DRAM reset signal output, active low.	O	VDD2H_LPDDR
C1_LPDDR_ZN	C1 Calibration reference, connected to VDDQ through a 240Ω ±1% resistor.	I/O	VDDQ_LPDDR
C1_LPDDRA_CA0	C1 Rank A CA0	O	VDDQ_LPDDR
C1_LPDDRA_CA1	C1 Rank A CA1	O	VDDQ_LPDDR
C1_LPDDRA_CA2	C1 Rank A CA2	O	VDDQ_LPDDR
C1_LPDDRA_CA3	C1 Rank A CA3	O	VDDQ_LPDDR
C1_LPDDRA_CA4	C1 Rank A CA4	O	VDDQ_LPDDR
C1_LPDDRA_CA5	C1 Rank A CA5	O	VDDQ_LPDDR
C1_LPDDRA_CLK_C	C1 Rank A differential clock output negative.	O	VDDQ_LPDDR
C1_LPDDRA_CLK_T	C1 Rank A differential clock output positive.	O	VDDQ_LPDDR
C1_LPDDRA_DMI0	C1 Rank A DM0 and DBI0	O	VDDQ_LPDDR
C1_LPDDRA_DMI1	C1 Rank A DM1 and DBI1	O	VDDQ_LPDDR
C1_LPDDRA_DQ0	C1 Rank A Data0	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ1	C1 Rank A Data1	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ2	C1 Rank A Data2	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ3	C1 Rank A Data3	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ4	C1 Rank A Data4	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ5	C1 Rank A Data5	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ6	C1 Rank A Data6	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ7	C1 Rank A Data7	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ8	C1 Rank A Data8	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ9	C1 Rank A Data9	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ10	C1 Rank A Data10	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ11	C1 Rank A Data11	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ12	C1 Rank A Data12	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ13	C1 Rank A Data13	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ14	C1 Rank A Data14	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ15	C1 Rank A Data15	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS0_C	C1 Rank A Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS0_T	C1 Rank A Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS1_C	C1 Rank A Byte1 Data strobe negative	I/O	VDDQ_LPDDR

Signal Name	Description	Type	Power Supply
C1_LPDDRA_DQS1_T	C1 Rank A Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRA_LP4CKE0_LP5CS0	LPDDR5: C1 Rank A CS0 LPDDR4/x:C1 Rank A CKE0	O	VDD2H_LPDDR
C1_LPDDRA_LP4CKE1_LP5CS1	LPDDR5: C1 Rank A CS1 LPDDR4/x:C1 Rank A CKE1	O	VDD2H_LPDDR
C1_LPDDRA_LP4CS0_LP5CA6	LPDDR5: C1 Rank A CA6 LPDDR4/x:C1 Rank A CS0	O	VDDQ_LPDDR
C1_LPDDRA_LP4CS1	LPDDR5: Not used LPDDR4/x:C1 Rank A CS1	O	VDDQ_LPDDR
C1_LPDDRA_WCK0_C	C1 Rank A Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK0_T	C1 Rank A Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK1_C	C1 Rank A Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK1_T	C1 Rank A Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRB_CA0	C1 Rank B CA0	I/O	VDDQ_LPDDR
C1_LPDDRB_CA1	C1 Rank B CA1	I/O	VDDQ_LPDDR
C1_LPDDRB_CA2	C1 Rank B CA2	I/O	VDDQ_LPDDR
C1_LPDDRB_CA3	C1 Rank B CA3	I/O	VDDQ_LPDDR
C1_LPDDRB_CA4	C1 Rank B CA4	I/O	VDDQ_LPDDR
C1_LPDDRB_CA5	C1 Rank B CA5	I/O	VDDQ_LPDDR
C1_LPDDRB_CLK_C	C1 Rank B differential clock output negative.	I/O	VDDQ_LPDDR
C1_LPDDRB_CLK_T	C1 Rank B differential clock output positive.	I/O	VDDQ_LPDDR
C1_LPDDRB_DMI0	C1 Rank B DM0 and DBI0	I/O	VDDQ_LPDDR
C1_LPDDRB_DMI1	C1 Rank B DM1 and DBI1	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ0	C1 Rank B Data0	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ1	C1 Rank B Data1	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ2	C1 Rank B Data2	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ3	C1 Rank B Data3	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ4	C1 Rank B Data4	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ5	C1 Rank B Data5	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ6	C1 Rank B Data6	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ7	C1 Rank B Data7	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ8	C1 Rank B Data8	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ9	C1 Rank B Data9	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ10	C1 Rank B Data10	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ11	C1 Rank B Data11	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ12	C1 Rank B Data12	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ13	C1 Rank B Data13	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ14	C1 Rank B Data14	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ15	C1 Rank B Data15	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS0_C	C1 Rank B Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS0_T	C1 Rank B Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS1_C	C1 Rank B Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS1_T	C1 Rank B Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRB_LP4CKE0_LP5CS0	LPDDR5: C1 Rank B CS0 LPDDR4/x:C1 Rank B CKE0	I/O	VDD2H_LPDDR
C1_LPDDRB_LP4CKE1_LP5CS1	LPDDR5: C1 Rank B CS1 LPDDR4/x: C1 Rank B CKE1	I/O	VDD2H_LPDDR

Signal Name	Description	Type	Power Supply
C1_LPDDRB_LP4CS0_LP5CA6	LPDDR5: C1 Rank B CA6 LPDDR4/x: C1 Rank B CS0	I/O	VDDQ_LPDDR
C1_LPDDRB_LP4CS1	LPDDR5: Not used LPDDR4/x: C1 Rank B CS1	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK0_C	C1 Rank B Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK0_T	C1 Rank B Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK1_C	C1 Rank B Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK1_T	C1 Rank B Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
EMMC			
EMMC_RSTN_OUT	EMMC Flah reset output	O	VDDIO_EMMCSD / VDDIO33_EMMCSD
EMMC_DS	EMMC Data strobe: Generated by EMMC flash and used for data O and CRC status response O in HS400 mode.	I	
EMMC_CLK	EMMC Clock O	O	
EMMC_CMD	EMMC Command O	O	
EMMC_D0	EMMC data line0	I/O	
EMMC_D1	EMMC data line1	I/O	
EMMC_D2	EMMC data line2	I/O	
EMMC_D3	EMMC data line3	I/O	
EMMC_D4	EMMC data line4	I/O	
EMMC_D5	EMMC data line5	I/O	
EMMC_D6	EMMC data line6	I/O	
EMMC_D7	EMMC data line7	I/O	
SDIO			
SDIO0_CLK	SDIO0 Clock Output	O	VDDIO_EMMCSD / VDDIO33_EMMCSD
SDIO0_CMD	SDIO0 Command Output	O	
SDIO0_D0	SDIO0 data line0	I/O	
SDIO0_D1	SDIO0 data line1	I/O	
SDIO0_D2	SDIO0 data line2	I/O	
SDIO0_D3	SDIO0 data line3	I/O	
SDIO0_DETECT	SDIO0 SD card detect Input	I	
SDIO0_WRITE_PROT	SDIO0 SD card write protect Input	I	
SDIO1_CLK	SDIO1 Clock Output	O	VDDIO_EMMCSD / VDDIO33_EMMCSD
SDIO1_CMD	SDIO1 Command Output	O	
SDIO1_D0	SDIO1 data line0	I/O	
SDIO1_D1	SDIO1 data line1	I/O	
SDIO1_D2	SDIO1 data line2	I/O	
SDIO1_D3	SDIO1 data line3	I/O	
SDIO1_DETECT	SDIO1 SD card detect Input	I	
SDIO1_WRITE_PROT	SDIO1 SD card write protect Input	I	
SATA			
SATA_RXM	SATA Receiver Lane minus terminal	I	VP_SATA
SATA_RXP	SATA Receiver Lane positive terminal	I	VP_SATA
SATA_TXM	SATA Transmitter Lane minus terminal	O	VPTX_SATA
SATA_TXP	SATA Transmitter Lane positive terminal	O	VPTX_SATA

Signal Name	Description	Type	Power Supply
SATA_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_SATA
USB			
USB0_SSRX_M	USB 3.0 SuperSpeed receiver Lane minus terminal	I	VP_USB
USB0_SSRX_P	USB 3.0 SuperSpeed receiver positive terminal	I	VP_USB
USB0_SSTX_M	USB 3.0 SuperSpeed transmitter Lane minus terminal	O	VPTX_USB
USB0_SSTX_P	USB 3.0 SuperSpeed transmitter positive terminal	O	VPTX_USB
USB0_DM	USB2.0 Data minus	I/O	VDDIO33_USB
USB0_DP	USB2.0 Data positive	I/O	VDDIO33_USB
USB0_VBUS	30K 1% resistor connector to VBUS	I	VDDIO33_USB
USB0_ID	Test pin, do not connect anything	O	VDDIO33_USB
USB0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VDDIO33_USB
USB0_PWREN	USB Power enable	O	VDDIO18
USB1_SSRX_M	USB 3.0 SuperSpeed receiver Lane minus terminal	I	VP_USB
USB1_SSRX_P	USB 3.0 SuperSpeed receiver positive terminal	I	VP_USB
USB1_SSTX_M	USB 3.0 SuperSpeed transmitter Lane minus terminal	O	VPTX_USB
USB1_SSTX_P	USB 3.0 SuperSpeed transmitter positive terminal	O	VPTX_USB
USB1_DM	USB2.0 Data minus	I/O	VDDIO33_USB
USB1_DP	USB2.0 Data positive	I/O	VDDIO33_USB
USB1_VBUS	30K 1% resistor connector to VBUS	I	VDDIO33_USB
USB1_ID	Test pin, do not connect anything	O	VDDIO33_USB
USB1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VDDIO33_USB
USB1_PWREN	USB Power enable	O	VDDIO18
PCIE			
PCIE_CLKN	PCIE reference clock lane negative terminal	I	VP_PCIE
PCIE_CLKP	PCIE reference clock lane positive terminal	I	VP_PCIE
PCIE_RX0N	PCIE RX lane0 negative terminal	I	VP_PCIE
PCIE_RX0P	PCIE RX lane0 positive terminal	I	VP_PCIE
PCIE_RX1N	PCIE RX lane1 negative terminal	I	VP_PCIE
PCIE_RX1P	PCIE RX lane1 positive terminal	I	VP_PCIE
PCIE_RX2N	PCIE RX lane2 negative terminal	I	VP_PCIE
PCIE_RX2P	PCIE RX lane2 positive terminal	I	VP_PCIE
PCIE_RX3N	PCIE RX lane3 negative terminal	I	VP_PCIE
PCIE_RX3P	PCIE RX lane3 positive terminal	I	VP_PCIE
PCIE_TX0N	PCIE TX lane0 negative terminal	O	VP_PCIE
PCIE_TX0P	PCIE TX lane0 positive terminal	O	VP_PCIE
PCIE_TX1N	PCIE TX lane1 negative terminal	O	VP_PCIE
PCIE_TX1P	PCIE TX lane1 positive terminal	O	VP_PCIE
PCIE_TX2N	PCIE TX lane2 negative terminal	O	VP_PCIE
PCIE_TX2P	PCIE TX lane2 positive terminal	O	VP_PCIE
PCIE_TX3N	PCIE TX lane3 negative terminal	O	VP_PCIE
PCIE_TX3P	PCIE TX lane3 positive terminal	O	VP_PCIE
PCIE_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_PCIE
PCIE_CLKREQ_N	PCIE clock requirement	I/O	VDDIO18
PCIE_PERST_N	PCIE Reset signal	O	VDDIO18
PCIE_WAKE_N	PCIE Link Reactivation	I/O	VDDIO18
HDMI			
HDMI_TX0N	HDMI data lane0 negative terminal	O	VP_HDMI

Signal Name	Description	Type	Power Supply
HDMI_TX0P	HDMI data lane0 positive terminal	O	VP_HDMI
HDMI_TX1N	HDMI data lane1 negative terminal	O	VP_HDMI
HDMI_TX1P	HDMI data lane1 positive terminal	O	VP_HDMI
HDMI_TX2N	HDMI data lane2 negative terminal	O	VP_HDMI
HDMI_TX2P	HDMI data lane2 positive terminal	O	VP_HDMI
HDMI_TXCKN	HDMI reference clock lane negative terminal	O	VP_HDMI
HDMI_TXCKP	HDMI reference clock lane positive terminal	O	VP_HDMI
HDMI_HPD	HDMI Hot Plug Detect	I	VPH_HDMI
HDMI_REXT	Calibration reference, 1.62K 1% resistor connector to ground	Reference	VPH_HDMI
HDMI_CEC	Consumer Electronics Control, internal pull-up	I/O	VDDIO18
HDMI_SCL	HDMI SCL (DDC lock)	O	VDDIO18
HDMI_SDA	HDMI SDA (DDC data)	I/O	VDDIO18
MIPI DSI			
MIPI_DSI_CKN	MIPI DSI clock lane negative terminal	O	VPH_DSI
MIPI_DSI_CKP	MIPI DSI clock lane positive terminal	O	VPH_DSI
MIPI_DSI_D0N	MIPI DSI data lane0 negative terminal	O	VPH_DSI
MIPI_DSI_D0P	MIPI DSI data lane0 positive terminal	O	VPH_DSI
MIPI_DSI_D1N	MIPI DSI data lane1 negative terminal	O	VPH_DSI
MIPI_DSI_D1P	MIPI DSI data lane1 positive terminal	O	VPH_DSI
MIPI_DSI_D2N	MIPI DSI data lane2 negative terminal	O	VPH_DSI
MIPI_DSI_D2P	MIPI DSI data lane2 positive terminal	O	VPH_DSI
MIPI_DSI_D3N	MIPI DSI data lane3 negative terminal	O	VPH_DSI
MIPI_DSI_D3P	MIPI DSI data lane3 positive terminal	O	VPH_DSI
MIPI_DSI_ATB	MIPI DSI test Output	O	VPH_DSI
MIPI_DSI_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_DSI
MIPI CSI D-PHY			
MIPI_CSI0_CKN	MIPI CSI0 DPHY clock lane negative terminal	I	VPH_CSI01
MIPI_CSI0_CKP	MIPI CSI0 DPHY clock lane positive terminal	I	VPH_CSI01
MIPI_CSI0_D0N	MIPI CSI0 DPHY data lane0 negative terminal	I	VPH_CSI01
MIPI_CSI0_D0P	MIPI CSI0 DPHY data lane0 positive terminal	I	VPH_CSI01
MIPI_CSI0_D1N	MIPI CSI0 DPHY data lane1 negative terminal	I	VPH_CSI01
MIPI_CSI0_D1P	MIPI CSI0 DPHY data lane1 positive terminal	I	VPH_CSI01
MIPI_CSI0_ATB	MIPI CSI0 test Output	O	VPH_CSI01
MIPI_CSI0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI0_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI0_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI0_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI1_CKN	MIPI CSI1 DPHY clock lane negative terminal	I	VPH_CSI01
MIPI_CSI1_CKP	MIPI CSI1 DPHY clock lane positive terminal	I	VPH_CSI01
MIPI_CSI1_D0N	MIPI CSI1 DPHY data lane0 negative terminal	I	VPH_CSI01
MIPI_CSI1_D0P	MIPI CSI1 DPHY data lane0 positive terminal	I	VPH_CSI01
MIPI_CSI1_D1N	MIPI CSI1 DPHY data lane1 negative terminal	I	VPH_CSI01
MIPI_CSI1_D1P	MIPI CSI1 DPHY data lane1 positive terminal	I	VPH_CSI01
MIPI_CSI1_ATB	MIPI CSI1 test Output	O	VPH_CSI01
MIPI_CSI1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI1_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI1_XHS	Horizontal sync signal	O	VDDIO18

Signal Name	Description	Type	Power Supply
MIPI_CSI1_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI2_CKN	MIPI CSI2 DPHY clock lane negative terminal	I	VPH_CSI23
MIPI_CSI2_CKP	MIPI CSI2 DPHY clock lane positive terminal	I	VPH_CSI23
MIPI_CSI2_D0N	MIPI CSI2 DPHY data lane0 negative terminal	I	VPH_CSI23
MIPI_CSI2_D0P	MIPI CSI2 DPHY data lane0 positive terminal	I	VPH_CSI23
MIPI_CSI2_D1N	MIPI CSI2 DPHY data lane1 negative terminal	I	VPH_CSI23
MIPI_CSI2_D1P	MIPI CSI2 DPHY data lane1 positive terminal	I	VPH_CSI23
MIPI_CSI2_ATB	MIPI CSI2 test Output	O	VPH_CSI23
MIPI_CSI2_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI2_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI2_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI2_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI3_CKN	MIPI CSI3 DPHY clock lane negative terminal	I	VPH_CSI23
MIPI_CSI3_CKP	MIPI CSI3 DPHY clock lane positive terminal	I	VPH_CSI23
MIPI_CSI3_D0N	MIPI CSI3 DPHY data lane0 negative terminal	I	VPH_CSI23
MIPI_CSI3_D0P	MIPI CSI3 DPHY data lane0 positive terminal	I	VPH_CSI23
MIPI_CSI3_D1N	MIPI CSI3 DPHY data lane1 negative terminal	I	VPH_CSI23
MIPI_CSI3_D1P	MIPI CSI3 DPHY data lane1 positive terminal	I	VPH_CSI23
MIPI_CSI3_ATB	MIPI CSI3 test Output	O	VPH_CSI23
MIPI_CSI3_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI3_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI3_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI3_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI4_CKN	MIPI CSI4 DPHY clock lane negative terminal	I	VPH_CSI45
MIPI_CSI4_CKP	MIPI CSI4 DPHY clock lane positive terminal	I	VPH_CSI45
MIPI_CSI4_D0N	MIPI CSI4 DPHY data lane0 negative terminal	I	VPH_CSI45
MIPI_CSI4_D0P	MIPI CSI4 DPHY data lane0 positive terminal	I	VPH_CSI45
MIPI_CSI4_D1N	MIPI CSI4 DPHY data lane1 negative terminal	I	VPH_CSI45
MIPI_CSI4_D1P	MIPI CSI4 DPHY data lane1 positive terminal	I	VPH_CSI45
MIPI_CSI4_ATB	MIPI CSI4 test Output	O	VPH_CSI45
MIPI_CSI4_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI4_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI4_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI4_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI5_CKN	MIPI CSI5 DPHY clock lane negative terminal	I	VPH_CSI45
MIPI_CSI5_CKP	MIPI CSI5 DPHY clock lane positive terminal	I	VPH_CSI45
MIPI_CSI5_D0N	MIPI CSI5 DPHY data lane0 negative terminal	I	VPH_CSI45
MIPI_CSI5_D0P	MIPI CSI5 DPHY data lane0 positive terminal	I	VPH_CSI45
MIPI_CSI5_D1N	MIPI CSI5 DPHY data lane1 negative terminal	I	VPH_CSI45
MIPI_CSI5_D1P	MIPI CSI5 DPHY data lane1 positive terminal	I	VPH_CSI45
MIPI_CSI5_ATB	MIPI CSI5 test Output	O	VPH_CSI45
MIPI_CSI5_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI5_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI5_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI5_XVS	Vertical sync signal	O	VDDIO18
MIPI CSI C-PHY			
MIPI_CSI0_A0	MIPI CSI0 CPHY trio 0 wire A	I	VPH_CSI01

Signal Name	Description	Type	Power Supply
MIPI_CSI0_B0	MIPI CSI0 CPHY trio 0 wire B	I	VPH_CSI01
MIPI_CSI0_C0	MIPI CSI0 CPHY trio 0 wire C	I	VPH_CSI01
MIPI_CSI0_A1	MIPI CSI0 CPHY trio 1 wire A	I	VPH_CSI01
MIPI_CSI0_B1	MIPI CSI0 CPHY trio 1 wire B	I	VPH_CSI01
MIPI_CSI0_C1	MIPI CSI0 CPHY trio 1 wire C	I	VPH_CSI01
MIPI_CSI0_ATB	MIPI CSI0 test Output	O	VPH_CSI01
MIPI_CSI0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI0_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI0_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI0_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI1_A0	MIPI CSI1 CPHY trio 0 wire A	I	VPH_CSI01
MIPI_CSI1_B0	MIPI CSI1 CPHY trio 0 wire B	I	VPH_CSI01
MIPI_CSI1_C0	MIPI CSI1 CPHY trio 0 wire C	I	VPH_CSI01
MIPI_CSI1_A1	MIPI CSI1 CPHY trio 1 wire A	I	VPH_CSI01
MIPI_CSI1_B1	MIPI CSI1 CPHY trio 1 wire B	I	VPH_CSI01
MIPI_CSI1_C1	MIPI CSI1 CPHY trio 1 wire C	I	VPH_CSI01
MIPI_CSI1_ATB	MIPI CSI1 test Output	O	VPH_CSI01
MIPI_CSI1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI1_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI1_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI1_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI2_A0	MIPI CSI2 CPHY trio 0 wire A	I	VPH_CSI23
MIPI_CSI2_B0	MIPI CSI2 CPHY trio 0 wire B	I	VPH_CSI23
MIPI_CSI2_C0	MIPI CSI2 CPHY trio 0 wire C	I	VPH_CSI23
MIPI_CSI2_A1	MIPI CSI2 CPHY trio 1 wire A	I	VPH_CSI23
MIPI_CSI2_B1	MIPI CSI2 CPHY trio 1 wire B	I	VPH_CSI23
MIPI_CSI2_C1	MIPI CSI2 CPHY trio 1 wire C	I	VPH_CSI23
MIPI_CSI2_ATB	MIPI CSI2 test Output	O	VPH_CSI23
MIPI_CSI2_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI2_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI2_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI2_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI3_A0	MIPI CSI3 CPHY trio 0 wire A	I	VPH_CSI23
MIPI_CSI3_B0	MIPI CSI3 CPHY trio 0 wire B	I	VPH_CSI23
MIPI_CSI3_C0	MIPI CSI3 CPHY trio 0 wire C	I	VPH_CSI23
MIPI_CSI3_A1	MIPI CSI3 CPHY trio 1 wire A	I	VPH_CSI23
MIPI_CSI3_B1	MIPI CSI3 CPHY trio 1 wire B	I	VPH_CSI23
MIPI_CSI3_C1	MIPI CSI3 CPHY trio 1 wire C	I	VPH_CSI23
MIPI_CSI3_ATB	MIPI CSI3 test Output	O	VPH_CSI23
MIPI_CSI3_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI3_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI3_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI3_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI4_A0	MIPI CSI4 CPHY trio 0 wire A	I	VPH_CSI45
MIPI_CSI4_B0	MIPI CSI4 CPHY trio 0 wire B	I	VPH_CSI45
MIPI_CSI4_C0	MIPI CSI4 CPHY trio 0 wire C	I	VPH_CSI45
MIPI_CSI4_A1	MIPI CSI4 CPHY trio 1 wire A	I	VPH_CSI45

Signal Name	Description	Type	Power Supply
MIPI_CSI4_B1	MIPI CSI4 CPHY trio 1 wire B	I	VPH_CSI45
MIPI_CSI4_C1	MIPI CSI4 CPHY trio 1 wire C	I	VPH_CSI45
MIPI_CSI4_ATB	MIPI CSI4 test Output	O	VPH_CSI45
MIPI_CSI4_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI4_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI4_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI4_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI5_A0	MIPI CSI5 CPHY trio 0 wire A	I	VPH_CSI45
MIPI_CSI5_B0	MIPI CSI5 CPHY trio 0 wire B	I	VPH_CSI45
MIPI_CSI5_C0	MIPI CSI5 CPHY trio 0 wire C	I	VPH_CSI45
MIPI_CSI5_A1	MIPI CSI5 CPHY trio 1 wire A	I	VPH_CSI45
MIPI_CSI5_B1	MIPI CSI5 CPHY trio 1 wire B	I	VPH_CSI45
MIPI_CSI5_C1	MIPI CSI5 CPHY trio 1 wire C	I	VPH_CSI45
MIPI_CSI5_ATB	MIPI CSI5 test Output	O	VPH_CSI45
MIPI_CSI5_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI5_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI5_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI5_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI_XTRIG0	Sensor slave mode, XVS/XHS trigger mode I 0	I	VDDIO18
MIPI_CSI_XTRIG1	Sensor slave mode, XVS/XHS trigger mode I 1	I	VDDIO18
DVP			
DVP0_D[13:0]	DVP0 data	I	VPH_CSI
DVP0_PCLK	DVP0 Pixel CLK	O	VPH_CSI
DVP0_MCLK	DVP0 Sensor master clock Output	O	VPH_CSI
DVP0_XHS	DVP0 Horizontal sync signal	O	VPH_CSI
DVP0_XVS	DVP0 Vertical sync signal	O	VPH_CSI
DVP1_D[13:0]	DVP1 data	I	VPH_CSI
DVP1_PCLK	DVP1 Pixel CLK	O	VPH_CSI
DVP1_MCLK	DVP1 Sensor master clock Output	O	VPH_CSI
DVP1_XHS	DVP1 Horizontal sync signal	O	VPH_CSI
DVP1_XVS	DVP1 Vertical sync signal	O	VPH_CSI
GRMII			
RGMII0_CLK_125	External 125MHz Reference Clock I, this pin should be tie low if not used	I	VDDIO33_RGMII
RGMII0_INTB	Interrupt Input	I	VDDIO33_RGMII
RGMII0_MDC	Management Data Clock	O	VDDIO33_RGMII
RGMII0_MDIO	Management Data	I/O	VDDIO33_RGMII
RGMII0_RXCLK	Receive reference clock.It will be 125MHz, 25MHz, or 2.5MHz	I	VDDIO33_RGMII
RGMII0_RXD[4:0]	Receive Data. Data is transmitted from PHY to MAC via RXD[3:0]	I	VDDIO33_RGMII
RGMII0_RXDV	Receive Control Signal from PHY	I	VDDIO33_RGMII
RGMII0_TXCLK	Transmit reference clock.It will be 125MHz, 25MHz, or 2.5MHz	O	VDDIO33_RGMII
RGMII0_TXD[4:0]	Transmit Data	O	VDDIO33_RGMII
RGMII0_TXEN	Transmit Control Signal to PHY	O	VDDIO33_RGMII
RGMII1_CLK_125	External 125MHz Reference Clock I, this pin should be tie low if not used	I	VDDIO33_RGMII
RGMII1_INTB	Interrupt Input	I	VDDIO33_RGMII
RGMII1_MDC	Management Data Clock	O	VDDIO33_RGMII

Signal Name	Description	Type	Power Supply
RGMII1_MDIO	Management Data	I/O	VDDIO33_RGMII
RGMII1_RXCLK	Receive reference clock.It will be 125MHz, 25MHz, or 2.5MHz	I	VDDIO33_RGMII
RGMII1_RXD[4:0]	Receive Data. Data is transmitted from PHY to MAC via RXD[3:0]	I	VDDIO33_RGMII
RGMII1_RXDV	Receive Control Signal from PHY	I	VDDIO33_RGMII
RGMII1_TXCLK	Transmit reference clock.It will be 125MHz, 25MHz, or 2.5MHz	O	VDDIO33_RGMII
RGMII1_TXD[4:0]	Transmit Data	O	VDDIO33_RGMII
RGMII1_TXEN	Transmit Control Signal to PHY	O	VDDIO33_RGMII
System			
CHIP_MODE	Chip mode selection I,internal pull-down 0: Function mode 1: Test mode	I	VDDIO18
BOOT_SEL0	Boot selection	I	VDDIO18
BOOT_SEL1		I	VDDIO18
BOOT_SEL2		I	VDDIO18
BOOT_SEL3		I	VDDIO18
S_MODE	single die or dual die package selection 0: Dual die 1: Single die	I	VDDIO18
POR_SEL	Internal or external POR selection 0: Function mode 1: DFT mode	I	VDDIO18
POR_TIME_SEL0	Internal POR keep low voltage time delay selction 0	I	VDDIO18
POR_TIME_SEL1	Internal POR keep low voltage time delay selction 1		VDDIO18
LPDDR_REF_CLK	external clock I, PAD type is RGMII PAD, support 1.8V and 3.3V dual mode， default config is 1.8V mode, up to 200MHz.	TBD	AVDDHV_PLL0
DDR_REFCLK_SEL	DDR reference clock selection, internal PLL or external clock source. 0: External buck up reference clock 1: Internal pll clock	I I/O	VDDIO18
KEY_RESET_N	Reset I,active low,internal pull-up	I	VDDIO18
RST_OUT_N	Reset O,active low,internal pull-up	O	VDDIO18
XIN_24M	24MHz clk Input	I	VDDIO18
XOUT_24M	24MHz clk Output	O	VDDIO18
JTAG			
JTAG0_TCK	JATG0 Clock	I	VDDIO18
JTAG0_TDI	JATG0 Data In	I	VDDIO18
JTAG0_TDO	JATG0 Data Out	O	VDDIO18
JTAG0_TMS	JTAG0 Mode Select	I	VDDIO18
JTAG1_TCK	JATG1 Clock	I	VDDIO18
JTAG1_TDI	JATG1 Data In	I	VDDIO18
JTAG1_TDO	JATG1 Data Out	O	VDDIO18
JTAG1_TMS	JTAG1 Mode Select	I	VDDIO18
JTAG2_TCK	JATG2 Clock	I	VDDIO18
JTAG2_TDI	JATG2 Data In	I	VDDIO18
JTAG2_TDO	JATG2 Data Out	O	VDDIO18
JTAG2_TMS	JTAG2 Mode Select	I	VDDIO18
JTAG2_TRST	JTAG2 Reset	I	VDDIO18
SPI			

Signal Name	Description	Type	Power Supply
SPI0_CLK	SPI0 Serial Clock	O	VDDIO18
SPI0_D0	SPI0 data0	I/O	VDDIO18
SPI0_D1	SPI0 data1	I/O	VDDIO18
SPI0_D2	SPI0 data2	I/O	VDDIO18
SPI0_D3	SPI0 data3	I/O	VDDIO18
SPI0_CS_N	SPI0 Chip Select	O	VDDIO18
SPI1_CLK	SPI1 Serial Clock	O	VDDIO18
SPI1_D0	SPI1 data0	I/O	VDDIO18
SPI1_D1	SPI1 data1	I/O	VDDIO18
SPI1_D2	SPI1 data2	I/O	VDDIO18
SPI1_D3	SPI1 data3	I/O	VDDIO18
SPI1_CS0_N	SPI1 Chip Select 0	O	VDDIO18
SPI1_CS1_N	SPI1 Chip Select 1	O	VDDIO18
SPI2_CLK	SPI2 Serial Clock	O	VDDIO18
SPI2_D0	SPI2 data0	I/O	VDDIO18
SPI2_D1	SPI2 data1	I/O	VDDIO18
SPI2_D2	SPI2 data2	I/O	VDDIO18
SPI2_D3	SPI2 data3	I/O	VDDIO18
SPI2_CS0_N	SPI2 Chip Select 0	O	VDDIO18
SPI2_CS1_N	SPI2 Chip Select 1	O	VDDIO18
SPI3_CLK	SPI3 Serial Clock	I	VDDIO18
SPI3_DI	SPI3 data in	I	VDDIO18
SPI3_DO	SPI3 data out	O	VDDIO18
SPI3_CS_N	SPI3 Chip Select	I	VDDIO18
PWM			
FAN_TACH	Fan Speed Detection	I	VDDIO18
FAN_PWM	Fan Speed Control	O	VDDIO18
PWM1	PWM Output0	O	VDDIO18
PWM2	PWM Outpur1	O	VDDIO18
GPIO			
GPIO[31:0]	GPIO, With Interrupt	I/O	VDDIO18
GPIO[111:32]	GPIO, Without Interrupt	I/O	VDDIO18
LED			
SD0_LED_CONTROL	SD0 Data transmission indicator light	I/O	VDDIO18
SD1_LED_CONTROL	SD1 Data transmission indicator light	I/O	VDDIO18
EMMC_LED_CONTROL	EMMC Data transmission indicator light	I/O	VDDIO18
SATA_ACT_LED	SATA Data transmission indicator light	I/O	VDDIO18
I2S			
I2S_MCLK	I2S Master Clock, It is shared for I2S0, I2S1, I2S2	O	VDDIO18
I2S0_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S0_SDI	Audio Data Input	I	VDDIO18
I2S0_SDO	Audio Data Output	O	VDDIO18
I2S0_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels，0:left channel，1:right channel	I/O	VDDIO18
I2S1_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S1_SDI	Audio Data Input	I	VDDIO18
I2S1_SDO	Audio Data Output	O	VDDIO18
I2S1_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels，0:left channel，1:right channel	I/O	VDDIO18

Signal Name	Description	Type	Power Supply
I2S2_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S2_SDI	Audio Data Input	I	VDDIO18
I2S2_SDO	Audio Data Output	O	VDDIO18
I2S2_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels， 0:left channel， 1:right channel	I/O	VDDIO18
I2C			
I2C0_SDA	I2C0 serial data/address。	B	VDDIO18
I2C0_SCL	I2C0 serial clock。	B	VDDIO18
I2C1_SDA	I2C1 serial data/address。	B	VDDIO18
I2C1_SCL	I2C1 serial clock。	B	VDDIO18
I2C2_SDA	I2C2 serial data/address。	B	VDDIO18
I2C2_SCL	I2C2 serial clock。	B	VDDIO18
I2C3_SDA	I2C3 serial data/address。	B	VDDIO18
I2C3_SCL	I2C3 serial clock。	B	VDDIO18
I2C4_SDA	I2C4 serial data/address。	B	VDDIO18
I2C4_SCL	I2C4 serial clock。	B	VDDIO18
I2C5_SDA	I2C5 serial data/address。	B	VDDIO18
I2C5_SCL	I2C5 serial clock。	B	VDDIO18
I2C6_SDA	I2C6 serial data/address。	B	VDDIO18
I2C6_SCL	I2C6 serial clock。	B	VDDIO18
I2C7_SDA	I2C7 serial data/address。	B	VDDIO18
I2C7_SCL	I2C7 serial clock。	B	VDDIO18
I2C8_SDA	I2C8 serial data/address。	B	VDDIO18
I2C8_SCL	I2C8 serial clock。	B	VDDIO18
I2C9_SDA	I2C9 serial data/address。	B	VDDIO18
I2C9_SCL	I2C9 serial clock。	B	VDDIO18
I2C10_SDA	I2C10 serial data/address。	B	VDDIO18
I2C10_SCL	I2C10 serial clock。	B	VDDIO18
I2C11_SDA	I2C11 serial data/address。	B	VDDIO18
I2C11_SCL	I2C11 serial clock。	B	VDDIO18
UART			
UART0_RXD	UART0 receiver.	I	VDDIO18
UART0_TXD	UART0 transmitter.	O	VDDIO18
UART1_RXD	UART1 receiver.	I	VDDIO18
UART1_TXD	UART1 transmitter.	O	VDDIO18
UART1_RTSN	UART1 Request To Send.	O	VDDIO18
UART1_CTSN	UART1 Clear To Send.	I	VDDIO18
UART2_RXD	UART2 receiver.	I	VDDIO18
UART2_TXD	UART2 transmitter.	O	VDDIO18
UART3_RXD	UART3 receiver.	I	VDDIO18
UART3_TXD	UART3 transmitter.	O	VDDIO18
UART4_RXD	UART4 receiver.	I	VDDIO18
UART4_TXD	UART4 transmitter.	O	VDDIO18

2.4 Pin Multiplexing Table

Multiple usage pins are used to conserve pin consumption for different features. The EIC7700X devices can be used in many different applications but each application will not utilize all the on chip features. As a result, some of the features share the same pin. Most of the multiple usage pins can be used as a GPIO

pin as well.

Table 2-4 GPIOZ_x Multi-Function Pin

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
D11	BOOT_SEL0	BOOT_SEL0		GPIO107	
E11	BOOT_SEL1	BOOT_SEL1		GPIO108	
F11	BOOT_SEL2	BOOT_SEL2		GPIO109	
F12	BOOT_SEL3	BOOT_SEL3		GPIO110	
G11	CHIP_MODE	CHIP_MODE			
AN3	POR_SEL	POR_SEL			
G15	S_MODE	S_MODE		GPIO94	
AM2	GPIO29	POR_TIME_SEL0	EMMC_LED_CONTROL	GPIO29	
AL3	GPIO34	POR_TIME_SEL1	SD0_LED_CONTROL	GPIO34	
AK2	GPIO95	LPDDR_REFCLK_SEL		GPIO95	
AD2	FAN_PWM	FAN_PWM		GPIO68	
AD1	FAN_TACH	FAN_TACH		GPIO69	
AP3	KEY_RESET_N	KEY_RESET_N			
AN2	RST_OUT_N	RST_OUT_N			
H27	LPDDR_REF_CLK	LPDDR_REF_CLK			
AM4	GPIO0	GPIO0			
AL2	GPIO106	GPIO106			
AH3	GPIO11	GPIO11			
AL4	GPIO111	GPIO111			
AL1	GPIO27	GPIO27	SATA_ACT_LED		
AM1	GPIO28	GPIO28			
AG1	GPIO5	GPIO5	SPI2_D3		
AK4	GPIO92	I2C8_SCL	MIPI_CSI_XTRIG0	GPIO92	UART3_TX
AK1	GPIO93	I2C8_SDA	MIPI_CSI_XTRIG1	GPIO93	UART3_RX
AJ21	HDMI_CEC	HDMI_CEC			
AL21	HDMI_SCL	HDMI_SCL			
AK21	HDMI_SDA	HDMI_SDA			
AE5	I2C0_SCL	I2C0_SCL		GPIO44	
AF5	I2C0_SDA	I2C0_SDA		GPIO45	
AE3	I2C1_SCL	I2C1_SCL		GPIO46	
AE4	I2C1_SDA	I2C1_SDA		GPIO47	
AE1	I2C10_SCL	I2C10_SCL		GPIO102	

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
AE2	I2C10_SDA	I2C10_SDA		GPIO103	
AF4	I2C11_SCL	I2C11_SCL		GPIO104	
AF3	I2C11_SDA	I2C11_SDA		GPIO105	
AK8	I2C2_SCL	I2C2_SCL		GPIO48	
AL8	I2C2_SDA	I2C2_SDA		GPIO49	
AM8	I2C3_SCL	I2C3_SCL		GPIO50	
AL9	I2C3_SDA	I2C3_SDA		GPIO51	
AN8	I2C4_SCL	I2C4_SCL		GPIO52	
AP8	I2C4_SDA	I2C4_SDA		GPIO53	
AM9	I2C5_SCL	I2C5_SCL		GPIO54	
AL10	I2C5_SDA	I2C5_SDA		GPIO55	
AL5	I2S_MCLK	I2S_MCLK		GPIO22	
AN6	I2S0_BCLK	I2S0_BCLK		GPIO18	
AP7	I2S0_SDI	I2S0_SDI		GPIO20	
AP6	I2S0_SDO	I2S0_SDO		GPIO21	
AN7	I2S0_WCLK	I2S0_WCLK		GPIO19	
AP4	I2S1_BCLK	I2S1_BCLK		GPIO30	
AP5	I2S1_SDI	I2S1_SDI		GPIO32	
AN5	I2S1_SDO	I2S1_SDO		GPIO33	
AN4	I2S1_WCLK	I2S1_WCLK		GPIO31	
AK5	I2S2_BCLK	I2S2_BCLK		GPIO23	
AM5	I2S2_SDI	I2S2_SDI		GPIO25	
AM6	I2S2_SDO	I2S2_SDO		GPIO26	
AL6	I2S2_WCLK	I2S2_WCLK		GPIO24	
AF1	JTAG0_TCK	JTAG0_TCK	SPI2_CLK	GPIO1	
AF2	JTAG0_TDI	JTAG0_TDI	SPI2_D1	GPIO3	
AG2	JTAG0_TDO	JTAG0_TDO	SPI2_D2	GPIO4	
AH1	JTAG0_TMS	JTAG0_TMS	SPI2_D0	GPIO2	
AG4	JTAG1_TCK	JTAG1_TCK		GPIO7	
AG5	JTAG1_TDI	JTAG1_TDI		GPIO9	
AH5	JTAG1_TDO	JTAG1_TDO		GPIO10	
AH4	JTAG1_TMS	JTAG1_TMS		GPIO8	
AJ2	JTAG2_TCK	JTAG2_TCK		GPIO64	
AJ3	JTAG2_TDI	JTAG2_TDI		GPIO66	
AJ1	JTAG2_TDO	JTAG2_TDO		GPIO67	

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
AJ4	JTAG2_TMS	JTAG2_TMS		GPIO65	
AH2	JTAG2_TRST	JTAG2_TRST		GPIO17	
AJ12	MIPI_CSI0_MCLK	MIPI_CSI0_MCLK		GPIO72	
AJ11	MIPI_CSI0_XHS	MIPI_CSI0_XHS		GPIO71	
AK10	MIPI_CSI0_XVS	MIPI_CSI0_XVS		GPIO70	
AM11	MIPI_CSI1_MCLK	MIPI_CSI1_MCLK		GPIO75	
AL11	MIPI_CSI1_XHS	MIPI_CSI1_XHS		GPIO74	
AK11	MIPI_CSI1_XVS	MIPI_CSI1_XVS		GPIO73	
AM12	MIPI_CSI2_MCLK	MIPI_CSI2_MCLK		GPIO78	
AP12	MIPI_CSI2_XHS	MIPI_CSI2_XHS		GPIO77	
AN12	MIPI_CSI2_XVS	MIPI_CSI2_XVS		GPIO76	
AM13	MIPI_CSI3_MCLK	MIPI_CSI3_MCLK		GPIO81	
AL13	MIPI_CSI3_XHS	MIPI_CSI3_XHS		GPIO80	
AL12	MIPI_CSI3_XVS	MIPI_CSI3_XVS		GPIO79	
AM14	MIPI_CSI4_MCLK	MIPI_CSI4_MCLK		GPIO84	
AN13	MIPI_CSI4_XHS	MIPI_CSI4_XHS		GPIO83	
AP13	MIPI_CSI4_XVS	MIPI_CSI4_XVS		GPIO82	
AL14	MIPI_CSI5_MCLK	MIPI_CSI5_MCLK		GPIO87	
AK13	MIPI_CSI5_XHS	MIPI_CSI5_XHS		GPIO86	
AK14	MIPI_CSI5_XVS	MIPI_CSI5_XVS		GPIO85	
F14	MODE_SET0	SDIO0_DETECT		GPIO13	
G12	MODE_SET1	SDIO0_WRITE_PROT		GPIO14	
G13	MODE_SET2	SDIO1_DETECT		GPIO15	
G14	MODE_SET3	SDIO1_WRITE_PROT		GPIO16	
AH14	PCIE_CLKREQ_N	PCIE_CLKREQ_N			
AJ14	PCIE_PERST_N	PCIE_PERST_N			
AH17	PCIE_WAKE_N	PCIE_WAKE_N			
H4	RGMII0_CLK_125	RGMII0_CLK_125			
J2	RGMII0_INTB	RGMII0_INTB			
N2	RGMII0_MDC	RGMII0_MDC			
N1	RGMII0_MDIO	RGMII0_MDIO			
J5	RGMII0_RXCLK	RGMII0_RXCLK			
J3	RGMII0_RXD0	RGMII0_RXD0			
K3	RGMII0_RXD1	RGMII0_RXD1			
K4	RGMII0_RXD2	RGMII0_RXD2			

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
K5	RGMII0_RXD3	RGMII0_RXD3			
J4	RGMII0_RXDV	RGMII0_RXDV			
K1	RGMII0_TXCLK	RGMII0_TXCLK			
L1	RGMII0_TXD0	RGMII0_TXD0			
L2	RGMII0_TXD1	RGMII0_TXD1			
M1	RGMII0_TXD2	RGMII0_TXD2			
M2	RGMII0_TXD3	RGMII0_TXD3			
K2	RGMII0_TXEN	RGMII0_TXEN			
P1	RGMII1_CLK_125	RGMII1_CLK_125			
P2	RGMII1_INTB	RGMII1_INTB			
T4	RGMII1_MDC	RGMII1_MDC			
T5	RGMII1_MDIO	RGMII1_MDIO			
P3	RGMII1_RXCLK	RGMII1_RXCLK			
P5	RGMII1_RXD0	RGMII1_RXD0			
R5	RGMII1_RXD1	RGMII1_RXD1			
R4	RGMII1_RXD2	RGMII1_RXD2			
R3	RGMII1_RXD3	RGMII1_RXD3			
P4	RGMII1_RXDV	RGMII1_RXDV			
R1	RGMII1_TXCLK	RGMII1_TXCLK			
T1	RGMII1_TXD0	RGMII1_TXD0			
T2	RGMII1_TXD1	RGMII1_TXD1			
U1	RGMII1_TXD2	RGMII1_TXD2			
U2	RGMII1_TXD3	RGMII1_TXD3			
R2	RGMII1_TXEN	RGMII1_TXEN			
V1	SPI0_CLK	SPI0_CLK		GPIO97	
W1	SPI0_CS_N	SPI0_CS_N		GPIO96	
V2	SPI0_D0	SPI0_D0		GPIO98	
Y1	SPI0_D1	SPI0_D1		GPIO99	
W2	SPI0_D2	SPI0_D2		GPIO100	
Y2	SPI0_D3	SPI0_D3		GPIO101	
AA1	SPI1_CLK	SPI1_CLK		GPIO36	
AC1	SPI1_CS0_N	SPI1_CS0_N		GPIO35	
AB2	SPI1_CS1_N	SPI1_CS1_N	PWM2	GPIO41	
AA2	SPI1_D0	SPI1_D0	I2C9_SCL	GPIO37	UART4_TX
AC3	SPI1_D1	SPI1_D1	I2C9_SDA	GPIO38	UART4_RX

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
AC2	SPI1_D2	SPI1_D2	SD1_LED_CONTROL	GPIO39	
AB1	SPI1_D3	SPI1_D3	PWM1	GPIO40	
AJ5	SPI2_CS0_N	SPI2_CS0_N		GPIO6	
AJ6	SPI2_CS1_N	SPI2_CS1_N		GPIO12	
AC4	SPI3_CLK	SPI3_CLK		GPIO89	
AC5	SPI3_CS_N	SPI3_CS_N		GPIO88	
AD4	SPI3_DI	SPI3_DI		GPIO90	
AD5	SPI3_DO	SPI3_DO		GPIO91	
Y4	UART0_RX	UART0_RX		GPIO57	
Y3	UART0_TX	UART0_TX		GPIO56	
AA4	UART1_CTS	UART1_CTS	I2C6_SCL	GPIO60	
AB5	UART1_RTS	UART1_RTS	I2C6_SDA	GPIO61	
Y5	UART1_RX	UART1_RX		GPIO59	
AA5	UART1_TX	UART1_TX		GPIO58	
AB4	UART2_RX	UART2_RX	I2C7_SDA	GPIO63	
AB3	UART2_TX	UART2_TX	I2C7_SCL	GPIO62	
H6	USB0_PWREN	USB0_PWREN		GPIO42	
H5	USB1_PWREN	USB1_PWREN		GPIO43	
AP10	XIN_24M	XIN_24M			
AN10	XOUT_24M	XOUT_24M			

2.5 ESD Ratings

Table 2-5 Table 2-5 ESD Ratings

		VALUE	UNIT
$V_{(ESD)}$ Elecrogstatic discharge	Human body model(HBM) ESD stress voltage	±2000	V
	Charged device model(CDM) ESD stress voltage	±500	V

2.6 Soldering Process Recommendations

2.6.1 Requirements on Parameters of the Lead-Free Reflow Soldering Process

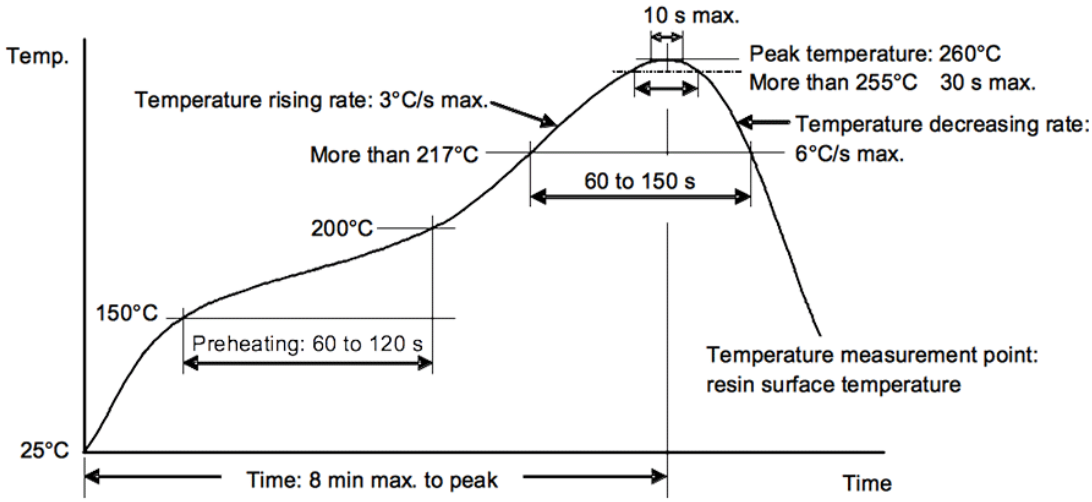


Figure 2-6 Curve of the lead-free reflow soldering process

Table 2-6 Describes parameters of the lead-free reflow soldering process.

Table 2-6 Parameters of the lead-free reflow soldering process

Zone	Duration	Heating Rate	Peak Temperature	Cooling Rate
Preheat zone (40°C–150°C or 104°F–302°F)	60s–150s	≤ 2.0°C/s (≤ 36°F/s)	-	-
Uniform temperature zone (150°C–200°C or 302°F–392°F)	60s–120s	< 1.0°C/s (< 34°F/s)	-	-
Reflow zone (greater than 217°C or 423°F)	60s–90s	-	230°C–260°C (446°F–500°F)	-
Cooling zone (Tmax to 180°C or 356°F)	-	-	-	1.0°C/s (34°F/s) ≤ Slope ≤ 4.0°C/s (39°F/s)

NOTE

- Preheat zone: The temperature range is 40°C–150°C (104°F–302°F), the heating rate is about 2°C/s (36°F/s), and the duration of this temperature zone is 60s–150s.
- Uniform temperature zone: The temperature range is 150°C–200°C (302°F–392°F), the temperature is increased steadily, the heating rate is less than 1°C/s (34°F/s), and the zone duration is 60s–120s. (Note: Slow heating is required for this zone. Otherwise, improper soldering easily occurs.)
- Reflow zone: The temperature increases from 217°C (423°F) to Tmax, and then decreases from Tmax to 217°C (423°F). The zone duration is 60s–90s.
- Cooling zone: The temperature decreases from Tmax to 180°C (356°F). The maximum cooling rate is 4°C/s (39°F/s).
- It should take no more than 6 minutes for the temperature to increase from the 25°C (77°F) ambient temperature to 250°C (482°F).
- The values on the reflow soldering curve shown in Figure 2-6 are recommended values. The values need to be adjusted on the client as required.
- Typically, the duration of the reflow zone is 60s–90s. For the boards with great heat capacity, the duration can be prolonged to 120s. For details about the requirements on package thermal resistance, see the IPC/JEDEC J-STD-020D standard. For details about the method of measuring the package temperature, see the JEP 140 standard.

Table 2-7 describes the temperature resistance standard for the lead-free package according to IPC/JEDEC 020D

Table 2-7 Temperature resistance standard for the lead-free package according to IPC/JEDEC 020D

Package Thickness	Temperature 1 (Package Volume < 350 mm^3 or 0.02 $in.^3$)	Temperature 2 (Package Volume= 350–2000 mm^3 or 0.02–0.12 $in.^3$)	Temperature 3 (Package Volume > 2000 mm^3 or 0.12 $in.^3$)
< 1.6 mm (0.06in.)	260°C (500°F)	260°C (500°F)	260°C (500°F)
1.6 mm to 2.5 mm(0.06 in. to 0.10in.)	260°C (500°F)	250°C (482°F)	245°C (473°F)
> 2.5 mm (0.10in.)	250°C (482°F)	245°C (473°F)	245°C (473°F)

The component soldering terminals (such as the solder ball and pin) and external heat sinks are not considered for volume calculation.

The method of measuring the reflow soldering process curve is as follows:

According to the JEP140 standard, to measure the package temperature, you are advised to place the temperature measuring probe of the thermocouple close to the chip surface if the chip package is thin, or to drill a hole on the package surface and place the temperature measuring probe of the thermocouple into the hole if the chip package is thick. The second method is recommended for all components because of the requirement on quantizing the component thickness. However, this method is not applicable if the chip package is too thin to drill a hole.

NOTE

To measure the temperature of the QFP-packaged chip, place the temperature measuring probe close to pins.

2.6.2 Requirements of Mixing Reflow Soldering

Lead-free components must be properly soldered during mixing reflow soldering. Table describes mixing reflow soldering parameters

Table 2-8 Mixing reflow soldering parameters

Zone		Lead BGA	Lead-free BGA	Other Components
Preheat zone (40°C–150°C or 104°F–302°F)	Duration	60s–150s		
	Heating up slope	< 2.5°C/s (37°F/s)		
Uniform temperature zone (150°C–183°C or 302°F–361°F)	Duration	30–90s		
	Heating up slope	< 1.0°C/s (34°F/s)		
Reflow zone (greater than 183°C or 361°F)	Peak temperature	210°C–240°C (410°F– 464°F)	220°C–240°C (428°F– 464°F)	210°C–245°C (410°F–473°F)
	Duration	30s–120s	60s–120s	30s–120s
Cooling zone (Tmax to 150°C or 302°F)	Cooling down slope	1.0ºC/s (34ºF/s) ≤ Slope ≤ 4.0ºC/s (39ºF/s)		

When the soldering curve is adjusted, the package thermal resistance requirements on the board components must be met. For details about the requirements on package thermal resistance, see the IPC/JEDEC J-STD-020D standard. For details about the method of measuring the package temperature, see the JEP 140 standard.

Table 2-9 describes the thermal resistance standard for the lead package according to the IPC/JEDEC 020D standard.

Table 2-9 Thermal resistance standard for the lead package

Package Thickness	Temperature 1 (Package Volume < 350 mm^3 or 0.02 $in.^3$)	Temperature 1 (Package Volume ≥ 350 mm^3 or 0.02 $in.^3$)
< 2.5 mm (0.10 in.)	235°C (455°F)	220°C (428°F)
≥ 2.5 mm (0.10 in.)	220°C (428°F)	220°C (428°F)

The component soldering terminals (such as the solder ball and pin) and external heat sinks are not considered for volume calculation.

According to the JEP 140 standard, the method of measuring the temperature of the package soldered with the mixing technology is the same as that for measuring the temperature of the package soldered with the lead-free technology. For details, see section "Requirements on Parameters of the Lead-Free Reflow Soldering Process."

2.7 Moisture-Sensitive Specifications

This chapter defines the usage principles for moisture-sensitive ICs to ensure that ICs are properly used. Related terms are explained as follows:

- Floor life: longest period during which a HiSilicon chip can be stored in the workshop below 30°C (86°F) and 60% relative humidity (RH), that is, the time from moisture barrier bag (MBB) unpacking to reflow soldering
- Desiccant: a material for absorbing moisture and keeping the product dry
- Humidity indicator card (HIC): a card that indicates the humidity status
- Moisture sensitivity level (MSL): a level for measuring the moisture degree. The MSL of this product is 3.
- MBB: a vacuum bag for protecting products against moisture
- Solder reflow: reflow soldering
- Shelf life: normal storage period of a product with the MBB

2.7.1 Moisture-Proof Packaging

2.7.1.1 Basic Information

The vacuum packaging materials include:

- An HIC
- An MBB
- Desiccant

Figure 2-7 Vacuum packaging materials

2.7.2 Incoming Inspection

When the vacuum bag is unpacked before SMT in the factories of customers or outsourcing vendors:

- If the largest indicator dot of the HIC is not in blue or khaki, rebake the product by referring to Table 2-7.
- If the 10% RH dot of the HIC is in blue or khaki, the product is dry. In this case, replace the desiccant and pack the product into a vacuum bag.
- If the 10% RH dot of the HIC is not in blue or khaki and the 5% RH dot is in red or light green, the product has become damp. In this case, rebake the product according to Table 2-7.

2.7.3 Storage and Usage

[Storage Environment]

You are advised to use vacuum packaging for the product and store it below 30°C (86°F) and 60% RH.

[Shelf Life]

Normal storage period of a product with the MBB

Below 30°C (86°F) and 60% RH, the shelf life of the product with vacuum packaging is less than or equal to 12 months.

[Floor Life]

Table 2-10 describes the floor life below 30°C (86°F) and 60% RH.

Table 2-10 Floor life

MSL	Floor life(out of bag) at factory ambient ≤ 30°C/60% RH or as stated
1	Unlimited at ≤ 30°C/85% RH
2	1 year
2a	4 weeks
3	168 hours
4	72 hours
5	48 hours
5a	24 hours
6	Mandatory bake before use, must be reflowed within the time limit specified on the label

[Usage of Moisture-Sensitive Products]

- If the product has been exposed to air for more than 2 hours at 30°C (86°F) or lower and at most 60% RH, rebake it and pack it into a vacuum bag.
- If the product has been exposed to air for no more than 2 hours at 30°C (86°F) or lower and at most 60% RH, replace the desiccant and pack the chip into a vacuum bag.

For details about other storage and usage rules, see the JEDEC J-STD-033A standard.

2.7.4 Rebaking

[Application Scope]

All moisture-sensitive ICs that need to be rebaked. Following lists the rebaking reference.

Table 2-11 Rebaking reference

Body Thickness	Level	Bake@125°C	Bake@90°C ≤ 5% RH	Bake@40°C ≤ 5% RH
≤ 1.4 mm	2a	3 hours	11 hours	5 days
	3	7 hours	23 hours	9 days
	4	7 hours	23 hours	9 days
	5	7 hours	24 hours	10 days

Body Thickness	Level	Bake@125°C	Bake@90°C ≤ 5% RH	Bake@40°C ≤ 5% RH
	5a	10 hours	24 hours	10 days
≤ 2.0 mm	2a	16 hours	2 days	22 days
	3	17 hours	2 days	23 days
	4	20 hours	3 days	28 days
	5	25 hours	4 days	35 days
	5a	40 hours	6 days	56 days
≤ 4.5 mm	2a	48 hours	7 days	67 days
	3	48 hours	8 days	67 days
	4	48 hours	10 days	67 days
	5	48 hours	10 days	67 days
	5a	48 hours	10 days	67 days

NOTE

- Table 2-11 lists the minimum rebaking time required for damp products.
- Low-temperature rebaking is recommended.
- For details, see the JEDEC standard.

2.8 Electrical Specifications

2.8.1 Temperature

Table 2-12 Operating environment parameters and requirements on the junction temperatures of EIC7700X

Table 2-12 Operating environment parameters and requirements

Chip	Junction Temperature for Working (°C)		Destructive Junction Temperature (°C)
	Min(°C)	Max(°C)	
EIC7700X	-20	105	125

2.8.2 Operating Conditions

2.8.2.1 Absolute Ratings

The table below gives the absolute maximum ratings. Exposure to stresses beyond those listed in this table may result in permanent device damage, unreliability or both.

Table 2-13 Absolute maximum rated voltage

Parameters	Related Power Group	Min	Max	Unit
VDD_CPU	CPU Core Power	-0.3	TBD	V
VDD_NPU	NPU Core Power	-0.3	TBD	V
VDD_SOC	SOC Core Power	-0.3	TBD	V
VDD_LPDDR	LPDDR Core Power	-0.3	0.88	V
VDDQ_LPDDR	LPDDR IO Power	-0.3	1.17	V
VDD2H_LPDDR	LPDDR VDD2H Power	-0.3	1.17	V
VAA_VDD2H_LPDDR0 VAA_VDD2H_LPDDR1	LPDDR PLL power（1.8V）	-0.3	1.98	V
VP_CSI01, VP_CSI23, VP_CSI45	MIPI CSI0, CSI1 Analog Power（0.8V)	-0.3	0.88	V
VP_DSI	MIPI DSI Analog Power（0.8V）	-0.3	0.88	V
VP_HDMI	HDMI Analog Power（0.8V）	-0.3	0.88	V
VP_PCIE	PCIE Analog Power（0.8V）	-0.3	0.88	V
VPTX_SATA	SATA Analog Power（0.8V）	-0.3	0.88	V
VP_SATA	SATA Analog Power（0.8V）	-0.3	0.88	V
VP_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
VPTX_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
DVDD_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
VP_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
VPTX_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
DVDD_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
VPH_CSI01, VPH_CSI23, VPH_CSI45	MIPI CSI0, CSI1 Analog Power（1.8V）	-0.3	1.98	V
VPH_DSI	MIPI DSI Analog Power（1.8V）	-0.3	1.98	V
VPH_HDMI	HDMI Analog Power（1.8V）	-0.3	1.98	V
VPH_PCIE	PCIE Analog Power（1.8V）	-0.3	1.98	V
VPH_SATA	SATA Analog Power（1.8V）	-0.3	1.98	V
VDDIO_EMMCSD	EMMC and SDIO 1.8V Power	-0.3	1.98	V
VDDIO18_RGMII	RGMII Internal logic 1.8V Power	-0.3	1.98	V
VDDIO18	1.8V IO Power	-0.3	1.98	V
AVDDHV_PLL	1.8V PLL Power	-0.3	1.98	V
VDD18_POR	Internal POR 1.8V Power	-0.3	1.98	V
VDD18_PVT_DDR	1.8V PVT Power For DDR	-0.3	1.98	V
VDD18_PVT_SOC	1.8V PVT Power For SOC	-0.3	1.98	V
VDDIO_OTP	OTP IO Voltage(1.8V)	-0.3	1.98	V
VDDIO33_RGMII	RGMII 1.8V IO Power	-0.3	3.63	V
VDDIO33_USB	USB Analog Power（3.3V）	-0.3	3.63	V
VDDIO33_EMMCSD	EMMC and SDIO 3.3V Power	-0.3	3.63	V

2.8.2.2 Recommended Operating Condition

Following table describes the recommended operating condition.

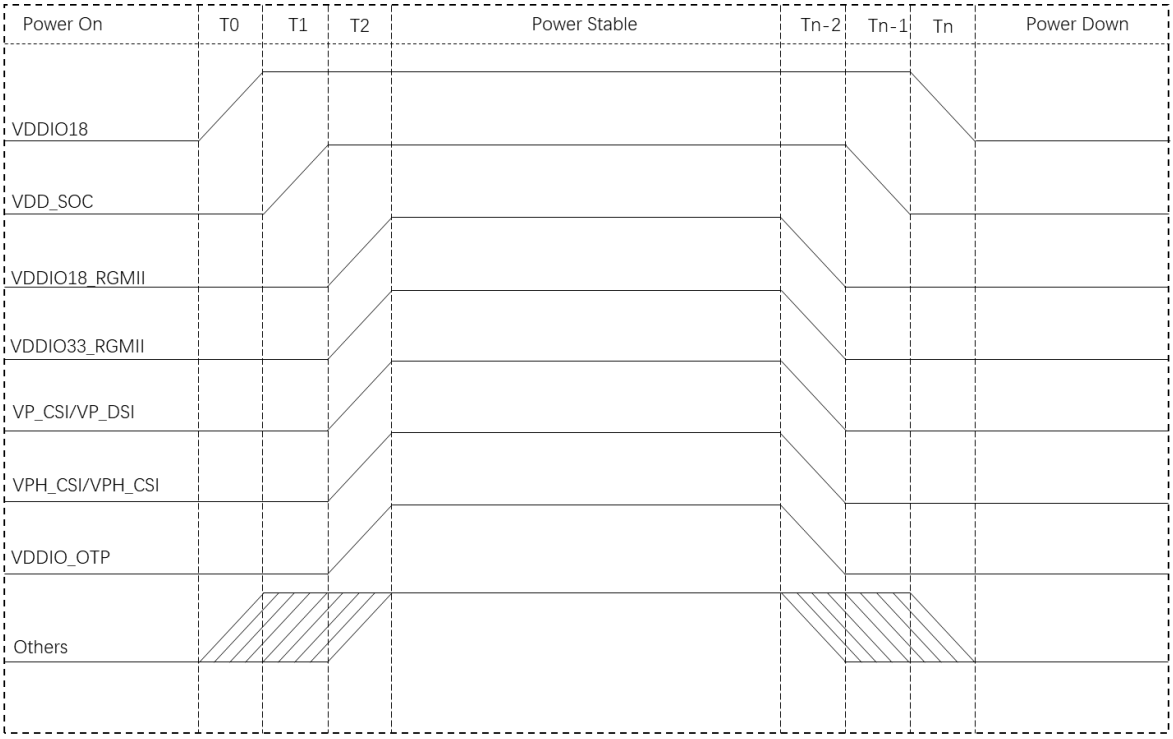
Table 2-14 Operating conditions for the power supply under normal voltage

Power Name	Description	Voltage			
		Min	Typ	Max	Unit
VDD_CPU	CPU Core Power	0.76	0.8	0.84	V
VDD_NPU	NPU Core Power	0.76	0.8	0.84	V
VDD_SOC	SOC Core Power	0.76	0.8	0.84	V
VDD_LPDDR	LPDDR Core Power	0.76	0.8	0.84	V
VDDQ_LPDDR	IO power For LPDDR4	0.76	0.8	0.84	V
	IO power For LPDDR4x	0.57	0.6	0.63	V
	IO power For LPDDR5	0.48	0.5	0.53	V
VDD2H_LPDDR	VDD2H For LPDDR4/LPDDR4x	1.05	1.1	1.16	V
	VDD2H For LPDDR5	1.00	1.05	1.10	V
VAA_VDD2H_LPDDR0 VAA_VDD2H_LPDDR1	LPDDR PLL power（1.8V）	1.71	1.8	1.89	V
VP_CSI01, VP_CSI23, VP_CSI45	MIPI CSI0, CSI1 Analog Power（0.8V)	0.76	0.8	0.84	V
VP_DSI	MIPI DSI Analog Power（0.8V）	0.76	0.8	0.84	V
VP_HDMI	HDMI Analog Power（0.8V）	0.76	0.8	0.84	V
VP_PCIE	PCIE Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_SATA	SATA Analog Power（0.8V）	0.76	0.8	0.84	V
VP_SATA	SATA Analog Power（0.8V）	0.76	0.8	0.84	V
VP_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
DVDD_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VP_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
DVDD_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPH_CSI01, VPH_CSI23, VPH_CSI45	MIPI CSI0, CSI1 Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_DSI	MIPI DSI Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_HDMI	HDMI Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_PCIE	PCIE Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_SATA	SATA Analog Power（1.8V）	1.71	1.8	1.89	V
VDDIO_EMMCSD	EMMC and SDIO 1.8V Power	1.71	1.8	1.89	V
VDDIO18_RGMII	RGMII Internal logic 1.8V Power	1.71	1.8	1.89	V
VDDIO18	1.8V IO Power	1.71	1.8	1.89	V
AVDDHV_PLL	1.8V PLL Power	1.71	1.8	1.89	V
VDD18_POR	Internal POR 1.8V Power	1.71	1.8	1.89	V
VDD18_PVT_DDR	1.8V PVT Power For DDR	1.71	1.8	1.89	V
VDD18_PVT_SOC	1.8V PVT Power For SOC	1.71	1.8	1.89	V
VDDIO_OTP	OTP IO Voltage(1.8V)	1.71	1.8	1.89	V
VDDIO33_RGMII	RGMII 1.8V IO Power	1.71	1.8	1.89	V
	RGMII 3.3V IO Power	3.135	3.3	3.465	V

Power Name	Description	Voltage			
VDDIO33_USB	USB Analog Power（3.3V）	3.135	3.3	3.465	V
VDDIO33_EMMCSD	EMMC and SDIO 3.3V Power	3.135	3.3	3.465	V

2.8.3 Power-On and Power-Off Sequences

Figure 2-8 describe power-on sequences, and the the power-off sequences



Note: The VDD18_POR must be powered on with the last group of 1.8V.

Figure 2-8 power-on sequences, and the power-off sequences

2.8.4 Normal GPIO Electrical Parameters

Table 2-16 Normal GPIO DC Electrical Specifications (VDDIO18 = 1.8 V)

Table 2-15 GPIO DC Electrical Specifications

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO18	Interface voltage	1.71	1.8	1.89	V	
V_{IH}	High-level input voltage	1.27	-	1.98	V	
V_{IL}	Low-level input voltage	–0.3	-	0.58	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	
V_{OH}	High-level output voltage	1.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.45	V	

Symbol	Description	Min	Typ	Max	Unit	Remarks
R_{SPU}	Internal Srtong pull-up resistor	2.8	3.5	4.5	kΩ	
R_{PU}	Internal pull-up resistor	18	25	38	kΩ	
R_{PD}	Internal pull-down	17	22	33	kΩ	
I_{OL}	High Level Output Current @VOH(max)	2.1	3.1	4.1	mA	DS[3:0] = '0000'
		4.6	6.7	8.9	mA	DS[3:0] = '0001'
		6.6	9.6	12.8	mA	DS[3:0] = '0010'
		8.8	12.9	17.2	.mA	DS[3:0] = '0011'
		12.2	18.0	24.0	mA	DS[3:0] = '0100'
		14.2	20.9	28	mA	DS[3:0] = '0101'
		15.7	23.2	31.1	mA	DS[3:0] = '0110'
		17.5	25.9	34.8	mA	DS[3:0] = '0111'
I_{OH}	High Level Output Current @VOH(max)	1.8	2.8	3.8	mA	DS[3:0] = '0000'
		3.9	5.9	8.1	mA	DS[3:0] = '0001'
		5.5	8.4	11.6	mA	DS[3:0] = '0010'
		7.4	11.3	15.6	mA	DS[3:0] = '0011'
		10.2	15.7	21.6	mA	DS[3:0] = '0100'
		11.9	18.3	25.1	mA	DS[3:0] = '0101'
		13.1	20.2	27.9	mA	DS[3:0] = '0110'
		14.6	22.6	31.2	mA	DS[3:0] = '0111'

2.8.5 LPDDR Electrical Parameters

Table 2-17 The different voltages of EIC7700X and LPDDR SDRAM in LPDDR4, LPDDR4X, LPDDR5 mode

Table 2-16 voltages of EIC7700X

	EIC7700X			SDRAM		
	VDDQ(V)	VDD2H(V)	VAA_VDD2H(V)	VDDQ(V)	VDD1(V)	VDD2H(V)
LPDDR4	0.8	1.1	1.8	1.1	1.8	1.1
LPDDR4X	0.6	1.1	1.8	0.6	1.8	1.1
LPDDR5	0.5	1.05	1.8	0.5	1.8	1.05

Table 2-17 DC electrical parameters for the signals which powered by VDDQ

Symbol	Parameter	Minimum	Typical	Maximum	Unit
IIZ	BP_DAT input leakage current			221.15	uA
Vref	Referenc evoltage	0.49*VDDQ	0.5*VDDQ	0.51*VDDQ	V
VIH-DC	Input high voltage for BP_DAT	Vref +0.020			V
VIL-DC	Input low voltage for BP_DAT			Vref – 0.020	V
VID-DC	Differential input voltage for abs(DQS_t – DQS_c)	0.1			V

Table 2-18 DC electrical parameters for the signals which powered by VDD2H

Symbol	Parameter	Minimum	Typical	Maximum	Unit
VIH-DC	Input high voltage	0.7*VDD2H			V
VIL-DC	Input low voltage			0.3*VDD2H	V

2.8.6 SDIO Electrical Parameters

Table 2-19 SDIO electrical parameters (3.3 V)

Symbol	Description	Min	Typ	Max	Unit
VDDIO	Power voltage	3.135	3.3	3.465	V
VOH	High-level output	0.75 x DVDD33	-	-	V
VOL	Low-level output	-	-	0.125 x VDDIO33	V
VIH	High-level input	1.68	-	-	V
VIL	Low-level input	-0.3	-	0.9	V

Table 2-20 SDIO electrical parameters (1.8 V)

Symbol	Description	Min	Typ	Max	Unit
DVDD	Power voltage	1.71	1.8	1.89	V
VOH	High-level output	VDDIO-0.45	-	-	V

VOL	Low-level output	-	-	0.45	V
VIH	High-level input	1.07	-	-	V
VIL	Low-level input	–0.3	-	0.68	V

2.8.7 RGMII Electrical Parameters

Table 2-21 RGMII electrical parameters (3.3 V)

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO33_RGMII	Interface voltage	3.135	3.3	3.3465	V	
V_{IH}	High-level input voltage	2	-	VDDIO+0.3	V	
V_{IL}	Low-level input voltage	–0.3	-	0.8	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	
V_{OH}	High-level output voltage	2.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.4	V	
R_{PU}	Internal pull-up resistor	24		34	kΩ	
R_{PD}	Internal pull-down	24		34	kΩ	
I_{OL}	High Level Output Current @VOH(max)	1.1	1.6	2.1	mA	DS[3:0] = '0000'
		2.1	3.1	4.1	mA	DS[3:0] = '0001'
		3.2	4.7	6.2	mA	DS[3:0] = '0010'
		4.2	6.2	8.2	mA	DS[3:0] = '0011'
		5.3	7.8	10.3	mA	DS[3:0] = '0100'
		6.4	9.3	12.3	mA	DS[3:0] = '0101'
		7.4	10.8	14.3	mA	DS[3:0] = '0110'
		8.4	12.2	16.2	mA	DS[3:0] = '0111'
I_{OH}	High Level Output Current @VOH(max)	2.1	3.6	5.2	mA	DS[3:0] = '0000'
		4.1	6.9	10.1	mA	DS[3:0] = '0001'
		6.4	10.7	15.7	mA	DS[3:0] = '0010'
		8.1	13.5	19.8	mA	DS[3:0] = '0011'
		10.4	17.3	25.4	mA	DS[3:0] = '0100'
		12.4	20.6	30.3	mA	DS[3:0] = '0101'
		14.4	23.9	35.1	mA	DS[3:0] = '0110'
		16.1	26.8	39.3	mA	DS[3:0] = '0111'

Table 2-22 GRMII electrical parameters (1.8 V)

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO33_RGMII	Interface voltage	1.71	1.8	1.89	V	

Symbol	Description	Min	Typ	Max	Unit	Remarks
V_{IH}	High-level input voltage	1.27	-	1.98	V	
V_{IL}	Low-level input voltage	–0.3	-	0.58	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	
V_{OH}	High-level output voltage	1.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.45	V	
R_{PU}	Internal pull-up resistor	24		34	kΩ	
R_{PD}	Internal pull-down	24		34	kΩ	

2.8.8 Oscillator Electrical Parameters

EIC7700X requires the 24MHz oscillator for generating the main clock source.

Table 2-23 Oscillator electrical parameters (1.8 V)

Symbol	Description	Min.	Typ.	Max.	Unit
Fo	Nominal Frequency		24		MHz
Δf/fo	Frequency Tolerance	-30		30	ppm
CL	Load Capacitance	7.5	12	12.5	pF
ESR	Equivalent Series Resistance			100	oHm

2.9 Interface Timing

2.9.1 Ethernet MAC Port Timings

2.9.1.1 RGMII Timings

Figure 2-9 shows the 1000 Mbit/s RX timing of the RGMII.

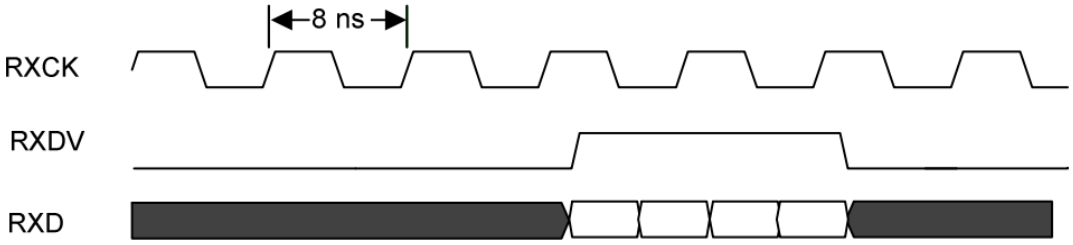


Figure 2-9 1000 Mbit/s RX timing of the RGMII

Figure 2-10 shows the 1000 Mbit/s TX timing of the RGMII.

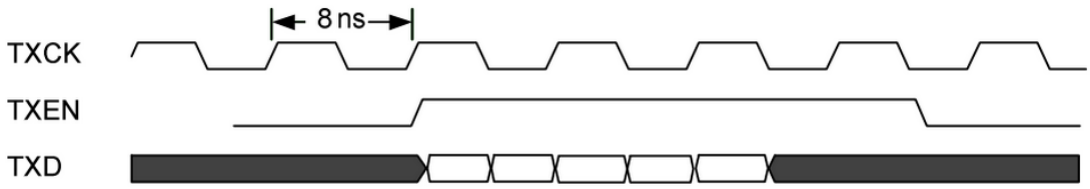


Figure 2-10 1000 Mbit/s TX timing of the RGMII

2.9.1.2 MDIO Interface Timings

Figure 2-11 shows the read timing of the MDIO interface.

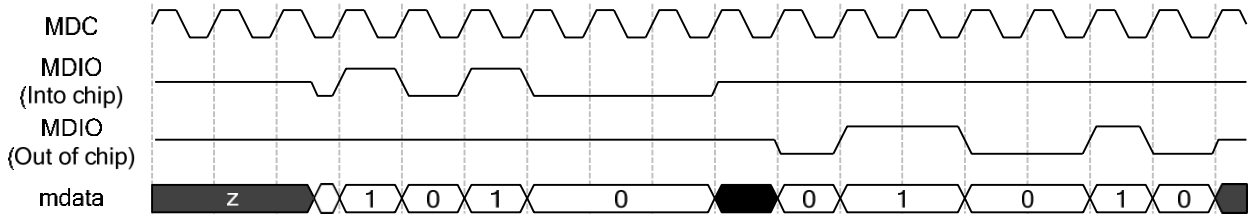


Figure 2-11 Read timing of the MDIO interface

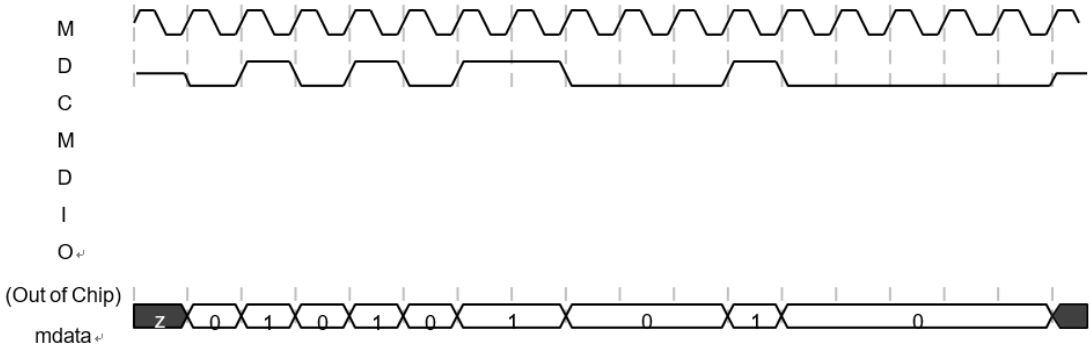


Figure 2-12 Write timing of the MDIO interface

Figure 2-13 illustrates the RX timing parameters of the MDIO interface.

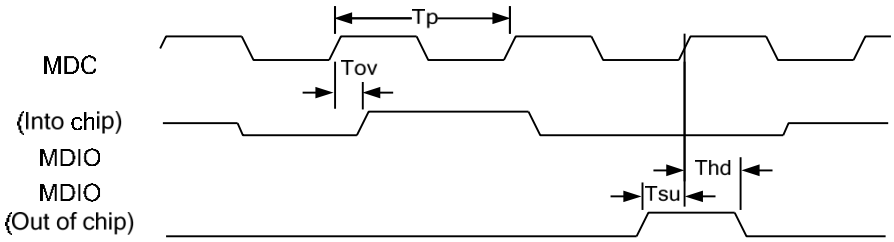


Figure 2-13 Timing parameters of the MDIO interface

Table 2-24 The timing parameters of the MDIO interface

Parameter	Symbol	Signal	Min	Max	Unit
MDIO data RX delay	Tov	MDIO	0	300	ns

MDIO clock cycle	Tp	MDCK	400	400	ns
MDIO data TX setup time	Tsu	MDIO	10	N/A	ns
MDIO data TX hold time	Thd	MDIO	10	N/A	ns

2.9.2 SPI Timings

In Figure 2-14 to Figure 2-16, the conventions are as follows:

- MSB = most significant bit
- LSB = least significant bit
- SPI_CK(0): spo = 0
- SPI_CK(1): spo = 1

Figure 2-14 shows the SPI clock (SPICK) timing.

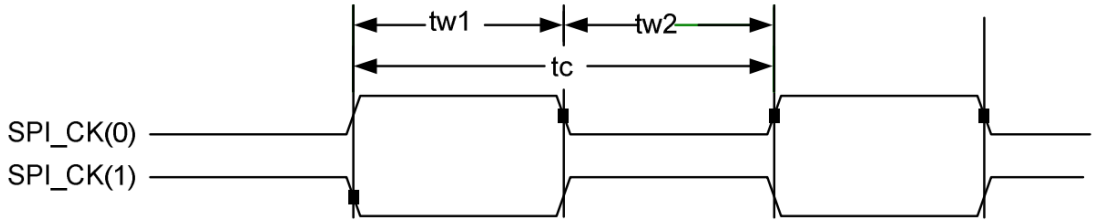


Figure 2-14 SPICK timing

Figure 2-15 and Figure 2-16 show the SPI timings in master mode.

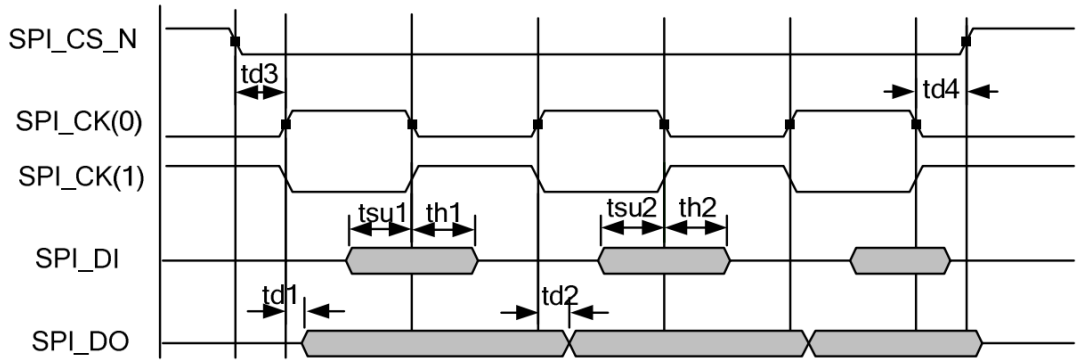


Figure 2-15 SPI timing in master mode (sph = 1)

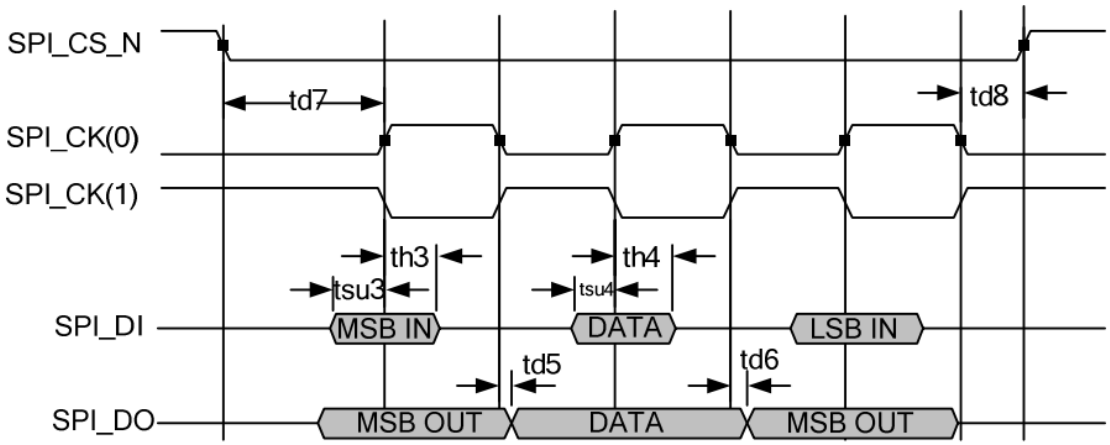


Figure 2-16 SPI timing in master mode (sph = 0)

3 System

3.1 Reset

3.1.1 Sources Of Reset

Table 3-1 description of the sources of reset

Sources of reset	description
internal por	You can use the POR_SEL pad to select internal por or external reset from KEY_RESET_N.
KEY_RESET_N	1. POR_SEL=1: select the internal por, EIC7700X has debounce logic and the debounce time is 10ms 2. POR_SEL=0: select the external por, debounce function is disabled 3. After por reset is released, EIC7700X will check whether the reset signal of KEY_RESET_N is deasert. If true, EIC7700X will continue to release the other reset after the period of debounce
watchdog	This signal is also output to the pad(RST_OUT_N). You can use this signal to reset other peripherals for system synchronization.
sw reset	This reset is generated by the software configuration 0x1AC0_FFE6. It is also output to the pad(RST_OUT_N).
debug_ndreset	For details about the debug_ndreset reset applications of MCPU, LPCPU, SCPU, and NPU, see the corresponding CPU description
lowpwr_rst_n[:]	Low-power reset for power domain. After powering on, the default value is 1, and all domains are powered on. It is output to other modules after AND with the reset from system reset from CSR.

3.1.2 Reset Requirements

A reset required for modules with bus reset, core logic reset and configuration interface reset, unless otherwise specified, is shown below:

1. When reset, the corresponding logic should be in the IDLE state, and there is no data transmission in progress, otherwise the module may not be able to return a response signal to the other party after being reset, resulting in the bus hanging.
2. It is recommended to enable the clock before releasing the reset of the module, so as to avoid the failure of some synchronous reset modules to reset.
3. For the released sequence, you can release the configuration interface first, and then release the bus interface and core logic. Make sure that all logic has been reset before starting the configuration and start-up work, so as to avoid the module being accessed by the pre-master before it is ready.

3.2 clock

Address space：0x5182_8000~0x5182_FFFF，32k.

3.2.1 Overview

A 24M clock serves as a pll reference clock input for the entire chip. There are a total of 7 PLLs, of which one is for the CPU and one for the DDR, and the rest are in the clock reset subsystem.

The following features are supported:

- Compatible with APB3.0 interface, bit width 32bit
- Contains 7 PLLs
- clock gating is Suport
- Clock division is supported
- Clock MUX is supported
- Soft reset of other subsystems is supported

3.2.2 Block diagram

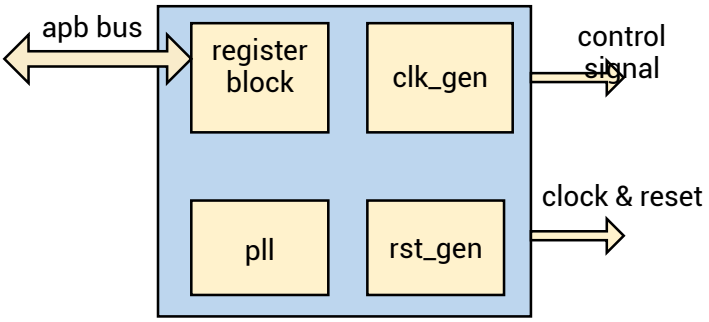


Figure 3-1 Clock and Reset block diagram

3.2.3 System clock

3.2.3.1 Clock structure

Each clock can be composed of any basic unit, and each basic unit contains at least one of PLL, MUX, DIV, and GATE

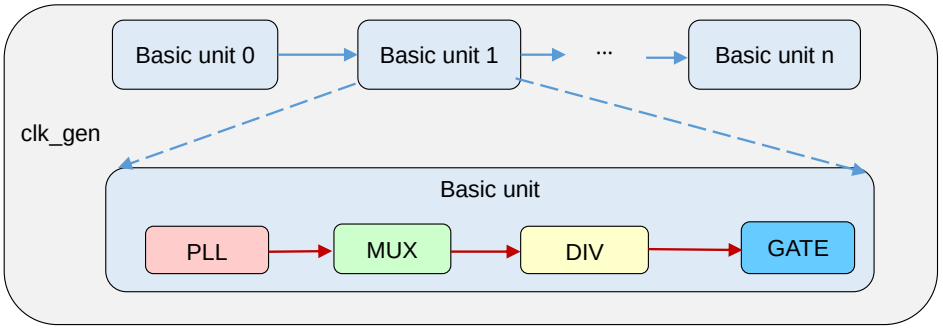


Figure 3-2 Clock structure

3.2.3.2 Clock solution

- N – NO, there is no such base unit
- Y – YES
- GM – glitch free mux
- M – static mux

"gate" Column:：
- 1: Gate is enabled by default
- 0: Gate is disenabled by default

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
source	clk_xtal_24m	24	24									
source	clk_spll0_fout1	1600	1600									
source	clk_spll0_fout2	800	800									1
source	clk_spll0_fout3	400	400									
source	clk_spll1_fout1	1500	1500									
source	clk_spll1_fout2	300	300									
source	clk_spll1_fout3	250	250									
source	clk_spll2_fout1	2080	2080									
source	clk_spll2_fout2	1040	1040									
source	clk_spll2_fout3	416	416									
source	clk_vpll_fout1	1188	1188									
source	clk_vpll_fout2	594	594									
source	clk_vpll_fout3	49.5	49.5									
source	clk_apll_fout1	983.04	983.04									
source	clk_apll_fout2	0	0									
source	clk_apll_fout3	0	0									

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
source	ext_mclk											
ddr	lpddr_ref_clk_bak	125	-									
internal	clk_1m	1	1	clk_xtal_24m			N	N	24	24	24	N
mcpu	clk_sys_cfg	200	200	clk_spll0_fout3			N	N	2	2	7	N
mipi dphy, hdmi	clk_mipi_txesc	20	20	clk_sys_cfg			N	N	10	10	10	N
noc	noc_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
noc	noc_nsp_clk	1040	1040	clk_spll2_fout1			N	N	2	2	7	1
mcpu	clk_u84_core_lp	400	400	clk_spll0_fout2			N	N	2	2	2	N
scpu	clk_scpu_core	800	800	clk_spll0_fout1			N	N	2	2	15	1
scpu	clk_scpu_bus	400	400	clk_scpu_core			N	N	2	2	2	1
lpcpu	clk_lpcpu_core	800	800	clk_spll0_fout1			N	N	2	2	15	1
lpcpu,mcpu, npu,secure	cpu_rtc_toggle	1	1	clk_1m			N	N	N	N	N	N
lpcpu	clk_lpcpu_bus	400	400	clk_lpcpu_core			N	N	2	2	2	1
3Dgpu	gpu_aclk	800	800	clk_spll0_fout1			N	N	2	2	15	0
3Dgpu	gpu_gray_clk	24	24	clk_xtal_24m			N	N	N	N	N	0
3Dgpu	gpu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_dsp_aclk_st1	1040	1040	clk_spll2_fout1	clk_spll0_fout1		GM	0	2	2	15	N
dsp	dspt_aclk	1040	1040	clk_dsp_aclk_st1			N	N	N	N	N	0
dsp	dspt_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_d2ddr_aclk	1040	1040	clk_spll2_fout1	clk_spll0_fout1		GM	0	2	2	15	N
tcu	tcu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	clk_ddr_aclk_st0	2080	1600	clk_spll2_fout1	clk_spll0_fout1		GM	1	N	N	N	N
internal	clk_ddr_aclk	1040	800	clk_ddr_aclk_st0			N	N	2	2	15	N
tcu	tcu_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
ddr	ddrt_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
ddr	ddrt0_p0_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p1_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p2_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p3_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p4_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_trace_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_trace_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p0_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p1_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p2_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p3_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p4_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
tcu	tcu_aclk_free	1040	800	clk_ddr_aclk			N	N	N	N	N	N
hsp	hsp_aclk	800	800	hsp_aclk_free			N	N	N	N	N	1
hsp	hsp_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	sata_phy_ref_clk	50	50	clk_spll1_fout2			N	N	6	6	15	N
internal	sata_phy_ref_clk_gated	125	50	sata_phy_ref_clk	lpddr_ref_clk_bak		M	0	N	N	N	1
sata	hsp_sata_phy_ref_clk	125	50	sata_phy_ref_clk_gated			N	N	N	N	N	N
usb	hsp_ref_in_clk	24	24	clk_xtal_24m			N	N	N	N	N	N
gmac	hsp_eth_app_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gmac	hsp_eth_csr_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gmac	hsp_eth0_core_clk	125	125	clk_spll1_fout3			N	N	2	2	127	N
gmac	hsp_eth1_core_clk	125	125	clk_spll1_fout3			N	N	2	2	127	N

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
internal	rmii_ref_clk_mux	125	50	rmii_ref_clk	lpddr_ref_clk_bak		M	0	6	6	6	N
gmac	hsp_rmii0_ref_clk	125	50	rmii_ref_clk_mux			N	N	N	N	N	1
gmac	hsp_rmii1_ref_clk	125	50	rmii_ref_clk_mux			N	N	N	N	N	1
emmc	hsp_mshc0_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc1_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc2_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc0_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
emmc	hsp_mshc1_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
emmc	hsp_mshc2_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
usb	hsp_usb0_suspend_clk	1	1	clk_1m			N	N	N	N	N	N
usb	hsp_usb1_suspend_clk	1	1	clk_1m			N	N	N	N	N	N
pcie	pciet_aclk	520	520	clk_spll2_fout2			N	N	2	2	15	1
pcie	pciet_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
pcie	pciet_cr_clk	100	100	clk_sys_cfg			N	N	2	2	2	0
internal	clk_npu_aclk_st1	800	800	clk_spll0_fout1			N	N	2	2	15	N
npu	npu_aclk	800	800	clk_npu_aclk_st1			N	N	N	N	N	0
npu	npu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_npu_llc_src0	800	800	clk_spll0_fout1			N	N	2	2	15	N
internal	clk_npu_llc_src1	1040	1040	clk_spll2_fout1			N	N	2	2	15	N
npu	npu_llc_aclk	1188	1188	clk_npu_llc_src0	clk_npu_llc_src1	clk_vpll_fout1	GM	2	N	N	N	0
internal	clk_npu_core_st1	1500	1500	clk_spll1_fout1	clk_vpll_fout1	clk_spll2_fout2	GM	0	1	1	15	N
npu	npu_clk	1500	1500	clk_npu_core_st1			N	N	N	N	N	0

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
noc	npu_noc_aclk	800	800	clk_npu_aclk_st1			N	N	N	N	N	N
internal	clk_vi_aclk_st1	1040	800	clk_spll0_fout1	clk_spll2_fout1		GM	0	2	2	15	N
video input	vi_aclk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	0
video input	vi_dig_dw_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		M	0	2	2	15	0
video input	vi_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
video input	vi_dvp_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		GM	0	2	2	15	0
video input	vi_dig_isp_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		GM	0	2	2	15	0
shutter	vi_ref_clk[0]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[1]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[2]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[3]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[4]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[5]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
Combo PHY	vi_phy_txclkesc	20	20	clk_mipi_txesc			N	N	N	N	N	0
Combo PHY	vi_phy_cfg_clk	24	24	clk_xtal_24m			N	N	N	N	N	0
video output	vo_aclk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	0
video output	vo_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
video output	vo_hdmi_iesmclk	200	200	clk_spll0_fout3			N	N	2	2	15	0
video output	vo_pixel_clk	594	594	clk_vpll_fout1	clk_spll2_fout2		M	0	2	2	63	1
video output	clk_vo_mclk_st1	98.3	98.3	clk_apll_fout1			N	N	10	10	255	N
video output	vo_i2s_mclk	98.3	98.3	clk_vo_mclk_st1	ext_mclk		M	0	N	N	N	0
video output	vo_cr_clk	20	20	clk_mipi_txesc			N	N	N	N	N	0
video output	vo_cec_clk	0.032	0.032	clk_vpll_fout2			N	N	18128	18128	18128	N
video output	vo_phy_txclkesc	20	20	clk_mipi_txesc			N	N	N	N	N	0
video codec	vc_aclk	1040	1040	clk_spll0_fout1	clk_spll2_fout1		GM	1	2	2	15	0
internal	pad_aud_mclk_o	98.3	98.3	clk_vo_mclk_st1			N	N	N	N	N	N

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
video codec	vc_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_vc_cdec_root	2080	1600	clk_spll0_fout1	clk_spll2_fout2		GM	0	N	N	N	N
video codec	vc_je_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	vc_jd_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	vc_ve_clk	693.33	533.33	clk_vc_cdec_root			N	N	3	3	15	0
video codec	vc_vd_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	clk_g2d_st1	1040	1040	clk_dsp_root			N	N	2	2	15	N
video codec	g2d_clk	1040	1040	clk_g2d_st1			N	N	N	N	N	0
video codec	g2d_aclk	1040	1040	clk_g2d_st1			N	N	N	N	N	0
internal	clk_pvt_inner	1.2	1.2	clk_xtal_24m			N	N	20	20	20	N
system	pvt_clk[0]	1.2	1.2	clk_pvt_inner			N	N	N	N	N	1
ddr	pvt_clk[1]	1.2	1.2	clk_pvt_inner			N	N	N	N	N	1
dma	clk_aondma_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	clk_aondma_axi_st3	800	24	clk_spll0_fout1	clk_xtal_24m		M	1	2	2	15	N
dma	aondma_aclk	800	24	clk_aondma_axi_st3			N	N	N	N	N	1
secure, lpcpu	aon_aclk	800	24	clk_aondma_axi_st3			N	N	N	N	N	1
TIMER0	timer_clk[0]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER1	timer_clk[1]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER2	timer_clk[2]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER3	timer_clk[3]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER0	timer_pclk[0]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER1	timer_pclk[1]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER2	timer_pclk[2]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER3	timer_pclk[3]	200	200	clk_sys_cfg			N	N	N	N	N	1
rtc	clk_rtc_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
rtc	clk_rtc	0.5	0.15625	clk_1m			N	N	2	64	64	N
i2c10	i2c10_pclk	200	200	clk_sys_cfg			N	N	N	N	N	1
i2c11	i2c11_pclk	200	200	clk_sys_cfg			N	N	N	N	N	1
secure	clk_pka_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
secure	clk_spacc_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
clk_crypto	clk_crypto	400	400	clk_spll0_fout1			N	N	4	4	15	1
clk_trng_cfg	clk_trng_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
otp	clk_otp_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
clmm	clmm_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gpio	clmm_deb_clk	1	1	clk_1m			N	N	N	N	N	N
lsp	lsp_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
noc	noc_wd_refclk	24	24	clk_sys_cfg			N	N	1	1	2	1
noc	rnoc_nsp_clk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	1
TIMER3-timer8	timer3_clk8	50	50	clk_vpll_fout3			N	N	N	N	N	1

3.2.3.3 PLL

3.2.3.3.1 features

- Input frequency range: 12MHz ~ 1200MHz
- Output frequency range: 9MHz ~ 2500MHz
- 24-bit fractional accuracy
- The lock signal is used to indicate that the output frequency has been successfully locked at the set value

3.2.3.3.2 configuration

In integer mode, the frequency is calculated by the following formula:

$$FOUT* = \frac{FREF \times FBDIV}{4 \times REFDIV \times (POSTDIV_A + 1) \times (POSTDIV_B + 1)}$$

In fractional mode, the frequency is calculated by the following formula:

$$FOUT* = \frac{FREF}{4 \times REFDIV} \times \frac{FBDIV + \frac{FRAC}{2^{24}}}{(POSTDIV_A + 1) \times (POSTDIV_B + 1)}$$

Note: POSTDIV_A should be bigger than POSTDIV_B

Configuration process:

1. Configure pllen = 0
2. Configure PLL working mode, fraction mode (dsmen=1) or integer mode (dsmen=0)
3. Configure clock frequency/bypass, and other information
4. Configure pllen = 1 and wait for lock = 1

3.2.3.3.3 PLL reference clock

Table 3-2 PLL reference clock

pll name	refclk name	refclk freq (MHz)	remarks
spll0	clk_xtal_24m	24	Input from external source
spll1	clk_xtal_24m	24	Input from external source
spll2	clk_xtal_24m	24	Input from external source
vpll	clk_xtal_24m	24	Input from external source
apll	clk_xtal_24m	24	Input from external source
cpll	clk_xtal_24m	24	Input from external source
dpll	clk_xtal_24m	24	Input from external source，default
	pinmux_lpddr_ref_clk_bak	-	Input directly by pad LPDDR_REF_CLK. The frequency depends on the input frequency

3.2.3.3.4 PLL spread-spectrum

The spll0/spll1/spll2/vpll/apll/cpll/dpll in the system have spread spectrum modulator (SSMOD) module, which can spread the output clock of the pll and reduce the EMI impact of the peak frequency. SSMOD is not recommended for clocks with long-term jitter requirements. Turning on ssmod causes a

maximum of 1.56% skew.

When using ssmod, the PLL must be configured to fractional mode:

1. Initialize the modulator with **RESET**=1'b1, **DISABLE_SSCG**=1'b1, and **RESETPTR**=1'b1
2. Initialize the PLL with **PD**=1'bl
3. Provide valid settings to **INTIN**(fbdiv), **FRACIN**, **DIVVAL**(ss_div), **SPREAD**, and **DOWNSPREAD**
4. Set the PLL to fration mode，**DSMPD** = 0，**DACPD** = 0（**DSMEN** = 1, **DACEN** = 1）
5. Keep **pllen**=0 at least 1us
6. Set **RESET**=1'b0 and **DISABLE_SSCG**=1'b0 to activate the modulator.
7. After the 1us delay, set **PD**=1'b0 and wait for the PLL to establish **LOCK**=1'b1
8. Once the PLL has asserted its LOCK signal, set **RESETPTR**=1'b0 and after 2 **CLKSSCG** periods, the modulator will begin to update the PLL divide settings.

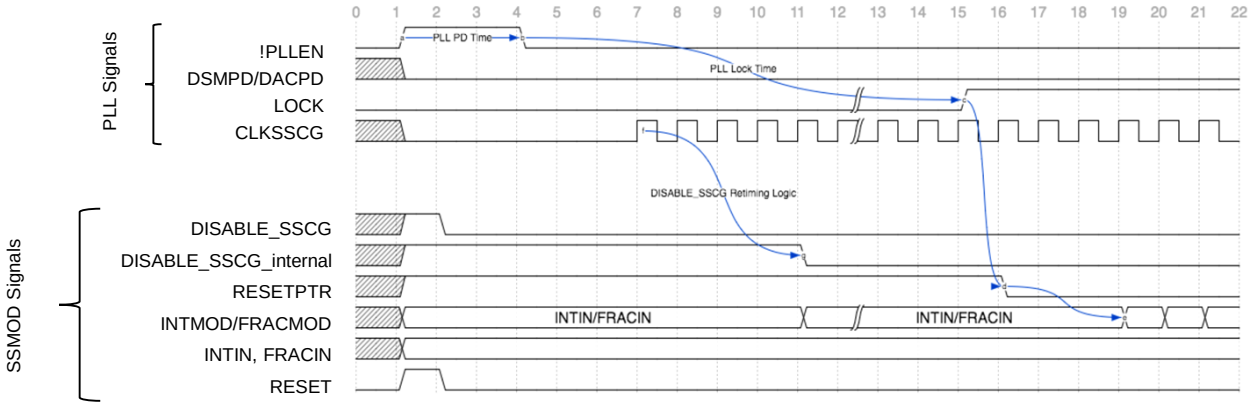


Figure 3-3 Timing diagram for programming SSMOD

3.2.3.4 Clock switching

The clock is switched as follows:

1. Static switching

Clock switching with ordinary MUX can only be configured before the subsystem reset is released, cannot be switched during work

2. Dynamic switching

The clock with glitch-free mux can be switched dynamically without glitches during the switching. If the clock has clock gating, it is recommended to gating first and then switching

3. Dynamic divider

The clock of dynamic frequency division supports dynamic configuration during operation, but the following requirements need to be followed:

- Gating the clock first
- Configure the dynamic divider
- Wait 32 cycles to release gating

3.2.4 Register specification

Address space: 0x51828000~0x5182FFFF, 32K

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h0000	spll0_cfg_0	spll0_fbdiv	31	20	RW	12'd266	
		reserved	19	18	RO	2'b0	
		spll0_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		spll0_dsmen	9	9	RW	1'b1	0: interger mode 1: fractional mode
		spll0_dacen	8	8	RW	1'b1	Factional noise caseling DAC in FRAC mode, 0: disable 1: enable
		spll0_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		spll0_frefcmlen	1	1	RW	1'b0	Feference clock select(do not change this bit)
		spll0_en	0	0	RW	1'b1	
16'h0004	spll0_cfg_1	reserved	31	28	RO	4'b0	
		spll0_frac	27	4	RW	24'd11184810	
		spll0_fouten	3	0	RW	4'b1110	
16'h0008	spll0_cfg_2	spll0_fout2_clken	31	31	RW	1'b1	
		spll0_postdiv3_b	30	28	RW	3'd0	
		reserved	27	27	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		spll0_postdiv3_a	26	24	RW	3'd3	
		reserved	23	23	RO	1'b0	
		spll0_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		spll0_postdiv2_a	18	16	RW	3'd1	
		reserved	15	15	RO	1'b0	
		spll0_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		spll0_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		spll0_postdiv0_b	6	4	RW	3'd0	
		spll0_postdiv0_a	3	1	RW	3'd0	
		spll0_postdiv0_pre	0	0	RW	1'b0	
16'h000c	spll0_dskewcal	spll0_dskewcalout	31	20	RO	12'd0	
		spll0_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		spll0_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		spll0_dskewcalbyp	2	2	RW	1'b0	
		spll0_dskewfastcal	1	1	RW	1'b0	
		spll0_dskewcalen	0	0	RW	1'b0	
16'h0010	spll0_ssc	reserved	31	17	RO	15'b0	
		spll0_ssc_spread	16	12	RW	5'd0	spread depth:

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							5'b00001:0.1%
							……
							5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		spll0_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		spll0_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread
							1'b1:down-spread
		spll0_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode
							1'b1:bypass spread mode
		spll0_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		spll0_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating
							1'b1:reset modulator
16'h0014	spll1_cfg_0	spll1_fbdiv	31	20	RW	12'd250	
		reserved	19	18	RO	2'b0	
		spll1_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		spll1_dsmen	9	9	RW	1'b0	
		spll1_dacen	8	8	RW	1'b0	
		spll1_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		spll1_frefcmlen	1	1	RW	1'b0	
		spll1_en	0	0	RW	1'b1	
16'h0018	spll1_cfg_1	reserved	31	28	RO	4'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		spll1_frac	27	4	RW	24'd0	
		spll1_fouten	3	0	RW	4'b1110	
16'h001C	spll1_cfg_2	reserved	31	31	RO	1'b0	
		spll1_postdiv3_b	30	28	RW	3'd0	
		reserved	27	27	RO	1'b0	
		spll1_postdiv3_a	26	24	RW	3'd5	
		reserved	23	23	RO	1'b0	
		spll1_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		spll1_postdiv2_a	18	16	RW	3'd4	
		reserved	15	15	RO	1'b0	
		spll1_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		spll1_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		spll1_postdiv0_b	6	4	RW	3'd0	
		spll1_postdiv0_a	3	1	RW	3'd0	
		spll1_postdiv0_pre	0	0	RW	1'b0	
16'h0020	spll1_dskewcal	spll1_dskewcalout	31	20	RO	12'd0	
		spll1_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		spll1_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		spll1_dskewcalbyp	2	2	RW	1'b0	
		spll1_dskewfastcal	1	1	RW	1'b0	
		spll1_dskewcalen	0	0	RW	1'b0	
16'h0024	spll1_ssc	reserved	31	17	RO	15'b0	
		spll1_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		spll1_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		spll1_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread 1'b1:down-spread
		spll1_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode 1'b1:bypass spread mode
		spll1_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		spll1_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating 1'b1:reset modulator
16'h0028	spll2_cfg_0	spll2_fbdiv	31	20	RW	12'd346	
		reserved	19	18	RO	2'b0	
		spll2_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		spll2_dsmen	9	9	RW	1'b1	
		spll2_dacen	8	8	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		spll2_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		spll2_frefcmlen	1	1	RW	1'b0	
		spll2_en	0	0	RW	1'b1	
16'h002c	spll2_cfg_1	reserved	31	28	RO	4'b0	
		spll2_frac	27	4	RW	24'd11184810	
		spll2_fouten	3	0	RW	4'b1110	
16'h0030	spll2_cfg_2	reserved	31	31	RO	1'b0	
		spll2_postdiv3_b	30	28	RW	3'd0	
		reserved	27	27	RO	1'b0	
		spll2_postdiv3_a	26	24	RW	3'd4	
		reserved	23	23	RO	1'b0	
		spll2_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		spll2_postdiv2_a	18	16	RW	3'd1	
		reserved	15	15	RO	1'b0	
		spll2_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		spll2_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		spll2_postdiv0_b	6	4	RW	3'd0	
		spll2_postdiv0_a	3	1	RW	3'd0	
		spll2_postdiv0_pre	0	0	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h0034	spll2_dskewcal	spll2_dskewcalout	31	20	RO	12'd0	
		spll2_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		spll2_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		spll2_dskewcalbyp	2	2	RW	1'b0	
		spll2_dskewfastcal	1	1	RW	1'b0	
		spll2_dskewcalen	0	0	RW	1'b0	
16'h0038	spll2_ssc	reserved	31	17	RO	15'b0	
		spll2_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		spll2_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		spll2_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread 1'b1:down-spread
		spll2_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode 1'b1:bypass spread mode
		spll2_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		spll2_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating 1'b1:reset modulator
16'h003c	vpll_cfg_0	vpll_fbdiv	31	20	RW	12'd198	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	18	RO	2'b0	
		vpll_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		vpll_dsmen	9	9	RW	1'b0	
		vpll_dacen	8	8	RW	1'b0	
		vpll_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		vpll_frefcmlen	1	1	RW	1'b0	
		vpll_en	0	0	RW	1'b1	
16'h0040	vpll_cfg_1	reserved	31	28	RO	4'b0	
		vpll_frac	27	4	RW	24'd0	
		vpll_fouten	3	0	RW	4'b1110	
16'h0044	vpll_cfg_2	reserved	31	31	RO	1'b0	
		vpll_postdiv3_b	30	28	RW	3'd3	
		reserved	27	27	RO	1'b0	
		vpll_postdiv3_a	26	24	RW	3'd5	
		reserved	23	23	RO	1'b0	
		vpll_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		vpll_postdiv2_a	18	16	RW	3'd1	
		reserved	15	15	RO	1'b0	
		vpll_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		vpll_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		vpll_postdiv0_b	6	4	RW	3'd0	
		vpll_postdiv0_a	3	1	RW	3'd0	
		vpll_postdiv0_pre	0	0	RW	1'b0	
16'h0048	vpll_dskewcal	vpll_dskewcalout	31	20	RO	12'd0	
		vpll_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		vpll_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		vpll_dskewcalbyp	2	2	RW	1'b0	
		vpll_dskewfastcal	1	1	RW	1'b0	
		vpll_dskewcalen	0	0	RW	1'b0	
16'h004c	vpll_ssc	reserved	31	17	RO	15'b0	
		vpll_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		vpll_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		vpll_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread 1'b1:down-spread
		vpll_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:bypass spread mode
		vpll_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		vpll_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating
							1'b1:reset modulator
16'h0050	apll_cfg_0	apll_fbdiv	31	20	RW	12'd163	
		reserved	19	18	RO	2'b0	
		apll_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		apll_dsmen	9	9	RW	1'b1	
		apll_dacen	8	8	RW	1'b1	
		apll_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		apll_frefcmlen	1	1	RW	1'b0	
		apll_en	0	0	RW	1'b1	
16'h0054	apll_cfg_1	reserved	31	28	RO	4'b0	
		apll_frac	27	4	RW	24'd14092861	
		apll_fouten	3	0	RW	4'b0010	
16'h0058	apll_cfg_2	reserved	31	31	RO	1'b0	
		apll_postdiv3_b	30	28	RW	3'd3	
		reserved	27	27	RO	1'b0	
		apll_postdiv3_a	26	24	RW	3'd4	
		reserved	23	23	RO	1'b0	
		apll_postdiv2_b	22	20	RW	3'd0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	19	RO	1'b0	
		apll_postdiv2_a	18	16	RW	3'd0	
		reserved	15	15	RO	1'b0	
		apll_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		apll_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		apll_postdiv0_b	6	4	RW	3'd0	
		apll_postdiv0_a	3	1	RW	3'd0	
		apll_postdiv0_pre	0	0	RW	1'b0	
16'h005c	apll_dskewcal	apll_dskewcalout	31	20	RO	12'd0	
		apll_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		apll_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		apll_dskewcalbyp	2	2	RW	1'b0	
		apll_dskewfastcal	1	1	RW	1'b0	
		apll_dskewcalen	0	0	RW	1'b0	
16'h0060	apll_ssc	reserved	31	17	RO	15'b0	
		apll_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	11	10	RO	2'b0	
		apll_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		apll_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread
							1'b1:down-spread
		apll_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode
							1'b1:bypass spread mode
		apll_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		apll_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating
							1'b1:reset modulator
16'h0064	mcput_pll_cfg_0	mcput_pll_fbdiv	31	20	RW	12'd266	
		reserved	19	18	RO	2'b0	
		mcput_pll_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		mcput_pll_dsmen	9	9	RW	1'b1	
		mcput_pll_dacen	8	8	RW	1'b1	
		mcput_pll_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		mcput_pll_frefcmlen	1	1	RW	1'b0	
		mcput_pll_en	0	0	RW	1'b1	
16'h0068	mcput_pll_cfg_1	reserved	31	28	RO	4'b0	
		mcput_pll_frac	27	4	RW	24'd11184810	
		mcput_pll_fouten	3	0	RW	4'b1010	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h006c	mcput_pll_cfg_2	reserved	31	31	RO	1'b0	
		mcput_pll_postdiv3_b	30	28	RW	3'd3	
		reserved	27	27	RO	1'b0	
		mcput_pll_postdiv3_a	26	24	RW	3'd7	
		reserved	23	23	RO	1'b0	
		mcput_pll_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		mcput_pll_postdiv2_a	18	16	RW	3'd1	
		reserved	15	15	RO	1'b0	
		mcput_pll_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		mcput_pll_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		mcput_pll_postdiv0_b	6	4	RW	3'd1	
		mcput_pll_postdiv0_a	3	1	RW	3'd7	
		mcput_pll_postdiv0_pre	0	0	RW	1'b0	
16'h0070	mcput_pll_dskewcal	mcput_pll_dskewcalout	31	20	RO	12'd0	
		mcput_pll_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mcput_pll_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		mcput_pll_dskewcalbyp	2	2	RW	1'b0	
		mcput_pll_dskewfastcal	1	1	RW	1'b0	
		mcput_pll_dskewcalen	0	0	RW	1'b0	
16'h0074	mcput_pll_ssc	reserved	31	17	RO	15'b0	
		mcput_pll_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		mcput_pll_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		mcput_pll_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread 1'b1:down-spread
		mcput_pll_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode 1'b1:bypass spread mode
		mcput_pll_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		mcput_pll_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating 1'b1:reset modulator
16'h0078	ddrt_pll_cfg_0	ddrt_pll_fbdiv	31	20	RW	12'd177	1066:177 800:133 533:88

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	18	RO	2'b0	
		ddrt_pll_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		ddrt_pll_dsmen	9	9	RW	1'b1	
		ddrt_pll_dacen	8	8	RW	1'b1	
		ddrt_pll_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		ddrt_pll_frefcmlen	1	1	RW	1'b0	
		ddrt_pll_en	0	0	RW	1'b1	
16'h007c	ddrt_pll_cfg_1	reserved	31	28	RO	4'b0	
		ddrt_pll_frac	27	4	RW	24'd11184810	1066:11184810
							800:5592405
							533:13981013
		ddrt_pll_fouten	3	0	RW	4'b0010	
16'h0080	ddrt_pll_cfg_2	reserved	31	31	RO	1'b0	
		ddrt_pll_postdiv3_b	30	28	RW	3'd0	
		reserved	27	27	RO	1'b0	
		ddrt_pll_postdiv3_a	26	24	RW	3'd7	
		reserved	23	23	RO	1'b0	
		ddrt_pll_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		ddrt_pll_postdiv2_a	18	16	RW	3'd3	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	15	15	RO	1'b0	
		ddrt_pll_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		ddrt_pll_postdiv1_a	10	8	RW	3'd1	
		reserved	7	7	RO	1'b0	
		ddrt_pll_postdiv0_b	6	4	RW	3'd0	
		ddrt_pll_postdiv0_a	3	1	RW	3'd0	
		ddrt_pll_postdiv0_pre	0	0	RW	1'b0	
16'h0084	ddrt_pll_dskewcal	ddrt_pll_dskewcalout	31	20	RO	12'd0	
		ddrt_pll_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		ddrt_pll_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		ddrt_pll_dskewcalbyp	2	2	RW	1'b0	
		ddrt_pll_dskewfastcal	1	1	RW	1'b0	
		ddrt_pll_dskewcalen	0	0	RW	1'b0	
16'h0088	ddrt_pll_ssc	reserved	31	17	RO	15'b0	
		ddrt_pll_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% ……

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		ddrt_pll_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		ddrt_pll_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread
							1'b1:down-spread
		ddrt_pll_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode
							1'b1:bypass spread mode
		ddrt_pll_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		ddrt_pll_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating
							1'b1:reset modulator
16'h00a4	pll_status	reserved	31	24	RO	8'b0	
		d2d_pll_dskewcalock	23	23	RO	1'b0	pll dskew lock,active hig
		ddrt_pll_dskewcalock	22	22	RO	1'b0	pll dskew lock,active hig
		mcput_pll_dskewcalock	21	21	RO	1'b0	pll dskew lock,active hig
		apll_dskewcalock	20	20	RO	1'b0	pll dskew lock,active hig
		vpll_dskewcalock	19	19	RO	1'b0	pll dskew lock,active hig
		spll2_dskewcalock	18	18	RO	1'b0	pll dskew lock,active hig
		spll1_dskewcalock	17	17	RO	1'b0	pll dskew lock,active hig
		spll0_dskewcalock	16	16	RO	1'b0	pll dskew lock,active hig
		reserved	15	8	RO	8'b0	
		d2d_pll_lock	7	7	RO	1'b0	pll lock,active hig
		ddrt_pll_lock	6	6	RO	1'b0	pll lock,active hig

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mcput_pll_lock	5	5	RO	1'b0	pll lock,active hig
		apll_lock	4	4	RO	1'b0	pll lock,active hig
		vpll_lock	3	3	RO	1'b0	pll lock,active hig
		spll2_lock	2	2	RO	1'b0	pll lock,active hig
		spll1_lock	1	1	RO	1'b0	pll lock,active hig
		spll0_lock	0	0	RO	1'b0	pll lock,active hig
16'h00b0	d2d_pll_cfg_0	d2d_pll_fbdiv	31	20	RW	12'd250	
		reserved	19	18	RO	2'b0	
		d2d_pll_refdiv	17	12	RW	6'd1	
		reserved	11	10	RO	2'b0	
		d2d_pll_dsmen	9	9	RW	1'b0	
		d2d_pll_dacen	8	8	RW	1'b0	
		d2d_pll_byp	7	4	RW	4'h0	
		reserved	3	2	RO	2'b0	
		d2d_pll_frefcmlen	1	1	RW	1'b0	
		d2d_pll_en	0	0	RW	1'b1	
16'h00b4	d2d_pll_cfg_1	reserved	31	28	RO	4'b0	
		d2d_pll_frac	27	4	RW	24'd0	
		d2d_pll_fouten	3	0	RW	4'b0010	
16'h00b8	d2d_pll_cfg_2	reserved	31	31	RO	1'b0	
		d2d_pll_postdiv3_b	30	28	RW	3'd0	
		reserved	27	27	RO	1'b0	
		d2d_pll_postdiv3_a	26	24	RW	3'd0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	23	23	RO	1'b0	
		d2d_pll_postdiv2_b	22	20	RW	3'd0	
		reserved	19	19	RO	1'b0	
		d2d_pll_postdiv2_a	18	16	RW	3'd1	
		reserved	15	15	RO	1'b0	
		d2d_pll_postdiv1_b	14	12	RW	3'd0	
		reserved	11	11	RO	1'b0	
		d2d_pll_postdiv1_a	10	8	RW	3'd0	
		reserved	7	7	RO	1'b0	
		d2d_pll_postdiv0_b	6	4	RW	3'd0	
		d2d_pll_postdiv0_a	3	1	RW	3'd3	
		d2d_pll_postdiv0_pre	0	0	RW	1'b1	
16'h00bc	d2d_pll_dskewcal	d2d_pll_dskewcalout	31	20	RO	12'd0	
		d2d_pll_dskewcalin	19	8	RW	12'd0	
		reserved	7	7	RO	1'b0	
		d2d_pll_dskewcalcnt	6	4	RW	3'd0	
		reserved	3	3	RO	1'b0	
		d2d_pll_dskewcalbyp	2	2	RW	1'b0	
		d2d_pll_dskewfastcal	1	1	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		d2d_pll_dskewcalen	0	0	RW	1'b0	
16'h00c0	d2d_pll_ssc	reserved	31	17	RO	15'b0	
		d2d_pll_ssc_spread	16	12	RW	5'd0	spread depth: 5'b00001:0.1% …… 5'b11111:3.1%
		reserved	11	10	RO	2'b0	
		d2d_pll_ssc_div	9	4	RW	6'd0	fmod=ssclk/(128*ssc_div)
		d2d_pll_ssc_dnspread	3	3	RW	1'b0	1'b0:center-spread 1'b1:down-spread
		d2d_pll_ssc_disable	2	2	RW	1'b1	1'b0:use spread mode 1'b1:bypass spread mode
		d2d_pll_ssc_resetptr	1	1	RW	1'b1	active high for reset wave tale pointer
		d2d_pll_ssc_reset	0	0	RW	1'b1	1'b0:modulator is operating 1'b1:reset modulator
16'h100	noc_clk_ctrl	noc_nsp_clken	31	31	RW	1'b1	Clk: noc_nsp_clk noc nsp clock enable, 1'b0:clock gated 1'b1:clock enable
		noc_wdref_clken	30	30	RW	1'b1	Clk: noc_wd_refclk clock gate enable, 1'b0:clock gated 1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		rnoc_nsp_clken	29	29	RW	1'b1	Clk: rnoc_nsp_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	28	20	RO	9'd0	
		noc_wdref_divsor	19	4	RW	16'd1	Clk: noc_wd_refclk 0,1: no divisior 2:2 divisior …… n:n divisior
		reserved	3	3	RO	1'b0	
		nocnsp_divsor	2	0	RW	3'd2	Clk: noc_nsp_clk 0,1,2:2 divisior …… n:n divisior
16'h104	bootspi_clk_ctrl	bootspi_clken	31	31	RW	1'b1	Clk: clk_bootspi clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	10	RO	21'b0	
		bootspi_divsor	9	4	RW	6'd20	Clk: clk_bootspi_st3 bootspi interface clock divisor 0,1,2:2 divisior ……

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							n:n divisior
		reserved	3	1	RO	3'b0	
		bootspi_clk_sel	0	0	RW	1'b1	Clk: clk_bootspi_st3 1: XTAL 24M Selected,
							0: internal function clock(50M)
16'h108	bootspi_cfgclk_ctrl	bootspi_cfg_clken	31	31	RW	1'b1	Clk: clk_bootspi_cfg clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h10c	scpu_coreclk_ctrl	scpu_core_clken	31	31	RW	1'b1	Clk: clk_scpu_core clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	8	RO	23'b0	
		scpu_core_divsor	7	4	RW	4'd2	Clk: clk_scpu_core scpu core clock divisor
							0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		scpu_core_clksel	0	0	RW	1'b1	clk source selection.
							1: XTAL24M,
							0: internal function clock.(400M)

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h110	scpu_busclk_ctrl	scpu_bus_clken	31	31	RW	1'b1	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	17	RO	14'b0	
		scpu_bus_clksel	16	16	RW	1'b0	reserved,always 2:1 ratio. scpu bus clock ratio selection. 1-1:1 ratio, 0-2:1 ratio
		reserved	15	0	RO	16'b0	
16'h114	lpcpu_coreclk_ctrl	lpcpu_core_clken	31	31	RW	1'b1	Clk: clk_lpcpu_core clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	8	RO	23'b0	
		lpcpu_core_divsor	7	4	RW	4'd2	Clk: clk_lpcpu_core lpcpu core clock divisor 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		lpcpu_core_clksel	0	0	RW	1'b1	clk source selection. 1: XTAL24M, 0: internal function clock.(400M)
16'h118	lpcpu_busclk_ctrl	lpcpu_bus_clken	31	31	RW	1'b1	Clk: clk_lpcpu_bus clock gate enable,

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	17	RO	14'b0	
		lpcpu_bus_clksel	16	16	RW	1'b0	reserved
		reserved	15	0	RO	16'b0	
16'h11C	tcu_aclk_ctrl	tcu_aclk_clken	31	31	RW	1'b1	Clk: tcu_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h120	tcu_cfg_ctrl	tcu_cfg_clken	31	31	RW	1'b1	Clk: tcu_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h124	ddr0_clk_ctrl	reserved	31	24	RO	8'b0	
		ddr_aclk_divsor	23	20	RW	4'd2	Clk: clk_ddr_aclk ddr axi clock divisor
							0,1,2:2 divisior
							……
							n:n divisior
		reserved	19	17	RO	3'b0	
		ddr_aclksrc_sel	16	16	RW	1'b1	Clk: clk_ddr_aclk_st0 ddr AXI clock source selection.

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							0-clk_spll2_fout1, (1040M)
							1-clk_spll0_fout1.(800M)
		reserved	15	10	RO	6'd0	
		ddr_cfg_clken	9	9	RW	1'b1	Clk: ddrt_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		ddr0_aclk_clken	8	4	RW	5'h1f	Clk: ddrt0_pN_aclk, N = 0/1/2/3/4 clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		ddrt_pll_bypass_en	3	3	RW	1'b0	DDR PHY PLL bypass enable, high active.
		ddrt_core_clk_div_sel	2	2	RW	1'b0	In bypass mode, divisior for ddr core clock
							0:4 divisior
							1:8 divisior
		ddrt_rx_test_clk_en	1	1	RW	1'b1	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		ddr0_trace_clken	0	0	RW	1'b1	Clk: ddrt0_trace_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h128	ddr1_clk_ctrl	reserved	31	10	RO	22'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	9	9	RO	1'b0	
		ddr1_aclk_clken	8	4	RW	5'h1f	Clk: ddrt1_pN_aclk, N = 0/1/2/3/4 clock gate enale,
							1'b0:clock gated
							1'b1:clock enable
		reserved	3	3	RO	1'b0	DDR PHY PLL bypass enable, high active.
		reserved	2	2	RO	1'b0	
		reserved	1	1	RO	1'b0	
		ddr1_trace_clken	0	0	RW	1'b1	Clk: ddrt1_trace_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h12c	gpu_aclk_ctrl	gpu_aclk_clken	31	31	RW	1'b0	Clk: gpu_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		gpu_aclk_divsor	7	4	RW	4'd2	Clk: gpu_aclk gpu axi clock divisor
							0,1,2:2 divisior
							……
							n:n divisior

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	3	0	RO	4'b0	
16'h130	gpu_cfg_ctrl	gpu_cfg_clken	31	31	RW	1'b0	Clk: gpu_cfg_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h134	gpu_gray_ctrl	gpu_gray_clken	31	31	RW	1'b0	Clk: gpu_gray_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h138	dsp_aclk_ctrl	dsp_aclk_clken	31	31	RW	1'b0	Clk: dspt_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		dsp_aclk_divsor	7	4	RW	4'd2	Clk: clk_dsp_aclk_st1 dsp axi clock divisor 0,1,2:2 …… n:n divisior
		reserved	3	1	RO	3'b0	
		dsp_aclksrc_sel	0	0	RW	1'b0	Clk: clk_dsp_aclk_st1

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							DSP AXI clock source selection.
							0-clk_spll2_fout1, (1040M)
							1-clk_spll0_fout1.(800M)
16'h13c	dsp_cfg_ctrl	dsp_cfg_clken	31	31	RW	1'b0	Clk: dspt_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	27	RO	4'd0	
		scu_dspt_div_en	26	23	RW	4'd0	not used
		scu_dspt_div_sel	22	19	RW	4'd0	divisor in dsp
							[19]: divisior in dsp core0 1'b0:no divisior 1'b1:2 divisior
							[20]: divisior in dsp core1 1'b0:no divisior 1'b1:2 divisior
							[21]: Not usesd

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							[22]: Not uesd
		reserved	18	0	RO	31'b0	
16'h140	d2d_aclk_ctrl	d2d_aclk_clken	31	31	RW	1'b0	Clk: d2d_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		d2d_aclk_divsor	7	4	RW	4'd2	Clk: clk_d2ddr_aclk d2d axi clock divisor 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		d2d_aclksrc_sel	0	0	RW	1'b0	Clk: clk_d2ddr_aclk D2D AXI clock source selection. 0-clk_spll2_fout1, (1040M) 1-clk_spll0_fout1.(800M)
16'h144	d2d_clk_ctrl	reserved	31	1	RO	31'b0	
		d2d_cfg_clken	0	0	RW	1'b0	Clk: d2d_cfg_clk

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h148	hsp_aclk_ctrl	hsp_aclk_clken	31	31	RW	1'b1	Clk: hsp_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		hsp_aclk_divsor	7	4	RW	4'd2	Clk: hsp_aclk_free hsp axi clock divisor
							0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	2	RO	2'b0	
		scu_hsp_dma0_clk_test_en	1	1	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		scu_hsp_dma0_clk_en	0	0	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h14c	hsp_cfg_ctrl	hsp_cfg_clken	31	31	RW	1'b1	Clk: hsp_cfg_clk clock gate enable,
							1'b0:clock gated

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:clock enable
		scu_hsp_pclk_en	30	30	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	29	0	RO	30'b0	
16'h150	sata_rbc_ctrl	reserved	31	31	RO	1'b0	
		reserved	30	6	RO	25'b0	
		sata_rbc_divsor	5	3	RW	3'd1	not used
		reserved	2	1	RO	2'b0	
		sata_rbc_clken	0	0	RW	1'b1	Clk: hsp_sata_rbc_clk not used
16'h154	sata_oob_ctrl	sata_oob_clken	31	31	RW	1'b1	Clk: sata_phy_ref_clk_gated gating oob,pmalive,phy_ref
		reserved	30	10	RO	21'b0	
		hsp_sata_phy_bak_sel	9	9	RW	1'b0	Clk: sata_phy_ref_clk_gated 1'b0:sel pll clock 1'b1:sel 50m bak clock
		sata_oob_divsor	8	4	RW	5'd6	not used
		sata_phy_ref_clk_divsor	3	0	RW	4'd6	Clk: sata_phy_ref_clk 0,1,2:2 divisior …… n:n divisior
16'h158	eth0_ctrl	rmii0_ref_clken	31	31	RW	1'b1	Clk: hsp_rmii0_ref_clk clock gate enable,

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	11	RO	20'b0	
		eth0_divsor	10	4	RW	7'd2	Clk: hsp_eth0_core_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	3	RO	1'b0	
		hsp_rmii_ref_bak_sel	2	2	RW	1'b0	Clk: rmii_ref_clk_mux 1'b0:sel pll clock
							1'b1:sel 50m bak clock
		hsp_eth_core_bak_sel	1	1	RW	1'b0	1'b0:sel pll clock
							1'b1:sel 50m bak clock
		hsp_eth0_clk_en	0	0	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h15c	eth1_ctrl	rmii1_ref_clken	31	31	RW	1'b1	Clk: hsp_rmii1_ref_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	11	RO	20'b0	
		eth1_divsor	10	4	RW	7'd2	Clk: hsp_eth1_core_clk 0,1,2:2 divisior
							……

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							n:n divisior
		reserved	3	1	RO	3'b0	
		hsp_eth1_clk_en	0	0	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h160	mshc0_coreclk_ctrl	reserved	31	31	RO	1'b0	
		reserved	30	20	RO	11'b0	
		reserved	19	17	RO	3'd0	
		mshc0_coreclk_en	16	16	RW	1'b0	Clk: hsp_mshc0_core_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		mshcore0_divsor	15	4	RW	12'd2	Clk: hsp_mshc0_core_clk 0,1: no divisior 2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		mshc0_coreclk_sel	0	0	RW	1'b0	Clk: hsp_mshc0_core_clk 0: clk_spll0_fout3 selected as source clock.(200M) 1: clk_spll2_fout3(208M)
16'h164	mshc1_coreclk_ctrl	reserved	31	31	RO	1'b0	
		reserved	30	20	RO	11'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	17	RO	3'd0	
		mshc1_coreclk_en	16	16	RW	1'b0	Clk: hsp_mshc1_core_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		mshcore1_divsor	15	4	RW	12'd2	0,1:no divisior
							2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		mshc1_coreclk_sel	0	0	RW	1'b0	Clk: hsp_mshc1_core_clk 0: clk_spll0_fout3 selected as source clock.(200M)
							1: clk_spll2_fout3(208M)
16'h168	mshc2_coreclk_ctrl	reserved	31	31	RO	1'b0	
		reserved	30	20	RO	11'b0	
		reserved	19	17	RO	3'd0	
		mshc2_coreclk_en	16	16	RW	1'b0	Clk: hsp_mshc1_core_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		mshcore2_divsor	15	4	RW	12'd2	Clk: hsp_mshc1_core_clk 0,1: no divisior
							2:2 divisior

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		mshc2_coreclk_sel	0	0	RW	1'b0	Clk: hsp_mshc1_core_clk 0: clk_spll0_fout3 selected as source clock.(200M)
							1: clk_spll2_fout3(208M)
16'h16c	mshc_usb_slwclk	reserved	31	6	RO	26'b0	
		reserved	5	5	RO	1'b0	
		reserved	4	4	RO	1'b0	
		reserved	3	3	RO	1'b0	
		reserved	2	2	RO	1'b0	
		reserved	1	1	RO	1'b0	
		reserved	0	0	RO	1'b0	
16'h170	pcie_aclk_ctrl	pcie_aclk_clken	31	31	RW	1'b1	Clk: pciet_aclk
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		pcie_aclk_divsor	7	4	RW	4'd2	Clk: pciet_aclk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h174	pcie_cfg_ctrl	pcie_cfg_clken	31	31	RW	1'b1	Clk: pciet_cfg_clk

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	30	2	RO	29'b0	
		pciet_aux_clken	1	1	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pciet_cr_clken	0	0	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h178	npu_aclk_ctrl	npu_aclk_clken	31	31	RW	1'b0	Clk: npu_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		npu_cfg_clken	30	30	RW	1'b0	Clk: npu_cfg_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	29	20	RO	10'b0	
		reserved	19	8	RO	12'b0	
		npu_aclk_divsor	7	4	RW	4'd2	Clk: clk_npu_aclk_st1 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		npu_aclksrc_sel	0	0	RW	1'b0	reserved
16'h17c	npu_llc_ctrl	npu_llc_clken	31	31	RW	1'b0	Clk: npu_llc_aclk

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	12	RO	8'b0	
		npu_llcsrc1_divsor	11	8	RW	4'd2	Clk: clk_npu_llc_src1 0,1,2:2 divisior
							……
							n:n divisior
		npu_llcsrc0_divsor	7	4	RW	4'd2	Clk: clk_npu_llc_src0 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	2	RO	2'b0	
		npu_llclk_sel	1	0	RW	2'd2	Clk: npu_llc_aclk 0-divided by clk_spll0_fout11, (800M)
							1-divide by clk_spll2_fout1, (1040M)
							2-clk_vpll_fout1(1188M)
16'h180	npu_core_ctrl	npu_core_clken	31	31	RW	1'b0	Clk: npu_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
							npu_core_clk e31 clock
		e31_core_clken	30	30	RW	1'b0	clock gate enable,

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b0:clock gated
							1'b1:clock enable
							note:not used
		reserved	29	20	RO	10'b0	
		reserved	19	16	RO	4'd0	
		npu_e31clk_divsor	15	12	RW	4'd2	0,1:no divisior
							2:2 divisior
							……
							n:n divisior
							note:not used
		reserved	11	10	RO	2'b0	
		npu_e31clk_sel	9	8	RW	2'd1	0-divided by clk_spll1_fout1, (1500M)
							1-divide by clk_vpll_fout1, (1188M)
							2-clk_spll2_fout2(1040M)
							note:not used
		npu_coreclk_divsor	7	4	RW	4'd2	Clk: clk_npu_core_st1 0,1: no divisior
							2:2 divisior
							……
							n:n divisior
		reserved	3	2	RO	2'b0	
		npu_coreclk_sel	1	0	RW	2'd0	Clk: clk_npu_core_st1 0-divided by clk_spll1_fout1, (1500M)
							1-divide by clk_vpll_fout1, (1188M)

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							2-clk_spll2_fout2(1040M)
16'h184	vi_dwclk_ctrl	vi_dw_clken	31	31	RW	1'b0	Clk: vi_dig_dw_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		vi_dw_divsor	7	4	RW	4'd2	Clk: vi_dig_dw_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vi_dwclksrc_sel	0	0	RW	1'b0	Clk: vi_dig_dw_clk 0-clk_vpll_fout1, (594M)
							1-clk_spll0_fout1(800M)
16'h188	vi_aclk_ctrl	vi_aclk_clken	31	31	RW	1'b0	Clk: vi_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		vi_cfg_clken	30	30	RW	1'b0	Clk: vi_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	29	20	RO	10'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	8	RO	12'b0	
		vi_aclk_divsor	7	4	RW	4'd2	Clk: clk_vi_aclk_st1 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vi_aclksrc_sel	0	0	RW	1'b0	Clk: clk_vi_aclk_st1 0-clk_spll0_fout1, (800M)
							1-clk_spll2_fout1.(1040M)
16'h18c	vi_dig_isp_clk_ctrl	vi_dig_isp_clken	31	31	RW	1'b0	Clk: vi_dig_isp_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		vi_dig_isp_divsor	7	4	RW	4'd2	Clk: vi_dig_isp_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vi_dig_isp_src_sel	0	0	RW	1'b0	Clk: vi_dig_isp_clk 0-clk_vpll_fout1, (594M)
							1-clk_spll0_fout1.(800M)
16'h190	vi_dvp_clk_ctrl	vi_dvp_clken	31	31	RW	1'b0	Clk: vi_dvp_clk clock gate enable,

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		vi_dvp_divsor	7	4	RW	4'd2	Clk: vi_dvp_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vi_dvp_src_sel	0	0	RW	1'b0	Clk: vi_dvp_clk 0-clk_vpll_fout1, (594M)
							1-clk_spll0_fout1(800M)
16'h194	vi_shutter0	vi_shutter_clken0	31	31	RW	1'b0	Clk: vi_ref_clk[0] clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor0	10	4	RW	7'd8	Clk: vi_ref_clk[0] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h198	vi_shutter1	vi_shutter_clken1	31	31	RW	1'b0	Clk: vi_ref_clk[1]

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor1	10	4	RW	7'd8	Clk: vi_ref_clk[1] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h19C	vi_shutter2	vi_shutter_clken2	31	31	RW	1'b0	Clk: vi_ref_clk[2] clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor2	10	4	RW	7'd8	Clk: vi_ref_clk[2] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1a0	vi_shutter3	vi_shutter_clken3	31	31	RW	1'b0	Clk: vi_ref_clk[3] clock gate enable,
							1'b0:clock gated

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor3	10	4	RW	7'd8	Clk: vi_ref_clk[3] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1a4	vi_shutter4	vi_shutter_clken4	31	31	RW	1'b0	Clk: vi_ref_clk[4] clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor4	10	4	RW	7'd8	Clk: vi_ref_clk[4] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1a8	vi_shutter5	vi_shutter_clken5	31	31	RW	1'b0	Clk: vi_ref_clk[5] clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	11	RO	9'b0	
		vi_shutter_divsor5	10	4	RW	7'd8	Clk: vi_ref_clk[5] 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1ac	vi_phy_clkctrl	reserved	31	2	RO	30'b0	
		vi_phycfg_clken	1	1	RW	1'b0	Clk: vi_phy_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		vi_txesc_clken	0	0	RW	1'b0	Clk: vi_phy_txclkesc clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h1b0	vo_aclk_ctrl	vo_aclk_clken	31	31	RW	1'b1	Clk: vo_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		vo_cfg_clken	30	30	RW	1'b1	Clk: vo_cfg_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	29	20	RO	10'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	19	8	RO	12'b0	
		vo_aclk_divsor	7	4	RW	4'd2	0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vo_aclksrc_sel	0	0	RW	1'b0	0:from spll0_fout1.(800M)
							1:from spll2 fout1.(1040M)
16'h1b4	vo_iesmclk_ctrl	vo_iesmclk_clken	31	31	RW	1'b0	Clk: vo_hdmi_iesmclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		vo_iesmclk_divsor	7	4	RW	4'd2	Clk: vo_hdmi_iesmclk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1b8	vo_pixel_ctrl	vo_pixel_clken	31	31	RW	1'b1	Clk: vo_pixel_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	10	RO	10'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		vo_pixel_divsor	9	4	RW	6'd2	Clk: vo_pixel_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vo_pixelsrc_sel	0	0	RW	1'b0	Clk: vo_pixel_clk 0-clk_vpll_fout1, (594M)
							1-clk_spll2_fout2.(520M)
16'h1bc	vo_mclk_ctrl	vo_mclk_clken	31	31	RW	1'b0	Clk: vo_i2s_mclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	12	RO	19'b0	
		vo_mclk_divsor	11	4	RW	8'd10	Clk: clk_vo_mclk_st1 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	1	RO	3'b0	
		vo_mclk_sel	0	0	RW	1'b0	Clk: vo_i2s_mclk 0-internal mclk,
							1-external mclk
16'h1c0	vo_phy_clkctrl	vo_cec_clk_divsor	31	16	RW	16'd18128	Clk: vo_cec_clk 0,1,2:2 divisior
							……
							n:n divisior

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							cec clock dividor,cec clock must set to 32.760K~32.769K
		reserved	15	3	RO	13'd0	
		vo_cec_sel	2	2	RW	1'b0	not used
		vo_cr_clken	1	1	RW	1'b0	Clk: vo_cr_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		vo_txesc_clken	0	0	RW	1'b0	Clk: vo_phy_txclkesc clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h1c4	vc_aclk_ctrl	vc_aclk_clken	31	31	RW	1'b0	Clk: vc_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		vc_aclk_divsor	7	4	RW	4'd2	Clk: vc_aclk 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		vcaclk_root_sel	0	0	RW	1'b1	Clk: vc_aclk 0-clk_spll0_fout1, (800M)

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1-clk_spll2_fout1.(1040M)
16'h1c8	vcdec_rootclk_ctrl	reserved	31	1	RO	31'b0	
		vcdec_root_sel	0	0	RW	1'b0	Clk: clk_vc_cdec_root 0-clk_spll0_fout1, (800M)
							1-clk_spll2_fout1.(1040M)
16'h1cc	g2d_ctrl	g2d_aclken	31	31	RW	1'b0	Clk: g2d_aclk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		g2d_clken	30	30	RW	1'b0	Clk: g2d_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	29	29	RO	1'b0	
		scu_g2d_pclk_en	28	28	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	27	20	RO	8'b0	
		reserved	19	8	RO	12'b0	
		g2d_divsor	7	4	RW	4'd2	Clk: clk_g2d_st1 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h1d0	vc_clken_ctrl	reserved	31	6	RO	26'b0	
		scu_vc_ve_pclk_en	5	5	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		scu_vc_vd_pclk_en	4	4	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		scu_vc_mon_pclk_en	3	3	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		scu_vc_je_pclk_en	2	2	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		scu_vc_jd_pclk_en	1	1	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		vc_cfg_clken	0	0	RW	1'b0	Clk: vc_cfg_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h1d4	je_clk_ctrl	je_clken	31	31	RW	1'b0	Clk: vc_je_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		je_aclk_divsor	7	4	RW	4'd2	Clk: vc_je_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1d8	jd_clk_ctrl	jd_clken	31	31	RW	1'b0	Clk: jd_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		jd_aclk_divsor	7	4	RW	4'd2	Clk: jd_clk 0,1,2:2 divisior
							……
							n:n divisior
		reserved	3	0	RO	4'b0	
16'h1dc	vd_clk_ctrl	vd_clken	31	31	RW	1'b0	Clk: vd_clk clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		vd_aclk_divsor	7	4	RW	4'd2	Clk: vd_clk 0,1,2:2 divisior …… n:n divisior
		reserved	3	0	RO	4'b0	
16'h1e0	ve_clk_ctrl	ve_clken	31	31	RW	1'b0	Clk: ve_clk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	20	RO	11'b0	
		reserved	19	8	RO	12'b0	
		ve_aclk_divsor	7	4	RW	4'd3	Clk: ve_clk 0,1,2:2 divisior …… n:n divisior
		reserved	3	0	RO	4'b0	
16'h1e4	aon_dma_clk_ctrl	aondma_axi_clken	31	31	RW	1'b1	Clk: aondma_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		aondma_cfg_clken	30	30	RW	1'b1	Clk: clk_aondma_cfg clock gate enable, 1'b0:clock gated 1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		aon_aclk_clken	29	29	RW	1'b1	Clk: aon_aclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	28	8	RO	21'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		aondma_divsor	7	4	RW	4'd2	Clk: clk_aondma_axi_st3 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		aondma_axi_clksel	0	0	RW	1'b1	Clk: clk_aondma_axi_st3 0-interal clk divided, (800M) 1-xtal24 MHz clock.
16'h1e8	timer_clk_ctrl	reserved	31	9	RO	23'b0	
		timer3_clk8_clken	8	8	RW	1'b1	Clk: timer3_clk8 timer3's counter7 clock from clk_vpll_fout3, which is 49.5MHz to generate 90KHz time strap for AVC.
		timer_pclk_clken	7	4	RW	4'b1111	Clk: timer_pclk[N], N=0/1/2/3 clock gate enable, 1'b0:clock gated 1'b1:clock enable
		timer_clk_clken	3	0	RW	4'b1111	Clk: timer_clk[N], N = 0/1/2/3

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
16'h1ec	rtc_clk_ctrl	aon_rtc_divsor	31	21	RW	11'd64	Clk: clk_rtc the divsor of aon rtc ip
							note:the source of aon rtc clock is cpu rtc toggle
							0,1,2:2 divisior
							……
							n:n divisior
		rtc_toggle_divsor	20	16	RW	5'd24	clk： clk_1m the divsor of cpu rtc toggle
							0,1:bypass
							2:2 divisior
							……
							n:n divisior
		reserved	15	3	RO	13'b0	
		rtc_cfg_clken	2	2	RW	1'b1	Clk: clk_rtc_cfg clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		rtc_clken	1	1	RW	1'b1	Clk: clk_rtc clock gate enable,
							1'b0:clock gated
							1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		rtc_clk_sel	0	0	RW	1'b0	not used
16'h1f0	pka_clk_ctrl	pka_clken	31	31	RW	1'b1	Clk: clk_pka_cfg clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h1f4	spacc_clk_ctrl	spacc_clken	31	31	RW	1'b1	Clk: clk_spacc_cfg clock gate enable, 1'b0:clock gated 1'b1:clock enable
		crypto_clken	30	30	RW	1'b1	Clk: clk_crypto clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	29	20	RO	10'b0	
		reserved	19	8	RO	12'b0	
		crypto_divsor	7	4	RW	4'd4	Clk: clk_crypto 0,1,2:2 divisior …… n:n divisior
		reserved	3	0	RO	4'b0	
16'h1f8	trng_clk_ctrl	trng_clken	31	31	RW	1'b1	Clk: clk_trng_cfg clock gate enable, 1'b0:clock gated

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h1fc	otp_clk_ctrl	otp_clken	31	31	RW	1'b1	Clk: clk_otp_cfg clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h200	lsp_clk_en0	scu_lsp_wdt_pclk_en	31	28	RW	4'b1111	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		scu_lsp_ssi_pclk_en	27	26	RW	2'b11	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		scu_lsp_timer_pclk_en	25	25	RW	1'b1	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	24	22	RO	3'b000	
		scu_lsp_uart_pclk_en	21	17	RW	5'h1F	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		scu_lsp_i2c_pclk_en	16	7	RW	10'h3FF	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	6	2	RO	5'd0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_lsp_pvt_pclk_en	1	1	RW	1'b1	clock gate enable, 1'b0:clock gated 1'b1:clock enable scu_lsp_pvt_pclk_en:lsp pvt pclk en
		scu_lsp_fan_pclk_en	0	0	RW	1'b1	clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h204	lsp_clk_en1	reserved	31	18	RO	14'b0	
		pvt_clken	17	16	RW	2'b11	clock gate enable, 1'b0:clock gated 1'b1:clock enable Clk: pvt_clk[0] pvt_clken[0]:lsp pvt pclk en Clk: pvt_clk[1] pvt_clken[1]:ddr pvt pclk en
		scu_lsp_mbox_pclk_en	15	0	RW	16'hffff	clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h208	u84_clk_ctrl	u84_core_clk_en	31	28	RW	4'b1111	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		u84_trace_clk_en	27	24	RW	4'b1111	clock gate enable, 1'b0:clock gated 1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		u84_trace_com_clk_en	23	23	RW	1'b1	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	22	21	RO	2'b0	
		u84_busclk_ratio_sel	20	20	RW	1'b0	0-core:bus=2:1, 1-core:bus=1:1
		reserved	19	2	RO	18'b0	
		u84_core_clk_sel	1	0	RW	2'd2	0: cpu pll fout2, 1: core_lp clock, (400M) 2: XTAL 24MHz.
16'h20c	syscfg_clk_ctrl	reserved	31	7	RO	25'b0	
		syscfg_divsor	6	4	RW	3'd2	clk: clk_sys_cfg 0,1,2:2 divisior …… n:n divisior
		reserved	3	1	RO	3'b0	
		syscfg_clk_sel	0	0	RW	1'b1	0-clk_spll0_fout3 used to clock division, (200M) 1-xtal 24MHz clock
16'h210	i2c0_clk_ctrl	i2c10_pclken	31	31	RW	1'b1	Clk: i2c10_pclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	0	RO	31'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h214	i2c1_clk_ctrl	i2c11_pclken	31	31	RW	1'b1	Clk: i2c11_pclk clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	30	0	RO	31'b0	
16'h280	dft_clk_ctrl	reserved	31	27	RO	5'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft1000m_clken	26	26	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft500m_clken	25	25	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft400m_clken	24	24	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft250m_clken	23	23	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft200m_clken	22	22	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		pcie_dft150m_clken	21	21	RW	1'b0	clock gate enable,

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b0:clock gated
							1'b1:clock enable
		pcie_dft125m_clken	20	20	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	19	14	RO	6'b0	
		d2d_dft800m_clken	13	13	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		d2d_dft1000m_clken	12	12	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	11	10	RO	2'b0	
		vo_dft600m_clken	9	9	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		vo_dft300m_clken	8	8	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable
		reserved	7	6	RO	2'b0	
		vi_dft500m_clken	5	5	RW	1'b0	clock gate enable,
							1'b0:clock gated
							1'b1:clock enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		vi_dft400m_clken	4	4	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		reserved	3	3	RO	1'b0	
		hsp_dft600m_clken	2	2	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		hsp_dft300m_clken	1	1	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
		hsp_dft150m_clken	0	0	RW	1'b0	clock gate enable, 1'b0:clock gated 1'b1:clock enable
16'h300	sys_swrst_value	sw_sysrst_value	31	0	RW	32'b0	system software reset value, system will be reset thru configing 0x1AC0FFE6.
16'h304	clr_rst_status	reserved	31	1	RO	31'b0	
		clr_rstsrc_status	0	0	WO	1'b0	clear the reset stauts, high active
16'h308	die_status	reserved	31	24	RO	8'b0	
		die_num	23	16	RO	8'b0	the quantity number of DIE in one pacakge
		die_ordinal	15	8	RO	8'b0	Die order number for current die.
		chip_rstsrc_status	7	0	RO	8'b0	chip reset source status. High active Casued by: [0]-inernal POR reset [1]-external die POR reset, from another

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							die.
							[2]-key reset
							[3]-watchdog reset
							[4]-system software reset
							[5]-external die's reset, fron die0
							[6]-u84 debug_ndreset
							[7]-scpu debug_ndreset
16'h30C	clr_boot_info	reserved	31	1	RO	31'b0	
		clr_boot_flag	0	0	WO	1'b0	the U84 or scpu could be reset thru software while boot flag cleared while boot from either.
16'h310	scpu_boot_address	scpu_reset_vector	31	0	RW	32'b0	if boot from u84,this is the scpu boot address
							if boot from scpu,this field is unuse.
16'h314	lpcpu_boot_address	lpcpu_reset_vector	31	0	RW	32'b0	lpcpu boot address
16'h318	npue31_boot_address	reserved	31	0	RO	32'b0	reserved
16'h31C	u84_boot_address0_hi	reserved	31	9	RO	23'b0	
		u84_reset_vector0_hi	8	0	RW	9'b0	if boot from scpu,this is the u84 boot address[40:32]
							if boot from u84,this field not used
16'h320	u84_boot_address0_low	u84_reset_vector0_low	31	0	RW	32'h8000_0000	if boot from scpu,this is the u84 boot addres[31:0]
							if boot from u84,this field not used
16'h324	u84_boot_address1_hi	reserved	31	9	RO	23'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		u84_reset_vector1_hi	8	0	RW	9'b0	if boot from scpu,this is the u84 boot addres[40:32] if boot from u84,this field not used
16'h328	u84_boot_address1_low	u84_reset_vector1_low	31	0	RW	32'h8000_0000	if boot from scpu,this is the u84 boot addres[31:0] if boot from u84,this field not used
16'h32C	u84_boot_address2_hi	reserved	31	9	RO	23'b0	
		u84_reset_vector2_hi	8	0	RW	9'b0	if boot from scpu,this is the u84 boot addres[40:32] if boot from u84,this field not used
16'h330	u84_boot_address2_low	u84_reset_vector2_low	31	0	RW	32'h8000_0000	if boot from scpu,this is the u84 boot addres[31:0] if boot from u84,this field not used
16'h334	u84_boot_address3_hi	reserved	31	9	RO	23'b0	
		u84_reset_vector3_hi	8	0	RW	9'b0	if boot from scpu,this is the u84 boot addres[40:32] if boot from u84,this field not used
16'h338	u84_boot_address3_low	u84_reset_vector3_low	31	0	RW	32'h8000_0000	if boot from scpu,this is the u84 boot addres[31:0] if boot from u84,this field not used
16'h33C	boot_sel_stat	reserved	31	4	RO	28'b0	
		boot_sel_status	3	0	RO	4'b0	
16'h340	boot_spi_cfg	reserved	31	1	RO	31'b0	
		spi_boot_mode	0	0	RW	1'b0	1'b0:enable spi controller boot mode,boot mode dicide by boot sel pin

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:disable spi controller boot mode
16'h400	snoc_rst_ctrl	reserved	31	17	RO	15'b0	
		sw_snoc_aon_arst_n	16	16	RW	1'b1	snoc aon aclk domain sw reset
		sw_snoc_d2d_arst_n	15	15	RW	1'b1	snoc d2d aclk domain sw reset
		sw_snoc_ddrc0_p1_arst_n	14	14	RW	1'b1	snoc ddr0 port1 aclk domain sw reset
		sw_snoc_ddrc0_p2_arst_n	13	13	RW	1'b1	snoc ddr0 port2 aclk domain sw reset
		sw_snoc_ddrc1_p1_arst_n	12	12	RW	1'b1	snoc ddr1 port1 aclk domain sw reset
		sw_snoc_ddrc1_p2_arst_n	11	11	RW	1'b1	snoc ddr1 port2 aclk domain sw reset
		sw_snoc_dspt_arst_n	10	10	RW	1'b1	snoc dsp aclk domain sw reset
		sw_snoc_jtag_prst_n	9	9	RW	1'b1	not used.
		sw_snoc_npu_arst_n	8	8	RW	1'b1	snoc npu aclk domain sw reset
		sw_snoc_pciet_prst_n	7	7	RW	1'b1	snoc pcie pclk domain sw reset
		sw_snoc_pciet_xmrst_n	6	6	RW	1'b1	snoc pcie aclk domain xm sw reset
		sw_snoc_pciet_xsrst_n	5	5	RW	1'b1	snoc pcie pclk domain xs sw reset
		sw_snoc_u84_arst_n	4	4	RW	1'b1	snoc u84 aclk domain sw reset
		sw_snoc_tcu_arst_n	3	3	RW	1'b1	snoc tcu aclk domain sw reset
		sw_rnoc_nsp_rstn	2	2	RW	1'b1	snoc rnoc nsp logic sw reset
		sw_noc_cfg_rstn	1	1	RW	1'b1	not used,only reset at powerup
		sw_noc_nsp_rstn	0	0	RW	1'b1	snoc mnoc nsp logic sw reset

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h404	gpu_rst_ctrl	reserved	31	5	RO	27'b0	
		sw_gpu_spu_rstn	4	4	RW	1'b0	
		sw_gpu_jones_rstn	3	3	RW	1'b0	
		sw_gpu_gray_rstn	2	2	RW	1'b0	
		sw_gpu_cfg_rstn	1	1	RW	1'b0	
		sw_gpu_axi_rstn	0	0	RW	1'b0	
16'h408	dsp_rst_ctrl	reserved	31	8	RO	24'b0	
		sw_dsp_div_rstn	7	4	RW	4'b0	For reset the divisior in dsp
		reserved	3	3	RO	1'b0	
		sw_dsp_div4_rstn	2	2	RW	1'b0	For reset the division in dsp, at the same time, reset the apb slave and dsp csr for each dsp core
		sw_dsp_cfg_rstn	1	1	RW	1'b0	
		sw_dsp_axi_rstn	0	0	RW	1'b0	
16'h40c	d2d_rst_ctrl	reserved	31	8	RO	24'b0	
		sw_d2d_subsys_rst_n	7	7	RW	1'b1	reset d2d subsys
		sw_d2d_core_rst_n	6	6	RW	1'b0	
		sw_d2d_tx_rst_n	5	5	RW	1'b0	
		sw_d2d_rx_rst_n	4	4	RW	1'b0	
		sw_d2d_raw_pcs_rst_n	3	3	RW	1'b0	
		sw_d2d_prst_n	2	2	RW	1'b0	
		sw_d2d_cfg_rstn	1	1	RW	1'b0	
		sw_d2d_axi_rstn	0	0	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h410	ddr_rst_ctrl	reserved	31	27	RO	5'b0	
		sw_ddr_prstn	26	26	RW	1'b0	
		reserved	25	25	RO	1'b0	
		sw_ddr0_pmp_soft_rstn	24	24	RW	1'b0	not used, discard csr
		sw_ddr_core_rstn	23	23	RW	1'b0	
		sw_ddr0_trace_arstn	22	22	RW	1'b0	
		sw_ddr_cfg_rstn	21	21	RW	1'b0	
		sw_ddr0_p_arstn	20	16	RW	5'b00000	
		reserved	15	11	RO	5'b0	
		reserved	10	10	RO	1'b0	
		reserved	9	9	RO	1'b0	
		sw_ddr1_pmp_soft_rstn	8	8	RW	1'b0	not used,discard csr
		reserved	7	7	RO	1'b0	
		sw_ddr1_trace_arstn	6	6	RW	1'b0	
		reserved	5	5	RO	1'b0	
		sw_ddr1_p_arstn	4	0	RW	5'b00000	
16'h414	tcu_rst_ctrl	reserved	31	21	RO	11'b0	
		tbu_rstn	20	4	RW	17'b0	Reserved.
		reserved	3	2	RO	1'b0	
		sw_tcu_cfg_rstn	1	1	RW	1'b1	
		sw_tcu_axi_rstn	0	0	RW	1'b1	
16'h418	npu_rst_ctrl	reserved	31	7	RO	25'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_npu_llc_rstn	6	6	RW	1'b0	
		sw_npu_e31dbg_rstn	5	5	RW	1'b0	reserved
		sw_npu_e31bus_rstn	4	4	RW	1'b0	reserved
		sw_npu_e31core_rstn	3	3	RW	1'b0	
		sw_npu_core_rstn	2	2	RW	1'b0	
		sw_npu_cfg_rstn	1	1	RW	1'b0	
		sw_npu_axi_rstn	0	0	RW	1'b0	
16'h41c	hspdma_rst_ctrl	reserved	31	28	RO	4'b0	
		sw_hsp_sata_arstn	27	27	RW	1'b0	
		sw_hsp_eth0_arstn	26	26	RW	1'b0	
		sw_hsp_eth1_arstn	25	25	RW	1'b0	
		sw_hsp_dma_arstn	24	24	RW	1'b0	
		sw_hsp_emmc_arstn	23	23	RW	1'b0	
		sw_hsp_sd0_arstn	22	22	RW	1'b0	
		sw_hsp_sd1_arstn	21	21	RW	1'b0	
		sw_hsp_dma_prstn	20	20	RW	1'b0	
		sw_hsp_emmc_prstn	19	19	RW	1'b0	
		sw_hsp_sd0_prstn	18	18	RW	1'b0	
		sw_hsp_sd1_prstn	17	17	RW	1'b0	
		sw_usb1_vaux_rstn	16	16	RW	1'b0	
		sw_usb0_vaux_rstn	15	15	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_hsp_dma0_rstn	14	14	RW	1'b0	
		sw_dma0_rst_n	13	13	RW	1'b0	
		sw_sata_rbc_rstn	12	12	RW	1'b0	not used,reserved
		sw_sata_pmalive_rstn	11	11	RW	1'b0	
		sw_sata_oob_rstn	10	10	RW	1'b0	
		sw_sata_asic0_rstn	9	9	RW	1'b0	not used,reserved
		sw_mshc_txrx_rstn	8	6	RW	3'b0	
		sw_mshc_phy_rstn	5	3	RW	3'b0	
		sw_hsp_por_rstn	2	2	RW	1'b0	
		sw_hsp_cfg_rstn	1	1	RW	1'b0	
		sw_hsp_axi_rstn	0	0	RW	1'b0	the hsp xbar axi reset
16'h420	pcie_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_pcie_perst_n	2	2	RW	1'b0	
		sw_pcie_powerup_rstn	1	1	RW	1'b0	
		sw_pcie_cfg_rstn	0	0	RW	1'b0	
16'h424	i2c_rst_ctrl	reserved	31	10	RO	22'b0	
		sw_i2c_rst_n	9	0	RW	10'b0	
16'h428	fan_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_fan_rst_n	0	0	RW	1'b1	
16'h42C	pvt_rst_ctrl	reserved	31	2	RO	30'd0	
		sw_pvt_rst_n	1	0	RW	2'b11	sw_pvt_rst_n[0]:lsp pvt reset
							sw_pvt_rst_n[1]:ddr pvt reset

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h430	mbox_rst_ctrl	reserved	31	16	RO	16'b0	
		sw_mbox_rst_n	15	0	RW	16'hffff	
16'h434	uart_rst_ctrl	reserved	31	5	RO	27'b0	
		sw_uart_rst_n	4	0	RW	5'b11111	
16'h43C	timer_rst_ctrl	reserved	31	1	RO	31'b0	timer in LSP
		sw_timer_rst_n	0	0	RW	1'b1	
16'h440	ssi_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_ssi_rst_n	1	0	RW	2'b11	
16'h444	wdt_rst_ctrl	reserved	31	4	RO	28'b0	
		sw_wdt_rst_n	3	0	RW	4'b1111	
16'h448	lsp_cfgrst_ctrl	reserved	31	1	RO	31'b0	
		sw_lsp_cfg_rstn	0	0	RW	1'b1	
16'h44c	u84_rst_ctrl	reserved	31	12	RO	20'b0	
		sw_u84_trace_rstn	11	8	RW	4'b0	
		reserved	7	7	RO	1'b0	
		sw_u84_tracecom_rstn	6	6	RW	1'b0	
		sw_u84_dbg_rstn	5	5	RW	1'b0	
		sw_u84_bus_rstn	4	4	RW	1'b0	
		sw_u84_core_rstn	3	0	RW	4'b0	
16'h450	scpu_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_scpu_dbg_rstn	2	2	RW	1'b0	
		sw_scpu_bus_rstn	1	1	RW	1'b0	
		sw_scpu_core_rstn	0	0	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h454	lpcpu_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_lpcpu_dbg_rstn	2	2	RW	1'b0	
		sw_lpcpu_bus_rstn	1	1	RW	1'b0	
		sw_lpcpu_core_rstn	0	0	RW	1'b0	
16'h458	vc_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_vc_moncfg_rstn	2	2	RW	1'b0	
		sw_vc_axi_rstn	1	1	RW	1'b0	
		sw_vc_cfg_rstn	0	0	RW	1'b0	
16'h45C	jd_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_jd_axi_rstn	1	1	RW	1'b0	
		sw_jd_cfg_rstn	0	0	RW	1'b0	
16'h460	je_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_je_axi_rstn	1	1	RW	1'b0	
		sw_je_cfg_rstn	0	0	RW	1'b0	
16'h464	vd_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_vd_axi_rstn	1	1	RW	1'b0	
		sw_vd_cfg_rstn	0	0	RW	1'b0	
16'h468	ve_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_ve_axi_rstn	1	1	RW	1'b0	
		sw_ve_cfg_rstn	0	0	RW	1'b0	
16'h46C	g2d_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_g2d_axi_rstn	2	2	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_g2d_cfg_rstn	1	1	RW	1'b0	
		sw_g2d_core_rstn	0	0	RW	1'b0	
16'h470	vi_rst_ctrl	reserved	31	3	RO	29'b0	
		sw_vi_dwe_rstn	2	2	RW	1'b0	
		sw_vi_cfg_rstn	1	1	RW	1'b0	
		sw_vi_axi_rstn	0	0	RW	1'b0	
16'h474	dvp_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_vi_dvp_rstn	0	0	RW	1'b0	
16'h478	isp0_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_vi_isp0_rstn	0	0	RW	1'b0	
16'h47C	isp1_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_vi_isp1_rstn	0	0	RW	1'b0	
16'h480	shutter_rst_ctrl	reserved	31	6	RO	26'b0	
		sw_vi_shutter_rstn	5	0	RW	6'b0	
16'h484	vo_phyrst_ctrl	reserved	31	6	RO	26'b0	
		sw_vo_hdmi_rstn	5	5	RW	1'b0	
		sw_hdmi_phyctrl_rstn	4	4	RW	1'b0	
		sw_vo_hdmi_prstn	3	3	RW	1'b0	
		reserved	2	2	RO	1'b0	
		sw_vo_prstn	1	1	RW	1'b0	
		sw_vo_mipi_prstn	0	0	RW	1'b0	
16'h488	vo_i2srst_ctrl	reserved	31	2	RO	30'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_vo_i2s_prstn	1	1	RW	1'b0	
		sw_vo_i2s_rstn	0	0	RW	1'b0	
16'h48C	vo_rst_ctrl	reserved	31	4	RO	28'b0	
		sw_vo_dc_prstn	3	3	RW	1'b0	
		sw_vo_dc_rstn	2	2	RW	1'b0	
		sw_vo_cfg_rstn	1	1	RW	1'b0	
		sw_vo_axi_rstn	0	0	RW	1'b0	
16'h490	bootspi_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_bootspi_rstn	1	1	RW	1'b1	
		sw_bootspi_hrstn	0	0	RW	1'b1	
16'h494	i2c1_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_i2c1_prstn	0	0	RW	1'b1	
16'h498	i2c0_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_i2c0_prstn	0	0	RW	1'b1	
16'h49C	dma1_rst_ctrl	reserved	31	2	RO	31'b0	
		sw_dma1_hrstn	1	1	RW	1'b1	
		sw_dma1_arstn	0	0	RW	1'b1	
16'h4A0	fprt_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_fp_prt_hrstn	0	0	RW	1'b1	
16'h4A4	hblock_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_hblock_hrstn	0	0	RW	1'b1	
16'h4A8	secsr_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_secsr_hrstn	0	0	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h4AC	otp_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_otp_prstn	0	0	RW	1'b1	reserved
16'h4B0	pka_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_pka_hrstn	0	0	RW	1'b1	
16'h4B4	spacc_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_spacc_rstn	0	0	RW	1'b1	
16'h4B8	trng_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_trng_hrstn	0	0	RW	1'b1	
16'h4C0	timer0_rst_ctrl	reserved	31	9	RO	23'b0	
		sw_timer0_prstn	8	8	RW	1'b1	
		sw_timer0_rstn	7	0	RW	8'hff	
16'h4C4	timer1_rst_ctrl	reserved	31	9	RO	23'b0	
		sw_timer1_prstn	8	8	RW	1'b0	
		sw_timer1_rstn	7	0	RW	8'b0	
16'h4C8	timer2_rst_ctrl	reserved	31	9	RO	23'b0	
		sw_timer2_prstn	8	8	RW	1'b0	
		sw_timer2_rstn	7	0	RW	8'b0	
16'h4cc	timer3_rst_ctrl	reserved	31	9	RO	23'b0	
		sw_timer3_prstn	8	8	RW	1'b0	
		sw_timer3_rstn	7	0	RW	8'b0	
16'h4D0	rtc_rst_ctrl	reserved	31	1	RO	31'b0	
		sw_rtc_rstn	0	0	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h4D4	mnoc_rst_ctrl	reserved	31	7	RO	25'b0	
		sw_mnoc_ddrc0_p3_arst_n	6	6	RW	1'b1	
		sw_mnoc_ddrc1_p3_arst_n	5	5	RW	1'b1	
		sw_mnoc_gpu_arst_n	4	4	RW	1'b1	
		sw_mnoc_hsp_arst_n	3	3	RW	1'b1	
		sw_mnoc_cfg_rst_n	2	2	RW	1'b1	
		sw_mnoc_vc_arst_n	1	1	RW	1'b1	
		sw_mnoc_snoc_nsp_rst_n	0	0	RW	1'b1	mnoc nsp logic sw reset
16'h4D8	rnoc_rst_ctrl	reserved	31	6	RO	29'b0	
		sw_rnoc_ddrc0_p4_arst_n	5	5	RW	1'b1	
		sw_rnoc_ddrc1_p4_arst_n	4	4	RW	1'b1	
		sw_rnoc_cfg_rst_n	3	3	RW	1'b1	
		sw_rnoc_snoc_nsp_rst_n	2	2	RW	1'b1	rnoc nsp logic sw reset
		sw_rnoc_vi_arst_n	1	1	RW	1'b1	
		sw_rnoc_vo_arst_n	0	0	RW	1'b1	
16'h4DC	cnoc_rst_ctrl	reserved	31	16	RO	16'b0	
		sw_cnoc_aon_cfg_rst_n	15	15	RW	1'b1	
		sw_cnoc_clmm_cfg_rst_n	14	14	RW	1'b1	
		sw_cnoc_cfg_rst_n	13	13	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_cnoc_d2d_cfg_rst_n	12	12	RW	1'b1	
		sw_cnoc_ddrt0_cfg_rst_n	11	11	RW	1'b1	
		sw_cnoc_ddrt1_cfg_rst_n	10	10	RW	1'b1	
		sw_cnoc_dspt_cfg_rst_n	9	9	RW	1'b1	
		sw_cnoc_gpu_cfg_rst_n	8	8	RW	1'b1	
		sw_cnoc_hsp_cfg_rst_n	7	7	RW	1'b1	
		sw_cnoc_lsp_cfg_rst_n	6	6	RW	1'b1	
		sw_cnoc_npu_cfg_rst_n	5	5	RW	1'b1	
		sw_cnoc_pciet_cfg_rst_n	4	4	RW	1'b1	
		sw_cnoc_tcu_cfg_rst_n	3	3	RW	1'b1	
		sw_cnoc_vc_cfg_rst_n	2	2	RW	1'b1	
		sw_cnoc_vi_cfg_rst_n	1	1	RW	1'b1	
		sw_cnoc_vo_cfg_rst_n	0	0	RW	1'b1	
16'h4E0	lnoc_rst_ctrl	reserved	31	4	RO	28'b0	
		sw_lnoc_ddrc0_p0_arst_n	3	3	RW	1'b1	
		sw_lnoc_ddrc1_p0_arst_n	2	2	RW	1'b1	
		sw_lnoc_npu_llc_arst_n	1	1	RW	1'b1	
		sw_lnoc_cfg_rst_n	0	0	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h4E4	pipe_rst_ctrl	reserved	31	25	RO	9'b0	
		sw_vc2mnoc_pipe_rst_n	24	24	RW	1'b1	top pipe cell reset
		sw_gpu2mnoc_pipe_rst_n	23	23	RW	1'b1	top pipe cell reset
		sw_vo2rnoc_pipe_rst_n	22	22	RW	1'b1	top pipe cell reset
		sw_vi2rnoc_pipe_rst_n	21	21	RW	1'b1	top pipe cell reset
		sw_pcie2tcu_pipe_rst_n	20	20	RW	1'b1	top pipe cell reset
		sw_dsp2tcu_pipe_rst_n	19	19	RW	1'b1	top pipe cell reset
		sw_dsp2snoc_pipe_rst_n	18	18	RW	1'b1	top pipe cell reset
		sw_snoc2dsp_pipe_rst_n	17	17	RW	1'b1	top pipe cell reset
		sw_mnoc2ddr0_pipe_rst_n	16	16	RW	1'b1	top pipe cell reset
		sw_rnoc2ddr0_pipe_rst_n	15	15	RW	1'b1	top pipe cell reset
		sw_rnoc2ddr1_pipe_rst_n	14	14	RW	1'b1	top pipe cell reset
		sw_lnoc2ddr0_pipe_rst_n	13	13	RW	1'b1	top pipe cell reset
		sw_lnoc2ddr1_pipe_rst_n	12	12	RW	1'b1	top pipe cell reset
		sw_mnoc2snoc_pipe_rst_n	11	11	RW	1'b1	top pipe cell reset
		sw_MCPU2snoc_sp_pipe_rst_n	10	10	RW	1'b1	top pipe cell reset
		sw_MCPU2snoc_pipe_rst_n	9	9	RW	1'b1	top pipe cell reset
		sw_snoc2d2d_pipe_rst_n	8	8	RW	1'b1	top pipe cell reset

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		sw_snoc2ddr0_p2_pipe_rst_n	7	7	RW	1'b1	top pipe cell reset
		sw_snoc2ddr1_p2_pipe_rst_n	6	6	RW	1'b1	top pipe cell reset
		sw_snoc2ddr0_p1_pipe_rst_n	5	5	RW	1'b1	top pipe cell reset
		sw_snoc2ddr1_p1_pipe_rst_n	4	4	RW	1'b1	top pipe cell reset
		sw_snoc2mnoc_pipe_rst_n	3	3	RW	1'b1	top pipe cell reset
		sw_snoc2pcie_pipe_rst_n	2	2	RW	1'b1	top pipe cell reset
		sw_tbu2mnoc_pipe_rst_n	1	1	RW	1'b1	top pipe cell reset
		sw_tbu2snoc_pipe_rst_n	0	0	RW	1'b1	top pipe cell reset
16'h4E8	tbu_rst_ctrl	reserved	31	4	RO	28'b0	
		sw_tbu_vi_axi_rst_n	3	3	RW	1'b1	top tbu reset
		sw_tbu_hsp_arst_n	2	2	RW	1'b1	top tbu reset
		sw_tbu_pciet_tbu3_rst_n	1	1	RW	1'b1	top tbu reset
		sw_tbu_aon_arst_n	0	0	RW	1'b1	top tbu reset
16'h4Ec	reset_status	reserved	31	1	RO	28'b0	
		ext_d2d_subsys_rstn_status	0	0	RO	1'b1	d2d subsys reset status of the opposite die
16'h4F0	test_rst_ctrl	reserved	31	2	RO	30'b0	
		sw_spi_slv_rstn	1	1	RW	1'b1	spi slave module spi clock domain reset
		sw_testmux_rstn	0	0	RW	1'b1	testmux reset

3.3 System control register

The system control registers include JTAG bypass selection, interrupt selection, timer enablement, etc.

Address space: 0x5181_0000~0x5181_7FFF, 32k

Table 3-3 System Register Description

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h0000	dynm_csr_en	Reserved	31	4	RO	28'd0	reserved
		mcpu_sp0_dynm_csr_en	3	3	RW	1'b0	
		mcpu_sp1_dynm_csr_en	2	2	RW	1'b0	
		mcpu_mp_dynm_csr_en	1	1	RW	1'b0	
		mcpu_d2d_dynm_csr_en	0	0	RW	1'b0	
16'h0004	dynm_csr_gnt	Reserved	31	4	RO	28'd0	
		mcpu_sp0_dynm_csr_gnt	3	3	RO	1'b0	
		mcpu_sp1_dynm_csr_gnt	2	2	RO	1'b0	
		mcpu_mp_dynm_csr_gnt	1	1	RO	1'b0	
		mcpu_d2d_dynm_csr_gnt	0	0	RO	1'b0	
16'h0008	cpu_qos	scu_cpu_sp1_arqos	31	28	RW	4'h0	
		scu_cpu_sp1_awqos	27	24	RW	4'h0	
		scu_cpu_sp0_arqos	23	20	RW	4'h0	
		scu_cpu_sp0_awqos	19	16	RW	4'h0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_cpu_mp_arqos	15	12	RW	4'h0	
		scu_cpu_mp_awqos	11	8	RW	4'h0	
		scu_d2d_arqos	7	4	RW	4'h0	not used
		scu_d2d_awqos	3	0	RW	4'h0	not used
16'h0324	noc_cfg0	Reserved	31	17	RO	15'd0	
		rnoc_ddrt1_p4_wd_en	16	16	RW	1'b0	noc watch dog enable
		rnoc_ddrt0_p4_wd_en	15	15	RW	1'b0	noc watch dog enable
		mnoc_ddrt1_p3_wd_en	14	14	RW	1'b0	noc watch dog enable
		mnoc_ddrt0_p3_wd_en	13	13	RW	1'b0	noc watch dog enable
		snoc_pciet_wd_en	12	12	RW	1'b0	noc watch dog enable
		snoc_npu_wd_en	11	11	RW	1'b0	noc watch dog enable
		reserved	10	10	RO	1'b0	noc watch dog enable
		snoc_dspt_wd_en	9	9	RW	1'b0	noc watch dog enable
		snoc_ddrt1_p2_wd_en	8	8	RW	1'b0	noc watch dog enable
		snoc_ddrt1_p1_wd_en	7	7	RW	1'b0	noc watch dog enable
		snoc_ddrt0_p2_wd_en	6	6	RW	1'b0	noc watch dog enable
		snoc_ddrt0_p1_wd_en	5	5	RW	1'b0	noc watch dog enable
		reserved	4	4	RO	1'b0	noc watch dog enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		snoc_aon_wd_en	3	3	RW	1'b0	noc watch dog enable
		lnoc_ddrt1_p0_wd_en	2	2	RW	1'b0	noc watch dog enable
		lnoc_ddrt0_p0_wd_en	1	1	RW	1'b0	noc watch dog enable
		scu_npu_noc_interlv_en	0	0	RW	1'b0	npu noc interleave enable
16'h0328	noc_cfg1	Reserved	31	21	RO	12'd0	
		cnoc_vo_wd_en	20	20	RW	1'b0	noc watch dog enable
		cnoc_vi_wd_en	19	19	RW	1'b0	noc watch dog enable
		cnoc_vc_wd_en	18	18	RW	1'b0	noc watch dog enable
		cnoc_tcu_wd_en	17	17	RW	1'b0	noc watch dog enable
		cnoc_pciet_x_wd_en	16	16	RW	1'b0	noc watch dog enable
		cnoc_pciet_p_wd_en	15	15	RW	1'b0	noc watch dog enable
		cnoc_npu_wd_en	14	14	RW	1'b0	noc watch dog enable
		cnoc_mcput_d2d_wd_en	13	13	RW	1'b0	noc watch dog enable
		cnoc_lsp_apb6_wd_en	12	12	RW	1'b0	noc watch dog enable
		cnoc_lsp_apb4_wd_en	11	11	RW	1'b0	noc watch dog enable
		cnoc_lsp_apb3_wd_en	10	10	RW	1'b0	noc watch dog enable
		cnoc_lsp_apb2_wd_en	9	9	RW	1'b0	noc watch dog enable
		cnoc_hsp_wd_en	8	8	RW	1'b0	noc watch dog enable

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		cnoc_gpu_wd_en	7	7	RW	1'b0	noc watch dog enable
		cnoc_dspt_wd_en	6	6	RW	1'b0	noc watch dog enable
		cnoc_ddrt1_phy_wd_en	5	5	RW	1'b0	noc watch dog enable
		cnoc_ddrt1_ctrl_wd_en	4	4	RW	1'b0	noc watch dog enable
		cnoc_ddrt0_phy_wd_en	3	3	RW	1'b0	noc watch dog enable
		cnoc_ddrt0_ctrl_wd_en	2	2	RW	1'b0	noc watch dog enable
		cnoc_aon_wd_en	1	1	RW	1'b0	noc watch dog enable
		clmm_wd_en	0	0	RW	1'b0	noc watch dog enable
16'h0330	dsp_cfg	Reserved	31	31	RO	1'b0	
		reserved	30	27	RO	4'd0	
		Reserved	26	23	RO	4'd0	
		Reserved	22	19	RO	4'd0	
		Reserved	18	18	RO	1'b0	
		scu_dspt_tmod	17	17	RW	1'b0	not used
		scu_dspt_vdd	16	16	RW	1'b0	not used
		scu_dspt_dbgen	15	12	RW	4'd0	arm core sight signal,not used
		scu_dspt_niden	11	8	RW	4'd0	arm core sight signal,not used
		scu_dspt_spiden	7	4	RW	4'd0	arm core sight signal,not used

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_dspt_spniden	3	0	RW	4'd0	arm core sight signal,not used
16'h0354	mcput_cfg8	mcput_debug_ndreset_en	31	31	RW	1'b1	debug_ndreset reset whole chip enable, high active. And it will reset whole chip while both debug_ndreset and debug_ndreset_en high.
		mcput_debug_systemjtag_mfr_id	30	20	RW	11'h489	
		mcput_debug_systemjtag_part_number	19	4	RW	16'd0	
		mcput_debug_systemjtag_version	3	0	RW	4'd0	
16'h0368	lpcpu_cfg0	Reserved	31	31	RO	1'b0	
		lpcpu_jtag_mfr_id	30	20	RW	11'h489	
		lpcpu_jtag_part_number	19	4	RW	16'd1	
		lpcpu_jtag_version	3	0	RW	4'h0	
16'h036c	scpu_cfg	Reserved	31	31	RO	1'b0	
		scpu_jtag_mfr_id	30	20	RW	11'h489	
		scpu_jtag_part_number	19	4	RW	16'd2	
		scpu_jtag_version	3	0	RW	4'h0	
16'h0370	dma_cfg	Reserved	31	8	RO	24'd0	
		scu_lsp_i2c1_dma_sel	7	7	RW	1'b0	1'b0:sel dma0 1'b1:sel dma1
		scu_lsp_i2c0_dma_sel	6	6	RW	1'b0	1'b0:sel dma0

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1'b1:sel dma1
		scu_lsp_uart0_dma_sel	5	5	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
		scu_lsp_ssi1_dma_sel	4	4	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
		scu_lsp_ssi0_dma_sel	3	3	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
		scu_vo_i2s2_dma_sel	2	2	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
		scu_vo_i2s1_dma_sel	1	1	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
		scu_vo_i2s0_dma_sel	0	0	RW	1'b0	1'b0:sel dma0
							1'b1:sel dma1
16'h03b8	pcie_cfg	Reserved	31	7	RO	25'd0	reserved
		scu_pciet_app_flr_pf_done	6	6	RW	1'b0	
		scu_pciet_slv_awmisc_info_nw	5	5	RW	1'b0	
		scu_pciet_slv_awmisc_info_ats	4	3	RW	2'b0	
		scu_pciet_slv_armisc_info_nw	2	2	RW	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_pciet_slv_armisc_info_ats	1	0	RW	2'b0	
16'h03bc	jtag_chain_ctrl	Reserved	31	6	RO	26'b0	
		scpu_jtag_bypass	5	5	RW	1'b0	not used
		vq7_jtag_bypass	4	4	RW	1'b0	VQ7 DSP JTAG bypassed, and it not connected in the JTAG chain, high active.
		reserved	3	3	RW	1'b1	
		npue21_jtag_bypass	2	2	RW	1'b1	NPU E21 JTAG bypassed, and it not connected in the JTAG chain, high active.
		u84_jtag_bypass	1	1	RW	1'b0	U84 JTAG bypassed, and it not connected in the JTAG chain, high active.
		lpcpu_jtag_bypass	0	0	RW	1'b1	LPCPU E21 JTAG bypassed, and it not connected in the JTAG chain, high active.
16'h03c0	aon_int_mux	Reserved	31	17	RO	15'b0	
		aon_i2c0_int_sel	16	16	RW	1'b1	i2c0 interrupt destination selection. 0- interrupt routed to, U84 1- interrupt routed to LPCPU
		aon_i2c1_int_sel	15	15	RW	1'b1	i2c1 interrupt destination selection. 0- interrupt routed to, U84 1- interrupt routed to LPCPU
		aon_rtc_int_sel	14	13	RW	2'b01	RTC interrupt destination selection.

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							2-interrupt routed to SCPU
							GPIO interrupt destination selection.
		aon_gpio_int_sel	12	12	RW	1'b1	0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							bootspi interrupt destination selection.
		aon_spi_int_sel	11	10	RW	2'b0	0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							2/3: interrupt routed to SCPU
							DMA in aon interrupt destination selection.
		aon_dma_int_sel	9	9	RW	1'b0	0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							DDR MPMP interrupt destination selection.
		ddr_mpmp_int_sel	8	8	RW	1'b1	0- interrupt routed to, U84
							1- interrupt routed to SCPU
							timer0 interrupt destination selection.
		aon_timer0_int_sel	7	6	RW	2'b00	0- interrupt routed to, U84

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							1- interrupt routed to LPCPU
							2/3: interrupt routed to SCPU
		aon_timer1_int_sel	5	4	RW	2'b00	timer1 interrupt destination selection.
							0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							2/3: interrupt routed to SCPU
		aon_timer2_int_sel	3	2	RW	2'b01	timer2 interrupt destination selection.
							0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							2/3: interrupt routed to SCPU
		aon_timer3_int_sel	1	0	RW	2'b10	timer3 interrupt destination selection.
							0- interrupt routed to, U84
							1- interrupt routed to LPCPU
							2/3: interrupt routed to SCPU
16'h03CC	noc_ddrt_lowpower	reserved	31	31	RO	1'b0	
		rnoc_ddrt1_p4_cactive	30	30	RO	1'b0	
		rnoc_ddrt1_p4_csysack	29	29	RO	1'b0	
		rnoc_ddrt1_p4_csysreq	28	28	RW	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mnoc_ddrt1_p3_cactive	27	27	RO	1'b0	
		mnoc_ddrt1_p3_csysack	26	26	RO	1'b0	
		mnoc_ddrt1_p3_csysreq	25	25	RW	1'b1	
		snoc_ddrt1_p2_cactive	24	24	RO	1'b0	
		snoc_ddrt1_p2_csysack	23	23	RO	1'b0	
		snoc_ddrt1_p2_csysreq	22	22	RW	1'b1	
		snoc_ddrt1_p1_cactive	21	21	RO	1'b0	
		snoc_ddrt1_p1_csysack	20	20	RO	1'b0	
		snoc_ddrt1_p1_csysreq	19	19	RW	1'b1	
		snoc_ddrt1_p0_cactive	18	18	RO	1'b0	
		snoc_ddrt1_p0_csysack	17	17	RO	1'b0	
		snoc_ddrt1_p0_csysreq	16	16	RW	1'b1	
		reserved	15	15	RO	1'b0	
		rnoc_ddrt0_p4_cactive	14	14	RO	1'b0	
		rnoc_ddrt0_p4_csysack	13	13	RO	1'b0	
		rnoc_ddrt0_p4_csysreq	12	12	RW	1'b1	
		mnoc_ddrt0_p3_cactive	11	11	RO	1'b0	
		mnoc_ddrt0_p3_csysack	10	10	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mnoc_ddrt0_p3_csysreq	9	9	RW	1'b1	
		snoc_ddrt0_p2_cactive	8	8	RO	1'b0	
		snoc_ddrt0_p2_csysack	7	7	RO	1'b0	
		snoc_ddrt0_p2_csysreq	6	6	RW	1'b1	
		snoc_ddrt0_p1_cactive	5	5	RO	1'b0	
		snoc_ddrt0_p1_csysack	4	4	RO	1'b0	
		snoc_ddrt0_p1_csysreq	3	3	RW	1'b1	
		snoc_ddrt0_p0_cactive	2	2	RO	1'b0	
		snoc_ddrt0_p0_csysack	1	1	RO	1'b0	
		snoc_ddrt0_p0_csysreq	0	0	RW	1'b1	
16'h03D0	npu_tbu_power	Reserved	31	16	RO	16'b0	
		scu_npu_tbu_qreqn_pd	15	15	RW	1'b1	
		scu_npu_tbu_qacceptn_pd	14	14	RO	1'b0	
		scu_npu_tbu_qdeny_pd	13	13	RO	1'b0	
		scu_npu_tbu_qactive_pd	12	12	RO	1'b0	
		scu_npu_tbu_qreqn_cg	11	11	RW	1'b1	
		scu_npu_tbu_qacceptn_cg	10	10	RO	1'b0	
		scu_npu_tbu_qdeny_cg	9	9	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_npu_tbu_qactive_cg	8	8	RO	1'b0	
		reserved	7	0	RO	2'b0	
16'h03D4	vc_tlb_qchanel	vc_vd_tlb_qreqn_pd	31	31	RW	1'b1	
		vc_vd_tlb_qacceptn_pd	30	30	RO	1'b0	
		vc_vd_tlb_qdeny_pd	29	29	RO	1'b0	
		vc_vd_tlb_qactive_pd	28	28	RO	1'b0	
		vc_vd_tlb_qreqn_cg	27	27	RW	1'b1	
		vc_vd_tlb_qacceptn_cg	26	26	RO	1'b0	
		vc_vd_tlb_qdeny_cg	25	25	RO	1'b0	
		vc_vd_tlb_qactive_cg	24	24	RO	1'b0	
		vc_ve_tlb_qreqn_pd	23	23	RW	1'b1	
		vc_ve_tlb_qacceptn_pd	22	22	RO	1'b0	
		vc_ve_tlb_qdeny_pd	21	21	RO	1'b0	
		vc_ve_tlb_qactive_pd	20	20	RO	1'b0	
		vc_ve_tlb_qreqn_cg	19	19	RW	1'b1	
		vc_ve_tlb_qacceptn_cg	18	18	RO	1'b0	
		vc_ve_tlb_qdeny_cg	17	17	RO	1'b0	
		vc_ve_tlb_qactive_cg	16	16	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		vc_jd_tlb_qreqn_pd	15	15	RW	1'b1	
		vc_jd_tlb_qacceptn_pd	14	14	RO	1'b0	
		vc_jd_tlb_qdeny_pd	13	13	RO	1'b0	
		vc_jd_tlb_qactive_pd	12	12	RO	1'b0	
		vc_jd_tlb_qreqn_cg	11	11	RW	1'b1	
		vc_jd_tlb_qacceptn_cg	10	10	RO	1'b0	
		vc_jd_tlb_qdeny_cg	9	9	RO	1'b0	
		vc_jd_tlb_qactive_cg	8	8	RO	1'b0	
		vc_je_tlb_qreqn_pd	7	7	RW	1'b1	
		vc_je_tlb_qacceptn_pd	6	6	RO	1'b0	
		vc_je_tlb_qdeny_pd	5	5	RO	1'b0	
		vc_je_tlb_qactive_pd	4	4	RO	1'b0	
		vc_je_tlb_qreqn_cg	3	3	RW	1'b1	
		vc_je_tlb_qacceptn_cg	2	2	RO	1'b0	
		vc_je_tlb_qdeny_cg	1	1	RO	1'b0	
		vc_je_tlb_qactive_cg	0	0	RO	1'b0	
16'h03D8	tbu0234_qchanel	tbu4_qreqn_pd	31	31	RW	1'b1	
		tbu4_qacceptn_pd	30	30	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		tbu4_qdeny_pd	29	29	RO	1'b0	
		tbu4_qactive_pd	28	28	RO	1'b1	
		tbu4_qreqn_cg	27	27	RW	1'b1	
		tbu4_qacceptn_cg	26	26	RO	1'b0	
		tbu4_qdeny_cg	25	25	RO	1'b0	
		tbu4_qactive_cg	24	24	RO	1'b1	
		tbu3_qreqn_pd	23	23	RW	1'b1	
		tbu3_qacceptn_pd	22	22	RO	1'b0	
		tbu3_qdeny_pd	21	21	RO	1'b0	
		tbu3_qactive_pd	20	20	RO	1'b1	
		tbu3_qreqn_cg	19	19	RW	1'b1	
		tbu3_qacceptn_cg	18	18	RO	1'b0	
		tbu3_qdeny_cg	17	17	RO	1'b0	
		tbu3_qactive_cg	16	16	RO	1'b1	
		tbu2_qreqn_pd	15	15	RW	1'b1	
		tbu2_qacceptn_pd	14	14	RO	1'b0	
		tbu2_qdeny_pd	13	13	RO	1'b0	
		tbu2_qactive_pd	12	12	RO	1'b1	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		tbu2_qreqn_cg	11	11	RW	1'b1	
		tbu2_qacceptn_cg	10	10	RO	1'b0	
		tbu2_qdeny_cg	9	9	RO	1'b0	
		tbu2_qactive_cg	8	8	RO	1'b1	
		tbu0_qreqn_pd	7	7	RW	1'b1	
		tbu0_qacceptn_pd	6	6	RO	1'b0	
		tbu0_qdeny_pd	5	5	RO	1'b0	
		tbu0_qactive_pd	4	4	RO	1'b1	
		tbu0_qreqn_cg	3	3	RW	1'b1	
		tbu0_qacceptn_cg	2	2	RO	1'b0	
		tbu0_qdeny_cg	1	1	RO	1'b0	
		tbu0_qactive_cg	0	0	RO	1'b1	
16'h03DC	tbu7_tcu_qchannel	reserved	31	8	RO	24'd0	
		tcu_qreqn_pd	7	7	RW	1'b1	Attention: minimum value 1, it will be 1 in hardware actually while set to 0.
							[7:0]: MMU SSID
							[15:0]: MMU SID.
		tcu_qacceptn_pd	6	6	RO	1'b0	similar as above.

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		tcu_qdeny_pd	5	5	RO	1'b0	
		tcu_qactive_pd	4	4	RO	1'b1	
		tcu_qreqn_cg	3	3	RW	1'b1	
		tcu_qacceptn_cg	2	2	RO	1'b0	
		tcu_qdeny_cg	1	1	RO	1'b0	
		tcu_qactive_cg	0	0	RO	1'b1	
16'h03F0	ddrt_trace_ctrl	reserved	31	4	RO	28'b0	
		ddrt0_scu_trace_trigger	3	3	RW	1'b0	
		ddrt0_scu_trace_finish	2	2	RO	1'b0	
		ddrt1_scu_trace_trigger	1	1	RW	1'b0	
		ddrt1_scu_trace_finish	0	0	RO	1'b0	
16'h03F4	ddrt01_axi_lp	Reserved	31	8	RO	24'b0	
		ddrt1_csysdeny	7	7	RO	1'b0	
		ddrt1_cactive	6	6	RO	1'b0	
		ddrt1_csysack	5	5	RO	1'b0	
		ddrt1_csysreq	4	4	RW	1'b1	
		ddrt0_csysdeny	3	3	RO	1'b0	
		ddrt0_cactive	2	2	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		ddrt0_csysack	1	1	RO	1'b0	
		ddrt0_csysreq	0	0	RW	1'b1	
16'h03F8	dsp_tbu_lp	dsp_reserved	31	31	RW	1'b1	
		dsp_reserved	30	30	RO	1'b0	
		dsp_reserved	29	29	RO	1'b0	
		dsp_reserved	28	28	RO	1'b1	
		dsp_reserved	27	27	RW	1'b1	
		dsp_reserved	26	26	RO	1'b0	
		dsp_reserved	25	25	RO	1'b0	
		dsp_reserved	24	24	RO	1'b1	
		dsp_reserved	23	23	RW	1'b1	
		dsp_reserved	22	22	RO	1'b0	
		dsp_reserved	21	21	RO	1'b0	
		dsp_reserved	20	20	RO	1'b1	
		dsp_reserved	19	19	RW	1'b1	
		dsp_reserved	18	18	RO	1'b0	
		dsp_reserved	17	17	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		dsp_reserved	16	16	RO	1'b1	
		dsp1_tbu_qreqn_pd	15	15	RW	1'b1	
		dsp1_tbu_qacceptn_pd	14	14	RO	1'b0	
		dsp1_tbu_qdeny_pd	13	13	RO	1'b0	
		dsp1_tbu_qactive_pd	12	12	RO	1'b1	
		dsp1_tbu_qreqn_cg	11	11	RW	1'b1	
		dsp1_tbu_qacceptn_cg	10	10	RO	1'b0	
		dsp1_tbu_qdeny_cg	9	9	RO	1'b0	
		dsp1_tbu_qactive_cg	8	8	RO	1'b1	
		dsp0_tbu_qreqn_pd	7	7	RW	1'b1	
		dsp0_tbu_qacceptn_pd	6	6	RO	1'b0	
		dsp0_tbu_qdeny_pd	5	5	RO	1'b0	
		dsp0_tbu_qactive_pd	4	4	RO	1'b1	
		dsp0_tbu_qreqn_cg	3	3	RW	1'b1	
		dsp0_tbu_qacceptn_cg	2	2	RO	1'b0	
		dsp0_tbu_qdeny_cg	1	1	RO	1'b0	
		dsp0_tbu_qactive_cg	0	0	RO	1'b1	
16'h03FC	mmu_clear_int	reserved	31	17	RO	15'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mmu_int_clear	16	0	W1P	17'b0	MMU interrupt clear pulse, write 1 to clear the MMU interrupts. Bit mapping as below:
							[0]: tbu0_u84_ras_intr_clr, U84 interrupt number272
							[1]: tbu0_u84_pmu_intr_clr, U84 interrupt number273
							[2]: tbu2_u84_ras_intr_clr, U84 interrupt number274
							[3]: tbu2_u84_pmu_intr_clr, U84 interrupt number275
							[4]: tbu3_u84_ras_intr_clr, U84 interrupt number276
							[5]: tbu3_u84_pmu_intr_clr, U84 interrupt number277
							[6]: tbu4_u84_ras_intr_clr, U84 interrupt number278
							[7]: tbu4_u84_pmu_intr_clr, U84 interrupt number279
							[8]: tcu_u84_event_q_irpt_s_clr, U84 interrupt number350
							[9]: tcu_u84_event_q_irpt_ns_clr, U84 interrupt number351
							[10]: tcu_u84_pri_q_irpt_ns_clr, U84 interrupt number352

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							[11]: tcu_u84_cmd_sync_irpt_ns_clr, U84 interrupt number353
							[12]: tcu_u84_cmd_sync_irpt_s_clr, U84 interrupt number354
							[13]: tcu_u84_global_irpt_ns_clr, U84 interrupt number355
							[14]: tcu_u84_global_irpt_s_clr, U84 interrupt number356
							[15]: tcu_u84_ras_irpt_clr , U84 interrupt number357
							[16]: tcu_u84_pmu_irpt_clr, U84 interrupt number358
16'h0400	aon_timer_pause	timer3_pause	31	24	RW	8'b0	
		timer2_pause	23	16	RW	8'b0	
		timer1_pause	15	8	RW	8'b0	
		timer0_pause	7	0	RW	8'b0	
16'h0404	audio_mclk_oectrl	reserved	31	1	RO	31'b0	
		audio_mclk_oe	0	0	RW	1'b1	I2S MCLK output enable, high active.
16'h0600	hsp_status	Reserved	31	9	RO	23'd0	
		hsp_scu_dw_ahb_hmaster_data	8	5	RO	4'h0	
		hsp_scu_dw_ahb_hmster	4	1	RO	4'h0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		hsp_scu_dw_ahb_hmasterlock	0	0	RO	1'h0	
16'h0604	vc_status	Reserved	31	1	RO	31'd0	
		vc_scu_ve_idle	0	0	RO	1'b0	
16'h0608	mcpu_status	Reserved	31	17	RO	15'd0	
		mcput_debug_ndreset	16	16	RO	1'b0	
		mcput_cease_from_tile_0	15	15	RO	1'b0	
		mcput_cease_from_tile_1	14	14	RO	1'b0	
		mcput_cease_from_tile_2	13	13	RO	1'b0	
		mcput_cease_from_tile_3	12	12	RO	1'b0	
		mcput_halt_from_tile_0	11	11	RO	1'b0	
		mcput_halt_from_tile_1	10	10	RO	1'b0	
		mcput_halt_from_tile_2	9	9	RO	1'b0	
		mcput_halt_from_tile_3	8	8	RO	1'b0	
		mcput_wfi_from_tile_0	7	7	RO	1'b0	
		mcput_wfi_from_tile_1	6	6	RO	1'b0	
		mcput_wfi_from_tile_2	5	5	RO	1'b0	
		mcput_wfi_from_tile_3	4	4	RO	1'b0	
		mcput_debug_from_tile_0	3	3	RO	1'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		mcput_debug_from_tile_1	2	2	RO	1'b0	
		mcput_debug_from_tile_2	1	1	RO	1'b0	
		mcput_debug_from_tile_3	0	0	RO	1'b0	
16'h060c	sec_status	Reserved	31	3	RO	29'd0	
		scpu_jtag_locken	2	2	RO	1'b0	
		sec_otp_done	1	1	RO	1'b0	
		sec_security_en	0	0	RO	1'b0	1'b0:non-security mode 1'b1:security mode
16'h0614	lpcpu_status	Reserved	31	6	RO	26'd0	
		lpcpu_debug_ndreset_en	5	5	RW	1'b0	debug_ndreset reset whole chip enable, high active. And it will reset whole chip while both debug_ndreset and debug_ndreset_en high.
		lpcpu_debug_ndreset	4	4	RO	1'b0	
		lpcpu_halt_from_tile_0	3	3	RO	1'b0	
		lpcpu_wfi_from_tile_0	2	2	RO	1'b0	
		lpcpu_cease_from_tile_0	1	1	RO	1'b0	
		lpcpu_debug_from_tile_0	0	0	RO	1'b0	
16'h0618	scpu_status	Reserved	31	6	RO	26'd0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scpu_debug_ndreset_en	5	5	RW	1'b1	debug_ndreset reset whole chip enable, high active. And it will reset whole chip while both debug_ndreset and debug_ndreset_en high.
		scpu_debug_ndreset	4	4	RO	1'b0	
		scpu_halt_from_tile_0	3	3	RO	1'b0	
		scpu_wfi_from_tile_0	2	2	RO	1'b0	
		scpu_cease_from_tile_0	1	1	RO	1'b0	
		scpu_debug_from_tile_0	0	0	RO	1'b0	
16'h061c	aon_peri_status	Reserved	31	3	RO	29'd0	
		rtc_en	2	2	RO	1'h0	
		i2c1_en	1	1	RO	1'h0	
		i2c0_en	0	0	RO	1'h0	
16'h0620	die_status	Reserved	31	22	RO	10'd0	
		por_waitime_sel	21	20	RO	2'd0	
		die_mode_strapped	19	19	RO	1'b0	
		ddr_refclk_sel_strapped	18	18	RO	1'b0	
		otp_progmode_strapped	17	17	RO	1'b0	
		repair_rd_done	16	16	RO	1'b1	
		die_num	15	8	RO	8'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_die_id	7	0	RO	8'b0	
16'h0624	lsp_status	Reserved	31	4	RO	28'd0	
		lsp_scu_timer_en	3	0	RO	4'b0	
16'h0628	pcie_status	Reserved	31	7	RO	25'd0	
		pciet_scu_flr_pf_active	6	6	RO	1'h0	
		pciet_scu_mstr_awmisc_info_nw	5	5	RO	1'h0	
		pciet_scu_mstr_awmisc_info_ats	4	3	RO	2'h0	
		pciet_scu_mstr_armisc_info_nw	2	2	RO	1'h0	
		pciet_scu_mstr_armisc_info_ats	1	0	RO	2'h0	
16'h062C	usb_curpwr_state	Reserved	31	21	RO	11'b0	
		usb0_current_pwrstate_u2pmu	20	19	RO	2'b0	
		usb0_pmegen_u2pmu	18	18	RO	1'b0	
		usb0_constate_u2pmu	17	17	RO	1'b0	
		usb0_current_pwrstate_u3pmu	16	15	RO	2'b0	
		usb0_pmegen_u3pmu	14	14	RO	1'b0	
		usb0_constate_u3pmu	13	13	RO	1'b0	
		usb0_pwrsw_ctrl	12	12	RW	1'b0	
		Reserved	11	9	RO	3'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		usb1_current_pwrstate_u2pmu	8	7	RO	2'b0	
		usb1_pmegen_u2pmu	6	6	RO	1'b0	
		usb1_constate_u2pmu	5	5	RO	1'b0	
		usb1_current_pwrstate_u3pmu	4	3	RO	2'b0	
		usb1_pmegen_u3pmu	2	2	RO	1'b0	
		usb1_constate_u3pmu	1	1	RO	1'b0	
		usb1_pwrsw_ctrl	0	0	RW	1'b0	
16'h0630	usb0_debug_u2pmu_low	usb0_debug_u2pmu_0to31	31	0	RO	32'b0	
16'h0634	usb0_debug_u2pmu_med	usb0_debug_u2pmu_32to63	31	0	RO	32'b0	
16'h0638	usb0_debug_u2pmu_hi	Reserved	31	4	RO	28'b0	
		usb0_debug_u2pmu_64to67	3	0	RO	4'b0	
16'h063C	usb0_debug_u3pmu_low	usb0_debug_u3pmu_0to31	31	0	RO	32'b0	
16'h0640	usb0_debug_u3pmu_hi	Reserved	31	7	RO	25'b0	
		usb0_debug_u3pmu_32to38	6	0	RO	7'b0	
16'h0644	usb0_pmu_config	Reserved	31	30	RO	2'b0	
		usb0_pmu_cfgstrap	29	0	RW	30'b0	
16'h0648	usb1_debug_u2pmu_low	usb1_debug_u2pmu_0to31	31	0	RO	32'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h064C	usb1_debug_u2pmu_med	usb1_debug_u2pmu_32to63	31	0	RO	32'b0	
16'h0650	usb1_debug_u2pmu_hi	Reserved	31	4	RO	28'b0	
		usb1_debug_u2pmu_64to67	3	0	RO	4'b0	
16'h0654	usb1_debug_u3pmu_low	usb1_debug_u3pmu_0to31	31	0	RO	32'b0	
16'h0658	usb1_debug_u3pmu_hi	Reserved	31	7	RO	25'b0	
		usb1_debug_u3pmu_32to38	6	0	RO	7'b0	
16'h065C	usb1_pmu_config	Reserved	31	30	RO	2'b0	
		usb1_pmu_cfgstrap	29	0	RW	30'b0	
16'h0660	aon_timers_en	timer0_en	31	24	RO	8'b0	
		timer1_en	23	16	RO	8'b0	
		timer2_en	15	8	RO	8'b0	
		timer3_en	7	0	RO	8'b0	
16'h0668	test_reg_0	test_reg0	31	0	RW	32'b0	not used
16'h066C	test_reg_1	test_reg1	31	0	RW	32'b0	not used
16'h0670	test_reg_2	test_reg2	31	0	RW	32'b0	not used
16'h0674	test_reg_3	test_reg3	31	0	RW	32'b0	not used
16'h0678	anaip_debug_ctrl	aon_test_mux_div_en	31	16	RW	16'b0	enable the test mux clock dividor

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							bit16 -> test_out0
							…
							Bit31 -> test_out15
		aon_test_mux_sel	15	0	RW	16'b0	
16'h067C	aon_i2c0_status	Reserved	31	22	RO	10'd0	
		aon_i2c0_debug_s_gen	21	21	RO	1'b0	
		aon_i2c0_debug_p_gen	20	20	RO	1'b0	
		aon_i2c0_debug_data	19	19	RO	1'b0	
		aon_i2c0_debug_addr	18	18	RO	1'b0	
		aon_i2c0_debug_rd	17	17	RO	1'b0	
		aon_i2c0_debug_wr	16	16	RO	1'b0	
		aon_i2c0_debug_hs	15	15	RO	1'b0	
		aon_i2c0_debug_master_act	14	14	RO	1'b0	
		aon_i2c0_debug_slave_act	13	13	RO	1'b0	
		aon_i2c0_debug_addr_10bit	12	12	RO	1'b0	
		reserved	11	9	RO	3'd0	
		aon_i2c0_debug_mst_cstate	8	4	RO	5'd0	
		aon_i2c0_debug_slv_cstate	3	0	RO	4'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
16'h0680	aon_i2c1_status	Reserved	31	22	RO	10'd0	
		aon_i2c1_debug_s_gen	21	21	RO	1'b0	
		aon_i2c1_debug_p_gen	20	20	RO	1'b0	
		aon_i2c1_debug_data	19	19	RO	1'b0	
		aon_i2c1_debug_addr	18	18	RO	1'b0	
		aon_i2c1_debug_rd	17	17	RO	1'b0	
		aon_i2c1_debug_wr	16	16	RO	1'b0	
		aon_i2c1_debug_hs	15	15	RO	1'b0	
		aon_i2c1_debug_master_act	14	14	RO	1'b0	
		aon_i2c1_debug_slave_act	13	13	RO	1'b0	
		aon_i2c1_debug_addr_10bit	12	12	RO	1'b0	
		reserved	11	9	RO	3'b0	
		aon_i2c1_debug_mst_cstate	8	4	RO	5'b0	
		aon_i2c1_debug_slv_cstate	3	0	RO	4'b0	
16'h0694	d2d_stauts	d2d_status_misc	31	8	RO	24'd0	D2D status info. [0]: d2d_srds_cpu_sys_halt flag, high active.
		Reserved	7	5	RO	3'd0	
		d2d_serdes_status_i	4	4	RO	1'b0	D2D serdes tx phy ready status input, which

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
							input from another die.
		Reserved	3	1	RO	3'd0	
		d2d_serdes_status_o	0	0	RW	1'b0	D2D serdes tx phy ready status output, which output to another die.
16'h0698	d2d_ctrl	d2d_ctrl_misc	31	0	RW	32'd0	control signal to D2D subsystem
16'h1000	scpu_qos	reserved	31	16	RO	16'd0	
		scu_scpu_sp1_arqos	15	12	RW	4'h0	
		scu_scpu_sp1_awqos	11	8	RW	4'h0	
		scu_scpu_sp0_arqos	7	4	RW	4'h0	
		scu_scpu_sp0_awqos	3	0	RW	4'h0	
16'h1004	scpu_sid	scu_aon_scpu_awmmusid	31	24	RW	8'b0	
		scu_aon_scpu_awmmussid	23	16	RW	8'b0	
		scu_aon_scpu_armmusid	15	8	RW	8'b0	
		scu_aon_scpu_armmussid	7	0	RW	8'b0	
16'h2000	lpcpu_qos	reserved	31	16	RO	16'd0	
		scu_lpcpu_sp1_arqos	15	12	RW	4'h0	
		scu_lpcpu_sp1_awqos	11	8	RW	4'h0	
		scu_lpcpu_sp0_arqos	7	4	RW	4'h0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_lpcpu_sp0_awqos	3	0	RW	4'h0	
16'h2004	lpcpu_sid	scu_aon_lpcpu_awmmusid	31	24	RW	8'b0	
		scu_aon_lpcpu_awmmussid	23	16	RW	8'b0	
		scu_aon_lpcpu_armmusid	15	8	RW	8'b0	
		scu_aon_lpcpu_armmussid	7	0	RW	8'b0	
16'h3000	dma1_qos	reserved	31	8	RO	24'd0	
		scu_dma1_arqos	7	4	RW	4'h0	
		scu_dma1_awqos	3	0	RW	4'h0	
16'h3004	dma1_sid	scu_aon_dma1_awmmusid	31	24	RW	8'b0	
		scu_aon_dma1_awmmussid	23	16	RW	8'b0	
		scu_aon_dma1_armmusid	15	8	RW	8'b0	
		scu_aon_dma1_armmussid	7	0	RW	8'b0	
16'h4000	sec_qos	reserved	31	8	RO	24'd0	
		scu_sec_arqos	7	4	RW	4'h0	
		scu_sec_awqos	3	0	RW	4'h0	
16'h4004	sec_sid	scu_aon_crypt_awmmusid	31	24	RW	8'b0	
		scu_aon_crypt_awmmussid	23	16	RW	8'b0	
		scu_aon_crypt_armmusid	15	8	RW	8'b0	

Offset	Reg Name	Field	MSB	LSB	Access	Default	Description
		scu_aon_crypt_armmussid	7	0	RW	8'b0	

3.4 Processor

3.4.1 MCPU

3.4.1.1 MCPU Overview

The EIC7700X MCPU includes 4 64-bit RISC–V cores, along with the necessary functional units required to support the cores. These units include a Core-Local Interruptor (CLINT) to support local interrupts, a Platform-Level Interrupt Controller (PLIC) to support platform interrupts, physical memory protection, a Debug unit to support a JTAG-based debugger host connection, Nexus 5001 compliant instruction trace for non-invasive debug and performance profiling, and a local crossbar that integrates the various components together.

The EIC7700X MCPU memory system consists of a Data Cache and Instruction Cache, with coherent L1 caches, private L2 Cache, shared L3 Cache, and a directory based coherence manager. The EIC7700X MCPU also includes a Front Port, which allows external masters to be coherent with the L1 memory system and access to the TIMs, thereby removing the need to maintain coherence in software for any external agents. All memories, including caches and TIMs, support Single Error Correction, Double Error Detection (SECDED) ECC to provide improved reliability and address safety critical applications.

The EIC7700X MCPU is guaranteed to be compatible with all applicable RISC–V standards, and this document should be read together with the official RISC–V user-level, privileged, and external debug architecture specifications.

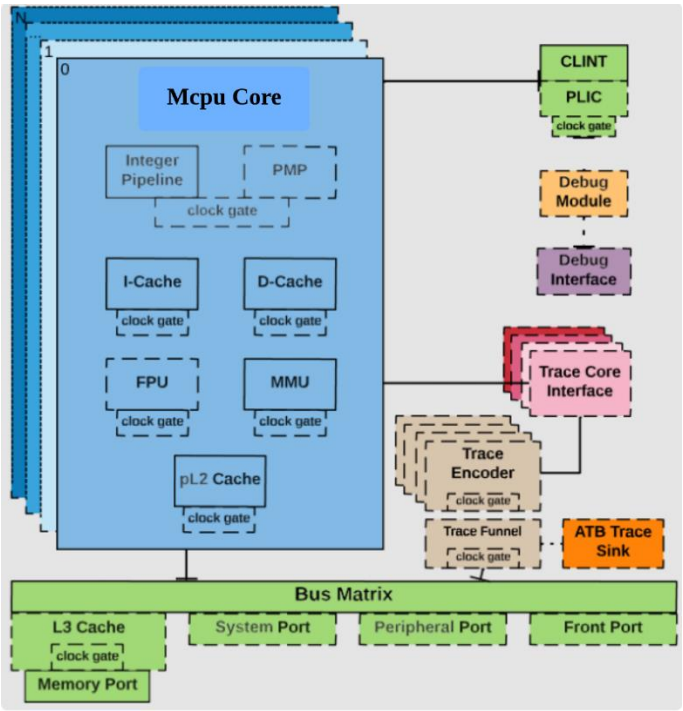


Figure 3-4 MCPU

Table 3-4 MCPU feature summary

Feature	Description
ISA	RV64GC_Zba_Zbb_Sscofpmf
Custom Instruction Extension (SCIE)	Present
Privilege Modes	M, U, HS, VU, VS

Feature	Description
L1 Instruction Cache	32 KiB 4-way instruction cache
L1 Data Cache	32 KiB 4-way data cache
Private L2 (pL2) Cache	256 KiB 8-way pL2 cache with 2 banks
L3 Cache	4 MiB 16-way L3 cache with 4 banks
ECC Support	Single error correction, double error detection on the L1 data cache, pL2 cache, and L3 cache
Physical Memory Protection	8 regions with a granularity of 4096 bytes
Memory Management Unit	Sv48 virtual memory support with fully-associative 32-entry L1 data and instruction TLBs, and a 4-way 512-entry unified TLB

- **EIC7700X RISC-V Core**

The EIC7700X MCPU includes four 64-bit EIC7700X RISC–V cores, which each have a triple-issue, out-of-order, 13 stage RISC–V processor. The EIC7700X core is guaranteed to be compatible with all applicable RISC–V standards.

Each EIC7700X core is configured to support the RV64I base ISA, as well as the Multiply (M), Atomic(A), Single-Precision Floating Point (F), Double-Precision Floating Point (D), Compressed (C), CSR Instructions (Zicsr), Instruction-Fetch Fence (Zifencei), Address Calculation (Zba), Basic Bit Manipulation (Zbb), and Count Overflow and Mode-Based Filtering (Sscofpmf) RISC–V extensions. This is captured by the RISC–V extension string: RV64GC_Zba_Zbb_Sscofpmf.

The EIC7700X MCPU also supports machine, hypervisor, user, virtual supervisor, and virtual user privilege modes, in conjunction with Physical Memory Protection (PMPs). The microarchitecture also incorporates a branch prediction unit that is composed of a 32-entry Branch Target Buffer (BTB), a 9.1 KiB-entry Branch History Table (BHT), a 16-entry Return Address Stack (RAS), 512-entry Indirect Jump Target Predictor (IJTP), and a 16-entry Return Instruction Predictor.

The EIC7700X MCPU includes an IEEE 754-2008 compliant Floating-Point Unit. The floating-point pipeline sits in conjunction with the integer pipeline; however, it is longer, increasing the total pipeline

stages to 7.

- **Memory System**

The EIC7700X MCPU memory system has a Level 1 memory system optimized for high performance. The instruction subsystem consists of a 32 KiB, 4-way instruction cache. The data subsystem is comprised of a high performance 32 KiB, 4-way L1 data cache. In addition to the L1 memory system, the EIC7700X MCPU has a per hart 256 KiB, 8-way private L2 cache.

- **Interrupts**

The EIC7700X MCPU provides the standard RISC–V M-mode timer and software interrupts via the Core-Local Interruptor (CLINT). The EIC7700X MCPU also includes a RISC–V standard Platform-Level Interrupt Controller (PLIC), which supports 520 global interrupts with 7 priority levels pre-integrated with the on core-complex peripherals.

- **Debug Support**

The EIC7700X MCPU provides external debugger support over an industry-standard JTAG

port, including 4 hardware-programmable breakpoints per hart.

- **Compliance**

The EIC7700X MCPU is compliant to the following versions of the various RISC–V specifications:

Table 3-5 RISC-V Specification Compliance

ISA	Version	Status
RV64I Base Integer Instruction Set	2.1	Ratified
Extensions	Version	Status
M Standard Extension for Integer Multiplication and Division	2.0	Ratified
A Standard Extension for Atomic Instruction	2.1	Ratified
F Standard Extension for Single-Precision Floating-Point	2.2	Ratified
D Standard Extension for Double-Precision Floating-Point	2.2	Ratified
C Standard Extension for Compressed Instruction	2.0	Ratified
Zicsr Standard Extension for Control and Status Register (CSR) Instructions	2.0	Ratified
Zifencei Standard Extension for Instruction-Fetch Fence	2.0	Ratified
Zba Standard Extension for Address Calculation	1.0	Ratified
Zbb Standard Extension for Basic Bit Manipulation	1.0	Ratified
Sscofpmf Standard Extension for Count Overflow and Mode-Based Filtering	0.1	Ratified
Privilege Mode	Version	Status
Machine-Level ISA	1.11	Ratified
User-Level ISA	1.11	Ratified
Supervisor-Level ISA	1.11	Ratified
Hypervisor-Level ISA	0.6	Frozen
Devices	Version	Status
The RISC-V Debug Specification	1.0	Frozen
RISC-V Platform-Level Interrupt Controller (PLIC) Specification	—	—

3.4.1.2 Instruction Memory System

The regions of executable memory consist of all directly addressable memory in the system. The memory includes any volatile or non-volatile memory located off the Core Complex ports, and includes the on-core-complex L3 LIM and L3 Zero Device.

Table 3-6 Executable Memory Regions for the EIC7700X MCPU

Base	Top	Description
0x0001_8000	0x0001_8FFF	Trace Funnel

Base	Top	Description
0x0010_0000	0x0010_0FFF	Trace Encoder 0
0x0010_1000	0x0010_1FFF	Trace Encoder 1
0x0010_2000	0x0010_2FFF	Trace Encoder 2
0x0010_3000	0x0010_3FFF	Trace Encoder 3
0x0010_4000	0x0010_7FFF	Hart 0 Private L2 Cache
0x0010_8000	0x0010_BFFF	Hart 1 Private L2 Cache
0x0010_C000	0x0010_FFFF	Hart 2 Private L2 Cache
0x0011_0000	0x0011_3FFF	Hart 3 Private L2 Cache
0x0170_0000	0x0170_0FFF	Hart 0 Bus-Error Unit
0x0170_1000	0x0170_1FFF	Hart 1 Bus-Error Unit
0x0170_2000	0x0170_2FFF	Hart 2 Bus-Error Unit
0x0170_3000	0x0170_3FFF	Hart 3 Bus-Error Unit
0x0200_0000	0x0200_FFFF	CLINT
0x0201_0000	0x0201_3FFF	L3 Cache Controller
0x0800_0000	0x083F_FFFF	L3 LIM
0x0C00_0000	0x0FFF_FFFF	PLIC
0x1001_0000	0x1001_0FFF	Burst Bundler on Front Port
0x1003_0000	0x1003_3FFF	TL-UH to TL-C Adapter
0x1A00_0000	0x1A3F_FFFF	L3 Zero Device
0x4000_0000	0x7FFF_FFFF	System Port 0 (1 GiB)
0x8000_0000	0x7F_FFFF_FFFF	Memory Port 0 (510 GiB)

3.4.1.3 Hardware Prefetcher

The Hardware Prefetcher (HWPF) is a region-based sequential stride prefetcher and is tightly integrated in the LSU. It enables memory-level parallelism (MLP) and hides memory latency. The HWPF relies on patter detection of loads and sends hints that either go to L1 MSHRs or directly to L2. It is comprised of the HWPF module with individual prefetch engines and a HWPF issue queue.

The HWPF uses a single state machine that sends both L1 and L2 prefetches, depending on how far the tail pointer and respective demand pointer are separated.

The first L1 demand request allocates with the line address that consists of the region's virtual address and offset. The second L1 demand request calculates the stride based on the current and previous addresses. The third L1 demand request confirms the stride and starts sending prefetches equal to the number programmed in the initialDist of the HWPF CSR, up to a maximum of 64 L1 prefetch requests. Each L1 prefetch request establishes an MSHR and sends an AcquireBlock.NtoB request to the downstream cache hierarchy. Each L2 prefetch request establishes an L2 HWPF queue and sends a Hint to L2.

Table 3-7 Hardware Prefetcher 0 CSR

Hardware Prefetcher 0 CSR				
CSR	0x7C3			
Bits	Field Name	Attr.	Rst.	Description
0	enable	RW	0x0	Enable the hardware prefetcher
[6:1]	window	RW	0x0	Address comparison window
[12:7]	initialDist	RW	0x0	Prefetch beginning distance
[18:13]	maxAllowedDist	RW	0x0	Maximum allowed prefetch distance
[24:19]	linToExpThrd	RW	0x0	Threshold when HWPF distance increase changes from linear to exponential
[28:25]	qFullnessThrdL1	RW	0x0	Threshold fraction/16 of L1 MSHRs to stop sending hints to L1
[33:29]	hitCacheThrdL1	RW	0x0	Threshold number of L1 HWPF hits in L1 cache tag for evicting HWPF entry
[37:34]	hitMSHRThrdL1	RW	0x0	Threshold number of loads hitting L1 HWPF allocated L1 MSHRs or L1 HWPFs hitting demand allocated L1 MSHRs to increase aggressiveness
38	issueBubble	RW	0x0	Prefetch request to load/store tag read force block LS issue to inject a bubble
[44:39]	maxL1PFDist	RW	0x0	Threshold for directly sending prefetches to L2 instead of L1 MSHR
[46:45]	forgiveThrd	RW	0x0	Threshold for forgiving loads with mismatching strides when HWPF is in trained state. Setting to 0 (default) disabled the feature.
[49:47]	numL1PFIssQEnt	RW	0x0	Number of HWPF issue queue entries dedicated for L1 prefetches
[63:50]	Reserved	RO	0x0	

Table 3-8 Hardware Prefetcher 1 CSR

Hardware Prefetcher 1 CSR				
CSR	0x7C4			
Bits	Field Name	Attr.	Rst.	Description
0	l2pfEnable	RW	0x0	Enable L2 prefetches
[4:1]	qFullnessThrdL2	RW	0x0	Threshold fraction/16 of L2 MSHRs to stop sending hints to L2
[9:5]	hitCacheThrdL2	RW	0x0	Threshold number of L2 HWPF hits in L2 cache tag to stop sending hints to L2

Hardware Prefetcher 1 CSR				
CSR	0x7C4			
Bits	Field Name	Attr.	Rst.	Description
[13:10]	hitMSHRThrdL2	RW	0x0	Threshold number of L1 HWPF hitting L2 HWPF L2 MSHRs to increase aggressiveness
[16:14]	numL2PFIssQEnt	RW	0x0	Number of HWPF issue queue entries dedicated for L2 prefetches
[31:17]	Reserved	RO	0x0	

The HWPF needs to be initialized in the boot software prior to use. EIC7700X initialization software writes 0x104095c1be241 to 0x7C3 and 0x38c84e to 0x7C4 and recommends these values as a starting point. Once initialized, the HWPF can be tuned on a per-configuration basis as needed.

3.4.1.4 Memory Map

The memory map is divided into different regions covering on-core-complex memory, system memory, peripherals, and empty holes. Physical memory attributes (PMAs) describe the proper- ties of the accesses that can be made to each region in the memory map. These properties encompass the type of access that may be performed: execute, read, or write. As well as other optional attributes related to the access, such as supported access size, alignment, atomic operations, and cacheability.

External memory map regions are accessed through a specific port type and that port type is used to define the PMAs. The port types are Memory, Peripheral, and System. Memory map regions defined for internal memory and internal control regions also have a predefined PMA based on the underlying contents of the region.

Table 3-9 Physical Memory Attributes for External Region

Port Type	Access Properties	Attributes
Memory Port	Read, Write, Execute	Atomics+LR/SC, Data Cacheable, Instruction Cacheable, Instruction Speculation
System Port	Read, Write, Execute	Instruction Cacheable

Table 3-10 Physical Memory Attributes for Internal Regions

Region	Access Properties	Attributes
Burst Bundler on Front Port	Read, Write, Execute	Atomics
Bus Blocker on TL256D2D_In (TL-C slave)	Read, Write	Atomics
Bus Blocker on TL256D2D_Out (TL-C master)	Read, Write	Atomics

Region	Access Properties	Attributes
Bus Blocker on TL64D2D_Out (TL-UH master)	Read, Write	Atomics
Bus-Error Unit	Read, Write, Execute	Atomics
CLINT	Read, Write, Execute	Atomics
Debug	None	N/A
Error Device	Read, Write, Execute	Atomics
Error Device	Read, Write, Execute	Atomics, Instruction Cacheable
L3 Cache Controller	Read, Write, Execute	Atomics
L3 LIM	Read, Write, Execute	Atomics
L3 Zero Device	Read, Write, Execute	Atomics, Data Cacheable, Instruction Cacheable
PLIC	Read, Write, Execute	Atomics
Private L2 Cache	Read, Write, Execute	Atomics
TL-UH to TL-C Adapter	Read, Write, Execute	Atomics
Test Indicator	Read, Write	Atomics
Trace Encoder	Read, Write, Execute	Atomics
Trace Funnel	Read, Write, Execute	Atomics
Unmapped	None	N/A

All memory map regions support word, half-word, and byte size data accesses.

3.4.1.5 Virtual Memory

The EIC7700X MCPU supports virtual memory by a Memory Management Unit (MMU). The MMU supports the Sv48 address mode as described in the RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

The translation lookaside buffers (TLBs) are address translation caches within the MMU. There are both level 1 and level 2 TLB entries. Level 1 entries contain separate instruction buffers (ITLB) and data buffers (DTLB). The ITLB and DTLB each contain 32 entries, which are fully associative. Level 2 TLB entries are unified and contain 512 TLB entries. Level 2 TLB are all 4 KiB pages.

- **Supervisor Address Translation and Protection Register (satp)**

The supervisor address translation and protection register (satp) is a 64-bit read/write register used to

control supervisor address translation and protection.

Table 3-11 Supervisor Address Translation and Protection Register

Supervisor Address Translation and Protection Register (satp)			
CSR	0x180		
Bits	Field Name	Attr.	Description
[43:0]	PPN	WARL	Holds the physical page number of the root page table, which is the supervisor physical address divided by 4 KiB
[59:44]	ASID	WARL	An address space identifier used to facilitate address-translation fences on a per-address-space basis
[63:60]	MODE	WARL	Determines the selected address-translation scheme. 0x0 - Bare mode, no translation enabled 0x8 - Page-based 38-bit virtual addressing (Sv39) 0x9 - Page-based 48-bit virtual addressing (Sv48)

When satp.MODE=0x0, supervisor virtual addresses are equal to supervisor physical addresses, and there is no additional memory protection beyond the physical memory protection scheme. In this case, the remaining fields in satp have no effect.

Note that writing satp does not imply any ordering constraints between page-table updates and subsequent address translations. If the new address space's page tables have been modified, or if an ASID is reused, it may be necessary to execute an SFENCE.VMA instruction after writing satp, which will:

1. Synchronize page table writes and address translation hardware for higher privilege levels

2. Guarantee previous stores are ordered before all subsequent references from the hart to the memory management data structures

3. Flush L1 and L2 unified TLB entry

- **Virtual Supervisor Address Translation and Protection Register (vsatp)**

The vsatp register is a 64-bit read/write register that is the virtual supervisor mode replica of the supervisor register satp. When virtualization is enabled, instructions that normally read or modify satp accesses vsatp instead. vsatp controls VS-stage address translation, the first stage of two-stage translation for guest virtual addresses.

Table 3-12 Virtual Supervisor Address Translation and Protection Register

Virtual Supervisor Address Translation and Protection Register (vsatp)			
CSR	0x280		
Bits	Field Name	Attr.	Description
[43:0]	PPN	WARL	Holds the physical page number of the root page table, which is the supervisor physical address divided by 4 KiB
[59:44]	ASID	WARL	An address space identifier used to facilitate address-translation fences on a per-address-space basis
[63:60]	MODE	WARL	Determines the selected address-translation scheme. 0x0 - Bare mode, no translation enabled 0x8 - Page-based 38-bit virtual addressing (Sv39) 0x9 - Page-based 48-bit virtual addressing (Sv48)

- **Hypervisor Guest Address Translation and Protection Register (hgatp)**

The hgatp register is a 64-bit read/write register that controls G-stage address translation and protection, the second stage of two-stage translation for guest virtual addresses.

Table 3-13 Hypervisor Guest Address Translation and Protection Register

Hypervisor Guest Address Translation and Protection Register (hgatp)			
CSR	0x680		
Bits	Field Name	Attr.	Description
[43:0]	PPN	WARL	Holds the physical page number of the root page table, which is the supervisor physical address divided by 4 KiB
[57:44]	VMID	WARL	A virtual machine identifier used to facilitate address-translation fences on a per-virtual-machine basis
[59:58]	Reserved	WARL	
[63:60]	MODE	WARL	Determines the selected address-translation scheme for guest physical addresses. 0x0 - Bare mode, no translation enabled 0x8 - Page-based 41-bit virtual addressing (Sv39x4) 0x9 - Page-based 50-bit virtual addressing (Sv48x4)

- **Virtual-to-Physical Address Mapping Example**

The following figure is a high-level view of how a virtual address is mapped to a physical address for a Linux application. When the Linux kernel creates a process, it will allocate multiple pages of physical memory to store the code and data. TLB MMU is used to:

• Translate the virtual addresses to physical addresses

• Provide uniform virtual memory layout for a user application

• Protect user applications unauthorized access to other address space

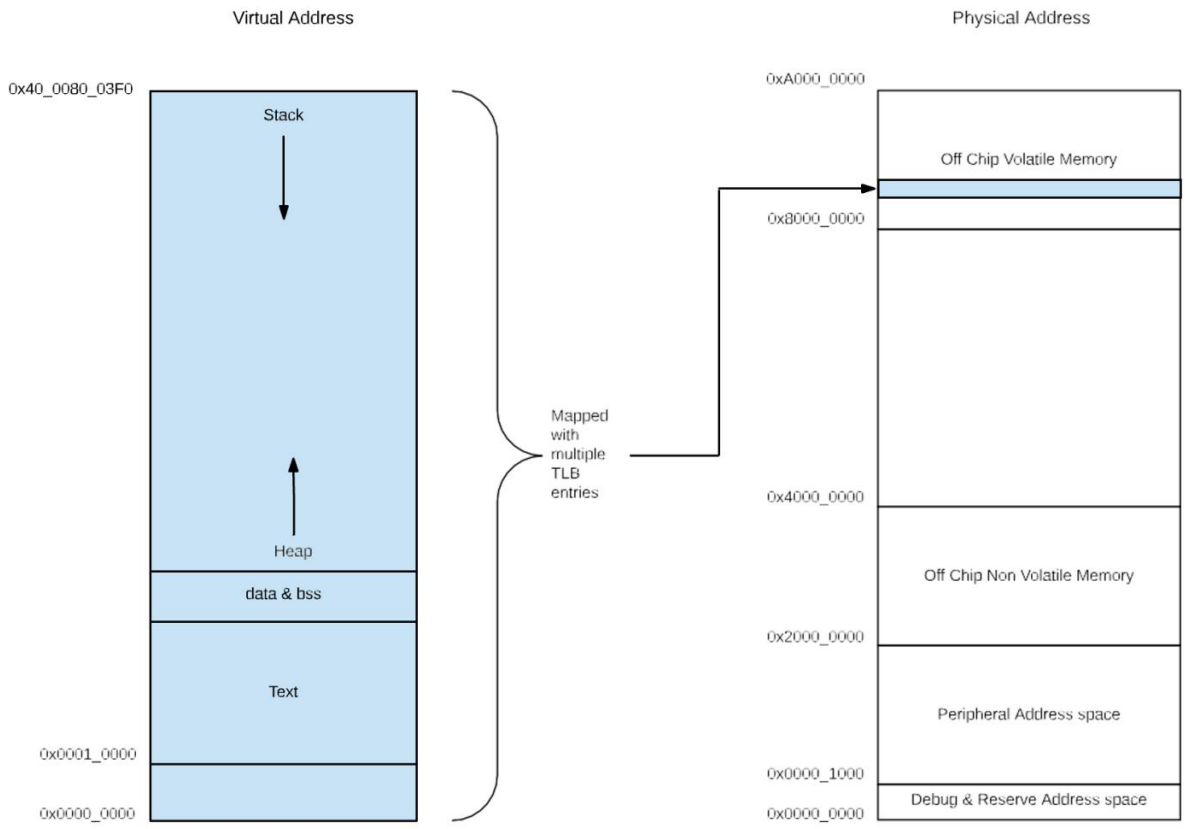


Figure 3-5 Linux User Application Memory Map Example

In this example, code beginning at virtual address 0x0001_0000 needs to be mapped to an address in off-chip volatile memory.

When the hart tries to execute instructions at this address, it needs to use a matched TLB page table entry to perform a virtual address to physical address translation. If it cannot be located in the L1 instruction TLB, or the L2 unified TLB, the hart will start hardware table walk from the TLB page table base address. The page table base address is obtained by multiplying satp.PPN by the L2 PAGESIZE (4 KiB).

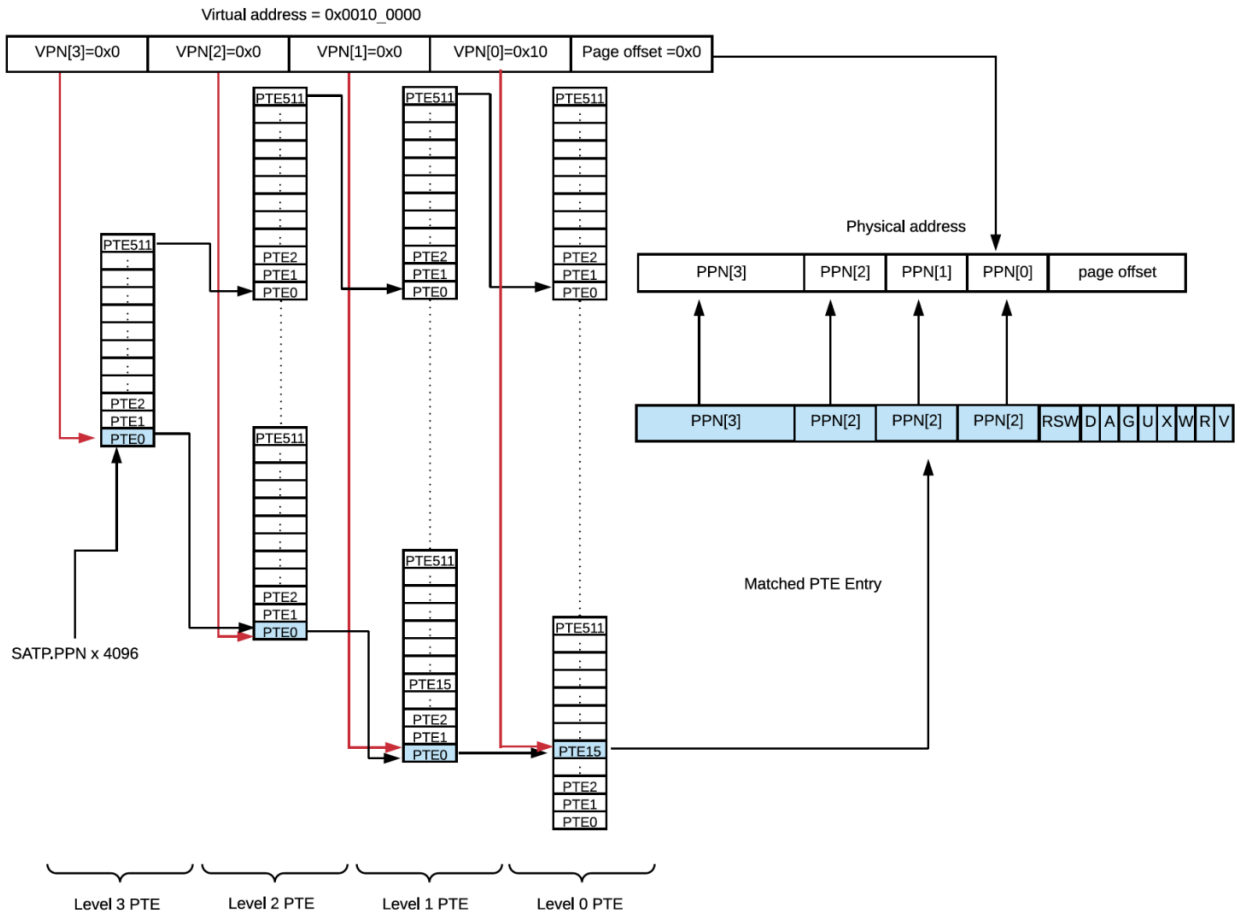


Figure 3-6 Hardware Table Walk Example

The TLB MMU will execute a page table walk to determine the correct mapping for a particular virtual address. Page table entries are pointers to the next level page table if the page is marked as not Readable (R=0), not Writable (W=0), and not Executable (X=0). Otherwise, it is a leaf PTE.

In this example, there are 4 levels of page table entries. The hart will start the hardware table walk from the level 3 page table entry. When there is a match in level 0 page table entry, the virtual address will map to a physical address. The physical page number is combined with the page offset to give the complete physical address.

A description of the PTE configuration bits can be found in Table below.

Table 3-14 PTE Configuration Bits

Bits	Description
0	V: Valid 0x0 - Page table entry not valid 0x1 - Page table entry valid
1	R: Readable 0x0 - Page table entry not readable 0x1 - Page table entry readable

Bits	Description
2	W: Writable 0x0 - Page table entry not writable 0x1 - Page table entry writable
3	X: Executable 0x0 - Page table entry not executable 0x1 - Page table entry executable
4	U: User mode access 0x0 - No access to user mode software 0x1 - Access granted to user mode software
5	G: Global mapping 0x0 - This mapping does not exist globally 0x1 - This mapping exists globally
6	A: Accessed 0x0 - Leaf page table entry has not been read, written, or fetched since the last time A was cleared 0x1 - Leaf page table entry has been read, written, or fetched since the last time A was cleared
7	D: Dirty 0x0 - The virtual page has not been written since the last time D was cleared 0x1 - The virtual page has been written since the last time D was cleared
[9:8]	RSW: Supervisor software use X - Open for supervisor software use

- **Page Table Configurations**

Read, write, and execute permissions for Sv48 are summarized in Table below. The value PTE.V=1 indicates the PTE is valid, while PTE.V=0 means all other bits in PTE are don't cares, and software can use these freely. The value PTE.R=1 indicates the page is readable. Likewise, PTE.W=1 indicates the page is writable, while PTE.X=1 means the page is executable. When PTE.V=0, PTE.R=0, and PTE.W=0, this indicates the PTE is a pointer to the next level page table, otherwise it is a leaf PTE. If a page is marked writable, it must also be marked readable. Combinations of PTE.W=1 and PTE.R=0 are not currently supported.

Table 3-15 PTE Encoding fields

X	W	R	Meaning
0	0	0	Pointer to next level of page table
0	0	1	Read-only page

X	W	R	Meaning
0	1	0	Reserved
0	1	1	Read-write page
1	0	0	Execute-only page
1	0	1	Read-execute page
1	1	0	Reserved
1	1	1	Read-write-execute page

A fetch page-fault exception will occur if an instruction is fetched from a page that does not have execute permissions. A load page-fault exception will occur if a load or load-reserved instruction falls within a page without read permissions. A store page-fault exception will occur if a store, store-conditional, or AMO instruction falls within a page without write permissions.

The value PTE.U=1 indicates the page is accessible to user mode. Supervisor mode software may also perform loads and stores to a page marked with PTE.U=1, but only if sstatus.SUM=1. The sstatus.SUM bit modifies the privilege of supervisor mode loads and stores to virtual mem- ory. Supervisor mode software may not execute code on any page marked with PTE.U=1.

Two schemes to manage the A and D bits are permitted:

- **When a virtual page is accessed and the A bit is clear, or is written and the D bit is clear, a page-fault exception is raised**
- **When a virtual page is accessed and the A bit is clear, or is written and the D bit is clear, the corresponding bit(s) are set in the PTE. The PTE update is atomic with respect to other accesses to the PTE, and memory access will not occur until the PTE update is visible globally**

For non-leaf PTEs, the D, A, and U bits are reserved for future use and must be cleared by software for forward compatibility.

It is important to note the EIC7700X MCPU does not automatically set the accessed (A) and dirty (D) bits in a Sv48 Page Table Entry (PTE). Instead, the EIC7700X MCPU MMU will raise a page fault exception for a read to a page with PTE.A=0 or a write to a page with PTE.D=0.

3.4.1.6 MMU and PMP

The TLB MMU is disabled by default out of reset. All accessed regions have a 1:1 virtual-to-physical address mapping when the MMU is disabled. If the PMP is not yet enabled, all access permissions out of reset are determined by the static PMA values.

- **Physical Memory Protection (PMP)**

The EIC7700X PMP supports 8 regions with a minimum region size of 4096 bytes. This section describes how PMP concepts in the RISC–V architecture apply to the EIC7700X MCPU. For additional information on the PMP refer to The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

- **PMP Functional Description**

The EIC7700X PMP unit has 8 regions and a minimum granularity of 4096 bytes. Access to each region is controlled by an 8-bit pmpXcfg field and a corresponding pmpaddrX register. Overlapping regions are permitted, where the lower numbered pmpXcfg and pmpaddrX registers take priority over higher numbered regions. The EIC7700X PMP unit implements the architecturally defined pmpcfgY CSR pmpcfg0, supporting 8 regions. pmpcfg2 is implemented but hardwired to zero. Access to pmpcfg1 or pmpcfg3 results in an illegal instruction exception.

The PMP registers may only be programmed in M-mode. Ordinarily, the PMP unit enforces permissions on S-mode and U-mode accesses. However, locked regions additionally enforce their permissions on M-mode.

- **PMP Region Locking**

The PMP allows for region locking whereby, once a region is locked, further writes to the configuration and address registers are ignored. Locked PMP entries may only be unlocked with a system reset. A region may be locked by setting the L bit in the pmpXcfg register.

In addition to locking the PMP entry, the L bit indicates whether the R/W/X permissions are enforced on machine mode accesses. When the L bit is clear, the R/W/X permissions apply to S- mode and U-mode.

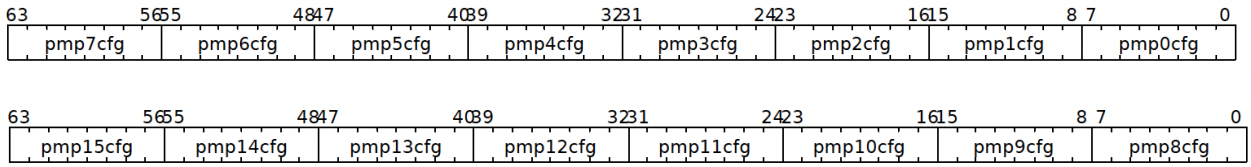
- **PMP Registers**

Each PMP region is described by an 8-bit pmpXcfg field, used in association with a 64-bit pmpaddrX register that holds the base address of the protected region. The range of each region depends on the Addressing (A) mode described in the next section. The pmpXcfg fields reside within 64-bit pmpcfgY CSRs.

Each 8-bit pmpXcfg field includes a read, write, and execute bit, plus a two bit address-matching field A, and a Lock bit, L. Overlapping regions are permitted, where the lowest numbered PMP entry wins for that region.

- **PMP Configuration Registers**

For RV64 architectures, pmpcfg1 and pmpcfg3 are not implemented. This reduces the footprint since pmpcfg2 already contains configuration fields pmp8cfg through pmp11cfg for both RV32 and RV64.



The pmpcfgY and pmpaddrX registers are only accessible via CSR specific instructions such as csrr for reads, and csrw for writes.

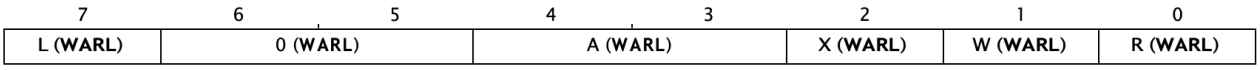


Table 3-16 pmpXcfg Bitfield Description

Bits	Description
0	R: Read Permissions 0x0 - No read permissions for this region 0x1 - Read permission granted for this region

Bits	Description
1	W: Write Permissions 0x0 - No write permissions for this region 0x1 - Write permission granted for this region
2	X: Execute permissions 0x0 - No execute permissions for this region 0x1 - Execute permission granted for this region
[4:3]	A: Address matching mode 0x0 - PMP Entry disabled. No PMP protection applied for any privilege level. 0x1 - Top of range (TOR) region defined by two adjacent pmpaddr registers. The upper limit of region X is defined by pmpaddrX, and the base of the region is defined by pmpaddr(X-1). Address 'a' matches the region if [pmpaddr(X-1) ≤ a < pmpaddrX]. If pmp0cfg defines a TOR region, then the base address of that region is 0x0, and pmpaddr0 defines the upper limit. Supports only a four byte granularity. 0x2 - Reserved 0x3 - Naturally aligned power-of-two region (NAPOT), ≥ 8 bytes. When this setting is programmed, the low bits of the pmpaddrX register encode the size, while the upper bits encode the base address right shifted by two. There is a zero bit in between, we will refer to as the least significant zero bit (LSZB).
7	L: Lock Bit 0x0 - PMP Entry Unlocked, no permission restrictions applied to machine mode. PMP entry only applies to S and U modes. 0x1 - PMP Entry Locked, permissions enforced for all privilege levels including machine mode. Writes to pmpXcfg and pmpcfgY are ignored and can only be cleared with system reset.
Note: The combination of R=0 and W=1 is not currently implemented.	

Out of reset, the PMP register fields A and L are set to 0. All other hart state is unspecified by The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Some examples follow using NAPOT address mode.

Table 3-17 pmpaddrX Encoding Examples for A=NAPOT

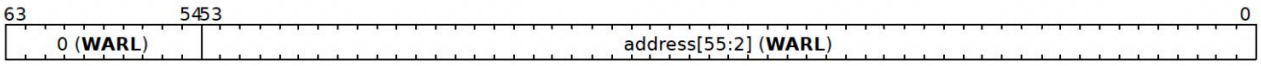
Base Address	Region Size*	LSZB Position	pmpaddrX Value
0x4000_0000	8 B	0	(0x1000_0000 | 1'b0)
0x4000_0000	32 B	2	(0x1000_0000 | 3'b011)
0x4000_0000	4 KB	9	(0x1000_0000 | 10'b01_1111_1111)

0x4000_0000	64 KB	13	(0x1000_0000 | 14'b01_1111_1111_1111)
0x4000_0000	1 MB	17	(0x1000_0000 | 18'b01_1111_1111_1111_1111)
*Region size is 2(LSZB+3).			

- **PMP Address Registers**

The PMP has 8 address registers. Each address register pmpaddrX correlates to the respective pmpXcfg field. Each address register contains the base address of the protected region right shifted by two, for a minimum 4-byte alignment.

The maximum encoded address bits per The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11 are [55:2].



- **PMP and PMA**

The PMP values are used in conjunction with the Physical Memory Attributes (PMAs). Since the PMAs are static and not configurable, the PMP can only revoke read, write, or execute permissions to the PMA regions if those permissions already apply statically.

- **PMP Programming Overview**

The PMP registers can only be programmed in machine mode. The pmpaddrX register should be first programmed with the base address of the protected region, right shifted by two. Then, the pmpcfgY register should be programmed with the properly configured 64-bit value containing each properly aligned 8-bit pmpXcfg field. Fields that are not used can be simply written to 0, marking them unused.

- **Programming Example**

The following example shows a machine mode only configuration where PMP permissions are applied to three regions of interest, and a fourth region covers the remaining memory map. Recall that lower numbered pmpXcfg and pmpaddrX registers take priority over higher numbered regions. This rule allows higher numbered PMP registers to have blanket coverage over the entire memory map while allowing lower numbered regions to apply permissions to specific regions of interest. The following example shows a 64 KB Flash region at base address 0x0, a 32 KB RAM region at base address 0x2000_0000, and finally a 4 KB peripheral region at base address base 0x3000_0000. The rest of the memory map is reserved space.

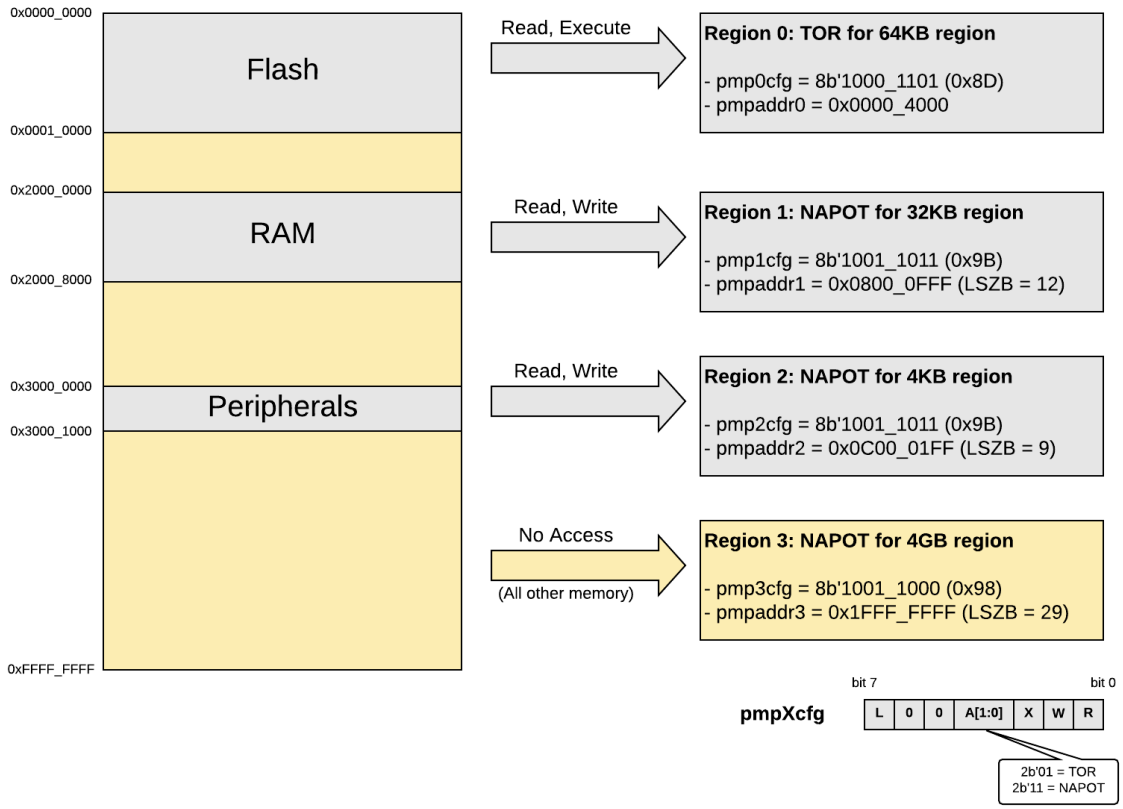


Figure 3-7 PMP Example Block Diagram

- **PMP Access Scenarios**

The L, R, W, and X bits only determine if an access succeeds if all bytes of that access are covered by that PMP entry. For example, if a PMP entry is configured to match the four-byte range 0xC–0xF, then an 8-byte access to the range 0x8–0xF will fail, assuming that PMP entry is thehighest-priority entry that matches those addresses.

While operating in machine mode when the lock bit is clear (L=0), if a PMP entry matches all bytes of an access, the access succeeds. If the lock bit is set (L=1) while in machine mode, then the access depends on the permissions set for that region. Similarly, while in Supervisor mode or User mode, the access depends on permissions set for that region.

Failed read or write accesses generate a load or store access exception, and an instruction access fault would occur on a failed instruction fetch. When an exception occurs while attempting to execute from a region without execute permissions, the fault occurs on the fetch and not the branch, so the mepc CSR will reflect the value of the targeted protected region, and not the address of the branch.

It is possible for a single instruction to generate multiple accesses, which may not be mutually atomic. If at least one access generated by an instruction fails, then an exception will occur. It might be possible that other accesses from a single instruction will succeed, with visible side effects. For example, references to virtual memory may be decomposed into multiple accesses.

On some implementations, misaligned loads, stores, and instruction fetches may also be decomposed into multiple accesses, some of which may succeed before an access exception occurs. In particular, a portion of a misaligned store that passes the PMP check may become visible, even if another portion fails the PMP check. The same behavior may manifest for floating-point stores wider than XLEN bits (e.g., the FSD instruction in RV32D), even when the store address is naturally aligned.

- **PMP and Paging**

The Physical Memory Protection mechanism is designed to compose with the page-based virtual memory systems described in The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

- **PMP Limitations**

In a system containing multiple harts, each hart has its own PMP device. The PMP permissions

on a hart cannot be applied to accesses from other harts in a multi-hart system. In addition, Front Port allow external bus masters access to the full memory map of the system. The PMP cannot prevent access from external bus masters on the Front Port.

- **Behavior for Regions without PMP Protection**

If a non-reserved region of the memory map does not have PMP permissions applied, then by default, supervisor or user mode accesses will fail, while machine mode access will be allowed. Access to reserved regions within a device's memory map (an interrupt controller for example) will return 0x0 on reads, and writes will be ignored. Access to reserved regions outside of a device's memory map without PMP protection will result in a bus error. The bus error can generate an interrupt to the hart using the Bus-Error Unit (BEU).

- **Cache Flush Behavior on PMP Protected Region**

When a line is brought into cache and the PMP is set up with the lock (L) bit asserted to protect a part of that line, a data cache flush instruction will generate a store access fault exception if the flush includes any part of the line that is protected. The cache flush instruction does an invalidate and write-back, so it is essentially trying to write back to the memory location that is protected. If a cache flush occurs on a part of the line that was not protected, the flush will succeed and not generate an exception. If a data cache flush is required without a write-back, use the cache discard instruction instead, as this will invalidate but not write back the line.

- **Defined CSRs**

The following tables describe the currently defined CSRs, categorized by privilege level. The usage of the CSRs below is implementation specific. CSRs are only accessible when operating within a specific access mode (user mode, debug mode, supervisor mode, or machine mode). Therefore, attempts to access a non-existent CSR raise an illegal instruction exception, and attempts to access a CSR without appropriate privilege level or to write a read-only register also raise illegal instruction exceptions.

Table 3-18 User Mode CSRs

Number	Privilege	Name	Description
User Trap Setup			
0x000	RW	ustatus	User status register.
0x004	RW	uie	User interrupt-enable register.
0x005	RW	utvec	User trap handler base address.
User Trap Handling			
0x040	RW	uscratch	Scratch register for use trap handlers.
0x041	RW	uepc	User exception program counter.
0x042	RW	ucause	User trap cause.
0x043	RW	ubadaddr	User bad address.
0x044	RW	uip	User interrupt pending.

Number	Privilege	Name	Description
User Floating-Point CSRs			
0x001	RW	fflags	Floating-Point Accrued Exceptions.
0x002	RW	frm	Floating-Point Dynamic Rounding Mode.
0x003	RW	fcsr	Floating-Point Control and Status Register (frm + fflags).
User Counter/Timers			
0xC00	RO	cycle	Cycle counter for RDCYCLE instruction.
0xC01	RO	time	Timer for RDTIME instruction.
0xC02	RO	instret	Instructions-retired counter for RDINSTRET instruction.
0xC03	RO	hpmcounter3	Performance-monitoring counter.
0xC04	RO	hpmcounter4	Performance-monitoring counter.
		…	
0xC1F	RO	hpmcounter31	Performance-monitoring counter.

Table 3-19 Supervisor Mode CSRs

Number	Privilege	Name	Description
Supervisor Trap Setup			
0x100	RW	sstatus	Supervisor status register.
0x102	RW	sedeleg	Supervisor exception delegation register.
0x103	RW	sideleg	Supervisor interrupt delegation register.
0x104	RW	sie	Supervisor interrupt-enable register.
0x105	RW	stvec	Supervisor trap handler base address.
0x106	RW	scounteren	Supervisor counter enable.
Supervisor Trap Handling			
0x140	RW	sscratch	Scratch register for supervisor trap handlers.
0x141	RW	sepc	Supervisor exception program counter.
0x142	RW	scause	Supervisor trap cause.
0x143	RW	stval	Supervisor bad address or instruction.
0x144	RW	sip	Supervisor interrupt pending.
Supervisor Protection and Translation			
0x180	RW	satp	Supervisor address translation and protection.
Supevisor Interrupts and Exceptions			
0xDA0	RO	scountovf	32-bit read-only register that contains shadow copies of the OF bits in the 29 mhpmevent CSRs.

Table 3-20 Machine Mode CSRs

Number	Privilege	Name	Description
Machine Information Registers			
0xF11	RO	mvendorid	Vendor ID.
0xF12	RO	marchid	Architecture ID.
0xF13	RO	mimpid	Implementation ID.
0xF14	RO	mhartid	Hardware thread ID.
Machine Trap Setup			
0x300	RW	mstatus	Machine status register.
0x301	RW	misa	ISA and extensions.
0x302	RW	medeleg	Machine exception delegation register.
0x303	RW	mideleg	Machine interrupt delegation register.
0x304	RW	mie	Machine interrupt-enable register.
0x305	RW	mtvec	Machine trap-handler base address.
0x306	RW	mcounteren	Machine counter enable.
Machine Trap Handling			
0x340	RW	mscratch	Scratch register for machine trap handlers.
0x341	RW	mepc	Machine exception program counter.
0x342	RW	mcause	Machine trap cause.
0x343	RW	mtval	Machine bad address or instruction.
0x344	RW	mip	Machine interrupt pending.
Machine Memory Protection			
0x3A0	RW	pmpcfg0	Physical memory protection configuration.
0x3A1	RW	pmpcfg1	Physical memory protection configuration, RV32 only.
0x3A2	RW	pmpcfg2	Physical memory protection configuration.
0x3A3	RW	pmpcfg3	Physical memory protection configuration, RV32 only.
0x3B0	RW	pmpaddr0	Physical memory protection address register.
0x3B1	RW	pmpaddr1	Physical memory protection address register.
		…	
0x3BF	RW	pmpaddr15	Physical memory protection address register.

Number	Privilege	Name	Description
Machine Counter/Timers			
0xB00	RW	mcycle	Machine cycle counter.
0xB02	RW	minstret	Machine instruction-retired counter.
Machine Counter Setup			
0x320	RW	mcountinhibit	Machine counter-inhibit register.
0x323	RW	mhpmevent3	Machine performance-monitoring event selector.
0x324	RW	mhpmevent4	Machine performance-monitoring event selector.
		…	
0x33F	RW	mhpmevent31	Machine performance-monitoring event selector.
Debug/Trace Register (shared with Debug Mode)			
0x7A0	RW	tselect	Debug/Trace trigger register select.
Number	Privilege	Name	Description
0x7A1	RW	tdata1	First Debug/Trace trigger data register.
0x7A2	RW	tdata2	Second Debug/Trace trigger data register.
0x7A3	RW	tdata3	Third Debug/Trace trigger data register.

Table 3-21 Debug Mode Registers

Number	Privilege	Name	Description
0x7B0	RW	dcsr	Debug control and status register.
0x7B1	RW	dpc	Debug PC.
0x7B2	RW	dscratch	Debug scratch register.

3.4.1.7 Zicsr Extension: Control and Status Register Instructions

- **Control and Status Registers**

The control and status registers (CSRs) are only accessible using variations of the CSRR (Read) and CSRRW (Write) instructions. Only the CPU executing the csr instruction can read or write these registers, and they are not visible by software outside of the core they reside on. The standard RISC–V ISA sets aside a 12-bit encoding space (csr[11:0]) for up to 4,096 CSRs. Attempts to access a non-existent CSR raise an illegal instruction exception. Attempts to access a CSR without appropriate privilege level or to write a read-only register also raise an illegal instruction exception. A read/write register may also contain some bits that are read-only, in which case, writes to the read-only bits are ignored. Each core functionality has its own con- trol and status registers which are described in the following section.

- **Defined CSRs**

The following tables describe the currently defined CSRs, categorized by privilege level. The usage of the CSRs below is implementation specific. CSRs are only accessible when operating within a specific

access mode (user mode, debug mode, supervisor mode, or machine mode). Therefore, attempts to access a non-existent CSR raise an illegal instruction exception, and attempts to access a CSR without appropriate privilege level or to write a read-only register also raise illegal instruction exceptions.

Table 3-22 User Mode CSRs

Number	Privilege	Name	Description
User Trap Setup			
0x000	RW	ustatus	User status register.
0x004	RW	uie	User interrupt-enable register.
0x005	RW	utvec	User trap handler base address.
User Trap Handling			
0x040	RW	uscratch	Scratch register for use trap handlers.
0x041	RW	uepc	User exception program counter.
0x042	RW	ucause	User trap cause.
0x043	RW	ubadaddr	User bad address.
0x044	RW	uip	User interrupt pending.
User Floating-Point CSRs			
0x001	RW	fflags	Floating-Point Accrued Exceptions.
0x002	RW	frm	Floating-Point Dynamic Rounding Mode.
0x003	RW	fcsr	Floating-Point Control and Status Register (frm + fflags).
User Counter/Timers			
0xC00	RO	cycle	Cycle counter for RDCYCLE instruction.
0xC01	RO	time	Timer for RDTIME instruction.
0xC02	RO	instret	Instructions-retired counter for RDINSTRET instruction.
0xC03	RO	hpmcounter3	Performance-monitoring counter.
0xC04	RO	hpmcounter4	Performance-monitoring counter.
		…	
0xC1F	RO	hpmcounter31	Performance-monitoring counter.

Table 3-23 Supervisor Mode CSRs

Number	Privilege	Name	Description
Supervisor Trap Setup			

Number	Privilege	Name	Description
0x100	RW	sstatus	Supervisor status register.
0x102	RW	sedeleg	Supervisor exception delegation register.
0x103	RW	sideleg	Supervisor interrupt delegation register.
0x104	RW	sie	Supervisor interrupt-enable register.
0x105	RW	stvec	Supervisor trap handler base address.
0x106	RW	scounteren	Supervisor counter enable.
Supervisor Trap Handling			
0x140	RW	sscratch	Scratch register for supervisor trap handlers.
0x141	RW	sepc	Supervisor exception program counter.
0x142	RW	scause	Supervisor trap cause.
0x143	RW	stval	Supervisor bad address or instruction.
0x144	RW	sip	Supervisor interrupt pending.
Supervisor Protection and Translation			
0x180	RW	satp	Supervisor address translation and protection.
Supevisor Interrupts and Exceptions			
0xDA0	RO	scountovf	32-bit read-only register that contains shadow copies of the OF bits in the 29 mhpmevent CSRs.

Table 3-24 Machine Mode CSRs

Number	Privilege	Name	Description
Machine Information Registers			
0xF11	RO	mvendorid	Vendor ID.
0xF12	RO	marchid	Architecture ID.
0xF13	RO	mimpid	Implementation ID.
0xF14	RO	mhartid	Hardware thread ID.
Machine Trap Setup			
0x300	RW	mstatus	Machine status register.
0x301	RW	misa	ISA and extensions.
0x302	RW	medeleg	Machine exception delegation register.
0x303	RW	mideleg	Machine interrupt delegation register.
0x304	RW	mie	Machine interrupt-enable register.

Number	Privilege	Name	Description
0x305	RW	mtvec	Machine trap-handler base address.
0x306	RW	mcounteren	Machine counter enable.
Machine Trap Handling			
0x340	RW	mscratch	Scratch register for machine trap handlers.
0x341	RW	mepc	Machine exception program counter.
0x342	RW	mcause	Machine trap cause.
0x343	RW	mtval	Machine bad address or instruction.
0x344	RW	mip	Machine interrupt pending.
Machine Memory Protection			
0x3A0	RW	pmpcfg0	Physical memory protection configuration.
0x3A1	RW	pmpcfg1	Physical memory protection configuration, RV32 only.
0x3A2	RW	pmpcfg2	Physical memory protection configuration.
0x3A3	RW	pmpcfg3	Physical memory protection configuration, RV32 only.
0x3B0	RW	pmpaddr0	Physical memory protection address register.
0x3B1	RW	pmpaddr1	Physical memory protection address register.
		…	
0x3BF	RW	pmpaddr15	Physical memory protection address register.
Machine Counter/Timers			
0xB00	RW	mcycle	Machine cycle counter.
0xB02	RW	minstret	Machine instruction-retired counter.
Machine Counter Setup			
0x320	RW	mcountinhibit	Machine counter-inhibit register.
0x323	RW	mhpmevent3	Machine performance-monitoring event selector.
0x324	RW	mhpmevent4	Machine performance-monitoring event selector.
		…	
0x33F	RW	mhpmevent31	Machine performance-monitoring event selector.
Debug/Trace Register (shared with Debug Mode)			
0x7A0	RW	tselect	Debug/Trace trigger register select.

Table 3-25 Machine Mode CSRs

Number	Privilege	Name	Description
0x7A1	RW	tdata1	First Debug/Trace trigger data register.
0x7A2	RW	tdata2	Second Debug/Trace trigger data register.
0x7A3	RW	tdata3	Third Debug/Trace trigger data register.

Table 3-26 Debug Mode Registers

Number	Privilege	Name	Description
0x7B0	RW	dcsr	Debug control and status register.
0x7B1	RW	dpc	Debug PC.
0x7B2	RW	dscratch	Debug scratch register.

- **RISC–V Status Registers**

The following sections further describe the RISC–V status registers.

- **Machine Status Register (mstatus)**

The mstatus register keeps track of and controls the hart's current operating state, including whether or not interrupts are enabled. Note that this is not a complete description of mstatus as it contains fields related to interrupts. For the full description of mstatus, consult the The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-27 Machine Status Register (partial)

Machine Status Register (mstatus)			
CSR	**0x300**		
Bits	**Field Name**	**Attr.**	**Description**
[14:13]	FS[1:0]	WARL	Floating-point context status
[16:15]	XS[1:0]	RO	Additional user-mode extensions and associated state status
17	MPRV	RW	Modify privilege level
18	SUM	RW	Permit supervisor user memory access
19	MXR	RW	Make executable readable
20	TVM	WARL	Trap virtual memory
21	TW	WARL	Timeout wait
22	TSR	WARL	Trap SRET
[31:23]	Reserved	WPRI	

Machine Status Register (mstatus)			
CSR	0x300		
Bits	Field Name	Attr.	Description
[33:32]	UXL[1:0]	WARL	User mode XLEN
[35:34]	SXL[1:0]	WARL	Supervisor mode XLEN
[62:36]	Reserved	WPRI	
63	SD	RO	FS dirty state status

Supervisor Status Register (sstatus)

Similar to machine mode, supervisor mode has a register dedicated to keeping track of the hart's current state called sstatus. sstatus is effectively a restricted view of mstatus, in that changes made to sstatus are reflected in mstatus and vice-versa, with the exception of the machine mode fields, which are not visible in sstatus.

Note that this is not a complete description of sstatus as it also contains fields related to interrupts. For the full description of sstatus, consult the The RISC–V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-28 Supervisor Status Register (partial)

Supervisor Status Register (sstatus)			
CSR	0x100		
Bits	Field Name	Attr.	Description
[14:13]	FS[1:0]	WARL	Floating-point unit state status
[16:15]	XS[1:0]	RO	Additional user-mode extensions and associated state status
17	Reserved	WPRI	
18	SUM	RW	Permit supervisor user memory access
19	MXR	RW	Make executable readable
[31:20]	Reserved	WPRI	
[33:32]	UXL[1:0]	WARL	User mode XLEN
[62:34]	Reserved	WPRI	
63	SD	RO	FS dirty state status

H Extension: Hypervisor-Level ISA

The RISC–V Hypervisor Extension virtualizes the supervisor-level architecture to support effi- cient hosting of guest operating systems atop a type-1 or type-2 hypervisor. The Hypervisor Extension changes supervisor mode into a hypervisor-extended supervisor mode, or hypervisor mode (HS-mode),

where a hypervisor or hosting-capable operating system runs.

- **Privilege Modes**

The current virtualization mode (V) indicates whether a hart is currently executing in a guest system. When V=1, the hart is either in virtual S-mode (VS-mode) or virtual U-mode (VU-mode) atop a guest OS running in VS-mode. When V=0, the hart is either in M-mode, HS-mode, or in U-mode atop an OS running in HS-mode. The virtualization mode also indicates whether two- stage address translation is active (V=1) or inactive (V=0). Table below describes the possible privilege modes with the Hypervisor Extension.

Table 3-29 Privilege Modes with the Hypervisor Extension

Virtualization Mode	Nominal Privilege	Name	Two-Stage Translation
0	U	User mode (U)	Off
0	S	Hypervisor mode (HS)	Off
0	M	Machine mode (M)	Off
1	U	Virtual user mode (VU)	On
1	S	Virtual supervisor mode (VS)	On

HS-mode is more privileged than VS-mode, and VS-mode is more privileged than VU-mode.

- **Hypervisor Extension CSRs**

The Hypervisor Extension adds additional CSRs to interact with the hypervisor and virtual supervisor level exception, interrupt, and address-translation subsystems. The hypervisor mode CSRs manage two-stage address translation and control the behavior of virtual supervisor mode guests. The virtual supervisor mode CSRs are replicas of the normal supervisor CSRs, taking over all functions of the usual supervisor CSRs while virtualization is enabled. Table below describes the CSRs added by the hypervisor extension.

Table 3-30 Hypervisor Extension CSRs

Number	Privilege	Name	Description
Machine Trap Handling			
0x34A	RW	mtinst	Machine trap instruction (transformed).
0x34B	RW	mtval2	Machine bad guest physical address.
Hypervisor Trap Setup			
0x600	RW	hstatus	Hypervisor status register.
0x602	RW	hedeleg	Hypervisor exception delegation register.
0x603	RW	hideleg	Hypervisor interrupt delegation register.
0x604	RW	hie	Hypervisor interrupt-enable register.
0x606	RW	hcounteren	Hypervisor counter enable.

Number	Privilege	Name	Description
0x607	RW	hgeie	Hypervisor guest external interrupt-enable register.
Hypervisor Trap Handling			
0x643	RW	htval	Hypervisor bad guest physical address.
0x644	RW	hip	Hypervisor interrupt pending.
0x645	RW	hvip	Hypervisor virtual interrupt pending.
0x64A	RW	htinst	Hypervisor trap instruction (transformed).
0xE12	RO	hgeip	Hypervisor guest external interrupt pending.
Hypervisor Protection and Translation			
0x680	RW	hgatp	Hypervisor guest address translation and protection.
Hypervisor Counter/Timer Virtualization Registers			
0x605	RW	htimedelta	Delta for VS/VU-mode timer.
Virtual Supervisor Registers			
0x200	RW	vsstatus	Virtual supervisor status register.
0x204	RW	vsie	Virtual supervisor interrupt-enable register.
0x205	RW	vstvec	Virtual supervisor trap handler base address.
0x240	RW	vsscratch	Virtual supervisor scratch register.
0x241	RW	vsepc	Virtual supervisor exception program counter.
0x242	RW	vscause	Virtual supervisor trap cause.
0x243	RW	vstval	Virtual supervisor bad address or instruction.
0x244	RW	vsip	Virtual supervisor interrupt pending.
0x280	RW	vsatp	Virtual supervisor address translation and protection.

- **Hypervisor Extension Status Registers**

The following sections further describe the RISC–V Hypervisor extension status registers.

- **Hypervisor Status Register (hstatus)**

A virtual supervisor mode guest has a register dedicated to keeping track of the hart's current state called hstatus, which functions analogously to mstatus. A summary of the hstatus fields related to hart state is provided in Table below. Note that this is not a complete description of hstatus as it also contains fields related to interrupts.

Table 3-31 Hypervisor Status Register (partial)

Hypervisor Status Register (hstatus)			
CSR	0x600		
Bits	Field Name	Attr.	Description
9	HU	RW	Hypervisor user mode
[19:10]	Reserved	WPRI	
20	VTVM	RW	Virtual supervisor mode trap virtual memory
21	VTW	RW	Virtual supervisor mode timeout wait
22	VTSR	RW	Virtual supervisor mode trap SRET
[31:23]	Reserved	WPRI	
[33:32]	VSXL[1:0]	WARL	Virtual supervisor mode XLEN

- **Virtual Supervisor Status Register (vsstatus)**

The vsstatus register is the virutal supervisor mode replica of the supervisor register sstatus. When virtualization is enabled, instructions that normally read or modify sstatus instead access vsstatus. A summary of the vsstatus fields related to hart state is provided in Table below. Note that this is not a complete description of vsstatus as it also contains fields related to interrupts.

Table 3-32 Virtual Supervisor Status Register (partial)

Virtual Supervisor Status Register (vsstatus)			
CSR	0x200		
Bits	Field Name	Attr.	Description
[14:13]	FS[1:0]	WARL	Floating-point unit state status
[16:15]	XS[1:0]	RO	Additional user-mode extensions and associated state status
17	Reserved	WPRI	
18	SUM	RW	Permit supervisor virtual user memory access
19	MXR	RW	Make executable readable
[31:20]	Reserved	WPRI	
[33:32]	UXL[1:0]	WARL	Virtual user mode XLEN
[62:34]	Reserved	WPRI	
63	SD	RO	FS dirty state status

- **Base Counters and Timers**

RISC–V ISAs provide a set of up to 32×64-bit performance counters and timers that are accessi- ble via unprivileged 64-bit read-only CSR registers 0xC00–0xC1F. The first three of these (CYCLE, TIME, and INSTRET) have dedicated functions; while the remaining counters, if imple- mented, provide programmable event counting.

The EIC7700X MCPU implements mcycle, mtime, and minstret counters, which have dedicated functions: cycle count, real-time clock, and instructions-retired, respectively. The timer functionality is based on the mtime register. Additionally, the EIC7700X MCPU implements event counters in the form of mhpmcounter, which is used to monitor user requested events

Table 3-33 Timer and Counter CSRs

Number	Privilege	Name	Description
0xC00	RO	cycle	Cycle counter for RDCYCLE instruction
0xC01	RO	time	Timer for RDTIME instruction
0xC02	RO	instret	Instruction-retired counter for RDINSTRET instruction

- **Floating-Point Control and Status Registers**

Floating-Point Control and Status Register, fcsr, is a RISC–V control and status register (CSR). The register selects the dynamic rounding mode for floating-point arithmetic operations and holds the accrued exception flags.

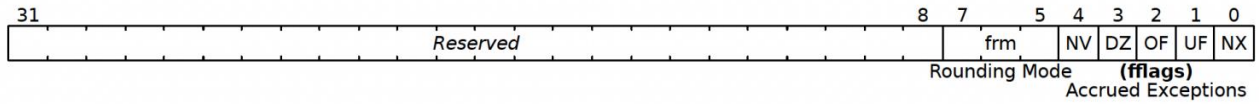


Table 3-34 Accrued Exception Flags

Flag Mnemonic	Flag Meaning
NV	Invalid Operation
DZ	Divide by Zero
OF	Overflow
UF	Underflow
NX	Inexact

- **Rounding Modes**

Floating-point operations use either a static rounding mode encoded in the instruction, or a dynamic rounding mode held in frm. A value of 111 in the instruction's rm field selects the dynamic rounding mode held in frm. If frm is set to an invalid value (101–111), any subsequent attempt to execute a floating-point operation with a dynamic rounding mode will raise an illegal instruction exception. Some instructions, including widening conversions, have the rm field, but are nevertheless unaffected by the rounding mode. Software should set their rm field to RNE (000).

Table 3-35 Floating-Point Rounding Modes

Mode	Mnemonic	Meaning
000	RNE	Round to Nearest, ties to Even.
001	RTZ	Round towards Zero.
010	RDN	Round Down (towards - ∞).
011	RUP	Round Up (towards + ∞).
100	RMM	Round to Nearest, ties to Max Magnitude.
101		Invalid. Reserved for future use.
110		Invalid. Reserved for future use.
111	DYN	In instruction's rm field, selects dynamic rounding mode; In Rounding Mode register, Invalid.

3.4.1.8 Ports

This section describes the Port interfaces to the EIC7700X MCPU core.

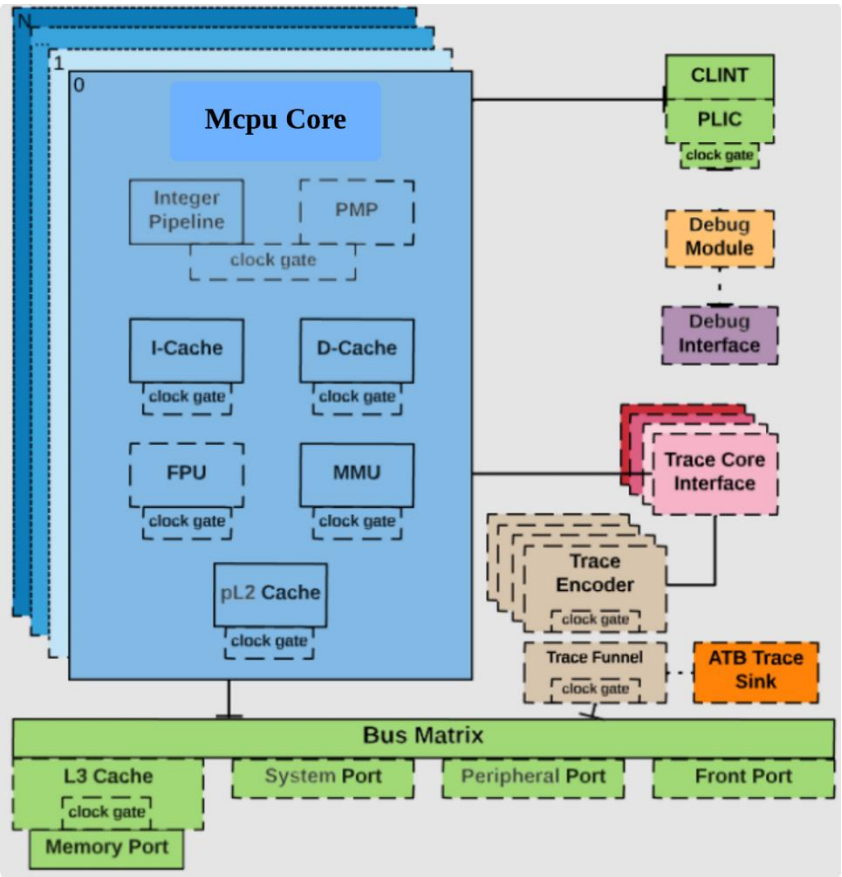


Figure 3-8 EIC7700X MCPU

- **Front Port**

The Front Port can be used by external masters to read from and write into the memory system utilizing

any Memory Port in the Core Complex. Transactions cannot be routed from the Front Port to any System or Peripheral Ports.

If a Front Port access targets the Memory Port, a coherency manager is responsible for maintaining coherency with the L1, pL2, and L3 caches. A read access can be returned directly from the caches without generating an external bus access. If a write from the Front Port targets a location in the L1 data cache or pL2 cache, it results in the line being evicted and invalidated. The write will then allocate to the L3 cache.

Any Front Port access that targets the Memory Port and results in an L1 and pL2 cache miss will allocate to the L3 cache.

- **Memory Port**

The Memory Port is used to interface with memory that offers the highest performance for the EIC7700X MCPU, such as DDR. It supports cacheable accesses for data and instructions.

- **System Port**

The System Port is used to interface with lower performance memory, like SRAM, memory mapped I/O (MMIO), and higher speed peripherals. The System Port also supports code execution.

Table below describes the basic properties of each port:

Table 3-36 Port Properties

Port Type	Supports Atomics?	Supports Bursts?	Cacheable?	Description
Front Port	No	Yes	No	Used by external masters to access devices and ports within the Core Complex
Memory Port	Yes	Yes	Yes	Used to access cacheable on-chip and off-chip memory
System Port	No	Yes	No	Used typically for high-bandwidth, uncached memory or devices
Peripheral Port	Yes	No	No	Used primarily for peripheral devices and low-bandwidth memories
• On the Memory Port, the core acquires the cache block and performs the atomic operation in the cache. The new value will be written back to memory when the cache evicts the block. • On the Peripheral Port, the core sends an atomic memory access to the bridge for that port, which then decomposes the request into a read-modify-write bus cycle.				

3.4.1.9 Physical Memory Attributes and Memory Map

This chapter describes the EIC7700X MCPU physical memory attributes and memory map. The memory map is divided into different regions covering on-core-complex memory, system memory, peripherals, and empty holes. Physical memory attributes (PMAs) describe the properties of the accesses that can be made to each region in the memory map. These properties encompass the type of access that may be performed: execute, read, or write. As well as other optional attributes related to the access, such as supported access size, alignment, atomic operations, and cacheability.

The assigned PMA properties and attributes for EIC7700X MCPU memory regions are shown in Table below.

Table 3-37 Physical Memory Attributes for External Regions

Port Type	Access Properties	Attributes
Memory Port	Read, Write, Execute	Atomics+LR/SC, Data Cacheable, Instruction Cacheable, Instruction Speculation
System Port	Read, Write, Execute	Instruction Cacheable

Table 3-38 Physical Memory Attributes for Internal Regions

Region	Access Properties	Attributes
Burst Bundler on Front Port	Read, Write, Execute	Atomics
Bus Blocker on TL256D2D_In (TL-C slave)	Read, Write	Atomics
Bus Blocker on TL256D2D_Out (TL-C master)	Read, Write	Atomics
Bus Blocker on TL64D2D_Out (TL-UH master)	Read, Write	Atomics
Bus-Error Unit	Read, Write, Execute	Atomics
CLINT	Read, Write, Execute	Atomics
Debug	None	N/A
Error Device	Read, Write, Execute	Atomics
Error Device	Read, Write, Execute	Atomics, Instruction Cacheable
L3 Cache Controller	Read, Write, Execute	Atomics
L3 LIM	Read, Write, Execute	Atomics
L3 Zero Device	Read, Write, Execute	Atomics, Data Cacheable, Instruction Cacheable
PLIC	Read, Write, Execute	Atomics
Private L2 Cache	Read, Write, Execute	Atomics
TL-UH to TL-C Adapter	Read, Write, Execute	Atomics
Test Indicator	Read, Write	Atomics
Trace Encoder	Read, Write, Execute	Atomics

Trace Funnel	Read, Write, Execute	Atomics
Unmapped	None	N/A

All memory map regions support word, half-word, and byte size data accesses.

Atomic access support enables the RISC–V standard Atomic (A) Extension for atomic instructions. The load-reserved (LR) and store-conditional (SC) instructions are only supported on the data cacheable region.

No region supports unaligned accesses. An unaligned access will generate the appropriate trap: instruction address misaligned, load address misaligned, or store/AMO address misaligned.

The Physical Memory Protection unit can control access properties based on address ranges, not ports. It has no control over the attributes of an address range, however.

- Memory Map

Note

The Debug and Error Device regions have special behavior. The Debug region is reserved for use from a Debugger, and all accesses to it from the core in non-Debug mode will trap. The Error Device will also trap all accesses.

The memory map of the EIC7700X MCPU is shown in Table below.

Table 3-39 EIC7700X MCPU Memory Map

Die	Function	Size	Base	Top	EIC7700X MCPU Port	Notes
D0/ D1	Local SRAM, MMIO, Peripherals	768GB	0x140_0000_0000	0x1FF_FFFF_FFFF	System Port 1	Non-coherent memory
D0/ D1	Interleaved DRAM	256GB	0x100_0000_0000	0x13F_FFFF_FFFF	System Port 1	DRAM interleaved with 256B granularity over D0 and D1 for non-coherent memory. Done by EIC7700X MCPU.
	D1 DRAM	128GB	0xE0_0000_0000	0xFF_FFFF_FFFF	System Port 1	Non-coherent
	D0 DRAM	128GB	0xC0_0000_0000	0xDF_FFFF_FFFF	System Port 1	Non-coherent
D1	MMIO, Peripherals	128GB	0xA0_0000_0000	0xBF_FFFF_FFFF	System Port 1	D1 higher MMIO/ peripherals
D0	MMIO, Peripherals	128GB	0x80_0000_0000	0x9F_FFFF_FFFF	System Port 1	D0 higher MMIO/ peripherals

D0/ D1	Interleaved DRAM	256GB	0x40_0000_0000	0x7F_FFFF_FFFF	Memory Port or TL256D2D	DRAM interleaved with 256B granularity over D0 and D1 for coherent memory.
	D1 DRAM	128GB	0x20_0000_0000	0x3F_FFFF_FFFF		Coherent
	D0 DRAM	126GB	0x00_8000_0000	0x1F_FFFF_FFFF		Coherent
D1	MMIO, Peripherals	512MB	0x00_6000_0000	0x00_7FFF_FFFF	System Port 0	D1 lower MMIO/ peripherals
D0	MMIO, Peripherals	512MB	0x00_4000_0000	0x00_5FFF_FFFF	System Port 0	D0 lower MMIO/ peripherals
D1	EIC7700X MCPU internals	512MB	0x00_2000_0000	0x00_3FFF_FFFF	Internal or TL64D2D	
D0	EIC7700X MCPU internals	512MB	0x00_0000_0000	0x00_1FFF_FFFF		

The memory map of a single EIC7700X MCPU Core Complex is shown in Table below

Table 3-40 EIC7700X MCPU Core Complex Memory Map. Physical Memory Attributes: R–Read, W–Write, X–Execute, I–Instruction Cacheable, D–Data Cacheable, A–Atomics

Base	Top	PMA	Description
0x0000_0000_0000	0x0000_0000_0FFF		Debug
0x0000_0000_1000	0x0000_0000_3FFF	RWX A	Error Device
0x0000_0000_4000	0x0000_0000_4FFF	RW A	Test Indicator
0x0000_0000_5000	0x0000_0001_7FFF	RWX A	Error Device
0x0000_0001_8000	0x0000_0001_8FFF	RWX A	Trace Funnel
0x0000_0001_9000	0x0000_000F_FFFF	RWX A	Error Device
0x0000_0010_0000	0x0000_0010_0FFF	RWX A	Trace Encoder 0
0x0000_0010_1000	0x0000_0010_1FFF	RWX A	Trace Encoder 1
0x0000_0010_2000	0x0000_0010_2FFF	RWX A	Trace Encoder 2
0x0000_0010_3000	0x0000_0010_3FFF	RWX A	Trace Encoder 3
0x0000_0010_4000	0x0000_0010_7FFF	RWX A	Hart 0 Private L2 Cache
0x0000_0010_8000	0x0000_0010_BFFF	RWX A	Hart 1 Private L2 Cache
0x0000_0010_C000	0x0000_0010_FFFF	RWX A	Hart 2 Private L2 Cache
0x0000_0011_0000	0x0000_0011_3FFF	RWX A	Hart 3 Private L2 Cache

Base	Top	PMA	Description
0x0000_0011_4000	0x0000_001F_FFFF	RWX A	Error Device
0x0000_0020_0000	0x0000_0020_0FFF	RW A	Bus Blocker on TL64D2D_Out (TL-UH master)
0x0000_0020_1000	0x0000_0020_1FFF	RWX A	Error Device
0x0000_0020_2000	0x0000_0020_2FFF	RW A	Bus Blocker on TL256D2D_Out (TL-C master)
0x0000_0020_3000	0x0000_0020_3FFF	RWX A	Error Device
0x0000_0020_4000	0x0000_0020_4FFF	RW A	Bus Blocker on TL256D2D_In (TL-C slave)
0x0000_0020_5000	0x0000_016F_FFFF	RWX A	Error Device
0x0000_0170_0000	0x0000_0170_0FFF	RWX A	Hart 0 Bus-Error Unit
0x0000_0170_1000	0x0000_0170_1FFF	RWX A	Hart 1 Bus-Error Unit
0x0000_0170_2000	0x0000_0170_2FFF	RWX A	Hart 2 Bus-Error Unit
0x0000_0170_3000	0x0000_0170_3FFF	RWX A	Hart 3 Bus-Error Unit
0x0000_0170_4000	0x0000_01FF_FFFF	RWX A	Error Device
0x0000_0200_0000	0x0000_0200_FFFF	RWX A	CLINT
0x0000_0201_0000	0x0000_0201_3FFF	RWX A	L3 Cache Controller
0x0000_0201_4000	0x0000_07FF_FFFF	RWX A	Error Device
0x0000_0800_0000	0x0000_083F_FFFF	RWX A	L3 LIM
0x0000_0840_0000	0x0000_0BFF_FFFF	RWX A	Error Device
0x0000_0C00_0000	0x0000_0FFF_FFFF	RWX A	PLIC
0x0000_1000_0000	0x0000_1000_2FFF	RWX A	Error Device
0x0000_1000_3000	0x0000_1000_3FFF	RWXI A	Error Device
0x0000_1000_4000	0x0000_1000_FFFF	RWX A	Error Device
0x0000_1001_0000	0x0000_1001_0FFF	RWX A	Burst Bundler on Front Port
0x0000_1001_1000	0x0000_1002_FFFF	RWX A	Error Device
0x0000_1003_0000	0x0000_1003_3FFF	RWX A	TL-UH to TL-C Adapter
0x0000_1003_4000	0x0000_19FF_FFFF	RWX A	Error Device
0x0000_1A00_0000	0x0000_1A3F_FFFF	RWXIDA	L3 Zero Device
0x0000_1A40_0000	0x0000_1FFF_FFFF	RWX A	Error Device
0x0000_2000_0000	0x0000_3FFF_FFFF		Die 1 EIC7700X MCPU Internals
0x0000_4000_0000	0x0000_7FFF_FFFF	RWXI	System Port 0 (1 GiB)
0x0000_8000_0000	0x007F_FFFF_FFFF	RWXIDA	Memory Port 0 (510 GiB)

Base	Top	PMA	Description
0x0080_0000_0000	0x01FF_FFFF_FFFF	RWXI	System Port 1 (1.5 TiB)
0x0200_0000_0000	0xFFFF_FFFF_FFFF		Unmapped

Region 0x0 to 0x1FFF_FFFF maps to the internals of the EIC7700X MCPU in Die 0. Each module of the EIC7700X MCPU in Die 1 is mapped to the same location with an offset of 0x2000_0000. For instance, the CLINT of Die 0 is accessible at 0x200_0000 and the CLINT of Die 1 is accessible at 0x2200_0000.

Note

The lower 2GB is carved out from the Memory Port on Region 0. The lower 1GB is used to access internal resources of the P550-MC Core Complex, and the other 1GB is allocated to the System Port.

- **Memory Interleaving**

To allow the two sockets to operate as a single cache-coherent Core Complex, the two SBuses can access all banks of the combined L3 over the D2D interface. The EIC7700X MCPU has a memory interleaving scheme, see Figure below, that consists of four memory regions, organized as follows:

- **Region 0 is a 126GB region starting at 2GB located on Die 0**
- **Region 1 is a 128GB region starting at 128GB located on Die 1**
- **Regions 2 and 3 are 128GB each, interleaved at a 256B granularity among Die 0 and Die 1 (with a hiccup every 256KB)**

Note

The lower 2GB is carved out from the Memory Port on Region 0. The lower 1GB is used to access internal resources of the P550-MC Core Complex, and the other 1GB is allocated to the System Port.

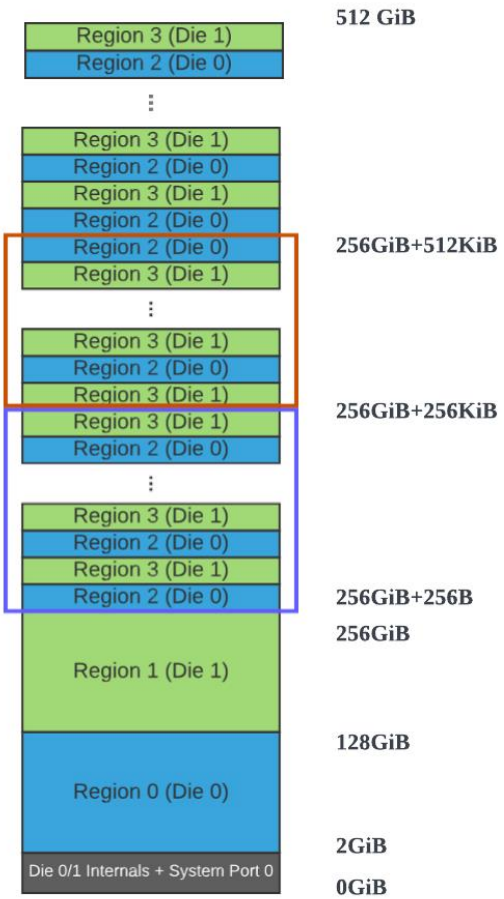


Figure 3-9 EIC7700X MCPU Memory Interleaving Scheme

The memory interleaving scheme on the Memory Port region follows the below rules:

- Region 0 on die 0: address(38) = 0 && address(37) = 0
- Region 1 on die 1: address(38) = 0 && address(37) = 1
- Region 2 on die 0: address(38) = 1 && address(8) XOR address(18) = 0
- Region 3 on die 1: address(38) = 1 && address(8) XOR address(18) = 1

3.4.1.10 Interrupts and Exceptions

This chapter describes how interrupt and exception concepts in the RISC-V architecture apply to the EIC7700X MCPU.

- **Interrupt Concepts**

Interrupts are asynchronous events that cause program execution to change to a specific location

in the software application to handle the interrupting event.

The EIC7700X MCPU supports machine mode, hypervisor-extended supervisor mode, and virtual supervisor mode interrupts. By default, all interrupts are handled in machine mode. For harts that support supervisor mode, it is possible to selectively delegate interrupts to supervisor mode. For harts that support virtual supervisor mode, it is possible to further delegate interrupts o virtual supervisor mode.

The Core Complex also has support for the following types of RISC-V interrupts: local and global. Local

interrupts are signaled directly to an individual hart with a dedicated interrupt exception code and fixed priority. This allows for reduced interrupt latency as no arbitration is required to determine which hart will service a given request and no additional memory accesses are required to determine the cause of the interrupt. Software and timer interrupts are local interrupts generated by the Core-Local Interruptor (CLINT). The EIC7700X MCPU contains no other local interrupt sources.

Global interrupts are routed through a Platform-Level Interrupt Controller (PLIC), which can direct interrupts to any hart in the system via the external interrupt. Decoupling global interrupts from the harts allow the design of the PLIC to be tailored to the platform, permitting a broad range of attributes like the number of interrupts and the prioritization and routing schemes.

- **Exception Concepts**

Exceptions are different from interrupts in that they typically occur synchronously to the instruction execution flow, and most often are the result of an unexpected event that results in the program to enter an exception handler. For example, if a hart is operating in supervisor mode and attempts to access a machine mode only Control and Status Register (CSR), it will immediately enter the exception handler and determine the next course of action. The exception code in the mstatus register will hold a value of 0x2, showing that an illegal instruction exception occurred. Based on the requirements of the system, the supervisor mode application may report an error and/or terminate the program entirely.

There are no specific enable bits to allow exceptions to occur since they are always enabled by default. However, early in the boot flow, software should set up mtvec.BASE to a defined value, which contains the base address of the default exception handler. All exceptions will trap to mtvec.BASE. Software must read the mcause CSR to determine the source of the exception, and take appropriate action.

Synchronous exceptions that occur from within an interrupt handler will immediately cause program execution to abort the interrupt handler and enter the exception handler. Exceptions within an interrupt handler are usually the result of a software bug and should generally be avoided since mepc and mcause CSRs will be overwritten from the values captured in the original interrupt context.

The RISC-V defined synchronous exceptions have a priority order which may need to be considered when multiple exceptions occur simultaneously from a single instruction. Table below describes the synchronous exception priority order.

Table 3-41 Exception Priority

Priority	Exception Code	Description
Highest	3	Instruction address breakpoint
	12	Instruction page fault
	20	Instruction guest-page fault
	1	Instruction access fault
	2	Illegal instruction
	22	Virtual instruction
	0	Instruction address misaligned
	8, 9, 10, 11	Environment call
	3	Environment break
	3	Load/Store/AMO address breakpoint
	6	Store/AMO address misaligned
	4	Load address misaligned

Priority	Exception Code	Description
	15	Store/AMO page fault
	13	Load page fault
	23	Store/AMO guest-page fault
	21	Load guest-page fault
Lowest	7	Store/AMO access fault
	5	Load access fault

Table 3-42 mcause Exception Codes

Interrupt	Exception Code	Description
1	0	Reserved
1	1	Supervisor software interrupt
1	2	Virtual supervisor software interrupt
1	3	Machine software interrupt
1	4	Reserved
1	5	Supervisor timer interrupt
1	6	Virtual supervisor timer interrupt
1	7	Machine timer interrupt
1	8	Reserved
1	9	Supervisor external interrupt
1	10	Virtual supervisor external interrupt
1	11	Machine external interrupt
1	12	Supervisor guest external interrupt
1	13	HPM Counter Overflow
1	14	Debug interrupt
1	≥15	Reserved
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault

Interrupt	Exception Code	Description
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode or VU-mode
0	9	Environment call from HS-mode
0	10	Environment call from VS-mode
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	Debug exception
0	15	Store/AMO page fault
0	16–19	Reserved
0	20	Instruction guest-page fault
0	21	Load guest-page fault
0	22	Virtual instruction
0	23	Store/AMO guest-page fault
0	≥24	Reserved

Some of the helpful CSRs for debugging exceptions and interrupts are described below:

Table 3-43 summary of Exception and Interrupt CSRs

CSR	Description
exception	Scope signal. Indicates the moment that an exception occurs in the write-back (commit) stage.
mcause	Contains the cause value of the exception/interrupt.
mepc	Contains the pc where the exception occurs.
mtval	If the cause is a load/store fault, this register has the value of the problematic address. If it is an invalid instruction, it provides the instruction that the core tried to execute.
mstatus	Contains the interrupt enables, privilege modes, and general status of execution.
mtvec	Contains the vector that the core will jump to when an exception occurs. If this is not a valid executable value, you may get a double exception when jumping to the exception handler, so it is important to look at all these registers when the exception FIRST occurs.

- **Trap Concepts**

The term trap describes the transfer of control in a software application, where trap handling typically executes in a more privileged environment. For example, a particular hart contains three privilege modes: machine, supervisor, and user. Each privilege mode has its own software execution environment including a dedicated stack area. Additionally, each privilege mode contains separate control and status registers (CSRs) for trap handling. While operating in user mode, a context switch is required to handle an event in supervisor mode. The software sets up the system for a context switch and then an ECALL instruction is executed, which synchronously switches control to the environment-call-from-user-mode exception handler.

The default mode out of reset is machine mode. Software begins execution at the highest privilege level, which allows all CSRs and system resources to be initialized before any privilege level changes. The steps below describe the required steps necessary to change privilege mode from machine to user mode, on a particular design that also includes supervisor mode.

1. Interrupts should first be disabled globally by writing mstatus.MIE to 0, which is the default reset value.

2. Write mtvec CSR with the base address of the machine mode exception handler. This is a required step in any boot flow.

3. Write mstatus.MPP to 0 to set the previous mode to user, which allows us to return to that mode.

4. Setup the Physical Memory Protection (PMP) regions to grant the required regions to user and supervisor mode, and optionally, revoke permissions from machine mode.

5. Write stvec CSR with the base address of the supervisor mode exception handler.

6. Write medeleg register to delegate exceptions to supervisor mode. Consider ECALL and page fault exceptions.

7. Write mstatus.FS to enable floating-point (if supported).

8. Store machine mode user registers to stack or to an application-specific frame pointer.

9. Write mepc with the entry point of user mode software

10. Execute mret instruction to enter user mode.

> **Note**
>
> There is only one set of user registers (`x1-x31`) that are used across all privilege levels, so application software is responsible for saving and restoring state when entering and exiting different levels.

- **Interrupt Block Diagram**

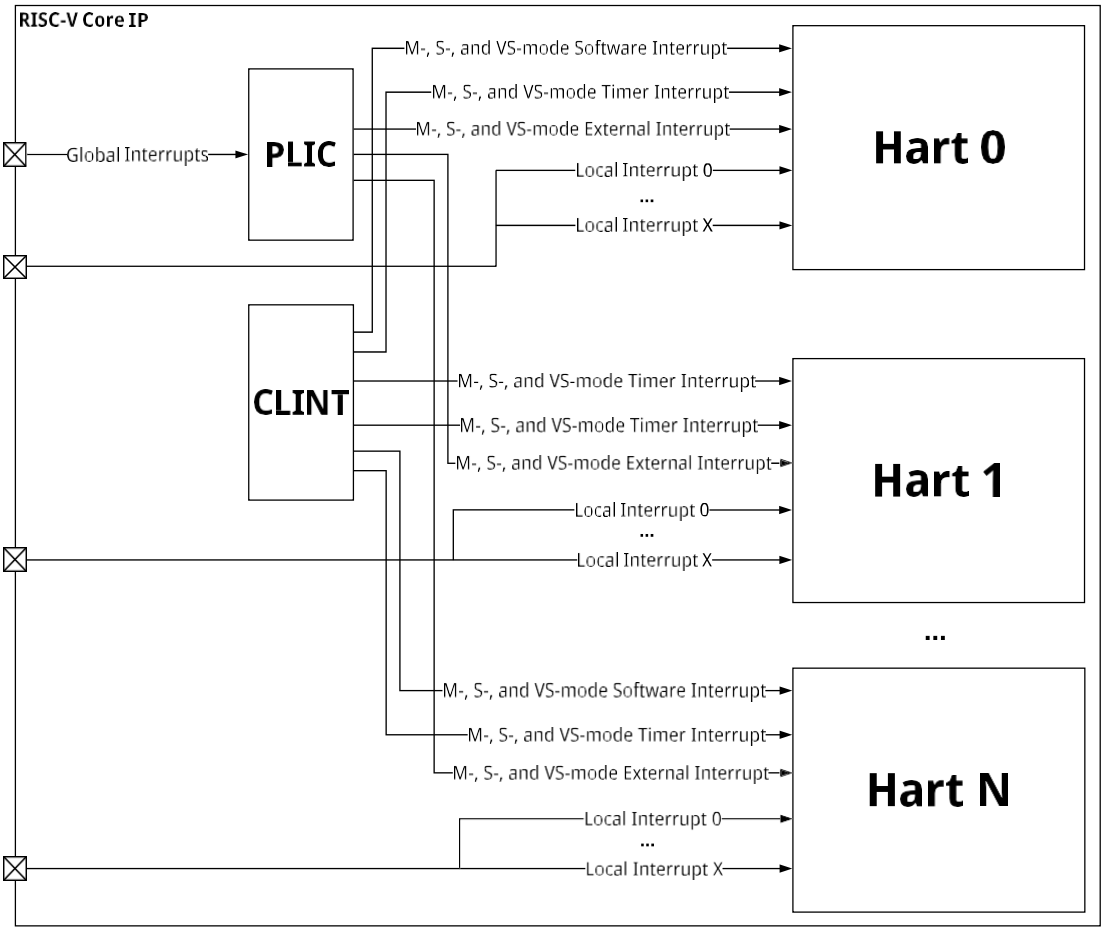


Figure 3-10 EIC7700X MCPU Interrupt Architecture Block Diagram

- **Local Interrupts**

Software interrupts (Interrupt ID #3) are triggered by writing the memory-mapped interrupt pending register msip for a particular hart. Other harts are able to write msip to trigger a software interrupt on any other hart in the EIC7700X MCPU. This allows for efficient interprocessor communication.

Timer interrupts (Interrupt ID #7) are triggered when the memory-mapped register mtime is greater than or equal to the global timebase register mtimecmp, and both registers are part of the CLINT memory map. mtimecmp can be written by other harts to set up timer interrupts. The mtime and mtimecmp registers are generally only available in machine mode, unless the PMP grants user or supervisor mode access to the memory-mapped region in which they reside.

Global interrupts are usually first routed to the PLIC, then into the hart using external interrupts (Interrupt ID #11).

- **Interrupt Operation**

Within a privilege mode M, if the associated global interrupt-enable {ie} is clear, then no interrupts will be taken in that privilege mode, but a pending-enabled interrupt in a higher privilege mode will preempt current execution. If {ie} is set, then pending-enabled interrupts at a higher interrupt level in the same privilege mode will preempt current execution and run the interrupt handler for the higher interrupt level.

When an interrupt or synchronous exception is taken, the privilege mode is modified to reflect the new privilege mode. The global interrupt-enable bit of the handler's privilege mode is cleared.

- **Interrupt Entry and Exit**

When an interrupt occurs:

• The value of mstatus.MIE is copied into mstatus.MPIE, and then mstatus.MIE is cleared, effectively disabling interrupts.

• The privilege mode prior to the interrupt is encoded in mstatus.MPP and the virtualization mode is encoded in mstatus.MPV.

• The current pc is copied into the mepc register, and then pc is set to the value specified by mtvec as defined by the mtvec.MODE.

At this point, control is handed over to software in the interrupt handler with interrupts disabled.

When an mret instruction is executed, the following occurs:

• The privilege mode is set to the value encoded in mstatus.MPP. If mstatus.MPP is not machine mode, the virtualization mode is set to the value encoded in mstatus.MPV.

• The global interrupt enable, mstatus.MIE, is set to the value of mstatus.MPIE.

• The pc is set to the value of mepc.

At this point, control is handed over to software.

- **Interrupt Control and Status Registers**

The EIC7700X MCPU specific implementation of interrupt CSRs is described below. For a complete description of RISC-V interrupt behavior and how to access CSRs, please consult The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

- **Machine Status Register (mstatus)**

The mstatus register keeps track of and controls the hart's current operating state, including whether or not interrupts are enabled. A summary of the mstatus fields related to interrupts in the EIC7700X MCPU is provided in Table below. Note that this is not a complete description of mstatus as it contains fields unrelated to interrupts. For the full description of mstatus, please consult The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-44 Machine Status Register (partial)

Machine Status Register (mstatus)			
CSR	0x300		
Bits	Field Name	Attr.	Description
0	Reserved	WPRI	
1	SIE	RW	Supervisor Interrupt Enable
2	Reserved	WPRI	
3	MIE	RW	Machine Interrupt Enable
4	Reserved	WPRI	
5	SPIE	RW	Supervisor Previous Interrupt Enable
6	Reserved	WPRI	
7	MPIE	RW	Machine Previous Interrupt Enable
8	SPP	RW	Supervisor Previous Privilege Mode
[10:9]	Reserved	WPRI	

Machine Status Register (mstatus)			
CSR	0x300		
Bits	Field Name	Attr.	Description
[12:11]	MPP[1:0]	RW	Machine Previous Privilege Mode
…			
38	GVA	RW	Guest Virtual Address
39	MPV	RW	Machine Previous Virtualization Mode

Interrupts are enabled by setting the MIE bit in mstatus. Prior to writing mstatus.MIE=1, it is recommended to first enable interrupts in mie.

The privilege mode encoding for mstatus.MPP is described in Table below.

Table 3-45 Encoding of RISC-V Privilege Levels

Level	Encoding	Name
0	00	User mode (U)
1	01	Supervisor mode (S)
2	10	Reserved
3	11	Machine mode (M)

For mstatus.MPV, 0 indicates that virtualization mode is enabled.

- **Machine Trap Vector (mtvec)**

The mtvec register has two main functions: defining the base address of the trap vector, and setting the mode by which the EIC7700X MCPU will process interrupts. For Direct and Vectored modes, the interrupt processing mode is defined in the MODE field of the mtvec register. The mtvec register is described in Table below.

Table 3-46 Machine Trap Vector Register

Machine Trap Vector Register (mtvec)			
CSR	0x305		
Bits	Field Name	Attr.	Description
[1:0]	MODE	WARL	MODE Sets the interrupt processing mode. The encoding for the EIC7700X MCPU
[63:2]	BASE[63:2]	WARL	Interrupt Vector Base Address. Operating in Direct Mode requires 4-byte alignment. Operating in Vectored Mode requires 256-byte alignment.

Table 3-47 Encoding of mtvec.MODE

MODE Field Encoding mtvec.MODE		
Value	**Mode**	**Description**
0x0	Direct	All asynchronous interrupts and synchronous exceptions set pc to BASE.
0x1	Vectored	Exceptions set pc to BASE, interrupts set pc to BASE + 4 × mcause.EXCCODE.
≥0x2	Reserved	

- **Mode Direct**

When operating in direct mode, all interrupts and exceptions trap to the mtvec.BASE address. Inside the trap handler, software must read the mcause register to determine what triggered the trap. The mcause register is described in Table below.

When operating in Direct Mode, BASE must be 4-byte aligned.

- **Mode Vectored**

While operating in vectored mode, interrupts set the pc to mtvec.BASE + 4 Å~ exception code (mcause.EXCCODE). For example, if a machine timer interrupt is taken, the pc is set to mtvec.BASE + 0x1C. Typically, the trap vector table is populated with jump instructions to transfer control to interrupt-specific trap handlers.

- **Machine Interrupt Enable (mie)**

Individual interrupts are enabled by setting the appropriate bit in the mie register. The mie register

is described in Table below.

Table 3-48 Machine Interrupt Enable Register

Machine Interrupt Enable Register (mie)			
CSR	**0x304**		
Bits	**Field Name**	**Attr.**	**Description**
0	Reserved	WPRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
2	VSSIE	RW	Virtual Supervisor Software Interrupt Enable
3	MSIE	RW	Machine Software Interrupt Enable
4	Reserved	WPRI	
5	STIE	RW	Supervisor Timer Interrupt Enable
6	VSTIE	RW	Virtual Supervisor Timer Interrupt Enable
7	MTIE	RW	Machine Timer Interrupt Enable

Machine Interrupt Enable Register (mie)			
CSR	0x304		
Bits	Field Name	Attr.	Description
8	Reserved	WPRI	
9	SEIE	RW	Supervisor External Interrupt Enable
10	VSEIE	RW	Virtual Supervisor External Interrupt Enable
11	MEIE	RW	Machine External Interrupt Enable
12	SGEIE	RW	Supervisor Guest External Interrupt Enable
13	LCOFIE	RW	Local Count Overflow Interrupt Enable
[63:14]	Reserved	WPRI	

- **Machine Interrupt Pending (mip)**

The machine interrupt pending (mip) register indicates which interrupts are currently pending. The mip register is described in Table below.

Table 3-49 Machine Interrupt Pending Register

Machine Interrupt Pending Register (mip)			
CSR	0x344		
Bits	Field Name	Attr.	Description
0	Reserved	WIRI	
1	SSIP	RW	Supervisor Software Interrupt Pending
2	VSSIP	RW	Virtual Supervisor Software Interrupt Pending
3	MSIP	RO	Machine Software Interrupt Pending
4	Reserved	WIRI	
5	STIP	RW	Supervisor Timer Interrupt Pending
6	VSTIP	RW	Virtual Supervisor Timer Interrupt Pending
7	MTIP	RO	Machine Timer Interrupt Pending
8	Reserved	WIRI	
9	SEIP	RW	Supervisor External Interrupt Pending
10	VSEIP	RW	Virtual Supervisor External Interrupt Pending
11	MEIP	RO	Machine External Interrupt Pending
12	SGEIP	RW	Supervisor Guest External Interrupt Pending
13	LCOFIP	RW	Local Count Overflow Interrupt Pending
[63:14]	Reserved	WPRI	

- **Machine Cause (mcause)**

When a trap is taken in machine mode, mcause is written with a code indicating the event that caused the trap. When the event that caused the trap is an interrupt, the most-significant bit of mcause is set to 1, and the least-significant bits indicate the interrupt number, using the same encoding as the bit positions in mip. For example, a Machine Timer Interrupt causes mcause to be set to 0x8000_0000_0000_0007. mcause is also used to indicate the cause of synchronous exceptions, in which case the most-significant bit of mcause is set to 0.

Table 3-50 Machine Cause Register

Machine Cause Register (mcause)			
CSR	0x342		
Bits	Field Name	Attr.	Description
[9:0]	EXCCODE	WLRL	A code identifying the last exception.
[62:10]	Reserved	WLRL	
63	Interrupt	WARL	1, if the trap was caused by an interrupt; 0 otherwise.

Note that there are scenarios where a misaligned load or store will generate an access exception instead of an address-misaligned exception. The access exception is raised when the mis- aligned access should not be emulated in a trap handler, e.g., emulating an access in an I/O region, as such emulation could cause undesirable side-effects.

- **Minimum Interrupt Configuration**

The minimum configuration needed to configure an interrupt is shown below.

• Write mtvec to configure the interrupt mode and the base address for the interrupt vector table.

• Enable interrupts in memory-mapped PLIC register space. The CLINT does not contain interrupt enable bits.

• Write mie CSR to enable the software, timer, and external interrupt enables for each privilege mode.

• Write mstatus to enable interrupts globally for each supported privilege mode.

- **Supervisor Mode Interrupts**

The EIC7700X MCPU supports the ability to selectively direct interrupts and exceptions to supervisor mode, resulting in improved performance by eliminating the need for additional mode changes.

This capability is enabled by the interrupt and exception delegation CSRs; mideleg and medeleg, respectively. Supervisor interrupts and exceptions can be managed via supervisor versions of the interrupt CSRs, specifically: stvec, sip, sie, and scause.

Machine mode software can also directly write to the sip register, which effectively sends an interrupt to supervisor mode. This is especially useful for timer and software interrupts as it may be desired to handle these interrupts in both machine mode and supervisor mode.

The delegation and supervisor CSRs are described in the sections below. The definitive resource for information about RISC-V supervisor interrupts is The RISC-V Instruction Set Manual, Volume II:

Privileged Architecture, Version 1.11.

- **Delegation Registers (mideleg and medeleg)**

By default, all traps are handled in machine mode. Machine mode software can selectively delegate interrupts and exceptions to supervisor mode by setting the corresponding bits in mideleg and medeleg CSRs. The exact mapping is provided in Tablev below and matches the mcause interrupt and exception codes defined in Table mcause Exception Codes.

Note that local interrupts may be delegated to supervisor mode. Virtual supervisor-level interrupts and guest external interrupts are always delegated past machine mode to hypervisor mode.

Table 3-51 Machine Interrupt Delegation Register

Machine Interrupt Delegation Register (mideleg)		
CSR	0x303	
Bits	Attr.	Description
0	WARL	Reserved
1	RW	Delegate Supervisor Software Interrupt
2	RW	Delegate Virtual Supervisor Software Interrupt
[4:3]	WARL	Reserved
5	RW	Delegate Supervisor Timer Interrupt
6	RW	Delegate Virtual Supervisor Timer Interrupt
[8:7]	WARL	Reserved
9	RW	Delegate Supervisor External Interrupt
10	RW	Delegate Virtual Supervisor External Interrupt
11	WARL	Reserved
12	RW	Delegate Supervisor Guest External Interrupt
13	RW	Delegate HPM Counter Overflow Interrupt
[63:14]	WARL	Reserved

Table 3-52 Machine Exception Delegation Register

Machine Exception Delegation Register (medeleg)		
CSR	0x302	
Bits	Attr.	Description
0	RW	Delegate Instruction Access Misaligned Exception
1	RW	Delegate Instruction Access Fault Exception
2	RW	Delegate Illegal Instruction Exception
3	RW	Delegate Breakpoint Exception

Machine Exception Delegation Register (medeleg)		
4	RW	Delegate Load Access Misaligned Exception
5	RW	Delegate Load Access Fault Exception
6	RW	Delegate Store/AMO Address Misaligned Exception
7	RW	Delegate Store/AMO Access Fault Exception
8	RW	Delegate Environment Call from U-Mode or VU-Mode
9	RW	Delegate Environment Call from HS-Mode
10	RW	Delegate Environment Call from VS-Mode
11	WARL	Reserved
12	RW	Delegate Instruction Page Fault
13	RW	Delegate Load Page Fault
14	WARL	Reserved
15	RW	Delegate Store/AMO Page Fault Exception
[19:16]	WARL	Reserved
20	RW	Delegate Instruction Guest-Page Fault
21	RW	Delegate Load Guest-Page Fault
22	RW	Delegate Virtual Instruction
23	RW	Delegate Store/AMO Guest-Page Fault
[63:24]	WARL	Reserved

- **Supervisor Status Register (sstatus)**

Similar to machine mode, supervisor mode has a register dedicated to keeping track of the hart's current state called sstatus. sstatus is effectively a restricted view of mstatus, in that changes made to sstatus are reflected in mstatus and viceversa, with the exception of the machine mode fields, which are not visible in sstatus.

A summary of the sstatus fields related to interrupts in the EIC7700X MCPU is provided in Table below. Note that this is not a complete description of sstatus as it also contains fields unrelated to interrupts, For the full description of sstatus, consult the The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-53 Supervisor Status Register (partial)

Supervisor Status Register (sstatus)			
CSR	0x100		
Bits	Field Name	Attr.	Description
0	Reserved	WPRI	
1	SIE	RW	Supervisor Interrupt Enable

[4:2]	Reserved	WPRI	
5	SPIE	RW	Supervisor Previous Interrupt Enable
[7:6]	Reserved	WPRI	
8	SPP	RW	Supervisor Previous Privilege Mode
[12:9]	Reserved	WPRI	

- **Supervisor Interrupt Enable Register (sie)**

Supervisor interrupts are enabled by setting the appropriate bit in the sie register. The EIC7700X MCPU sie register is described in Table below.

Table 3-54 Supervisor Status Register (partial)

Supervisor Interrupt Enable Register (sie)			
CSR	**0x104**		
Bits	**Field Name**	**Attr.**	**Description**
0	Reserved	WPRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
[4:2]	Reserved	WPRI	
5	STIE	RW	Supervisor Timer Interrupt Enable
[8:6]	Reserved	WPRI	
9	SEIE	RW	Supervisor External Interrupt Enable
[12:10]	Reserved	WPRI	
13	LCOFIE	RW	Local Count Overflow Interrupt Enable
[63:14]	Reserved	WPRI	

- **Supervisor Interrupt Pending (sip)**

The supervisor interrupt pending (sip) register indicates which interrupts are currently pending. The EIC7700X MCPU sip register is described in Table below.

Table 3-55 Supervisor Interrupt Pending Register

Supervisor Interrupt Enable Register (sie)			
CSR	**0x104**		
Bits	**Field Name**	**Attr.**	**Description**
0	Reserved	WPRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
[4:2]	Reserved	WPRI	

Supervisor Interrupt Enable Register (sie)			
CSR	0x104		
Bits	Field Name	Attr.	Description
5	STIE	RW	Supervisor Timer Interrupt Enable
[8:6]	Reserved	WPRI	
9	SEIE	RW	Supervisor External Interrupt Enable
[12:10]	Reserved	WPRI	
13	LCOFIE	RW	Local Count Overflow Interrupt Enable
[63:14]	Reserved	WPRI	

- **Supervisor Cause Register (scause)**

When a trap is taken in supervisor mode, scause is written with a code indicating the event that caused the trap. When the event that caused the trap is an interrupt, the most-significant bit of scause is set to 1, and the least-significant bits indicate the interrupt number, using the same encoding as the bit positions in sip. For example, a Supervisor Timer Interrupt causes scause to be set to 0x8000_0000_0000_0005.

scause is also used to indicate the cause of synchronous exceptions, in which case the most significant bit of scause is set to 0. Refer to Table scause Exception Codes for a list of synchronous exception codes.

Table 3-56 Supervisor Cause Register

Supervisor Cause Register (scause)			
CSR	0x142		
Bits	Field Name	Attr.	Description
[62:0]	EXCCODE	WLRL	A code identifying the last exception.
63	Interrupt	WARL	1 if the trap was caused by an interrupt; 0 otherwise.

Table 3-57 scause Exception Codes

Interrupt	Exception Code	Description
1	0	Reserved
1	1	Supervisor software interrupt
1	2	Virtual supervisor software interrupt
1	3–4	Reserved
1	5	Supervisor timer interrupt
1	6	Virtual supervisor timer interrupt

Interrupt	Exception Code	Description
1	7–8	Reserved
1	9	Supervisor external interrupt
1	10	Virtual supervisor external interrupt
1	11	Reserved
1	12	Supervisor guest external interrupt
1	13	HPM Counter Overflow
1	≥14	Reserved
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Reserved
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode or VU-mode
0	9	Environment call from HS-mode
0	10	Environment call from VS-mode
0	11	Reserved
0	12	Instruction page fault
0	13	Load page fault
0	14	Reserved
0	15	Store/AMO page fault
0	16–19	Reserved
0	20	Instruction guest-page fault
0	21	Load guest-page fault
0	22	Virtual instruction
0	23	Store/AMO guest-page fault
0	≥24	Reserved

- **Supervisor Trap Vector (stvec)**

By default, all interrupts trap to a single address defined in the stvec register. It is up to the interrupt handler to read scause and react accordingly. RISC-V and the EIC7700X MCPU also support the ability to optionally enable interrupt vectors. When vectoring is enabled, each interrupt defined in sie will trap to its own specific interrupt handler.

Vectored interrupts are enabled when the MODE field of the stvec register is set to 1.

Table 3-58 Supervisor Trap Vector Register

Supervisor Trap Vector Register (stvec)			
CSR	0x105		
Bits	Field Name	Attr.	Description
[1:0]	MODE	WARL	MODE determines whether or not interrupt vectoring is enabled.
[63:2]	BASE[63:2]	WARL	Interrupt Vector Base Address. Must be aligned on a 128-byte boundary when MODE=1. Note, BASE[1:0] is not present in this register and is implicitly 0.

Table 3-59 Encoding of stvec.MODE

MODE Field Encoding stvec.MODE		
Value	Name	Description
0x0	Direct	All exceptions and interrupts set pc to BASE
0x1	Vectored	Exceptions set pc to BASE, interrupts set pc to BASE + 4 × scause.EXCCODE
≥0x2	Reserved	

If vectored interrupts are disabled (stvec.MODE=0), all interrupts trap to the stvec.BASE address. If vectored interrupts are enabled (stvec.MODE=1), interrupts set the pc to stvec.BASE + 4 x exception code (scause.EXCCODE). For example, if a supervisor timer interrupt is taken, the pc is set to stvec.BASE + 0x14. Typically, the trap vector table is populated with jump instructions to transfer control to interrupt-specific trap handlers.

In vectored interrupt mode, BASE must be 128-byte aligned.

All supervisor external interrupts (global interrupts) are mapped to exception code of 9. Thus, when interrupt vectoring is enabled, the pc is set to address stvec.BASE + 0x24 for any global interrupt.

See Table *Supervisor Trap Vector Register* for a description of the stvec register. See Table *Encoding of stvec.MODE* for a description of the stvec.MODE field. See Table *scause Exception Codes* for the EIC7700X MCPU supervisor mode interrupt exception code values.

- **Delegated Interrupt Handling**

Upon taking a delegated trap, the following occurs:

•The value of sstatus.SIE is copied into sstatus.SPIE, then sstatus.SIE is cleared, effectively disabling interrupts.

•The current pc is copied into the sepc register, and then pc is set to the value of stvec. In the case where vectored interrupts are enabled, pc is set to stvec.BASE + 4 x exception code (scause.EXCCODE).

• The privilege mode prior to the interrupt is encoded in sstatus.SPP.

At this point, control is handed over to software in the interrupt handler with interrupts disabled. Interrupts can be re-enabled by explicitly setting sstatus.SIE or by executing an SRET instruction to exit the handler. When an SRET instruction is executed, the following occurs:

• The privilege mode is set to the value encoded in sstatus.SPP.

• The value of sstatus.SPIE is copied into sstatus.SIE.

• The pc is set to the value of sepc.

At this point, control is handed over to software.

- **Hypervisor and Virtual Supervisor Mode Interrupts**

The EIC7700X MCPU supports the ability to selectively direct interrupts and exceptions to hypervisor mode, resulting in improved performance by eliminating the need for additional mode changes. This capability is enabled by the interrupt and exception delegation CSRs; hideleg and hedeleg, respectively.

While virtualization is enabled (V=1), accesses to the normal supervisor CSRs instead access their corresponding virtual supervisor CSR, while the normal supervisor CSRs retain their values. If a corresponding virtual supervisor CSR doesn't exist, the normal supervisor CSR retains its original functionality with VS-mode and VU-mode substituting HS-mode and U-mode. Directly accessing the virtual supervisor CSRs while V=1 results in a virtual instruction exception.

Virtual supervisor mode interrupts are globally disabled when executing in user mode.

The delegation, hypervisor, and virtual supervisor CSRs are described in the sections below. The definitive resource for information about RISC-V hypervisor and virtual supervisor interrupts is The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

- **Hypervisor Trap Delegation Registers (hideleg and hedeleg)**

By default, all traps are handled in machine mode. Machine mode software can selectively delegate interrupts and exceptions to hypervisor mode by setting the corresponding bits in the mideleg and medeleg CSRs.

The hideleg and hedeleg CSRs allow these interrupts and exceptions to be further delegated to a VS-mode guest. The exact mapping is provided in Table *Hypervisor Interrupt Delegation Register* and Table *Hypervisor Exception Delegation Register* and matches the mcause interrupt and exception codes defined in Table *mcause Exception Codes*.

A synchronous trap that has been delegated to HS-mode via medeleg is further delegated to VS-mode if virtualization is enabled before the trap and the corresponding bit in hedeleg is set.

An interrupt that has been delegated to HS-mode via mideleg is further delegated to VS-mode if the corresponding bit in hideleg is set.

Table 3-60 Hypervisor Interrupt Delegation Register

Hypervisor Interrupt Delegation Register (hideleg)		
CSR	0x603	
Bits	Attr.	Description
[1:0]	WARL	Reserved
2	RW	Delegate Virtual Supervisor Software Interrupt
[5:3]	WARL	Reserved
6	RW	Delegate Virtual Supervisor Timer Interrupt
[9:7]	WARL	Reserved
10	RW	Delegate Virtual Supervisor External Interrupt
[63:11]	WARL	Reserved

Table 3-61 Hypervisor Exception Delegation Register

Hypervisor Exception Delegation Register (hedeleg)		
CSR	0x602	
Bits	Attr.	Description
0	RW	Delegate Instruction Access Misaligned Exception
1	RW	Delegate Instruction Access Fault Exception
2	RW	Delegate Illegal Instruction Exception
3	RW	Delegate Breakpoint Exception
4	RW	Delegate Load Access Misaligned Exception
5	RW	Delegate Load Access Fault Exception
6	RW	Delegate Store/AMO Address Misaligned Exception
7	RW	Delegate Store/AMO Access Fault Exception
8	RW	Delegate Environment Call from U-Mode or VU-Mode
[11:9]	WARL	Reserved
12	RW	Delegate Instruction Page Fault
13	RW	Delegate Load Page Fault
14	WARL	Reserved
15	RW	Delegate Store/AMO Page Fault Exception
[63:16]	WARL	Reserved

- **Hypervisor Status Register (hstatus)**

A virtual supervisor mode guest has a register dedicated to keeping track of the hart's current state called hstatus, which functions analogously to mstatus.

A summary of the hstatus fields related to interrupts in the EIC7700X MCPU is provided in Table below. Note that this is not a complete description of hstatus as it also contains fields unrelated to interrupts. For the full description of hstatus, consult the The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-62 Hypervisor Status Register (partial)

Hypervisor Status Register (hstatus)			
CSR	0x600		
Bits	Field Name	Attr.	Description
6	GVA	RW	Guest Virtual Address
7	SPV	RW	Supervisor Previous Virtualization Mode
8	SPVP	RW	Supervisor Previous Virtual Privilege

Similar to sstatus.SPP, hstatus.SPV is set to the value of the virtualization mode when a trap is taken. When virtualization is disabled (V=0) and an SRET instruction is executed, the virtual- ization mode is set to the value encoded in SPV.

When V=1 and a trap is taken into hypervisor mode, hstatus.SPVP is set to the nominal privi- lege mode at the time of the trap (same as sstatus.SPP). If V=0 before a trap, SPVP is left unchanged on trap entry.

The hstatus.GVA field is set to 1 for any trap that writes a guest virtual address to stval. For any other trap into hypervisor mode, GVA is set to 0.

- **Virtual Supervisor Status Register (vsstatus)**

The vsstatus register is the virtual supervisor mode replica of the supervisor register sstatus. When virtualization is enabled, instructions that normally read or modify sstatus instead access vsstatus.

A summary of the vsstatus fields related to interrupts in the EIC7700X MCPU is provided in Table below. Note that this is not a complete description of vsstatus as it also contains fields unrelated to interrupts. For the full description of vsstatus, consult the The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

Table 3-63 Virtual Supervisor Status Register (partial)

Virtual Supervisor Status Register (vsstatus)			
CSR	0x200		
Bits	Field Name	Attr.	Description
0	Reserved	WPRI	
1	SIE	RW	Supervisor Interrupt Enable

[4:2]	Reserved	WPRI	
5	SPIE	RW	Supervisor Previous Interrupt Enable
…			
8	SPP	RW	Supervisor Previous Privilege Mode

Interrupts are enabled by setting vsstatus.SIE and by enabling the desired individual interrupt in the vsie register

- **Hypervisor Interrupt Enable Register (hie)**

Virtual supervisor and hypervisor-specific interrupts are enabled by setting the appropriate bit in the hie register. The EIC7700X MCPU hie register is described in Table 3.3.1.10-24.

Table 3-64 Hypervisor Interrupt Enable Register

Hypervisor Interrupt Enable Register (hie)			
CSR	0x604		
Bits	Field Name	Attr.	Description
[1:0]	Reserved	WPRI	
2	VSSIE	RW	Virtual Supervisor Software Interrupt Enable
[5:3]	Reserved	WPRI	
6	VSTIE	RW	Virtual Supervisor Timer Interrupt Enable
[9:7]	Reserved	WPRI	
10	VSEIE	RW	Virtual Supervisor External Interrupt Enable
11	Reserved	WPRI	
12	SGEIE	RW	Supervisor Guest External Interrupt Enable
[63:13]	Reserved	WPRI	

- **Hypervisor Interrupt Pending (hip)**

The hypervisor interrupt pending (hip) register indicates which virtual supervisor and hypervi- sor-specific interrupts are currently pending. The EIC7700X MCPU hip register is described in Table below.

Table 3-65 Hypervisor Interrupt Pending Register

Hypervisor Interrupt Pending Register (hip)			
CSR	**0x644**		
Bits	**Field Name**	**Attr.**	**Description**
[1:0]	Reserved	WPRI	
2	VSSIP	RW	Virtual Supervisor Software Interrupt Pending

Hypervisor Interrupt Pending Register (hip)			
CSR	0x644		
Bits	Field Name	Attr.	Description
[5:3]	Reserved	WPRI	
6	VSTIP	WPRI	Virtual Supervisor Timer Interrupt Pending
[9:7]	Reserved	WPRI	
10	VSEIP	WPRI	Virtual Supervisor External Interrupt Pending
11	Reserved	WPRI	
12	SGEIP	RO	Supervisor Guest External Interrupt Pending
[63:13]	Reserved	WPRI	

- **Hypervisor Virtual Interrupt Pending (hvip)**

A hypervisor can write to hvip to indicate virtual interrupts intended for virtual supervisor mode. Setting each bit asserts the corresponding virtual supervisor level interrupt.

Table 3-66 Hypervisor Virtual Interrupt Pending Register

Hypervisor Virtual Interrupt Pending Register (hvip)			
CSR	0x644		
Bits	Field Name	Attr.	Description
[1:0]	Reserved	WPRI	
2	VSSIP	RW	Virtual Supervisor Software Interrupt Pending
[5:3]	Reserved	WPRI	
6	VSTIP	RW	Virtual Supervisor Timer Interrupt Pending
[9:7]	Reserved	WPRI	
10	VSEIP	RW	Virtual Supervisor External Interrupt Pending
[63:11]	Reserved	WPRI	

- **Hypervisor Guest External Interrupt Registers (hgeie and hgeip)**

Guest External Interrupts are interrupts directed to individual virtual machines at the VS-level. Each bit of hgeip summarizes all pending interrupts directed to a single virtual hart; software can distinguish specific pending interrupts from multiple devices.

Guest external interrupts are enabled by setting the appropriate bit in the hgeie register. The hgeip register indicates pending guest external interrupts for a given hart. The guest external interrupt number N corresponds with bit N in both hgeie and hgeip. The least-significant bits, after bit 0, are implemented first.

- **Virtual Supervisor Interrupt Enable Register (vsie)**

The vsie register is the virtual supervisor mode replica of the supervisor register sie. When virtualization is enabled, instructions that normally read or modify sie instead access vsie.

Table 3-67 Virtual Supervisor Interrupt Enable Register

Virtual Supervisor Interrupt Enable Register (vsie)			
CSR	0x204		
Bits	Field Name	Attr.	Description
0	Reserved	WIRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
[4:2]	Reserved	WIRI	
5	STIE	RW	Supervisor Timer Interrupt Enable
[8:6]	Reserved	WIRI	
9	SEIE	RW	Supervisor External Interrupt Enable
[63:10]	Reserved	WIRI	

When a virtual supervisor interrupt bit in hideleg is zero, the corresponding supervisor interrupt bit in vsie is read-only. Otherwise, the bit in vsie is an alias of the corresponding virtual super- visor bit in hie.

- **Virtual Supervisor Interrupt Pending (vsip)**

The vsip register is the virtual supervisor mode replica of the supervisor register sip. When virtualization is enabled, instructions that normally read or modify sip instead access vsip; however, interrupts directed to HS-level continue to be indicated in the HS-level sip register, not vsip.

Table 3-68 Virtual Supervisor Interrupt Pending Register

Virtual Supervisor Interrupt Pending Register (vsip)			
CSR	0x244		
Bits	Field Name	Attr.	Description
0	Reserved	WIRI	
1	SSIP	RW	Supervisor Software Interrupt Pending
[4:2]	Reserved	WIRI	
5	STIP	RW	Supervisor Timer Interrupt Pending
[8:6]	Reserved	WIRI	
9	SEIP	RW	Supervisor External Interrupt Pending
[63:10]	Reserved	WIRI	

When a virtual supervisor interrupt bit in hideleg is zero, the corresponding supervisor interrupt bit in vsip is read-only. Otherwise, the bit in vsip is an alias of the corresponding virtual super- visor bit in hip.

- **Interrupt Priorities**

Individual priorities of global interrupts are determined by the PLIC. EIC7700X MCPU interrupts are prioritized as follows, in decreasing order of priority:

- Machine external interrupts
- Machine software interrupts
- Machine timer interrupts
- Supervisor external interrupts
- Supervisor software interrupts
- Supervisor timer interrupts
- Supervisor guest external interrupts
- Virtual supervisor external interrupts
- Virtual supervisor software interrupts
- Virtual supervisor timer interrupts

- **RNMI CSRs**

These M-mode CSRs enable a resumable non-maskable interrupt (RNMI).

Table 3-69 RNMI CSRs

Number	Name	Description
0x350	mnscratch	Resumable Non-maskable scratch register
0x351	mnepc	Resumable Non-maskable EPC value
0x352	mncause	Resumable Non-maskable cause value
0x353	mnstatus	Resumable Non-maskable status

- The mnscratch CSR holds a 64-bit read-write register, which enables the NMI trap handler to save and restore the context that was interrupted.
- The mnepc CSR is a 64-bit read-write register, which, on entry to the NMI trap handler, holds the PC of the instruction that took the interrupt. The lowest bit of mnepc is hardwired to zero.
- The mncause CSR holds the reason for the NMI, with bit 63 set to 1, and the NMI cause encoded in the least-significant bits, or zero if NMI causes are not supported. The lower bits of mncause, defined as the exception_code, are as follows:

Table 3-70 mncause.exception_code Fields

mncause	NMI Cause	Function
1	Reserved	Reserved
2	RNMI input pin	External rnmi_N input
3	Bus error	RNMI caused by BEU

• The mnstatus CSR holds a two-bit field mnpp encoded in the same manner as mstatus.mpp, which, on entry to the trap handler, holds the privilege mode of the inter- rupted context.

• mnstatus also contains an mnpv field, defined analagously to mstatus.mpv, which holds the virtualization mode of the interrupted context.

• mnstatus also holds the one-bit field mnie indicating whether NMIs are currently enabled. This bit can only be cleared by hardware, but can be set by software to indicate a further NMI can be taken.

Table 3-71 mnstatus CSR Fields

Bits	Field Name	Description
3	mnie	RMNI interrupt enable
7	mnpv	RNMI previous virtualization mode
12:11	mnpp	RMNI previous priority level

- **RNMI Operation**

When an RNMI interrupt is detected, the interrupted PC is written to the mnepc CSR, the type of RNMI to the mncause CSR, and the privilege mode of the interrupted context to the mnstatus CSR. An internal microarchitectural state bit, rnmie, is cleared to indicate that the processor is in an RNMI handler and cannot take a new RNMI interrupt. When clear, the internal rnmie bit also disables all other interrupts.

> **Note**
>
> These interrupts are called non-maskable because software cannot mask the interrupts. However, for correct operation, other instances of the same interrupt must be held off until the handler is completed, hence the internal state bit.

The RNMI handler can resume original execution using the MNRET instruction, which restores the PC from mnepc, the privilege mode from mnstatus, and also sets the internal rnmie state bit, which re-enables other interrupts.

If the hart encounters an exception while the rnmie bit is clear, the exception state is written to mepc and mcause, mstatus.mpp is set to M-mode, and the hart jumps to the RNMI exception handler address.

> **Note**
>
> Traps in the RNMI handler can only be resumed if they occur while the handler was servicing an interrupt that occurred outside of machine mode.

3.4.1.11 Core-Local Interruptor (CLINT)

The EIC7700X MCPU CLINT complies with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11.

The CLINT has a small footprint and provides software, timer, and external interrupts directly to the hart. The CLINT block also holds memory-mapped control and status registers associated with software and timer interrupts.

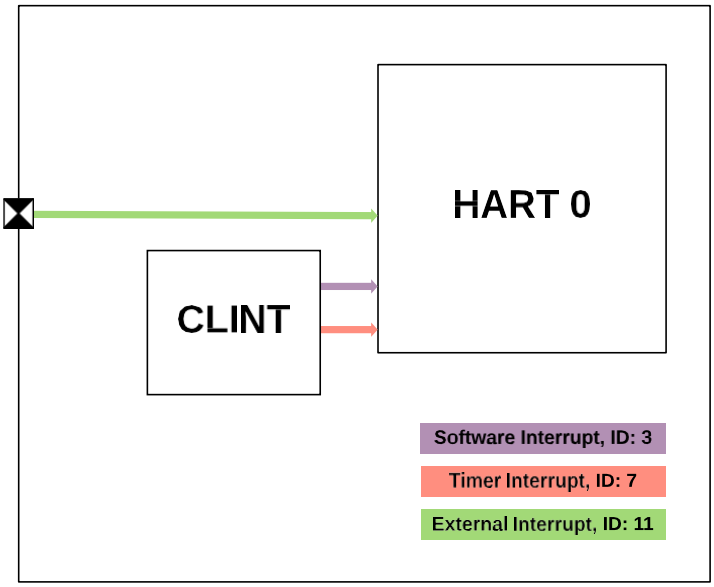


Figure 3-11 CLINT Block Diagram

- **CLINT Vector Table**

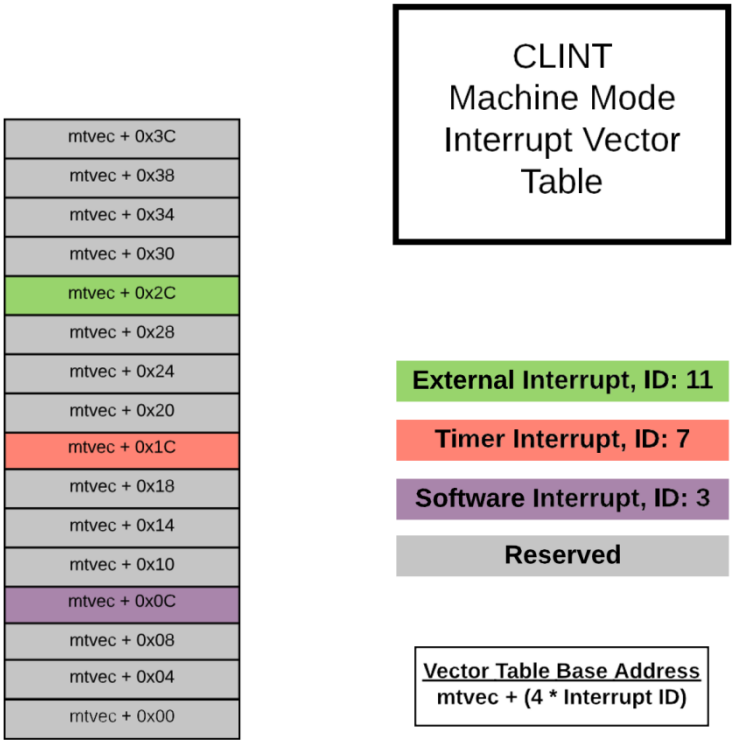


Figure 3-12 CLINT Interrupts and Vector Table

The CLINT vector table is populated with jump instructions, since hardware jumps to the index in the vector table first, then subsequently jumps to the handler. All exception types trap to the first entry in the table, which is mtvec.BASE.

An example CLINT vector table is shown below.

```
.weak default_exception_handler
.balign 4, 0
.global default_exception_handler

.weak software_handler
.balign 4, 0
.global software_handler

.weak timer_handler
.balign 4, 0
.global timer_handler

.weak external_handler
.balign 4, 0
.global external_handler

.option norvc
.weak __mtvec_clint_vector_table
#if __riscv_xlen == 32
.balign 128, 0
#else
.balign 256, 0
#endif
.global __mtvec_clint_vector_table
__mtvec_clint_vector_table:

IRQ_0:
        j default_exception_handler
IRQ_1:
        j default_vector_handler
IRQ_2:
        j default_vector_handler
IRQ_3:
        j software_handler
IRQ_4:
        j default_vector_handler
IRQ_5:
        j default_vector_handler
IRQ_6:
        j default_vector_handler
IRQ_7:
        j timer_handler
IRQ_8:
        j default_vector_handler
IRQ_9:
        j default_vector_handler
IRQ_10:
        j default_vector_handler
IRQ_11:
        j external_handler
IRQ_12:
        j default_vector_handler
IRQ_13:
        j default_vector_handler
IRQ_14:
        j default_vector_handler
IRQ_15:
        j default_vector_handler
```

Figure 3-13 CLINT Vector Table Example

- **CLINT Interrupt Sources**

The EIC7700X MCPU supports the standard RISC-V software, timer, and external interrupts.

Table 3-72 EIC7700X MCPU Interrupt IDs

EIC7700X MCPU Interrupt IDs		
ID	**Interrupt**	**Notes**
0	Reserved	
1	ssip	Supervisor Software Interrupt
2	Reserved	
3	msip	Machine Software Interrupt
4	Reserved	
5	stip	Supervisor Timer Interrupt
6	Reserved	
7	mtip	Machine Timer Interrupt
8	Reserved	
9	seip	Supervisor External Interrupt
10	Reserved	
11	meip	Machine External Interrupt

- **CLINT Memory Map**

Table below shows the memory map for CLINT on the EIC7700X MCPU. Note that there are no enable bits for specific interrupts within the CLINT memory map, as the enables for these interrupts reside in the mie CSR for each interrupt, and the mstatus.mie CSR bit, which enables all machine interrupts globally

Table 3-73 CLINT Memory Map

Address	Width	Attr.	Description	Notes
0x0200_0000	4B	RW	msip for hart 0	MSIP Registers (1-bit wide)
0x0200_0004	4B	RW	msip for hart 1	
0x0200_0008	4B	RW	msip for hart 2	
0x0200_000C	4B	RW	msip for hart 3	
0x0200_0010			Reserved	
…				
0x0200_3FFF				
0x0200_4000	8B	RW	mtimecmp for hart 0	MTIMECMP Registers
0x0200_4008	8B	RW	mtimecmp for hart 1	

Address	Width	Attr.	Description	Notes
0x0200_4010	8B	RW	mtimecmp for hart 2	
0x0200_4018	8B	RW	mtimecmp for hart 3	
0x0200_4020			Reserved	
…				
0x0200_BFF7				
0x0200_BFF8	8B	RW	mtime	Timer Register
0x0200_C000			Reserved	

- **Register Descriptions**

This section describes the functionality of the memory-mapped registers in the CLINT.

- **MSIP Register**

Machine mode software interrupts are generated by writing to the memory-mapped control reg- ister msip. Each msip register is a 32-bit wide WARL register, where the upper 31 bits are tied to 0. The least-significant bit is reflected in the MSIP bit of the mip CSR. Other bits in the msip register are hardwired to zero. On reset, each msip register is cleared to zero

Software interrupts are most useful for interprocessor communication in multi-hart systems, as harts may write each other's msip bits to effect interprocessor interrupts.

- **Timer Registers**

mtime is a 64-bit read-write register that contains the number of cycles counted from the rtc_toggle signal. A timer inter- rupt is pending whenever mtime is greater than or equal to the value in the mtimecmp register. The timer interrupt is reflected in the mtip bit of the mip register.

On reset, mtime is cleared to zero. The mtimecmp registers are not reset. Note that mtime is volatile and may be masked in IP-XACT.

- **Supervisor Mode Delegation**

By default, all interrupts trap to machine mode, including timer and software interrupts. In order for supervisor timer and software interrupts to trap directly to supervisor mode, supervisor timer and software interrupts must first be delegated to supervisor mode.

3.4.1.12 Platform-Level Interrupt Controller(PLIC)

This chapter describes the operation of the Platform-Level Interrupt Controller (PLIC) on the EIC7700X MCPU. The PLIC complies with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11 and can support a maximum of 520 external interrupt sources with 7 priority levels.

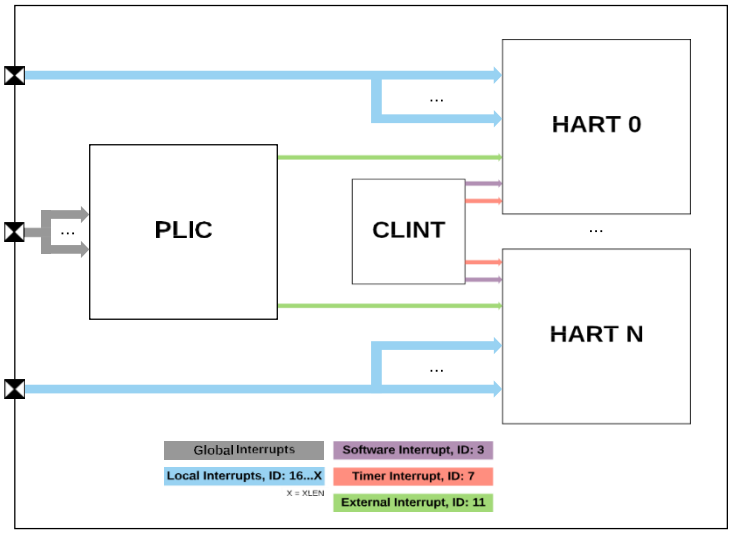


Figure 3-14 PLIC Multi-Core Block Diagram

- **Memory Map**

The memory map for the EIC7700X MCPU PLIC control registers is shown in Table below. The PLIC memory map only supports aligned 32-bit memory accesses.

Table 3-74 PLIC Memory Map

Address	Width	Attr.	Description
0x0C00_0000			Reserved
0x0C00_0004	4B	RW	Source 1 priority
...			
0x0C00_0820	4B	RW	Source 520 priority
0x0C00_0824			Reserved
...			
0x0C00_1000	4B	RO	Start of pending array
...			
0x0C00_1040	4B	RO	Last word of pending array
0x0C00_1044			Reserved
...			
0x0C00_2000	4B	RW	Start Hart 0 M-Mode interrupt enables
...			
0x0C00_2040	4B	RW	End Hart 0 M-Mode interrupt enables
0x0C00_2044			Reserved
...			

Address	Width	Attr.	Description
0x0C00_2080	4B	RW	Start Hart 0 S-Mode interrupt enables
...			
0x0C00_20C0	4B	RW	End Hart 0 S-Mode interrupt enables
0x0C00_20C4 ...			Reserved
0x0C00_2100	4B	RW	Start Hart 1 M-Mode interrupt enables
...			
0x0C00_2140	4B	RW	End Hart 1 M-Mode interrupt enables
0x0C00_2144 ...			Reserved
0x0C00_2180	4B	RW	Start Hart 1 S-Mode interrupt enables
...			
0x0C00_21C0	4B	RW	End Hart 1 S-Mode interrupt enables
0x0C00_21C4 ...			Reserved
0x0C00_2200	4B	RW	Start Hart 2 M-Mode interrupt enables
...			
0x0C00_2240	4B	RW	End Hart 2 M-Mode interrupt enables
0x0C00_2244 ...			Reserved
0x0C00_2280	4B	RW	Start Hart 2 S-Mode interrupt enables
...			
0x0C00_22C0	4B	RW	End Hart 2 S-Mode interrupt enables
0x0C00_22C4 ...			Reserved
0x0C00_2300	4B	RW	Start Hart 3 M-Mode interrupt enables
...			
0x0C00_2340	4B	RW	End Hart 3 M-Mode interrupt enables

0x0C00_2344 …			Reserved
0x0C00_2380	4B	RW	Start Hart 3 S-Mode interrupt enables
…			
0x0C00_23C0	4B	RW	End Hart 3 S-Mode interrupt enables
0x0C00_23C4 …			Reserved
0x0C1F_F000	1B	RW	PLIC global clock gating disable feature
0x0C1F_F001 …			Reserved
0x0C20_0000	4B	RW	Hart 0 M-Mode priority threshold
0x0C20_0004	4B	RW	Hart 0 M-Mode claim/complete
0x0C20_0008 …			Reserved
0x0C20_1000	4B	RW	Hart 0 S-Mode priority threshold
0x0C20_1004	4B	RW	Hart 0 S-Mode claim/ complete
0x0C20_1008 …			Reserved
0x0C20_2000	4B	RW	Hart 1 M-Mode priority threshold
Address	Width	Attr.	Description
0x0C20_2004	4B	RW	Hart 1 M-Mode claim/ complete
0x0C20_2008			Reserved
…			
0x0C20_3000	4B	RW	Hart 1 S-Mode priority threshold
0x0C20_3004	4B	RW	Hart 1 S-Mode claim/ complete
0x0C20_3008			Reserved
…			

0x0C20_4000	4B	RW	Hart 2 M-Mode priority threshold
0x0C20_4004	4B	RW	Hart 2 M-Mode claim/ complete
0x0C20_4008			Reserved
…			
0x0C20_5000	4B	RW	Hart 2 S-Mode priority threshold
0x0C20_5004	4B	RW	Hart 2 S-Mode claim/ complete
0x0C20_5008			Reserved
…			
0x0C20_6000	4B	RW	Hart 3 M-Mode priority threshold
0x0C20_6004	4B	RW	Hart 3 M-Mode claim/ complete
0x0C20_6008			Reserved
…			
0x0C20_7000	4B	RW	Hart 3 S-Mode priority threshold
0x0C20_7004	4B	RW	Hart 3 S-Mode claim/ complete
0x0C20_7008			Reserved
…			
0x1000_0000			End of PLIC Memory Map

- **Interrupt Sources**

The EIC7700X MCPU has a total of 520 global interrupt sources, in addition to the local interrupts described in Table *EIC7700X MCPU Interrupt IDs*. 511 of these are external global interrupts, and the remainder are driven by various on-chip devices as listed in Table below.

> **Note**
>
> In the *RISC-V Platform-Level Interrupt Controller Specification*, interrupt source 0 (ID 0) is unused, so the first usable PLIC Interrupt ID has a value of 1.

Table below describes the global interrupt sources on the EIC7700X MCPU.

Table 3-75 PLIC Interrupt Source Mapping

PLIC Interrupt ID	Source
0	Unused
1	L3 Cache DirError
2	L3 Cache DirFail
3	L3 Cache DataError
4	L3 Cache DataFail
5–515	External Global Interrupts
516	TL-UH to TL-C Adapter
517	Hart 0 Bus-Error Unit
518	Hart 1 Bus-Error Unit
519	Hart 2 Bus-Error Unit
520	Hart 3 Bus-Error Unit

Table below describes the mapping of external global interrupts to its corresponding top-level global_interrupts signal bit. This signal is positive-level triggered and not configurable.

Table 3-76 Mapping of global_interrupts Signal Bits to PLIC Interrupt ID

global_interrupts Signal	PLIC Interrupt ID	PLIC Pending / Enable Register
global_interrupts[0]	5	pending1[5] / enable1[5]
global_interrupts[1]	6	pending1[6] / enable1[6]
global_interrupts[2]	7	pending1[7] / enable1[7]
…		
global_interrupts[510]	515	pending17[3] / enable17[3]

- **Interrupt Priorities**

Each PLIC interrupt source can be assigned a priority by writing to its 32-bit memory-mapped priority register. The EIC7700X MCPU supports 7 levels of priority. A priority value of 0 is reserved to mean "never interrupt" and effectively disables the interrupt. Priority 1 is the low- est active priority, and priority 7 is the highest. Ties between global interrupts of the same prior- ity are broken by the Interrupt ID; interrupts with the lowest ID have the highest effective priority. See Table below for the detailed register description.

Table 3-77 PLIC Interrupt Priority Register

PLIC Interrupt Priority Register				
Base Address		0x0C00_0000 + 4 × Interrupt ID		
Bits	Field Name	Attr.	Rst.	Description
[2:0]	Priority	RW	X	Global interrupt priority
[31:3]	Reserved	RO	0x0	

- **Interrupt Pending Bits**

The current status of the interrupt source pending bits in the PLIC core can be read from the pending array, organized as 17 words of 32 bits. The pending bit for interrupt ID N is stored in bit (N mod 32) of word (N / 32). As such, the EIC7700X MCPU has 17 interrupt pending registers. Bit 0 of word 0, which represents the non-existent interrupt source 0, is hardwired to zero.

A pending bit in the PLIC core can be cleared by setting the associated enable bit then performing a claim.

Table 3-78 PLIC Interrupt Pending Register 1

PLIC Interrupt Pending Register 1				
Base Address		0x0C00_1000		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 0 Pending	RO	X	Non-existent global interrupt 0 is hardwired to zero
1	Interrupt 1 Pending	RO	X	Pending bit for global interrupt 1
2	Interrupt 2 Pending	RO	X	Pending bit for global interrupt 2
…				
31	Interrupt 31 Pending	RO	X	Pending bit for global interrupt 31

Table 3-79 PLIC Interrupt Pending Register 17

PLIC Interrupt Pending Register 17				
Base Address		0x0C00_1040		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 512 Pending	RO	X	Pending bit for global interrupt 512
…				
8	Interrupt 520 Pending	RO	X	Pending bit for global interrupt 520

PLIC Interrupt Pending Register 17				
Base Address		0x0C00_1040		
Bits	Field Name	Attr.	Rst.	Description
[31:9]	Reserved	WIRI	X	

- **Interrupt Enables**

Each global interrupt can be enabled by setting the corresponding bit in the enable registers. The enable registers are accessed as a contiguous array of 17 × 32-bit words, packed the same way as the pending bits. Bit 0 of enable word 0 represents the non-existent interrupt ID 0 and is hardwired to 0.

64-bit and 32-bit word accesses are supported by the enables array in RV64 systems.

Table 3-80 PLIC Interrupt Enable Register 1 for Hart 0 M-Mode

PLIC Interrupt Enable Register 1 for Hart 0 M-Mode				
Base Address		0x0C00_2000		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 0 Enable	RO	0x0	Non-existent global interrupt 0 is hardwired to zero
1	Interrupt 1 Enable	RW	X	Enable bit for global interrupt 1
2	Interrupt 2 Enable	RW	X	Enable bit for global interrupt 2
...				
31	Interrupt 31 Enable	RW	X	Enable bit for global interrupt 31

Table 3-81 PLIC Interrupt Enable Register 17 for Hart 0 M-Mode

PLIC Interrupt Enable Register 17 for Hart 0 M-Mode				
Base Address		0x0C00_2040		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 512 Enable	RW	X	Enable bit for global interrupt 512
...				
8	Interrupt 520 Enable	RW	X	Enable bit for global interrupt 520
[31:9]	Reserved	RO	0x0	

Table 3-82 PLIC Interrupt Enable Register 1 for Hart 0 S-Mode

PLIC Interrupt Enable Register 1 for Hart 0 S-Mode				
Base Address		0x0C00_2080		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 0 Enable	RO	0x0	Non-existent global interrupt 0 is hardwired to zero
1	Interrupt 1 Enable	RW	X	Enable bit for global interrupt 1
2	Interrupt 2 Enable	RW	X	Enable bit for global interrupt 2
…				
31	Interrupt 31 Enable	RW	X	Enable bit for global interrupt 31

Table 3-83 PLIC Interrupt Enable Register 17 for Hart 0 S-Mode

PLIC Interrupt Enable Register 17 for Hart 0 S-Mode				
Base Address		0x0C00_20C0		
Bits	Field Name	Attr.	Rst.	Description
0	Interrupt 512 Enable	RW	X	Enable bit for global interrupt 512
…				
8	Interrupt 520 Enable	RW	X	Enable bit for global interrupt 520
[31:9]	Reserved	RO	0x0	

- **PLIC Clock Gate Disable**

The PLIC implements a clock gating feature to gate the module clock node when not active. PLIC clock gating is disabled out of reset and should be enabled in startup code. Once enabled, clock is only available when there is activity on the PLIC control bus or on any interrupt line when the corresponding interrupt is not inflight.

Table 3-84 PLIC Clock Gate Disable Register

PLIC Clock Gate Disable Register				
Base Address		0x0C1F_F000		
Bits	Field Name	Attr.	Rst.	Description
0	disablePlicClockGateFeature	RW	0x1	Used to enable/disable PLIC clock gating feature. Clear to enable.

PLIC Clock Gate Disable Register				
Base Address		0x0C1F_F000		
Bits	Field Name	Attr.	Rst.	Description
[7:1]	Reserved	RO	0x0	

- **Priority Thresholds**

The EIC7700X MCPU supports setting of an interrupt priority threshold via the threshold register. The threshold is a WARL field, where the EIC7700X MCPU supports a maxi- mum threshold of 7.

The EIC7700X MCPU masks all PLIC interrupts of a priority less than or equal to threshold. For example, a threshold value of zero permits all interrupts with non-zero priority, whereas a value of 7 masks all interrupts. If the threshold register contains a value of 5, all PLIC interrupt configured with priorities from 1 through 5 will not be allowed to propagate to the CPU.

Table 3-85 PLIC Interrupt Priority Threshold Register for Hart 0 M-Mode

PLIC Interrupt Priority Threshold Register for Hart 0 M-Mode				
Base Address		0x0C20_0000		
Bits	Field Name	Attr.	Rst.	Description
[2:0]	Threshold	WARL	X	Sets the priority threshold
[31:3]	Reserved	RO	0x0	

Table 3-86 PLIC Interrupt Priority Threshold Register for Hart 0 S-Mode

PLIC Interrupt Priority Threshold Register for Hart 0 S-Mode				
Base Address		0x0C20_1000		
Bits	Field Name	Attr.	Rst.	Description
[2:0]	Threshold	WARL	X	Sets the priority threshold
[31:3]	Reserved	RO	0x0	

- **11.8 Interrupt Claim Process**

A EIC7700X MCPU hart can perform an interrupt claim by reading the claim_complete register, which returns the ID of the highest-priority pending interrupt or zero if there is no pending interrupt. A successful claim also atomically clears the corresponding pending bit on the interrupt source.

A EIC7700X MCPU hart can perform a claim at any time, even if the MEIP bit in its mip register is not set.

The claim operation is not affected by the setting of the priority threshold register.

- **11.9 Interrupt Completion**

A EIC7700X MCPU hart signals it has completed executing an interrupt handler by writing the interrupt

ID it received from the claim to the claim_complete register. The PLIC does not check whether the completion ID is the same as the last claim ID for that target. If the completion ID does not match an interrupt source that is currently enabled for the target, the completion is silently ignored.

Table 3-87 PLIC Claim/Complete Register for Hart 0 M-Mode

PLIC Claim/Complete Register for Hart 0 M-Mode				
Base Address		0x0C20_0004		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	Interrupt Claim/Complete for Hart 0 M-Mode	RW	X	A read of zero indicates that no interrupts are pending. A non-zero read contains the ID of the highest pending interrupt. A write to this register signals completion of the interrupt ID written. Consecutive reads will not return the same result. Note that this field is volatile and may be masked in IP-XACT.

Table 3-88 PLIC Claim/Complete Register for Hart 0 S-Mode

PLIC Claim/Complete Register for Hart 0 S-Mode				
Base Address		0x0C20_1004		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	Interrupt Claim/Complete for Hart 0 S-Mode	RW	X	A read of zero indicates that no interrupts are pending. A non-zero read contains the ID of the highest pending interrupt. A write to this register signals completion of the interrupt ID written. Consecutive reads will not return the same result. Note that this field is volatile and may be masked in IP-XACT.

The PLIC cannot forward a new interrupt to a hart that has claimed an interrupt, but has not yet finished the complete step of the interrupt handler. Thus, the PLIC does not support preemption of global interrupts to an individual hart.

Interrupt IDs for global interrupts routed through the PLIC are independent of the interrupt IDs for local interrupts. The PLIC handler may check for additional pending global interrupts once the initial claim/complete process has finished, prior to exiting the handler. This method could save additional PLIC save/restore context for global interrupts.

3.4.1.13 Private Level 2 Cache

In addition to the L1 memory system, the EIC7700X MCPU also includes a per hart Private L2 (pL2) Cache. The pL2 Cache communicates with both the upstream L1 caches and down- stream L3 cache or memory, enabling a high-performance cache subsystem. The pL2 Cache is responsible for managing requests from the L1 instruction and data caches of the core. The overall cache structure is depicted in Figure below. The features of the pL2 Cache are summarized in Table *Private L2 Cache Feature Set*.

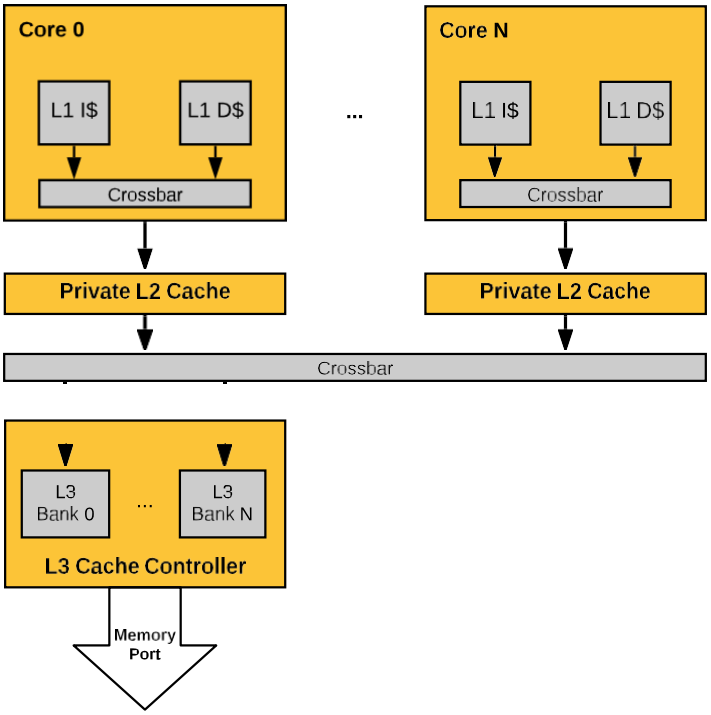


Figure 3-15 Cache Structure in the EIC7700X MCPU

The EIC7700X MCPU contains four pL2 Cache instances. Their addresses are shown in the table below.

Table 3-89 Private L2 Cache Instances

Instance	Base Address
Hart 0 Private L2 Cache	0x0010_4000
Hart 1 Private L2 Cache	0x0010_8000
Hart 2 Private L2 Cache	0x0010_C000
Hart 3 Private L2 Cache	0x0011_0000

Table 3-90 Private L2 Cache Feature Set

Features	Description
Size	256 KiB
Associativity	8 ways
Number of banks	2 banks
Line size	64 bytes
Coherence protocol	MESI Coherence Protocol
ECC protection	SECDED (9 bits ECC)
Latency	11-16 cycle load-to-use latency
Number of MSHRs	32 MSHRs

Table 3-91 Private L2 Cache Memory Map

Offset	Name	Description
0x0000	Config	Information about the cache configuration
0x0040	ECCInjectError	Inject an ECC error
0x0060	EccInjectErrorDone	ECC inject is done
0x0200	Cflush64	MMIO cache flush command register
0x0208	FlushCount	The number of outstanding CFLUSH requests
0x1008	pL2Config0	Config bits of private L2 cache
0x2000	pmEventSelect0	Performance monitor event select 0
0x2008	pmEventSelect1	Performance monitor event select 1
0x2010	pmEventSelect2	Performance monitor event select 2
0x2018	pmEventSelect3	Performance monitor event select 3
0x2020	pmEventSelect4	Performance monitor event select 4
0x2028	pmEventSelect5	Performance monitor event select 5
0x2800	pmClientFilter	Performance counter client disable mask
0x3000	pmEventCounter0	Performance monitor event counter 0
0x3008	pmEventCounter1	Performance monitor event counter 1
0x3010	pmEventCounter2	Performance monitor event counter 2
0x3018	pmEventCounter3	Performance monitor event counter 3
0x3020	pmEventCounter4	Performance monitor event counter 4
0x3028	pmEventCounter5	Performance monitor event counter 5

- **Private L2 Performance Monitor**

Similar to the hardware performance monitor (HPM) for the core, the pL2 Cache also has a pL2 performance monitoring (pL2PM) facility. It consists of a set of event-programmable counters and their event selector registers. The registers are available to control the behavior of the counters. The performance event selector and other control registers are configured in machine mode, and the event-programmable counters can be read in user mode.

- **PL2PM Event Selector Registers (pmEventSelectX)**

The pL2PM event selector CSRs (pmEventSelect0-5) follow the definition of mhpmeventX as defined in the RISC-V ISA. They are 64-bit bit WARL registers. To control the event type to count, these CSRs are used to program the corresponding event counters.

Table 3-92 PL2 Performance Monitor Event Selectors

Offset	Bits	Access	Description
0x2000	64	RW	pmEventSelect0
0x2008	64	RW	pmEventSelect1
0x2010	64	RW	pmEventSelect2
0x2018	64	RW	pmEventSelect3
0x2020	64	RW	pmEventSelect4
0x2028	64	RW	pmEventSelect5

The event selectors are partitioned into two fields: the lower 8 bits select an event class, and the upper bits form a mask of events in that class.

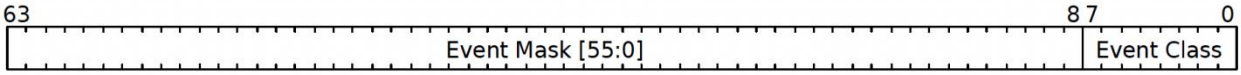


Figure 3-16 Event Selector Fields

The counter increments if the event corresponding to any set mask bit occurs.

- **PL2PM Counter Client Filter CSR (pmClientFilter)**

The pL2PM counter client filter register pmClientFilter is a 64-bit WARL register that controls which client's performance events are excluded from incrementing the performance monitor counters. However, some events might not be attributed to a particular client in a specific imple- mentation (for example, the pL2 release event in pL2 Cache design).

Table 3-93 PL2 Performance Monitor Counter Client Filter

Offset	Bits	Access	Description
0x2800	64	RW	pmClientFilter

The mapping between masters and their pL2 master IDs is shown in Table below.

Table 3-94 Master IDs in the Private L2 Cache

Master ID	Description
0	Fetch Unit
1	Load Store Unit Cache 0
2	Load Store Unit Cache 1

Master ID	Description
3	Load Store Unit IO
4	Load Store Unit HWPF

Note that the default value of pmClientFilter is 0, meaning all clients' events are counted. To disable event counting for a particular client, set its corresponding bit in pmClientFilter.

- **PL2 Performance Monitor Counters (pmEventCounterX)**

The pL2 Performance Monitor counters (pmEventCounter0-5), follow the definition of mhpmcounterX as defined in the RISC-V ISA. They are 64-bit WARL registers.

Table 3-95 PL2 Performance Monitor Counters

Offset	Bits	Access	Description
0x3000	64	RW	pmEventCounter0
0x3008	64	RW	pmEventCounter1
0x3010	64	RW	pmEventCounter2
0x3018	64	RW	pmEventCounter3
0x3020	64	RW	pmEventCounter4
0x3028	64	RW	pmEventCounter5

- Event Selector Encodings

Table below describes the event selector encodings available. Events are categorized into two classes based on the Event Class field encoded in pmEventSelectX[7:0]. One or more events

can be programmed by setting the respective Event Mask bit for a given event class. An event selector encoding of 0 means "count nothing". Multiple events will cause the counter to incre- ment any time any of the selected events occur.

Table 3-96 PL2PM pmEventSelect Register

pL2 Performance Monitor Event Register		
Inner Transaction Events, pmEventSelectX[7:0]=0x1		
Bits	Description	Can use ClientFilter?
0	PutFullData request from inner cache	Yes
1	PutPartialData request from inner cache	Yes
2	ArithmeticData / LogicalData request from inner cache	Yes
3	Get request from inner cache	Yes

pL2 Performance Monitor Event Register		
Inner Transaction Events, pmEventSelectX[7:0]=0x1		
Bits	Description	Can use ClientFilter?
4	PrefetchRead request from inner cache	Yes
5	PrefetchWrite request from inner cache	Yes
6	AcquireBlock.NtoB request from inner cache	Yes
7	AcquireBlock.NtoT request from inner cache	Yes
8	AcquireBlock.BtoT request from inner cache	Yes
9	AcquirePerm.NtoT request from inner cache	Yes
10	AcquirePerm.BtoT request from inner cache	Yes
11	Release.TtoB request from inner cache	Yes
12	Release.TtoN request from inner cache	Yes
13	Release.BtoN request from inner cache	Yes
14	ReleaseData.TtoB request from inner cache	Yes
15	ReleaseData.TtoN request from inner cache	Yes
16	ReleaseData.BtoN request from inner cache	Yes
17	ReleaseData.TtoT request from inner cache	Yes
18	ProbeBlock.toT request from inner cache	Yes
19	ProbeBlock.toB request from inner cache	Yes
20	ProbeBlock.toN request from inner cache	Yes
21	ProbePerm.toN request from inner cache	Yes
22	ProbeAck.TtoB request from inner cache	Yes
23	ProbeAck.TtoN request from inner cache	Yes
24	ProbeAck.BtoN request from inner cache	Yes
25	ProbeAck.TtoT request from inner cache	Yes
26	ProbeAck.BtoB request from inner cache	Yes
27	ProbeAck.NtoN request from inner cache	Yes
28	ProbeAckData.TtoB request from inner cache	Yes
29	ProbeAckData.TtoN request from inner cache	Yes
30	ProbeAckData.TtoT request from inner cache	Yes

pL2 Query Result Events, pmEventSelectX[7:0]=0x2		
Bits	**Description**	**Can use ClientFilter?**
0	PutFullData.Hit request from inner cache hits a valid line in pL2	Yes
1	PutPartialData.Hit request from inner cache hits a valid line in pL2	Yes
2	ArithmeticData / LogicalData request from inner cache hits a valid line in pL2	Yes
3	Get request from inner cache hits a valid line in pL2	Yes
4	PrefetchRead request from inner cache hits a valid line in pL2	Yes
5	AcquireBlock.NtoB request from inner cache hits a valid line in pL2	Yes
6	AcquirePerm.NtoT request from inner cache hits a valid line in pL2	Yes
7	Release.TtoB request from inner cache hits a valid line in pL2	Yes
8	ReleaseData.TtoB request from inner cache hits a valid line in pL2	Yes
9	ProbeBlock.toT request from outer cache hits a valid line in pL2	No
10	PutFullData request from inner cache hits a shared line in pL2	Yes
11	PutPartialData request from inner cache hits a shared line in pL2	Yes
12	ArithmeticData / LogicalData request from inner cache hits a shared line in pL2	Yes
13	Get request from inner cache hits a shared line in pL2	Yes
14	PrefetchRead request from inner cache hits a shared line in pL2	Yes
15	AcquireBlock.NtoB request from inner cache hits a shared line in pL2	Yes
16	AcquirePerm.NtoT request from inner cache hits a shared line in pL2	Yes
17	ProbeBlock.toT request from outer cache hits a shared line in pL2	No
18	ProbeBlock.toT request from outer cache hits a dirty line in pL2	No

Outer Transaction Events, pmEventSelectX[7:0]=0x3		
Bits	**Descripton**	**Can use ClientFilter?**
0	PutFullData request to outer cache	No
1	PutPartialData request to outer cache	No
2	ArithmeticData / LogicalData request to outer cache	No

Outer Transaction Events, pmEventSelectX[7:0]=0x3		
Bits	Descripton	Can use ClientFilter?
3	Get request to outer cache	No
4	PrefetchRead request to outer cache	No
5	PrefetchWrite request to outer cache	No
6	AcquireBlock.NtoB request to outer cache	No
7	AcquireBlock.NtoT request to outer cache	No
8	AcquireBlock.BtoT request to outer cache	No
9	AcquirePerm.NtoT request to outer cache	No
10	AcquirePerm.BtoT request to outer cache	No
11	Release.TtoB request to outer cache	No
12	Release.TtoN request to outer cache	No
13	Release.BtoN request to outer cache	No
14	ReleaseData.TtoB request to outer cache	No
15	ReleaseData.TtoN request to outer cache	No
16	ReleaseData.BtoN request to outer cache	No
17	ReleaseData.TtoT request to outer cache	No
18	ProbeBlock.toT request from outer cache	No
19	ProbeBlock.toB request from outer cache	No
20	ProbeBlock.toN request from outer cache	No
21	ProbePerm.toN request from outer cache	No
22	ProbeAck.TtoB request to outer cache	No
23	ProbeAck.TtoN request to outer cache	No
24	ProbeAck.BtoN request to outer cache	No
25	ProbeAck.TtoT request to outer cache	No
26	ProbeAck.BtoB request to outer cache	No
27	ProbeAck.NtoN request to outer cache	No
28	ProbeAckData.TtoB request to outer cache	No
29	ProbeAckData.TtoN request to outer cache	No
30	ProbeAckData.TtoT request to outer cache	No

Other pL2 Microarchitectural Events, pmEventSelectX[7:0]=0x4		
Bits	Description	Can use ClientFilter?

0	Hint request from inner cache hits an inflight miss request	Yes
1	Aquire / Get request from inner cache hits an inflight miss request	Yes
2	Put / ArithmeticData / LogicalData request from inner cache hits an inflight miss request	Yes
3	Hint / Acquire / Get request from inner cache replays	Yes
4	Put / ArithmeticData / LogicData request from inner cache replays	Yes
5	Probe request from outer cache replays	No
Note: "Inner cache" refers to the inner side of the pL2 cache, i.e., transactions from the core. "Outer cache" refers to the other side of the pL2 Cache, i.e., read/write transactions from the pL2 and probe requests from outside.		

- **Register Descriptions**

This section describes the functionality of the memory-mapped registers in the Private L2 Cache.

- **PL2 Cache Configuration Register (Config)**

The Config Register can be used to programmatically determine information regarding the cache size and organization.

Table 3-97 PL2 Cache Configuration Register

PL2 Cache Configuration Register (Config)				
Register Offset		**0x0**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	Banks	RO	0x2	Number of banks in the cache
[15:8]	Ways	RO	0x8	Number of ways per bank
[23:16]	lgSets	RO	0x9	Base-2 logarithm of the total number of sets
[31:24]	lgBlockBytes	RO	0x6	Base-2 logarithm of the bytes per cache block

- **ECC Error Injection Register (ECCInjectError)**

The ECCInjectError register can be used to inject single-bit or double-bit ECC errors into either the

tag/state or data SRAM. This function can be used to test error correction logic, mea- surement, and recovery. Note that when an uncorrectable ECC error is injected into either tag/ state or data SRAM, the program may lose data coherence and consistency, which may impact normal behavior.

Table 3-98 ECC Error Injection Register

ECC Error Injection Register (ECCInjectError)				
Register Offset		0x40		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	ECCToggleBit	RW	0x0	Toggle (corrupt) this bit index on the next cache operation
[15:8]	Reserved			
[17:16]	InjectionTarget	RW	0x0	Inject error in 0=data or 1=tag/state or 2=lru
18	InjectOnWrite	RW	0x0	Inject error on (0) read data or (1) write data
19	InjectionType	RW	0x0	Inject correctable (0) or uncorrectable (1) error
[31:20]	Reserved			

ECC injection only occurs on the next cacheable read or write. If software wants to know the status of the injection, it needs to poll the EccInjectErrorDone register.

- **ECC Error Injection Done Register (EccInjectErrorDone)**

ECCInjectErrorDone indicates if the ECC error injection is complete.

Table 3-99 ECC Error Injection Done Register

ECC Error Injection Register (ECCInjectError)				
Register Offset		0x40		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	ECCToggleBit	RW	0x0	Toggle (corrupt) this bit index on the next cache operation
[15:8]	Reserved			
[17:16]	InjectionTarget	RW	0x0	Inject error in 0=data or 1=tag/state or 2=lru
18	InjectOnWrite	RW	0x0	Inject error on (0) read data or (1) write data
19	InjectionType	RW	0x0	Inject correctable (0) or uncorrectable (1) error
[31:20]	Reserved			

- **PL2 Cache Flush Register (Cflush64)**

Writes to the Flush64_Command field generates a cache flush request; the response indicates that the pL2 cache has sunk the request, but may or may not perform it. This behavior is to enable multiple outstanding cache flush requests. If software wants to know the status of previously issued requests, it needs to poll the FlushCount register.

Table 3-100 PL2 Cache Flush Register

PL2 Cache Flush Register (Cflush64)				
Register Offset		**0x200**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[31:0]	Flush64_Address_Low	WO	0x0	For CFlush_PA_* commands, defines the lower 32-bit physical address. For CFlush_SetWay_* commands, defines the set number.
[55:32]	Flush64_Address_High	WO	0x0	For CFlush_PA_* commands, defines the upper 24-bit physical address. For CFlush_SetWay_* commands, defines the set number.
[63:56]	Flush64_Command	WO	0x0	The command of cache flush

Table below describes the defined cache flush commands.

Table 3-101 Cache Flush Commands

Message	**Bits**		**Description**
	[7:3]	[2:0]	
Reserved	0	0	
Reserved	0	1	
Reserved	0	2	
CFlush_PA_CleanInv	0	3	Evict the cache line of the target physical address
CFlush_SetWay_Refresh	1	0	Read the cache line of the specified set/way from cache, and write it back into cache
Reserved	1	1	
Reserved	1	2	
CFlush_SetWay_CleanInv	1	3	Evict the cache line of the target set/way

Message	Bits		Description
	[7:3]	[2:0]	
CFlush_Cache_Refresh	2	0	Read the cache line of each set/way from cache, and write it back into cache
Reserved	2	1	
Reserved	2	2	
CFlush_Cache_CleanInv	2	3	Evict all valid cache lines

When an illegal command is received, or an illegal set or way is given for a set/way target request, it is treated as a nop.

- **Flush Count Register (FlushCount)**

FlushCount is a read-only register that contains the number of outstanding cache flush requests. 0 indicates that all previously issued cache flush requests have been completed.

Table 3-102 PL2 Flush Count Register

PL2 Flush Count Register (FlushCount)				
Register Offset		0x208		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	FlushCount	RO	0x0	The number of outstanding cflush requests. Note that this field is volatile and may be masked in IP-XACT.

- **PL2 Configuration Register (pL2Config)**

This register contains bits to configure various features of the pL2 cache.

Table 3-103 PL2 Configuration 0 Register

PL2 Configuration 0 Register (pL2Config0)				
Register Address		0x1008		
Bits	Field Name	Attr.	Rst.	Description
[2:0]	Reserved	RO	0x0	
3	cleanEvictEnable	RW	0x1	When set, enable clean eviction from pL2 to an outer cache/ directory. This bit will affect the L3 cache miss rate.
[8:4]	Reserved	RO	0x0	
9	l2AvoidL1LineDisable	RW	0x1	When set, L2 victim can pick L1 cache lines as well. This bit will affect pL2

PL2 Configuration 0 Register (pL2Config0)				
Register Address		0x1008		
Bits	Field Name	Attr.	Rst.	Description
				cache miss rate.
[15:10]	Reserved	RO	0x0	
16	softwareEccInjectEnable	RW	0x0	When set, enable software ECC injection
17	errInjectOnWriteEnable	RW	0x1	When set, enable error injecton on tag/data RAM write (latent error). When clear, inject only on the next read. This is useful to test ECC correction logic.
18	errInjectOnReadEnable	RW	0x0	When set, enable error injection on tag/data RAM read.
19	dataUceInjectEnable	RW	0x0	When set, enable uncorrectable data error injection. When clear, inject only correctable data errors.
20	dirUceInjectEnable	RW	0x0	When set, enable uncorrectable metadata error injection. When clear, inject only correctable metadata errors.
[31:21]	Reserved	RO	0x0	

3.4.1.14 Bus-Error Unit

The Bus-Error Unit is a per-processor device that records erroneous events and reports them using platform-level and hart-local interrupts. Figure below shows the connections from the core to the BEU.

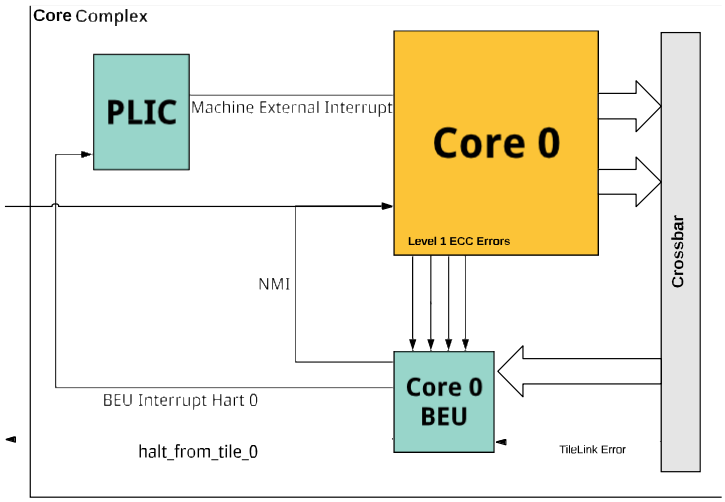


Figure 3-17 Bus-Error Unit Block Diagram

The BEU can be configured to generate interrupts on correctable memory errors, uncorrectable memory errors, and/or TileLink bus errors. When an error occurs, the BEU will hold the address of the error and a code to describe the error.

- **Memory Map**

The Bus-Error Unit memory map is shown in Table below.

Table 3-104 Bus-Error Unit Memory Map

Offset	Name	Attr.	Description
0x00	cause	RW	Cause of error event
0x08	value	RW	Physical address of error event
0x10	enable	RW	Event enable mask
0x18	plic_interrupt	RW	Platform-level interrupt enable mask
0x20	accrued	RW	Accrued event mask
0x28	local_interrupt	RW	Hart-local interrupt-enable mask

Each bit in enable, plic_interrupt, accrued, and local_interrupt is associated with an error event, described in Table below. The cause register represents which event occurred, and the value register contains the address of that error event. The value register will be updated when errors occur on memory that supports data cache. For example, the Memory Port has data cache support, where the System and Peripheral Ports do not.

- **Reportable Errors**

Table below lists the events that the Bus-Error Unit may report.

Table 3-105 Bus-Error Unit Error Events

Value	Description
0	No error
1	Instruction cache bus error
2	Reserved
3	Reserved
4	Reserved
5	Load/Store/PTW bus error
6	Data cache correctable ECC error
7	Data cache uncorrectable ECC error

8	Reserved
9	Private L2 bus error
10	Private L2 cache correctable ECC error
11	Private L2 cache uncorrectable ECC error

- **Functional Description**

When one of the events listed in Table above occurs, the Bus-Error Unit can record information about that event and can generate an interrupt to a platform-level interrupt controller (i.e., the PLIC) or locally to the hart. The enable register contains a mask of which events the BEU can record. Each bit in enable corresponds to an event in Table above. For example, if enable[5] is set, the BEU will record load, store, or PTW TileLink bus errors. If plic_interrupt[5] is also set, then a global interrupt will be asserted to the platform-level interrupt controller. Alternatively, if local_interrupt[5] is set, then a BEU error will result in a local interrupt to the hart. The cause register indicates the event the BEU has most recently recorded, e.g., a value of 5 indi- cates a load, store, or PTW TileLink bus error was recorded.

The cause value 0 is reserved to indicate "no error". cause is only written for events enabled in the enable register. Furthermore, cause is only written when its current value is 0; that is, if mul- tiple events occur, only the first one is latched, until software clears the cause register.

The value register supplies the physical address that caused the event, or 0 if the address is unknown. The BEU writes the value register whenever it writes the cause register, such as when an event enabled in the enable register occurs or when cause contains 0.

The bit position in the accrued register indicates which events have occurred since the last time it was cleared by software. Its format is the same as the enable register. The BEU sets bits in the accrued register whether or not they are enabled in the enable register.

- **BEU Global Interrupt**

The bit position in the plic_interrupt register indicates which accrued events should generate an interrupt into the PLIC. An interrupt is generated when the same bit is set in both accrued and plic_interrupt. The PLIC drives the machine external interrupt to the core, which has an interrupt ID of 11 (0xB). The exception code value, located in mcause (machine trap cause) CSR, will be 11 (0xB) when BEU interrupts are routed through the PLIC. This exception code value is independent of the PLIC interrupt number used to connect the BEU to the PLIC.

The external interrupt handler must clear the BEU accrued bit before exiting.

- **BEU Local Interrupt**

The BEU output signal connects to the non-maskable interrupt (NMI) logic. The local_interrupt register indicates which accrued events will cause a trap to the RNMI interrupt vector address. Within the NMI handler, the nmcause register can be read to determine if it was a BEU error, or an error triggered directly via the NMI rnmi_N input signal.

- **Global Interrupt Configuration**

In addition to writing the BEU registers to enable interrupts, the mstatus.MIE CSR bit should be written to enable Machine level interrupts globally. The mie.MEIE CSR bit should also be config- ured to enable external interrupts when plic_interrupt is enabled to route the PLIC interrupt from the BEU interrupts to the core. Likewise, the corresponding PLIC Hart N M-Mode interrupt enable register should be enabled, and the PLIC Hart N interrupt priority threshold must be set. Additionally, the PLIC interrupt priority

register must be set for the BEU's corresponding PLIC interrupt ID. Refer to the local interrupt chapter for more details regarding interrupt configurations.

3.4.1.15 TL-UH to TL-C Adapter

The TL-UH to TL-C Adapter services a stream of 64 bytes or smaller read and write requests to main memory using concurrently issued cache block accesses to achieve full throughput while maintaining FIFO ordering on the request stream.

The TL-UH to TL-C Adapter performs best when presented with 64-byte PutFull or Get accesses. Performance degrates with multiple concurrent accesses to the same block, therefore it is advisable to combine read/write requests with a Burst Bundler.

- **Memory Map**

The TL-UH to TL-C Adapter memory map is shown in Table below.

Table 3-106 TL-UH to TL-C Adapter Memory Map

Offset	Name	Description
0x0000	bridge_ctrl0	TL-UH to TL-C Adapter bridge_ctrl0
0x0004	bridge_ctrl1	TL-UH to TL-C Adapter bridge_ctrl1
0x0008	bridge_status0	TL-UH to TL-C Adapter bridge_status0
0x0020	error_log_en	TL-UH to TL-C Adapter error_log_en
0x0021	error_contain_en	TL-UH to TL-C Adapter error_contain_en
0x0022	error_signal_en	TL-UH to TL-C Adapter error_signal_en
0x0023	error_signal_pending	TL-UH to TL-C Adapter error_signal_pending
0x0024	error_signal_pending_set	TL-UH to TL-C Adapter error_signal_pending_set
0x0028	error_claim	TL-UH to TL-C Adapter error_claim
0x002C	error_lost	TL-UH to TL-C Adapter error_lost
0x0030	error_log_status	TL-UH to TL-C Adapter error_log_status
0x0038	error_log_address	TL-UH to TL-C Adapter error_log_address
0x2000	perfmon_global_ctrl	TL-UH to TL-C Adapter perfmon_global_ctrl
0x2008	perfmon_ctrl1	TL-UH to TL-C Adapter perfmon_ctrl1
0x2010	perfmon_ctrl2	TL-UH to TL-C Adapter perfmon_ctrl2
0x2018	perfmon_ctrl3	TL-UH to TL-C Adapter perfmon_ctrl3
0x2020	perfmon_ctrl4	TL-UH to TL-C Adapter perfmon_ctrl4
0x2800	perfmon_data0_s	TL-UH to TL-C Adapter perfmon_data0_s
0x2808	perfmon_data1_s	TL-UH to TL-C Adapter perfmon_data1_s
0x2810	perfmon_data2_s	TL-UH to TL-C Adapter perfmon_data2_s
0x2818	perfmon_data3_s	TL-UH to TL-C Adapter perfmon_data3_s

Offset	Name	Description
0x2820	perfmon_data4_s	TL-UH to TL-C Adapter perfmon_data4_s
0x3000	perfmon_data0_u	TL-UH to TL-C Adapter perfmon_data0_u
0x3008	perfmon_data1_u	TL-UH to TL-C Adapter perfmon_data1_u
0x3010	perfmon_data2_u	TL-UH to TL-C Adapter perfmon_data2_u
0x3018	perfmon_data3_u	TL-UH to TL-C Adapter perfmon_data3_u
0x3020	perfmon_data4_u	TL-UH to TL-C Adapter perfmon_data4_u

- **Register Descriptions**

This section describes the functionality of the memory-mapped registers in the TL-UH to TL-C Adapter.

- **Bridge Control Registers**

Table 3-107 TL-UH to TL-C Adapter Bridge Control Register 0

TL-UH to TL-C Adapter Bridge Control Register 0 (bridge_ctrl0)				
Register Offset		0x0		
Bits	Field Name	Attr.	Rst.	Description
0	en_clean_release_b_to_n	RW	0x0	Send a Release txn when ownership for a non-modified address is downgraded B to N.
1	en_clean_release_t_to_n	RW	0x0	Send a Release txn when ownership for a non-modified address is downgraded T to N.
2	mmio_single_issue	RW	0x0	Limit outstanding MMIO txn count to 1.
3	wait_to_reacquire_conflict	RW	0x0	After Release is complete, wait until conflicting txn is GO head to issue new Acquire.
4	en_probe_downgrade_t_to_b	RW	0x1	Downgrade T to B when Get has acquired a line in the T state and we receive a Probe.toB.
[7:5]	Reserved			

Table 3-108 TL-UH to TL-C Adapter Bridge Control Register 1

TL-UH to TL-C Adapter Bridge Control Register 1 (bridge_ctrl1)				
Register Offset		0x4		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	probe_timeout	RW	0x64	Max number of cycles to go without committing a new transaction before servicing a probe that hits an in-flight address.
[15:8]	Reserved			
[21:16]	buffer_limit	RW	0x3F	Use buffer IDs only up to this limit (0-based).
[31:22]	Reserved			

The bridge_ctrl1.buffer_limit field allows reducing the effective size of the tracker and data buffer in silicon. This field specifies the maximum buffer ID to use:

- 0: Only use buffer 0
- 2: Use buffers 0-2

The TL-UH to TL-C Adapter will watch for writes to this field and if buffer_limit is not equal to the currently in-use limit, then the TL-UH to TL-C Adapter will:

1. Stall all new requests
2. Wait for all outstanding transactions to complete
3. Update the in-use limit and reset all its pointers to buffer 0

- **Bridge Status Register**

Table 3-109 TL-UH to TL-C Adapter Bridge Status Register

TL-UH to TL-C Adapter Bridge Status Register (bridge_status0)				
Register Offset		0x8		
Bits	Field Name	Attr.	Rst.	Description
0	idle	RO	X	Indicates that TL-UH to TL-C Adapter is idle.
[7:1]	Reserved			

- **Error Registers**

error_contain_en controls the behavior of both correctable and uncorrectable errors. If correctable error containment is disabled, correctable errors are trated as uncorrectable errors for containment.

Table 3-110 TL-UH to TL-C Adapter Error Containment Enable Register

TL-UH to TL-C Adapter Error Containment Enable Register (error_contain_en)				
Register Offset		0x21		
Bits	Field Name	Attr.	Rst.	Description
0	buffer_corr	RW	0x1	Enable containment of data buffer correctable ECC errors (apply correction).
1	buffer_uncorr	RW	0x1	Enable containment of data buffer uncorrectable ECC errors (set the 'corrupt' bit).
[7:2]	Reserved			

error_signal_en controls signalling for both correctable and uncorrectable errors.

Table 3-111 TL-UH to TL-C Adapter Error Signal Enable Register

TL-UH to TL-C Adapter Error Signal Enable Register (error_signal_en)				
Register Offset		0x22		
Bits	Field Name	Attr.	Rst.	Description
0	buffer_corr	RW	0x1	Enable signaling of data buffer correctable ECC errors via interrupt.
1	buffer_uncorr	RW	0x1	Enable signaling of data buffer uncorrectable ECC errors via interrupt.
[7:2]	Reserved			

error_signal_pending provides software with a mechanism to read and clear the errors cur- rently being signaled.

Table 3-112 TL-UH to TL-C Adapter Error Signal Pending Register

TL-UH to TL-C Adapter Error Signal Pending Register (error_signal_pending)				
Register Offset		0x23		
Bits	Field Name	Attr.	Rst.	Description
0	buffer_corr	RW1C	0x0	Set by HW when data buffer correctable ECC error interrupt is asserted, write-1-to-clear for SW. Note that this field is volatile and may be masked in IP-XACT.
1	buffer_uncorr	RW1C	0x0	Set by HW when data buffer uncorrectable ECC error interrupt is asserted, write-1-to-clear for SW. Note that this field is volatile

				and may be masked in IP-XACT.
[7:2]	Reserved			

error_signal_pending_set allows software to manually set the error signal status.

Table 3-113 TL-UH to TL-C Adapter Error Signal Pending Register

TL-UH to TL-C Adapter Error Signal Pending Register (error_signal_pending_set)				
Register Offset		0x24		
Bits	Field Name	Attr.	Rst.	Description
0	buffer_corr	RW	0x0	Shadow of error_signal_pending.buffer_corr, provides write-1-to-set ability for SW (intended to be used to test error handling SW, not in normal error handler code). Note that this field is volatile and may be masked in IP-XACT.
1	buffer_uncorr	RW	0x0	Shadow of error_signal_pending.buffer_uncorr, provides write-1-to-set ability for SW (intended to be used to test error handling SW, not in normal error handler code). Note that this field is volatile and may be masked in IP-XACT.
[7:2]	Reserved			

error_claim and error_lost aid software handling of errors that overlap an interrupt handler. Software can write the error_claim register to tell hardware to stop updating the error log, and instead direct errors to the error_lost register.

Once software writes error_claim to indicate it is done handling the logged error, it must check error_lost to ensure that it did not miss an important error.

Table 3-114 TL-UH to TL-C Adapter Error Claim Register

TL-UH to TL-C Adapter Error Claim Register (error_claim)				
Register Offset		0x28		
Bits	Field Name	Attr.	Rst.	Description
0	lock	RW	0x0	Set this field to 1 to prevent HW from updating error_log_status/address while processing an error event. Errors occuring while error_lock=1 will be funnelled to error_lost.
[7:1]	Reserved			

Table 3-115 TL-UH to TL-C Adapter Error Lost Register

TL-UH to TL-C Adapter Error Lost Register (error_lost)				
Register Offset		0x2C		
Bits	Field Name	Attr.	Rst.	Description
0	valid	RW	0x0	Error lost is valid. Note that this field is volatile and may be masked in IP-XACT.
1	ovfl	RW	0x0	Error lost overflowed. Note that this field is volatile and may be masked in IP-XACT.
[3:2]	Reserved			
[7:4]	type	RW	0x0	Type of error lost. Note that this field is volatile and may be masked in IP-XACT. • 0x0 - Correctable data buffer error • 0x1 - Uncorrectable data buffer error

- **Error Log Registers**

error_log_en controls which errors are logged. During normal operation, error_log_status and error_log_address are written by hardware when errors are detected. In the case that the error log registers are alrady marked valid when an error occurs, the error log is updated to indi- cate overflow, and the highest-priority error that was unable to be logged is recorded. Uncor- rectable data errors have a higher priority than correctable data errors.

If the new error is higher priority than the currently logged error, then the new error will overwrite the old error in the log and the old error will be marked as the highest-priority overflow error. If the new error is the same or lower priority than the currently logged error, then the only bits that may be updated are the overflow bits.

Table 3-116 TL-UH to TL-C Adapter Error Log Enable Register

TL-UH to TL-C Adapter Error Log Enable Register (error_log_en)				
Register Offset		0x20		
Bits	Field Name	Attr.	Rst.	Description
0	buffer_corr	RW	0x1	Enable logging of data buffer correctable ECC errors.
1	buffer_uncorr	RW	0x1	Enable logging of data buffer uncorrectable ECC errors.
[7:2]	Reserved			

Table 3-117 TL-UH to TL-C Adapter Error Log Status Register

TL-UH to TL-C Adapter Error Log Status Register (error_log_status)				
Register Offset		0x30		
Bits	Field Name	Attr.	Rst.	Description
0	valid	RW	0x0	Error log is valid. Note that this field is volatile and may be masked in IP-XACT.
1	addr_valid	RW	0x0	Error log address is valid. Note that this field is volatile and may be masked in IP-XACT.
[3:2]	Reserved			
[7:4]	type	RW	0x0	Type of error logged. Note that this field is volatile and may be masked in IP-XACT. • 0x0 - Correctable data buffer error • 0x1 - Uncorrectable data buffer error
8	ovfl	RW	0x0	Error log overflowed. Note that this field is volatile and may be masked in IP-XACT.
[11:9]	Reserved			
[15:12]	ovfl_type	RW	0x0	Highest severity error that overflowed. Note that this field is volatile and may be masked in IP-XACT. • 0x0 - Correctable data buffer error • 0x1 - Uncorrectable data buffer error
[31:16]	ext	RW	0x0	Error-specific information. BUFFER_CORR/BUFFER_UNCORR encoding: [0] → error direction (0: data toward requester, 1: data toward memory); [1] → error was detected in data containing underfill read; [8:2] → SRAM row in error. Note that this field is volatile and may be masked in IP-XACT.

Table 3-118 TL-UH to TL-C Adapter Error Log Address Register

TL-UH to TL-C Adapter Error Log Address Register (error_log_address)				
Register Offset		0x38		
Bits	Field Name	Attr.	Rst.	Description
[38:0]	address	RW	0x0	Address corresponding to error_log_status error.

[63:39]	Reserved			

- **TL-UH to TL-C Adapter Performance Monitor Registers**

Table 3-119 TL-UH to TL-C Adapter Performance Monitor Global Control Register

TL-UH to TL-C Adapter Performance Monitor Global Control Register (perfmon_global_ctrl)				
Register Offset		0x2000		
Bits	Field Name	Attr.	Rst.	Description
0	en	RW	0x0	Enable the cycle counter and each configurable counter.
[31:1]	Reserved			
32	clk_en	RW	0x0	Enable performance monitor clocks. Must be set prior to en transitioning 0→1 for counts to be correct.
[63:33]	Reserved			

Table 3-120 TL-UH to TL-C Adapter Performance Monitor Control Register 1

TL-UH to TL-C Adapter Performance Monitor Control Register 1 (perfmon_ctrl1)				
Register Offset		0x2008		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	event_id	RW	0x0	Which event this counter counts. • 0x1 - Tracker allocation events (mask: [0]→coh_get, [1]→coh_put_full, [2]→coh_put_ptl, [3]→mmio_get, [4]→mmio_put_full, [5]→mmio_put_ptl) • 0x2 - Ack tracker allocation events (mask: [0]→coh, [1]→mmio) • 0x3 - Probe tracker allocation events • 0x4 - Tracker occupancy each cycle (mask: [0]→coh, [1]→mmio) • 0x5 - Ack tracker occupancy each cycle (mask: [0]→coh, [1]→mmio) • 0x6 - Probe tracker occupancy each cycle • 0x8 - Global observability stall cycles • 0xA - Probe timeout events

TL-UH to TL-C Adapter Performance Monitor Control Register 1 (perfmon_ctrl1)				
Register Offset		0x2008		
Bits	Field Name	Attr.	Rst.	Description
				• 0xB - Probe timeout cycles • 0xC - Conflict requests (mask: [0]→Rd, [1]→Wr) • 0x10 - Inward stream switches attributes from txn N-1 to N (AND mask: [0]→master, [1]→rd/wr, [2]→coh/mmio) • 0x11 - Successful transfers (mask: [0]→OutA, [1]→OutB, [2]→OutC, [3]→OutD, [4]→OutE, [5]→InA, [6]→InD, [8]→DataBufferAccessAckRd, [9]→DataBufferReleaseRd, [10]→DataBufferGrantWr, [11]→DataBufferPutWr, [12]→OutAResponse,
				[13]→OutARelease, [14]→OutAGlobObs, [15]→OutAPrefetch, [16]→OutCProbeAck, [17]→OutCRelease, [18]→InDTracker, [19]→InDPassthru) • 0x12 - Unsuccessful transfers (mask: [0]→OutA, [1]→OutB, [2]→OutC, [3]→OutD, [4]→OutE, [5]→InA, [6]→InD, [8]→DataBufferAccessAckRd, [9]→DataBufferReleaseRd, [10]→DataBufferGrantWr, [11]→DataBufferPutWr, [12]→OutAResponse, [13]→OutARelease, [14]→OutAGlobObs, [15]→OutAPrefetch, [16]→OutCProbeAck, [17]→OutCRelease, [18]→InDTracker, [19]→InDPassthru) • 0x13 - Probe CAM result (mask: [0]→miss, [1]→I, [2]→S, [3]→E, [4]→M) • 0x14 - Acquire reissued (or not) (mask: [0]→Rd@GO, [1]→Wr@GO, [2]→RD@Release/ issue, [3]→Wr@Release/issue, [4]→Rd@Release/skip, [5]→Wr@Release/skip)

[31:8]	event_mask_lo	RW	0xFFFFFF	Additional counter qualification mask (bits mean different things for different events).
[55:32]	event_mask_hi	RW	0xFFFFFF	Additional counter qualification mask (bits mean different things for different events).
[59:56]	edge_idx	RW	0x0	Which edge is being monitored (in the case that this control node services multiple TL edges).
[63:60]	Reserved			

Table 3-121 TL-UH to TL-C Adapter Performance Monitor Data Register

TL-UH to TL-C Adapter Performance Monitor Data Register (perfmon_data0_s)				
Register Offset		0x2800		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	count_lo	RW	0x0	Supervisor (R/W) copy of perfmon_data0_u.count_lo. Note that this field is volatile and may be masked in IP-XACT.
[63:32]	count_hi	RW	0x0	Supervisor (R/W) copy of perfmon_data0_u.count_hi. Note that this field is volatile and may be masked in IP-XACT.

Table 3-122 TL-UH to TL-C Adapter Performance Monitor Data Register

TL-UH to TL-C Adapter Performance Monitor Data Register (perfmon_data0_u)				
Register Offset		0x3000		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	count_lo	RO	0x0	Bits in lower 4B of counter that increments by 1 each cycle that perfmon_global_ctrl.en==1. Note that this field is volatile and may be masked in IP-XACT.
[63:32]	count_hi	RO	0x0	Bits in upper 4B of counter that increments by 1 each cycle that perfmon_global_ctrl.en==1. Note that this field is volatile and may be masked in IP-XACT.

Table 3-123 Master IDs in the TL-UH to TL-C Adapter

Master ID	Description
0	AXI4 Front Port ID#0 [W]
1	AXI4 Front Port ID#1 [W]
2	AXI4 Front Port ID#2 [W]
3	AXI4 Front Port ID#3 [W]
4	AXI4 Front Port ID#0 [R]
5	AXI4 Front Port ID#1 [R]
6	AXI4 Front Port ID#2 [R]
7	AXI4 Front Port ID#3 [R]

3.4.1.16 Level 3 Cache Controller

This chapter describes the functionality of the Level 3 Cache Controller used in the EIC7700X MCPU.

- **Level 3 Cache Controller Overview**

The Level 3 Cache Controller is used to provide access to fast copies of memory for masters in a Core Complex. The Level 3 Cache Controller also acts as a directory-based coherency manager.

The Level 3 Cache Controller offers extensive flexibility, as it allows for several features in addition to the Level 3 Cache functionality. These include memory-mapped access to L3 Cache RAM for disabled cache ways, scratchpad functionality, way masking and locking, ECC support with error tracking statistics, error injection, and interrupt signaling capabilities.

- **Functional Description**

The EIC7700X MCPU L3 Cache is a 4 MiB 16-way set-associative cache. It has a line size of 64 bytes and is read/write-allocate with a random replacement policy. The cache operates in write-back mode only. The L3 Cache is composed of 4 banks. This subdivision into banks helps facilitate increased available bandwidth between CPU masters and the L3 Cache, as each bank has its own dedicated 256-bit TL-C inner port. As such, multiple requests to different banks may proceed in parallel.

The outer port of the L3 Cache Controller is a 256-bit TL-C port shared among all banks and typically connected to a DDR controller. The outer Memory Port of the L3 Cache Controller is shared among all banks and typically connected to cacheable memory. The overall organization of the L3 Cache Controller is depicted in Figure below.

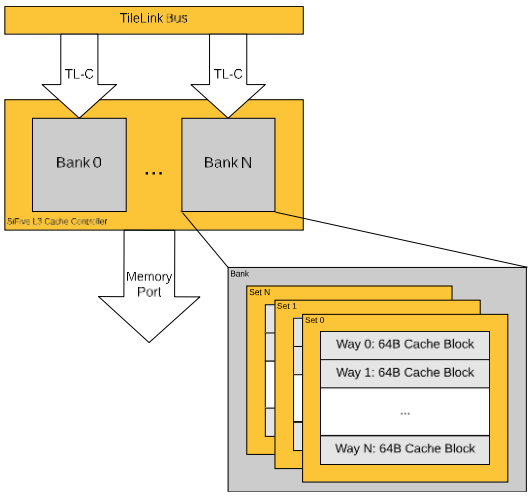


Figure 3-18 Organization of the L3 Cache Controller

- **Way Enable and the L3 Loosely-Integrated Memory (L3 LIM)**

The Level 3 Cache Controller allows for its SRAMs to act either as direct-addressed memory in the Core Complex address space or as a cache that is controlled by the L3 Cache Controller, which can contain a copy of any cacheable address.

When cache ways are disabled, they are addressable in the L3 Loosely-Integrated Memory (L3 LIM) address space as described in the EIC7700X MCPU memory map in Section 5.2. The L3 LIM is an uncacheable port into unused L3 SRAM and provides deterministic access time. It is neither cached by the L1 data cache nor memory backed, as it is just a dedicated software- addressable, low latency, uncached memory. Fetching instructions or data from the L3 LIM pro- vides deterministic behavior equivalent to an L3 Cache hit, with no possibility of a cache miss.

Accesses to the L3 LIM are always given priority over cache way accesses, which target the same L3 Cache bank.

Out of reset, all ways, except for way 0, are disabled. Cache ways can be enabled by writing to the WayEnable register. Once a cache way is enabled, it cannot be disabled unless the EIC7700X MCPU is reset. The highest numbered L3 cache way is mapped to the lowest L3 LIM address space, and way 1 occupies the highest L3 LIM address range. As L3 cache ways are enabled, the size of the L3 LIM address space shrinks. The map- ping of L3 cache ways to L3 LIM address space is shown in Figure below, where N is the number of L3 cache ways, each of size 256 KiB (0x0004_0000).

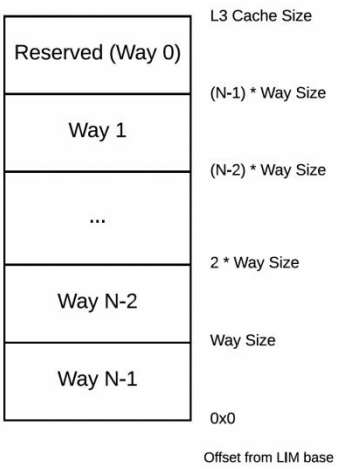


Figure 3-19 Mapping of L3 Cache Ways to L3 LIM Addresses

- **Way Masking and Locking**

The L3 Cache Controller can control the amount of cache memory a CPU master is able to allocate into by using the WayMaskN register. Note that WayMaskN registers only affect allocations, and reads can still occur to ways that are masked. As such, it becomes possible to lock down specific cache ways by masking them in all WayMaskN registers. In this scenario, all masters can still read data in the locked cache ways but cannot evict data.

The following example shows how to lock the L3 cache ways:

1. For the following example, assume the data to be locked is not present in any of the L3 ways.

2. Select an L3 way where the data is to be locked to.

3. From Table *Master IDs in the L3 Cache Controller*, select the Master ID(s) to be used to load the data that is to be locked into the L3.

4. Clear bit M of the WayMaskN register for the other masters, besides the master(s) selected in the step above, to enable allocation by the selected master(s) into locked way M. The WayMask0_M - WayMaskN_M registers are described in Table *L3 Cache Controller Memory Map*.

a. Use Table *L3 Cache Controller Memory Map* to locate the WayMaskN register(s).

5. Enable the WayMask registers for the selected master(s).

a. Set bit M of the WayMaskN register to 1, and clear other bits in that register, to allow master N to be able to allocate data only into way M of the L3.

6. Load the data to be locked into the L3 way M. Note that becaue the L3 is an inclusive cache, any load access by the master N will allocated the cache line for that access into the L3 cache.

7. Set the WayMaskN_M bit for this way to zero. At this point, no other master can evict data from this way.

8. If unlocking the data from way M is desired at some point in the future, set bit M in some or all of the WayMaskN registers.

- **L3 Zero Device**

The L3 Cache Controller has a dedicated scratchpad address region that allows for allo- cation into the cache using an address range that is not memory backed. This address region is denoted as the L3 Zero Device. Writes to the scratchpad region allocate into cache ways that are enabled and not masked.

A Zero Device ignores write data and always returns zero on reads. The EIC7700X MCPU provides a Zero Device behind the L3 Cache, similar to the Memory Port. When combined with locked L3 cache ways, which prevent eviction, locations within a Zero Device's address range appear to retain their value. This provides a mechanism to create L1 and L2 cacheable memory that are essentially backed by L3 SRAM until the way is released (and the value resets to zero). The L3 Zero Device is cacheable like the Memory Port. However, if dirty data is evicted and a write-back to the L3 Zero Device occurs, the Zero Device will discard the write data. Therefore, care must be taken with the scratchpad, as there is no memory backing this address space. Cache evictions from addresses in the scratchpad result in data loss.

The main advantage of the L3 Zero Device over the L3 LIM is that it is a cacheable region allow- ing for data stored to the scratchpad to also be cached in a master's L1 data cache and pL2 cache, which results in faster access.

To understand the difference between the L3 LIM and the L3 Zero Device, consider Figure below. Notice that the L3 LIM accesses the same blocks of memory as the main path into the L3 Cache, whereas the L3 Zero Device sits behind L3 Cache much like the Memory Port:

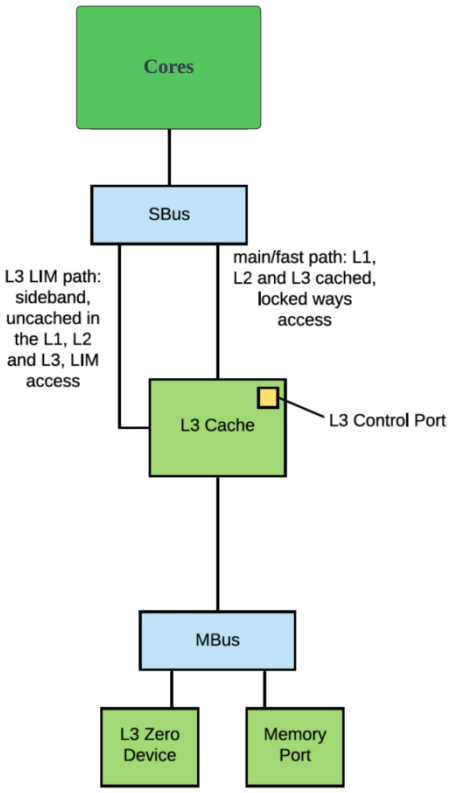


Figure 3-20 Difference between L3 LIM and L3 Zero Device

The recommended procedure for using the L3 Zero Device is as follows:

1. Use the WayEnable register to enable the desired cache ways

2. Designate a single master that will allocate into the scratchpad. For this procedure, we designate this master as Master S. All other masters (CPU and non-CPU) are denoted as Masters N.

3. Masters N: Write to the WayMaskN register to mask the ways that are to be used for the scratchpad. This prevents Masters N from evicting cache lines in the designated scratch- pad ways.

4. Master S: Write to the WayMaskN register to mask all ways except the ways that are to be used for the scratchpad. At this point, Master S should only be able to allocate into the cache ways meant to be used as a scratchpad.

5. Master S: Write scratchpad data into the L3 Zero Device address range

6. Master S: Repeat steps 4 and 5 for each way to be used as scratchpad

7. Master S: Use the WayMaskN register to mask the scratchpad ways for Master S so that it is no longer able to evict cache lines from the designated scratchpad ways

8. At this point, the scratchpad ways should contain the scratchpad data, with all masters able to read, write, and execute from this address space, and no masters able to evict the scratchpad contents

- **L3 Features Access Summary**

Table below describes the L3 features as a function of Way Enable and Way Mask.

Table 3-124 L3 Features Access Summary

Way Enable	Way Mask	Access Base Address	L3 Feature
0	X	0x0800_0000	LIM
1	0	0x8000_0000	Way Masking and Locking — Fast Read access
1	0	0x1A00_0000	Zero Device — Fast Read access from scratchpad
1	1	0x8000_0000	Way Masking and Locking — Write data mode
1	1	0x1A00_0000	Zero Device — Write data to scratchpad

- **L3 Performance Monitor**

Similar to the hardware performance monitor (HPM) for the core, the L3 Cache also has an L3 performance monitoring (L3PM) facility. It consists of a set of event-programmable counters and their event selector registers. The registers are available to control the behavior of the counters. The performance event selector and other control registers are configured in machine mode, and the event-programmable counters can be read in user mode.

- **L3PM Event Selector Registers (pmEventSelectX)**

The L3PM event selector CSRs (pmEventSelect0-5) follow the definition of mhpmeventX as defined in the RISC-V ISA. They are 64-bit bit WARL registers. To control the event type to count, these CSRs are used to program the corresponding event counters.

Table 3-125 L3 Performance Monitor Event Selectors

Offset	Bits	Access	Description
0x2000	64	RW	pmEventSelect0
0x2008	64	RW	pmEventSelect1
0x2010	64	RW	pmEventSelect2
0x2018	64	RW	pmEventSelect3
0x2020	64	RW	pmEventSelect4
0x2028	64	RW	pmEventSelect5

The event selectors are partitioned into two fields: the lower 8 bits select an event class, and the upper bits form a mask of events in that class.

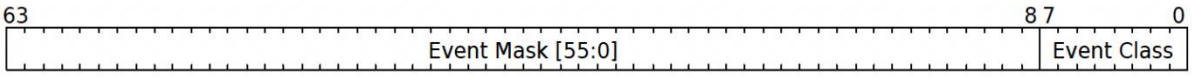


Figure 3-21 Event Selector Fields

The counter increments if the event corresponding to any set mask bit occurs.

- **L3PM Counter Client Filter CSR (pmClientFilterX)**

The L3PM counter client filter register (pmClientFilter0) is a 64-bit WARL register that con- trols which client's performance events are excluded from incrementing the performance moni- tor counters. However, some events might not be attributed to a particular client in a specific implementation (for example, the L3 release event in L3 Cache design).

Table 3-126 L3 Performance Monitor Counter Client Filter

Offset	Bits	Access	Description
0x2800	64	RW	pmClientFilter0

- **L3 Performance Monitor Counters (pmEventCounterX)**

The L3 Performance Monitor counters (pmEventCounter0-5), follow the definition of

mhpmcounterX as defined in the RISC-V ISA. They are 64-bit WARL registers.

Table 3-127 L3 Performance Monitor Counters

Offset	Bits	Access	Description
0x3000	64	RW	pmEventCounter0
0x3008	64	RW	pmEventCounter1
0x3010	64	RW	pmEventCounter2
0x3018	64	RW	pmEventCounter3
0x3020	64	RW	pmEventCounter4
0x3028	64	RW	pmEventCounter5

- **Event Selector Encodings**

Table below describes the event selector encodings available. Events are categorized into two classes based on the Event Class field encoded in pmEventSelectX[7:0]. One or more events can be programmed by setting the respective Event Mask bit for a given event class. An event selector encoding of 0 means "count nothing". Multiple events will cause the counter to incre- ment any time any of the selected events occur.

Table 3-128 L3PM pmEventSelect Register

L3 Performance Monitor Event Register		
Transaction Events, pmEventSelectX[7:0]=0x1		
Bits	Description	Can use ClientFilter?
0	PutFullData request from inner cache	Yes
1	PutPartialData request from inner cache	Yes
2	AtomicData request from inner cache	Yes
3	Get request from inner cache	Yes
4	PrefetchRead request from inner cache	Yes
5	PrefetchWrite request from inner cache	Yes
6	AcquireBlock.NtoB request from inner cache	Yes
7	AcquireBlock.NtoT request from inner cache	Yes
8	AcquireBlock.BtoT request from inner cache	Yes
9	AcquirePerm.NtoT request from inner cache	Yes
10	AcquirePerm.BtoT request from inner cache	Yes
11	Release.TtoB request from inner cache	Yes
12	Release.TtoN request from inner cache	Yes
13	Release.BtoN request from inner cache	Yes
14	ReleaseData.TtoB request from inner cache	Yes
15	ReleaseData.TtoN request from inner cache	Yes
16	ReleaseData.BtoN request from inner cache	Yes
17	ProbeBlock.toT request from outer cache	No
18	ProbeBlock.toB request from outer cache	No
19	ProbeBlock.toN request from outer cache	No

L3 Query Result Events, pmEventSelectX[7:0]=0x2		
Bits	Description	Can use ClientFilter?
0	PutFullData request from inner cache hits a valid line in L3	Yes
1	PutPartialData request from inner cache hits a valid line in L3	Yes
2	AtomicData request from inner cache hits a valid line in L3	Yes
3	Get request from inner cache hits a valid line in L3	Yes
4	Prefetch request from inner cache hits a valid line in L3	Yes

5	AcquireBlock request from inner cache hits a valid line in L3	Yes
6	AcquirePerm request from inner cache hits a valid line in L3	Yes
7	Release request from inner cache hits a valid line in L3	Yes
8	ReleaseData request from inner cache hits a valid line in L3	Yes
9	Probe request from outer cache hits a valid line in L3	No
10	PutFullData request from inner cache hits a shared line in L3	Yes
11	PutPartialData request from inner cache hits a shared line in L3	yes
12	AtomicData request from inner cache hits a shared line in L3	Yes
13	Get request from inner cache hits a shared line in L3	Yes
14	Prefetch request from inner cache hits a shared line in L3	Yes
15	AcquireBlock request from inner cache hits a shared line in L3	Yes
16	AcquirePerm request from inner cache hits a shared line in L3	Yes
17	Probe request from outer cache hits a shared line in L3	No
18	Probe request from outer cache hits a dirty line in L3	No

L3 Request Events, pmEventSelectX[7:0]=0x3		
Bits	**Description**	**Can use ClientFilter?**
0	AcquireBlock.NtoB request to outer cache, miss	No
1	AcquireBlock.NtoT request to outer cache, miss	No
2	AcquireBlock.BtoT request to outer cache, miss	No
3	AcquirePerm.NtoT request to outer cache, miss	No
4	AcquirePerm.BtoT request to outer cache, miss	No
5	Release.TtoB request to outer cache, eviction	No
6	Release.TtoN request to outer cache, eviction	No
7	Release.BtoN request to outer cache, eviction	No
8	ReleaseData.TtoB request to outer cache, not applicable	No
9	ReleaseData.TtoN request to outer cache, dirty eviction	No
10	ReleaseData.BtoN request to outer cache, not applicable	No
11	ProbeBlock.toT request to inner cache, code miss hits other harts	No
12	ProbeBlock.toB request to inner cache, load miss hits other harts	No
13	ProbeBlock.toN request to inner cache, store miss hits other harts	No

Other Events, pmEventSelectX[7:0]=0x4		
Bits	Description	Can use ClientFilter?
0	Hint request from inner cache hits an inflight miss request	Yes
Note: "Inner cache" refers to the inner side of the L3 cache, i.e., transactions from the core and Front Port. "Outer cache" refers to the other side of the L3 Cache, i.e., read/write transactions from the L3 and probe requests from outside.		

- **Setting up the pmClientFilterX Register**

Table Master *IDs in the L3 Cache Controller* shows the mapping of L3 Cache masters and their client ID.

Note that the default value of pmClientFilterX is 0, meaning all clients' events are counted. To disable event counting for a particular client, set its corresponding bit in pmClientFilterX.

- **Programming the pmEventSelect registers**

The following example shows the use of 6 performance counters:

```
// directory lookup events: L2 miss
*pmEventSelect0 = 0x01            | // Event Set 1
                  ( (0x01  << 6 )  |
                    (0x01  << 7 )  |
                    (0x01  << 8 )  |
                    (0x01  << 9 )  |
                    (0x01  << 10)
                  ) << 8;
// directory lookup results: L2 miss hit L3
*pmEventSelect1 = 0x02            | // Event Set 2
                  ( (0x01 << 5)    |
                    (0x01 << 6)
                  ) << 8;

// L2 request misses L3
*pmEventSelect4 = 0x03            | //   Event Set 3
                  ( (0x01 << 0)    |
                    (0x01 << 1)    |
                    (0x01 << 2)    |
                    (0x01 << 3)    |
                    (0x01 << 4)
                  ) << 8;
```

- **Error Correction Codes (ECC)**

The Level 3 Cache Controller supports ECC. ECC is applied to both categories of SRAM used, the data SRAMs and the metadata SRAMs (index, tag, and directory information). The data SRAMs use Single-Error Correcting, Double-Error Detecting (SECDED). The metadata SRAMs use Single-Error Correcting, Double-Error Detecting (SECDED).

Whenever a correctable error is detected, the cache immediately repairs the corrupted bit and writes it back to SRAM. This corrective procedure is completely invisible to application software. However, to support diagnostics, the cache records the address of the most recently corrected metadata and data errors. Whenever a new error is corrected, a counter is increased, and an interrupt is raised. There are independent addresses, counters, and interrupts for correctable metadata and data errors.

DirError, DirFail, DataError, and DataFail signals are used to indicate that an L3 meta- data, uncorrectable L3 metadata, L3 data, or uncorrectable L3 data error has occurred, respec- tively. These signals are connected to the PLIC and are cleared upon reading their respective count registers.

- **Coherence**

The L3 Cache is partially inclusive of the L1 instruction cache and is inclusive of the L1 data cache. When a block of data is allocated to the L1 cache, it is also allocated to the L3 Cache. When a block is evicted from the L1, the corresponding block in the L3 is then updated and marked dirty.

To understand how coherence is managed differently in the L3 Cache with respect to the L1 instruction and data caches, consider the following rules:

1. Only an instruction cache allocation from the Memory Port will land in the L3 Cache
2. An eviction from the L3 Cache does not cause an eviction from the instruction cache
3. An eviction from the instruction cache does not cause L3 Cache eviction either
4. A discard from the data cache does not invalidate the L3 Cache
5. Following a flush in the L3 Cache, the L3 Cache will back probe lines in L1 data cache

- **Memory Map**

The L3 Cache Controller memory map is shown in Table below.

Table 3-129 L3 Cache Controller Memory Map

Offset	Name	Description
0x0000	Config	Information about the Cache Configuration
0x0008	WayEnable	The index of the largest way which has been enabled. May only be increased.
0x0040	ECCInjectError	Inject an ECC Error
0x0100	DirECCFixLow	The low 32-bits of the most recent address to fail ECC
0x0104	DirECCFixHigh	The high 32-bits of the most recent address to fail ECC
0x0108	DirECCFixCount	Reports the number of times an ECC error occurred
0x0120	DirECCFailLow	The low 32-bits of the most recent address to fail ECC
0x0124	DirECCFailHigh	The high 32-bits of the most recent address to fail ECC
0x0128	DirECCFailCount	Reports the number of times an ECC error occurred
0x0140	DatECCFixLow	The low 32-bits of the most recent address to fail ECC
0x0144	DatECCFixHigh	The high 32-bits of the most recent address to fail ECC
0x0148	DatECCFixCount	Reports the number of times an ECC error occurred
0x0160	DatECCFailLow	The low 32-bits of the most recent address to fail ECC
0x0164	DatECCFailHigh	The high 32-bits of the most recent address to fail ECC
0x0168	DatECCFailCount	Reports the number of times an ECC error occurred
0x0200	Flush64	Flush the physical address equal to the 64-bit written data from the cache

Offset	Name	Description
0x0240	Flush32	Flush the physical address equal to the 32-bit written data << 4 from the cache
0x0800	WayMask0	Master 0 way enable mask register
0x0808	WayMask1	Master 1 way enable mask register
0x0810	WayMask2	Master 2 way enable mask register
0x0818	WayMask3	Master 3 way enable mask register
0x0820	WayMask4	Master 4 way enable mask register
0x0828	WayMask5	Master 5 way enable mask register
0x0830	WayMask6	Master 6 way enable mask register
0x0838	WayMask7	Master 7 way enable mask register
0x0840	WayMask8	Master 8 way enable mask register
0x0848	WayMask9	Master 9 way enable mask register
0x0850	WayMask10	Master 10 way enable mask register
0x0858	WayMask11	Master 11 way enable mask register
0x0860	WayMask12	Master 12 way enable mask register
0x0868	WayMask13	Master 13 way enable mask register
0x1000	FeatureDisable	Feature disable register
0x2000	pmEventSelect0	Performance monitor event select 0
0x2008	pmEventSelect1	Performance monitor event select 1
0x2010	pmEventSelect2	Performance monitor event select 2
0x2018	pmEventSelect3	Performance monitor event select 3
0x2020	pmEventSelect4	Performance monitor event select 4
0x2028	pmEventSelect5	Performance monitor event select 5

Table 3-130 L3 Cache Controller Memory Map

Offset	Name	Description
0x2800	pmClientFilter0	Performance counter client disable mask 0
0x3000	pmEventCounter0	Performance monitor event counter 0
0x3008	pmEventCounter1	Performance monitor event counter 1
0x3010	pmEventCounter2	Performance monitor event counter 2
0x3018	pmEventCounter3	Performance monitor event counter 3
0x3020	pmEventCounter4	Performance monitor event counter 4

0x3028	pmEventCounter5	Performance monitor event counter 5

- **Register Descriptions**

This section describes the functionality of the memory-mapped registers in the Level 3 Cache Controller.

- **L3 Cache Configuration Register (Config)**

The Config Register can be used to programmatically determine information regarding the cache size and organization.

Table 3-131 L3 Cache Configuration Register

L3 Cache Configuration Register (Config)				
Register Offset		**0x0**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	Banks	RO	0x4	Number of banks in the cache
[15:8]	Ways	RO	0x10	Number of ways per bank
[23:16]	lgSets	RO	0xA	Base-2 logarithm of the sets per bank
[31:24]	lgBlockBytes	RO	0x6	Base-2 logarithm of the bytes per cache block

- **Way Enable Register (WayEnable)**

The WayEnable register determines which ways of the Level 3 Cache Controller are enabled as cache. Cache ways that are not enabled are mapped into the EIC7700X MCPU's L3 LIM (Loosely-Integrated Memory).

This register is initialized to 0 on reset and may only be increased. This means that, out of reset, only a single L3 cache way is enabled, as one cache way must always remain enabled. Once a cache way is enabled, the only way to map it back into the L3 LIM address space is by a reset.

Table 3-132 Way Enable Register

Way Enable Register (WayEnable)				
Register Offset		**0x8**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[3:0]	WayEnable	WARL	0x0	The index of the largest way which has been enabled. May only be increased.
[7:4]	Reserved	RO	0x0	

- **ECC Error Injection Register (ECCInjectError)**

The ECCInjectError register can be used to insert an ECC error into either the backing data or metadata SRAM. This function can be used to test error correction logic, measurement, and recovery.

Table 3-133 ECC Error Injection Register

ECC Error Injection Register (ECCInjectError)				
Register Offset		0x40		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	ECCToggleBit	RW	0x0	Toggle (corrupt) this bit index on the next cache operation
[15:8]	Reserved			
16	ECCToggleType	RW	0x0	Toggle (corrupt) a bit in 0=data or 1=directory
[31:17]	Reserved			

- **ECC Directory Fix Registers (DirECCFix*)**

The DirECCFixHigh and DirECCFixLow registers are read-only registers that contain the address of the most recently corrected L2 metadata error. This field supplies only the portions of the address that correspond to the affected set, since all ways are corrected together.

The DirECCFixCount register is a read-only register that contains the number of corrected L3 metadata errors. Reading this register clears the DirError interrupt signal.

Note that these registers are volatile and may be masked in IP-XACT.

- **ECC Directory Fail Registers (DirECCFail*)**

The DirECCFailLow and DirECCFailHigh registers are read-only registers that contains the address of the most recent uncorrected L3 metadata error.

The DirECCFailCount register is a read-only register that contains the number of uncorrected L3 metadata errors. Reading this register clears the DirFail interrupt signal.

Note that these registers are volatile and may be masked in IP-XACT.

- **ECC Data Fix Registers (DatECCFix*)**

The DatECCFixLow and DatECCFixHigh registers are read-only registers that contain the address of the most recently corrected L3 data error.

The DataECCFixCount register is a read-only register that contains the number of corrected L2 data errors. Reading this register clears the DataError interrupt signal described.

Note that these registers are volatile and may be masked in IP-XACT.

- **ECC Data Fail Registers (DatECCFail*)**

The DatECCFailLow and DatECCFailHigh registers are a read-only registers that contain the address of the most recent uncorrected L3 data error.

The DatECCFailCount register is a read-only register that contains the number of uncorrected L2 data errors. Reading this register clears the DataFail interrupt signal described.

Note that these registers are volatile and may be masked in IP-XACT.

- **L3 Cache ECC Error Injection and Correction**

Writes to the ECCInjectError register, specifies a bit position within the combined data+ECC (or directoryEntry+ECC) granule to toggle or corrupt exactly 1 bit the next time an L3 entry is written.

The ECCToggleBit register specifies a hex value corresponding to which bit to toggle. For example, if you write a value of 0x47, bit 71 will be toggled (consequently, for a write to an L3 data RAM that has 64 bits of data and 8 bits of ECC, writing a value > 0x47 will have no effect).

The error will be injected (i.e., the bit will be toggled/corrupted) on the next write. When there are multiple data+ECC granules comprising a cache line, error injection will always be applied to the first of these granules. For instance, when you have a 64+8-bit granule and a 64-byte line, it takes eight granules to make up a full cache line, and errors will be injected into the specified bit in the first of these eight granules. For metadata, this is a non-issue since there is only one directoryEntry+ECC granule for each cache line.

The error will not be reported until the corrupted entry is subsequently read, and the error is detected/corrected; that is, this will be at an arbitrary time after the ECCInjectError Register is written, unless a test specifically ensures that it reads the particular set and way that was corrupted due to that write.

The address reported in the fix/fail address registers (DirECCFix and DirECCFail, respectively) will always be the cache-line-size-aligned address of the start of the cache line in which the error was detected.

For error injection on the data, if the L1-L3 interface (and consequently, the width of the L3 data R/W interface) is larger than the ECC granule, the designated bit will be toggled on multiple granules; the number of granules being as many granules as make up the data interface.

For example, if the L3 data interface is 128-bits wide and the data+ECC granule is 72-bits, two bits will get toggled—one per granule. This is, however, invisible to software since, when these bits are read, only a single corrected error will be reported, and the address will be the start of the cache line as stated above.

For the reported fix address (DirECCFix) on corrected errors in the metadata, only the address bits corresponding to the set (i.e., the index) are captured. In contrast, address bits correspond- ing to the tag are not captured and they will read out as 0.

- **Cache Flush Registers (Flush*)**

The EIC7700X MCPU L3 Cache Controller provides two registers that can be used for flushing specific cache blocks.

Flush64 is a 64-bit write-only register that flushes the cache block containing the address writ- ten. Flush32 is a 32-bit write-only register that flushes a cache block containing the written address left shifted by 4 bits. In both registers, all bits must be written in a single access for the flush to take effect.

The flush operation performs a write-back and invalidate, meaning the contents are written to memory and L3, pL2, and L1 cache lines are then invalidated.

- **Way Mask Registers (WayMask*)**

The WayMaskN register allows a master connected to the L3 Cache Controller to specify which L3 Cache ways can be evicted by Master N. Masters can still access memory cached in masked ways. The mapping between masters and their L3 master IDs is shown in Table Master IDs in the L3 Cache Controller.

At least one cache way must be enabled. It is recommended to set/clear bits in this register using atomic

operations.

Table 3-134 Way Mask 0 Register

ECC Error Injection Register (ECCInjectError)				
Register Offset		0x40		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	ECCToggleBit	RW	0x0	Toggle (corrupt) this bit index on the next cache operation
[15:8]	Reserved			
16	ECCToggleType	RW	0x0	Toggle (corrupt) a bit in 0=data or 1=directory
[31:17]	Reserved			

Table 3-135 Master IDs in the L3 Cache Controller

Master ID	Description
0	Debug
1	Hart 0 Private L2 Cache
2	Hart 1 Private L2 Cache
3	Hart 2 Private L2 Cache
4	Hart 3 Private L2 Cache
5	Trace Sink SBA
6	TL-UH to TL-C Adapter
7	inter_mem_slave_port_tl
8	inter_mem_slave_port_tl
9	inter_mem_slave_port_tl
10	inter_mem_slave_port_tl
11	inter_mem_slave_port_tl
12	inter_mem_slave_port_tl
13	inter_mem_slave_port_tl

- **L3 Cache Feature Disable Register (FeatureDisable)**

The FeatureDisable register controls the architectural clock gating functionality of the L3 Cache, which gates clock nodes when not active to reduce its clock tree and power. The FeatureDisable register controls whether trunk, regional, or "slow" regional clock gating features are enabled.

Table 3-136 L3 Cache Feature Disable Register

L3 Cache Feature Disable Register (FeatureDisable)				
Register Offset		0x1000		
Bits	Field Name	Attr.	Rst.	Description
0	disableTrunkClockGate	RW	0x1	When set, disables the trunk clock gating feature. When clear, dynamic gating of the L3 cache clock trunk is enabled. This clock gater incurs a 2-cycle penalty when exiting the WFI clock gated state and saves the most power. All cores in a Core Complex must be in WFI before this state is entered.
1	disableClockGate	RW	0x1	When set, disables all regional clock gating. When clear, non-aggressive clock gating is enabled. This clock gater has no penalty for exit, but less power savings than regional "slow" clock gating.
2	disableClockGateSlow	RW	0x1	When set, disables the regional clock gating "slow" feature. When clear, if regional clock gating is also enabled, the aggressive clock gating feature is enabled. This clock gater imposes a 1-cycle exit penalty from the gated state (on the first TL channel-A request), but saves more power than the no-penalty regional clock gate option.
[7:3]	Reserved			

- **Procedure to Flush the L3 Cache**

This section describes how to flush the L3 Cache using the Zero Device scratchpad. As the scratchpad resides in the L3 Cache, allocations made to the scratchpad can result in an eviction of a cache line that was allocated from the Memory Port. By controlling the WayMaskN register and targeted scratchpad address, it is possible to perform different flush operations.

As the scratchpad region is cacheable, it is necessary to perform a flush of the scratchpad region used to perform the L3 flush. Otherwise, if the targeted scratchpad line already exists in the L3 Cache, either through a direct access or speculation, it will prevent the desired L3 cache line from being evicted from the Memory Port when it is accessed. The steps outlined below include this necessary flush of the scratchpad region.

For flushing ranges of data, it is recommended to use the flush-by-address function in the L3 Controller space.

- **Flushing a Single Index+Way**

1. Write WayMaskN to allow evictions from only the desired way

2. Issue a FENCE instruction

3. Determine address A in the L3 scratchpad region that corresponds to the desired index to be flushed. For example, consider a 2 MiB 16-way cache with the Zero Device located at 0x0A00_000. To target

index 8, the valid addresss would be {0x0A00_0200, 0x0A02_0200, 0x0A04_0200, …, 0x0A1E_0200}.

4. Write address A to the L3 Controller Flush64 register

5. Issue a load from address A (or a store to address A)

6. (Optional) To force observation of the flush, write address A to the L3 Flush64 register and issue a FENCE instruction

7. Restore the WayMaskN register to the original value

8. Issue a FENCE instruction

- **Flushing the Entire L3 Cache**

1. Write WayMaskN to allow evictions from only way 0

2. Issue a FENCE instruction

3. Flush the first way-size of L3 scratchpad memory using a series of Flush64 operations. For example, consider a 2 MiB 16-way cache with the Zero Device located at 0x0A00_000. 2048 lines (128 KiB) in the region from 0xA000_0000-0xA001_FFC0 must be targeted with a Flush64 operation.

4. Issue a FENCE instruction

5. Access the first way-size of L3 scratchpad memory using a series of load or store opera- tions that correspond to each index of the L3 Cache. Considering the same cache example (2 MiB, 16-way), all 2048 lines in the region from 0x0A00_0000-0xA001_FFC0 must be accessed. Only one access per 64 B cache line is required. Each scratchpad access will cause an eviction if the corresponding cache index is dirty.

6. Issue a FENCE instruction

7. Repeat steps 1-6 for the next way to be flushed until all ways have been flushed. The region flushed in step 3 needs to advance by the cache way size, as does the region accessed in step 5. Considering the same example, address range 0xA002_0000-0xA003_FFC0 will be targeted for way 1 operations.

8. (Optional) To force observation of the complete flush, repeat steps 3 and 4

9. Restore the WayMaskN register to the original value

10. Issue a FENCE instruction Memory accesses from harts not performing the flush can inter- fere with the flush operation. It is recommended to put other harts into WFI or a holding loop until the L3 flush is completed.

3.4.1.17 Power Management

The following chapter describes power modes and establishes flows for powering up, powering down, and resetting the hardware of the EIC7700X MCPU.

- **Power Modes**

Power modes include normal run mode with the Power Dial option and wait-for-interrupt clock gating mode using the WFI instruction. Additionally, there is a full power down mode supported via the CEASE instruction. These modes are covered in detail below.

- **Run Mode**

The hart is fully operational in run mode, and EIC7700X designs include the option to include coarse-grained architectural clock gating. When this feature is enabled in the hart, configured instruction cache, data cache, integer pipeline, Debug Logic, and Floating-Point Unit (FPU) modules each contain their own clock gate. The clock gating feature will enable automatic clock gating of functional units when they are

inactive and allow the hart to gate its own clock(s) based on activity.

- **Power Control**
- **Core Clock**

To further reduce power while in run mode, users may choose to reduce external_source_for_core_N_clock, which is required to be changed synchronously to the rest of the clocks in the system. It is important to note that the clock relationships with the rest of the system must still be maintained if external_source_for_core_N_clock is reduced.

- **Power Dial**

To limit maximum power without frequency changes, Power Dial provides a method of scaling down dynamic power in a core. Power is reduced by restricting cycles allowed to advance instructions into the execution pipeline. This feature can typically be used for throttling high CPU usage applications.

Table 3-137 Power Dial CSR

Power Dial CSR				
CSR	0x7C8			
Bits	Field Name	Attr.	Rst.	Description
[3:0]	dutycycle	RW	0x0	(1 - value/16) portion of the peak instruction throughput (i.e., value=0 is no reduction)
[7:4]	Reserved	RO	0x0	

The Power Dial Register may only be programmed in M-mode. When written with a non-zero value, the register restricts peak instruction throughput to the indicated rate. A value of 0 has no effect on instruction throughput. The rate is calculated per 256-cycle period, so a Power Dial value of 1 restricts instruction throughput at a common point in the pipeline to 240 cycles of each 256-cycle period. Reducing the peak rate reduces the worst-case power while minimizing the impact on performance.

- **WFI Clock Gate Mode**

WFI clock gating mode can be entered by executing the WFI instruction. The assembly-level instruction is simply wfi and executing the C method using the GCC compiler can be accom- plished with asm("WFI").

- **WFI Wake Up**

Wake up from a WFI occurs when the hart receives any interrupt. Depending on the software configuration, the hart will either immediately enter the interrupt handler, or resume execution on the instruction immediately after the WFI.

If interrupts are enabled and mstatus.MIE=1, then the hart will wake when an interrupt is enabled and becomes pending, and immediately enter the interrupt handler. Upon exit from the interrupt handler, program execution will resume at the instruction following the WFI.

If interrupts are enabled but mstatus.MIE=0, then the hart will wake when an interrupt is enabled and becomes pending but will not enter the interrupt handler. It will simply resume at the instruction immediately after the WFI in this case.

To prevent an interrupt source from waking a hart, the enable bit for that interrupt must be writ- ten to 0 prior to executing the WFI instruction. If any interrupts are pending upon executing a WFI instruction, then the WFI is effectively treated as a NOP instruction.

- **CEASE Instruction for Power Down**

To fully power down, follow the steps described in Power-Down Flow, where the last step is to execute a CEASE instruction. Once the CEASE instruction is executed, the core will not retire another instruction until reset. The CEASE opcode is 0x30500073 and can be implemented in either assembly or C. To create an assembly-level function using GCC, consider the following example.

```
.global _cease
.type         _cease, @function
_cease:
    .word 0x30500073
    ret
```

The next example demonstrates how to implement the CEASE instruction within a function in C.

```
static inline void cease()
{
    __asm__ __volatile__ (".word 0x30500073" : : : "memory"); // CEASE
}
```

- **Hardware Reset**

The following list summarizes the hardware reset values required by The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.11 and applies to all EIC7700X MCPU designs.

1. Privilege mode is set to machine mode.

2. mstatus.MIE and mstatus.MPRV are required to be 0.

3. The misa register holds the full set of supported extensions for that implementation, and misa.MXL defaults to the widest supported ISA available, referred to as MXLEN.

4. The pc is set to the implementation specific reset vector.

5. The mcause register is set to 0x0 at reset.

6. The PMP configuration fields for address matching mode (A) and Lock (L) are set to 0, which defaults to no protection for any privilege level.

The internal state of the rest of the system should be completed by software early in the boot flow.

- **Early Boot Flow**

For the early stages of boot, some of the first things software must consider are listed below:

- The global pointer (gp or x3) user register should be initialized to the global_pointer$

linker generated symbol and not changed at any point in the application program.

- The stack pointer (sp or x2) user register should be also set up as a standard part of the boot flow.

- All other user registers (x1, x4 - x31) can be written to 0 upon initial power-on.

- The mtvec register holds the default exception handler base address, so it is important to set up this register early in the boot flow, so it points to a properly aligned, valid exception handler location.

- Zero out the bss section and copy data sections into RAM areas as needed.

- **Interrupt State During Early Boot**

Since mstatus.MIE defaults to 0, all interrupts are disabled globally out of reset. Prior to enabling interrupts globally through mstatus.MIE, consider the following:

- Ensure no timer interrupts are pending by checking the mip.MTIP bit. The mtime register is 0 out of

reset and starts running immediately. However, the mtimecmp register does not have a reset value.

If no timer interrupt is required, leave mie.MTIE equal to 0 prior to enabling global interrupt with mstatus.MIE.

If the application requires a timer interrupt, write mtimecmp to a value in the future for the next timer interrupt before enabling mstatus.MIE.

• Write the remaining bits in the mie CSR to the desired value to enable interrupts based on the requirements of the system. This register is not defined to have a reset value.

• Each msip register in the Core-Local Interruptor (CLINT) or Core-Local Interrupt Controller (CLIC) address space is reset to 0, so no specific initialization is required for local software interrupts.

Since msip is memory-mapped, any hart in the system may trigger a software interrupt on another hart, so this should be considered during the boot flow on a multi-hart system.

• If a Platform-Level Interrupt Controller (PLIC) exists, check the PLIC pending status. The PLIC memory mapped pending bits are read-only, so the pending status should be cleared at the source if they reset to a non-zero status. Then, enable the PLIC interrupts as required by the system prior to enabling interrupts in the system via mstatus.MIE.

If an L3 Cache or Bus-Error Unit (BEU) is present, these interrupt IDs begin at 128, so the enable bits may lie in a different region of the memory map than other PLIC enable bits in the design.

• Wipe down memory if enabled with ECC. This can be done by writing 0x0 to memory with either store instructions issued by the CPU or using a DMA controller. ECC errors are reported via the Bus-Error Unit (BEU).

• Check BEU registers to ensure no errors are reported and that the enable bits reflect the requirements of the application.

- **Other Boot Time Considerations**

• Write 0 to enable the appropriate bits in the Feature Disable CSR.

• Ensure the remaining bits in the mstatus CSR are written to the desired application spe- cific configuration at boot time.

• If a design includes user and supervisor privilege levels, initialize medeleg and mideleg registers to 0 until supervisor-level trap handling is set up correctly using stvec.

• The mcause, mepc, and mtval registers hold important information in the event of a synchronous exception. If the synchronous exception handler forces reset in the application, the contents of these registers can be checked to understand root cause.

• The PMP address and configuration CSRs are required to be initialized if user or supervisor privilege levels are part of the design. By default, user and supervisor modes have no permissions to the memory map unless explicitly granted by the PMP.

• The mcycle CSR is a 64-bit counter on both RV32 and RV64 systems, and it counts the number of cycles executed by the hart. It has an arbitrary value after reset and can be writ- ten as needed by the application.

• Instructions retired can be counted by the minstret register, and this also has an arbitrary value after reset. This can be written to any given value.

• The mhpmeventX CSR selects which hardware events to count, where the count is reflected in mhpmcounterX. At any point, the mhpmcounterX registers can be directly written to reset their value when the mhpmeventX register has the proper event selected.

• For cores with an MMU, ensure the satp register holds the correct configuration for address translation.

- There is no requirement for boot time initialization to any of the registers within the Debug Module, unless there is an application specific reason to do so.
- All other CSRs during boot time initialization should be considered based on system and application requirements.

- **Power-Down Flow**

Designate one core as "primary" and all others as "secondary". For EIC7700X MCPU, coordination with an "External Agent" is required.

1. External Agent: Wait for communication from the primary core to initiate the following steps:
   a. Stop sending inbound traffic (both transactions and interrupts) into the Core Complex.
   b. Wait until all outstanding requests to the Core Complex are completed, then
   c. Wait until cease_from_tile_N is high for the primary core and all secondary cores.
   d. Once cease_from_tile_N is high for the primary core and all secondary cores, apply reset to the entire Core Complex.
2. Primary core
   a. The following sequence should be executed in machine mode and NOT out of a remote ITIM/DTIM.
   b. Communicate with external agent to initiate cease power-down sequence.
   c. Poll external agent until steps 1.a and 1.b are completed.
   d. Disable all interrupts except those related to bus errors/memory corruption, and IPIs (if using enabled IPI to coordinate power-down sequence among cores).
      i. Copy contents of any TIMs/LIMs into external memory.
      ii. If there is an L3 cache, flush it.
      iii. If there is no L3 cache, but there is a pL2 cache, flush it.
   e. Inform all secondary cores to proceed.
   f. Wait until cease_from_tile_N is high for all secondary cores. Examples of how this can be accomplished:
      i. Have an off-core-complex memory-mapped register that tracks the state of the

      cease_from_tile_N signals. Primary core polls this register.

      ii. Wire the cease_from_tile_N signals back into interrupt wires. Corresponding interrupts can be disabled along with all others in the first step. Primary core polls the interrupt-pending bits for those interrupts.
   g. Disable all interrupts.
   h. Execute CEASE instruction.
3. Secondary cores:
   a. The following sequence should be executed in machine mode and NOT out of a remote ITIM/DTIM.
   b. Execute in an idle loop for notification sent in step 2.e (could be via polling on an MMIO-accessible mailbox, polling on an IPI (disabled), or waiting on an IPI (enabled)).
   c. Disable all interrupts except those related to bus errors/memory corruption, and IPIs (if using

enabled IPI to coordinate power-down sequence among cores).

d. Copy contents of any TIMs/LIMs into external memory.

e. If there is no L3 cache but there is a data cache, flush it using full-cache variant of

CFLUSH.D.L1 if available, or per-line variant if not.

f. Disable all interrupts.

g. Execute CEASE instruction.

3.4.1.18 Debug

This chapter describes the operation of EIC7700X MCPU debug hardware, which follows The RISC-V Debug Specification, Version 1.0. Currently, only interactive debug and hardware breakpoints are supported.

Debug Module

The Debug Module (DM) handles nearly all of the functions related to debugging. It is a slave to both the Debug Module Interface (DMI) coming from the probe, and a TileLink bus coming from the cores. From the perspective of the core, the DM appears as a 4KiB block in the memory map. The DM memory map as seen from the perspective of the core is shown in Table *Debug Module Memory Map from the Perspective of the Core*, and the register map from the perspective of the DMI is shown in Table below.

Most of the DM is clocked by debug_clock. The dmcontrol register is accessible when debug_clock is not running, mainly to be able to write to haltreq while the core is in reset due to ndreset. Doing so generates a debug interrupt and will interrupt the selected core immediately once it is out of reset or during a WFI instruction.

Table 3-138 Debug Module Memory Map Seen from the Debug Module Interface

DMI Address	Name	Description
0x04-0x0F	data0-data11	Read/Write DATA registers. 32-bit EIC7700X MCPU cores have 1 data register, 64-bit cores have 2. Note that these registers are volatile and may be masked in IP-XACT.
0x10	dmcontrol	Debug Module Control. See *Table Debug Module Status Register* for moren information.
0x11	dmstatus	Debug Module Status. See Table *Debug Module Status Register* for more information.
0x12	hartinfo	Hart Information. Unsupported, default value is 0.
0x13	haltsum1	Read-only. Halt Summary 1. Only present on systems with >32 harts. **Not used by EIC7700X MCPU**.
0x14	hawindowsel	Read/Write. Select which window of up to 32 harts is visible in hawindow. **Not used by EIC7700X MCPU**.
0x15	hawindow	Read/Write. Window of 32 harts to be selected, in addition to the one selected by hartsel. Bit 0 corresponds to hart 0. A 1 will select the corresponding hart.
0x16	abstractcs	Abstract Control and Status. See Table *Debug Module Control Register* for more information.

DMI Address	Name	Description
0x17	command	Initiate abstract command. See Table *Abstract Control and Status Register* for more information.
0x18	abstractauto	Selects whether access to particular DATA or PROGBUF locations will re-execute the last command. Used for block transfers or other repeating commands. See Table *Abstract Command Register* for more information.
0x20-0x2F	progbuf0-progbuf15	Read/Write PROGBUF registers.
0x32	dmcs2	Fields to set up and read back Halt Group or Resume Group configuration. Present by default on systems with more than 1 hart or with any external triggers. See Table *Abstract Command Autoexec Register* for more information.
0x38	sbcs	System Bus Access Control and Status
0x39	sbaddress0	System Bus Address 31:0
0x3A	sbaddress1	System Bus Address 63:32.
0x3C	sbdata0	System Bus Data 31:0
0x3D	sbdata1	System Bus Data 63:32
0x40	haltsum0	Read-only. Halt Summary 0. Bit n reads 1 if hart n is halted. Note that this register is volatile and may be masked in IP-XACT.

From the point of view of the core, the DM appears as a 4KiB block of memory. It is mapped into low memory so that memory references can use addresses relative to the $zero register.

Table 3-139 Debug Module Memory Map from the Perspective of the Core

TL Address	Name	Attr.	Description
0x100	HALTED	WO	Written with hartid by ROM code when hart gets a debug interrupt or reenters ROM due to EBREAK. Sets halted[hartid]. If an abstract command was running, writing this also clears busy.
0x104	GOING	WO	Written by ROM code when it begins executing a command started by FLAGS[hartid].go. Clears FLAGS[hartid].go.
0x108	RESUMING	WO	Written with hartid by hart when it is about to resume. Sets resumeack[hartid] and clears halted[hartid] and FLAGS[hartid].resume.
0x10C	EXCEPTION	WO	Written by hart when it encounters an exception in debug mode. Sets cmderr to "exception".

TL Address	Name	Attr.	Description
0x300	WHERETO	RO	JAL to ABSTRACT. This opcode is constructed by DM hardware and is needed because ABSTRACT is not a fixed address (depends on number of PROGBUF words selected in the configuration). Note that ABSTRACT is volatile and may be masked in IP-XACT.
contiguous	ABSTRACT	RO	2 words constructed by DM hardware based on abstract command written from DTM. • 0 - If transfer set, construct instruction to load/ store specific register to/from DATA[0] (32 bits) or DATA[1:0] (64 bits), else NOP. • 4 - If postexec set, then NOP to fall thru and execute PROGBUF, else EBREAK to return to ROM park loop. Note that this register is volatile and may be masked in IP-XACT.
contiguous	PROGBUF	RW	Configurable number (typically 16, max 16) of R/W words to be filled in by debugger and executed by hart. Note that this register is volatile and may be masked in IP-XACT.
0x380-0x3BF	DATA	RW	Configurable number (1 for 32-bit or 2 for 64-bit, max 12) of R/W words intended for use for data transfer between debugger and hart. Since it is contiguous with PROGBUF, the debugger may use DATA as an extension of PROGBUF. Note that these registers are volatile and may be masked in IP-XACT.
0x400-0x7FF	FLAGS	RO	One byte flag per hart. • Bit 0 (go): Set by writing an abstract command, cleared by ROM write to GOING. ROM will jump to WHERETO.

Table 3-140 Debug Module Memory Map from the Perspective of the Core

TL Address	Name	Attr.	Description
			• Bit 1 (resume): Set by writing 1 to resumereq[hartid]. Cleared by ROM write of hartid to RESUMING. ROM restores s0 then executes dret. Note that these registers are volatile and may be masked in IP-XACT.

TL Address	Name	Attr.	Description
0x800-0xFFF	ROM	RO	Debug interrupt or EBREAK enters at 0x800, saves s0, writes hartid to HALTED, then busy-waits for FLAGS[hartid] > 0. If FLAGS[hartid].go, write 0 to GOING, then jump to WHERETO. Else write hartid to RESUMING, then execute dret to return to user program. ROM Source Code: https://github.com/chipsalliance/rocket-chip/blob/master/scripts/debug_rom/debug_rom.S

- **Debug and Trigger Registers**

This section describes the per hart debug and trigger registers, which are mapped into the CSR space as follows:

Table 3-141 Debug and Trigger Registers

CSR	Name	Allowed Access Mode	Description
0x7B0	dcsr	Debug	Debug Control and Status Register
0x7B1	dpc	Debug	Debug PC. Stores execution address just before debug exception and to return to at dret.
0x7B2	dscratch0	Debug	Debug Scratch Register 0
0x7A0	tselect	Debug, Machine	Trigger Select. Most configs implement 2, 4, or 8 triggers. Triggers are all type 6 (address/data).
0x7A1	tdata1	Debug, Machine	Trigger Data 1, mcontrol6
0x7A2	tdata2	Debug, Machine	Trigger Data 2, the address for comparison
0x7A3	tdata3	Debug, Machine	Trigger Data 3

- **Debug Control and Status Register (dcsr)**

This register gives information about debug capabilities and status. Its detailed functionality is described in The RISC-V Debug Specification, Version 1.0.

Table 3-142 Debug Control and Status Register

Debug Control and Status Register (dcsr)			
CSR	0x7B0		
Bits	Field Name	Attr.	Description
[1:0]	prv	RW	Privilege level of processor prior to debug exception and to return to at dret.
2	step	RW	Set to 0x1 to single-step.
3	nmip	RO	Non-maskable interrupt pending. Not used by EIC7700X MCPU.
4	mprven	WARL	Not used by EIC7700X MCPU.
5	v	WARL	Extends the prv field with the virtualization mode prior to debug exception and to return to at dret.
[8:6]	cause	RO	Indicates cause of most recent debug exception.
9	stoptime	WARL	0x1 will stop timers in debug mode. Not used by EIC7700X MCPU (timers continue).
10	stopcount	WARL	0x1 will stop counters in debug mode. Not used by EIC7700X MCPU (counters continue).
11	stepie	WARL	Enable interrupts when stepping. Not used by EIC7700X MCPU (interrupts disabled).
12	ebreaku	RW	EBREAK instructions in U-mode enter debug mode (vs. breakpoint exception).
13	ebreaks	RW	EBREAK instructions in HS-mode enter debug mode.
14	Reserved		
15	ebreakm	RW	EBREAK instructions in M-mode enter debug mode.
16	ebreakvu	RW	EBREAK instructions in VU-mode enter debug mode.
17	ebreakvs	RW	EBREAK instructions in VS-mode enter debug mode.
[27:18]	Reserved		[31:28]

- **20.2.2 Debug PC (dpc)**

When entering debug mode, the current PC is copied here. When leaving debug mode, execution resumes at this PC.

- **20.2.3 Debug Scratch (dscratch)**

This register is generally reserved for use by Debug ROM in order to save registers needed by the code in Debug ROM. The debugger may use it as described in The RISC-V Debug Specifi- cation, Version 1.0.

- **20.2.4 Trigger Select Register (tselect)**

To support a large and variable number of triggers for tracing and breakpoints, they are accessed through one level of indirection where the tselect register selects which bank of three tdata1-3 registers are accessed via the other three addresses.

The tselect register has the format shown below:

Table 3-143 Trigger Select Register

Trigger Select Register (tselect)			
CSR	**0x7A0**		
Bits	**Field Name**	**Attr.**	**Description**
[63:0]	index	WARL	Selection index of triggers

The index field is a WARL field that does not hold indices of unimplemented triggers. Even if index can hold a trigger index, it does not guarantee the trigger exists. The type field of tdata1 must be inspected to determine whether the trigger exists.

- **Trigger Data Registers (tdata1-3)**

The tdata1-3 registers are 64-bit read/write registers selected from a larger underlying bank of triggers by the tselect register.

Table 3-144 Trigger Data Register 1

Trigger Data Register 1 (tdata1)			
CSR	**0x7A1**		
Bits	**Field Name**	**Attr.**	**Description**
[58:0]	Trigger-Specific Data		
59	dmode	WARL	Selects between debug mode (dmode=1) and machine mode (dmode=0) views of the registers, where only debug mode code can access the debug mode view of the triggers
[63:60]	type	WARL	The type of trigger selected by tselect • 0x0 - No such trigger • 0x1-0x5 - Reserved • 0x6 - Address/Data Match Trigger • ≥0x7 - Reserved

Table 3-145 Trigger Data Registers 2 and 3

Trigger Data Registers 2 and 3 (tdata2/3)			
CSR	0x7A2 - 0x7A3		
Bits	Field Name	Attr.	Description
[63:0]	Trigger-Specific Data		

Any attempt to read/write the tdata1-3 registers in machine mode when TSELECT.dmode=1 raises an illegal-instruction exception.

- **Breakpoints**

The EIC7700X MCPU supports four hardware breakpoint registers per hart, which can be flexibly shared between debug mode and machine mode.

When a breakpoint register is selected with tselect, the other CSRs access the following information for the selected breakpoint:

Table 3-146 Trigger CSRs When Used as Breakpoints

CSR Name	Breakpoint Alias	Description
tselect	tselect	Breakpoint selection index
tdata1	mcontrol6	Breakpoint match control
tdata2	maddress	Breakpoint match address
tdata3	N/A	Reserved

- **Breakpoint Match Control Register (mcontrol6)**

Each breakpoint control register is a read/write register laid out in Table below. This register is accessible as tdata1 when type is 0x6.

Table 3-147 Breakpoint Match Control Register

Breakpoint Match Control Register (mcontrol6)				
CSR	0x7A1			
Bits	Field Name	Attr.	Rst.	Description
0	R	WARL	0x0	Address match on load
1	W	WARL	0x0	Address match on store
2	X	WARL	0x0	Address match on instruction fetch
3	U	WARL	X	Address match on user mode
4	S	WARL	X	Address match on supervisor mode
5	Reserved	WPRI	X	

Breakpoint Match Control Register (mcontrol6)				
CSR	0x7A1			
Bits	Field Name	Attr.	Rst.	Description
6	M	WARL	X	Address match on machine mode
[10:7]	match	WARL	X	Breakpoint Address Match type • 0x0 - Single address • 0x1 - Power-of-2 range, limited to 64 bytes in EIC7700X MCPU implementations • 0x2 - ≥ address • 0x3 - < address • Others not supported by EIC7700X MCPU
11	chain	WARL	0x0	Chain adjacent conditions. When set, this trigger and the next must match at the same time to fire. Typically used for a range breakpoint using 2 triggers, one with match=0x2 and one with match=0x3. This is not a sequential trigger.
[15:12]	action	WARL	0x0	Breakpoint action to take
[19:16]	size	WARL	0x0	Size of the breakpoint. Fixed at 0, meaning accesses of any size that cover any part of the trigger address range will fire.
20	timing	WARL	0x0	Timing of the breakpoint. Fixed at 0, meaning breaks happen just before the event.
21	select	WARL	0x0	Perform match on address or data. Fixed at 0, meaning all triggers compare addresses only (no data value).
22	Reserved			
23	VU	WARL	X	Address match on virtual user mode
24	VS	WARL	X	Address match on virtual supervisor mode
[58:25]	Reserved			
59	dmode	RW	0x0	Debug-only access mode

Breakpoint Match Control Register (mcontrol6)				
CSR	0x7A1			
Bits	Field Name	Attr.	Rst.	Description
[63:60]	type	RO	0x6	Address/Data match type, always 0x6

The type field is a 4-bit read-only field holding the value 0x6 to indicate this is a breakpoint con- taining address match logic.

The action field is a 4-bit read-write WARL field that specifies the available actions when the address match is successful. The value 0 generates a breakpoint exception. The value 1 enters debug mode. The value 2 starts tracing. The value 3 stops tracing. The value 4 inserts a pro- gram trace sync message. The value 5 generates a trace external trigger out. Other actions are not implemented.

The R/W/X bits are individual WARL fields, and if set, indicate an address match should only be successful for loads, stores, and instruction fetches, respectively. All combinations of implemented bits must be supported.

The M/S/U/VS/VU bits are individual WARL fields, and if set, indicate that an address match should only be successful in the machine, hypervisor, user, virtual supervisor, and virtual user modes, respectively. All combinations of implemented bits must be supported.

The match field is a 4-bit read-write WARL field that encodes the type of address range for breakpoint address matching. Three different match settings are currently supported: exact, NAPOT, and arbitrary range. A single breakpoint register supports both exact address matches and matches with address ranges that are naturally aligned powers-of-two (NAPOT) in size.

Breakpoint registers can be paired to specify arbitrary exact ranges, with the lower-numbered breakpoint register giving the byte address at the bottom of the range and the higher-numbered breakpoint register giving the address 1 byte above the breakpoint range and using the chain bit to indicate both must match for the action to be taken.

NAPOT ranges make use of low-order bits of the associated breakpoint address register to encode the size of the range as follows:

Table 3-148 NAPOT Size Encoding

maddress	Match Type and Size
a…aaaaaa	Exactly 1 byte
a…aaaaa0	2-byte NAPOT range
a…aaaa01	4-byte NAPOT range
a…aaa011	8-byte NAPOT range
a…aa0111	16-byte NAPOT range
a…a01111	32-byte NAPOT range
…	
a01…1111	231-byte NAPOT range

maskmax6 determines the largest supported NAPOT range. The value of maskmax6 is the logarithm base 2 of the number of bytes in the largest supported NAPOT range. A value of 0 indi- cates that only exact address matches are supported (1-byte range). A value of 31 corresponds to the maximum NAPOT range, which is 231 bytes in size. The largest range is encoded in maddress with the 30 least-significant bits set to 1, bit 30 set to 0, and bit 31 holding the only address bit considered in the address comparison.

The value of maskmax6 is not directly observable, but can be determined via the following sequence:

1. Set match to 1 to select NAPOT mode

2. Read match. If the returned value is not 1, then NAPOT matching is not supported.

3. Write all ones to tdata2

4. Read tdata2. The value of maskmax6 is one more than the index of the most-significant zero bit. For example, if the read value is 0xFFFF_FFF7, bit 3 is zero, so maskmax6 is 4.

To provide breakpoints on an exact range, two neighboring breakpoints can be combined with the chain bit. The first breakpoint can be set to match on an address using match of 2 (greater than or equal). The second breakpoint can be set to match on address using match of 3 (less than). Setting the chain bit on the first breakpoint prevents the second breakpoint from firing unless they both match.

Note

When a chain includes both instruction trigger and data address trigger, the breakpoint does not fire. To work around this limitation, set a data trigger on any access to the data item. Then, in the GDB breakpoint command script, check whether the PC is the one you want and restart if not.

- **Breakpoint Match Address Register (maddress)**

Each breakpoint match address register is a 64-bit read/write register used to hold significant address bits for address matching and the unary-encoded address masking information for NAPOT ranges.

- **Breakpoint Execution**

Breakpoint traps are taken precisely. Implementations that emulate misaligned accesses in soft- ware will generate a breakpoint trap when either half of the emulated access falls within the address range. Implementations that support misaligned accesses in hardware must trap if any byte of an access falls within the matching range.

Debug mode breakpoint traps jump to the debug trap vector without altering machine mode registers.

Machine mode breakpoint traps jump to the exception vector with "Breakpoint" set in the mcause register and with badaddr holding the instruction or data address that caused the trap.

- **Sharing Breakpoints Between Debug and Machine Mode**

When debug mode uses a breakpoint register, it is no longer visible to machine mode (that is, the tdrtype will be 0). Typically, a debugger will leave the breakpoints alone until it needs them, either because a user explicitly requested one or because the user is debugging code in ROM.

- **Debug Memory Map**

This section describes the Debug Module's memory map when accessed via the regular system interconnect. The Debug Module is only accessible to debug code running in debug mode on a hart (or via a Debug Transport Module). The following addresses are offsets from the base address of the Debug Module. Note that the PMP must allow M-mode access to the Debug Module address range for debugging to be possible.

- **Debug RAM and Program Buffer (0x300–0x3FF)**

The EIC7700X MCPU has 16 32-bit words of program buffer for the debugger to direct a hart to execute arbitrary RISC-V code. Its location in memory can be determined by executing aiupc instructions and storing the result into the program buffer.

The EIC7700X MCPU has two 32-bit words of debug data RAM. This RAM space is used to pass data for the Access Register abstract command, as described in The RISC-V Debug Specification, Version 1.0. The EIC7700X MCPU supports only general-purpose register access when harts are halted. All other commands must be implemented by executing from the debug program buffer.

In the EIC7700X MCPU, both the program buffer and debug data RAM are general purpose RAM and are mapped contiguously in the Core Complex memory space. Therefore, additional data can be passed in the program buffer, and additional instructions can be stored in the debug data RAM.

Debuggers must not execute program buffer programs that access any Debug Module memory except defined program buffer and debug data addresses.

- **Debug ROM (0x800–0xFFF)**

This ROM region holds the debug routines on EIC7700X MCPU systems.

- **Debug Flags (0x100–0x110, 0x400–0x7FF)**

The flag registers in the Debug Module are used for the Debug Module to communicate with each hart. These flags are set and read used by the debug ROM and should not be accessed by any program buffer code. The specific behavior of the flags is not further documented here.

- **Safe Address**

In the EIC7700X MCPU, the Debug Module contains the Debug Module address range in the memory map. Memory accesses to these addresses raise access exceptions, unless the hart is in debug mode. This property allows a "safe" location for unprogrammed parts, as the default mtvec location is 0x0.

- **Instruction Trace Interface**

This section describes the interface between a EIC7700X MCPU core and its trace unit. The trace interface conveys information about instruction-retirement and exception events.

Table *Instruction Trace Interface Fields* lists the fields in a trace packet. The valid signal is 1 if and only if an instruction retires or traps (either by generating a synchronous exception or taking an interrupt). The remaining fields in the packet are only defined when valid is 1.

The iaddr field holds the address of the instruction that retired or trapped. If address translation is enabled, it is a virtual address, else it is a physical address. Virtual addresses narrower than 64 bits are sign-extended, and physical addresses narrower than 64 bits are zero-extended.

The insn field holds the instruction that retired or trapped. For instructions narrower than the maximum width, e.g., those in the RISC-V C extension, the unused high-order bits are zero- filled. The length of the instruction can be determined by examining the low-order bits of the instruction, as described in The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1. The width of the insn field, ILEN, is 32 bits for current implementations.

The priv field indicates the privilege mode at the time of instruction execution. On an exception, the next valid trace packet's priv field gives the privilege mode of the activated trap handler.

The width of the priv field, PRIVLEN, is 3, and it is encoded as shown in Table *Instruction Trace Interface Fields*.

Table 3-149 Encoding of priv

Value*	Description
0x0	User mode
0x1	Supervisor mode
0x3	Machine mode
0x4-0x7	Debug mode
*Unspecified values are reserved.	

The exception field is 0 if this packet corresponds to a retired instruction, or 1 if it corresponds to an exception or interrupt. In the former case, the cause and interrupt fields are undefined and the tval field is zero. In the latter case, the fields are set as follows:

- interrupt is 0 for synchronous exceptions and 1 for interrupts.
- cause supplies the exception or interrupt cause, as would be written to the lower CAUSE- LEN bits of the mcause CSR. For current implementations, CAUSELEN is log2(64).
- tval supplies the associated trap value, e.g., the faulting virtual address for address exceptions, as would be written to the mtval CSR.

Future optional extensions may define tval to provide ancillary information in cases where it currently supplies zero.

For cores that can retire N instructions per clock cycle, this interface is replicated N times. Lower-numbered entries correspond to older instructions. If fewer than N instructions retire, the valid packets need not be consecutive, i.e., there may be invalid packets between two valid packets. If one of the instructions is an exception, no younger instruction will be valid.

Table 3-150 Instruction Trace Interface Fields

Name	Description
valid	Indicates an instruction has retired or trapped.
iaddr[63:0]	The address of the instruction.
insn[31:0]	The instruction.
priv[2:0]	Privilege mode during execution.
exception	0, if the instruction retired; 1, if it trapped.
interrupt	0, if the exception was synchronous; 1, if interrupt.
cause[5:0]	Exception cause.
tval[63:0]	Exception data.

- **Debug Module Status Register (dmstatus)**

dmstatus holds the DM version number and other implementation information. Most importantly, it

contains status bits that indicate the current state of the selected hart(s).

Table 3-151 Debug Module Status Register

Debug Module Status Register (dmstatus)				
DMI Address		0x11		
Bits	Field Name	Attr.	Rst.	Description
[3:0]	version	RO	0x3	Implementation version number
4	Reserved	RO	0x0	
5	hasresethaltreq	RO	0x1	1 if resethaltreq exists
6	authbusy	RO	0x0	0=the Authenication Module is ready, 1=busy. Note that this field is volatile and may be masked in IP-XACT.
7	authenticated	RO	0x1	0=Debug Module locked, 1=the authentication check has passed. Note that this field is volatile and may be masked in IP-XACT.
8	anyhalted	RO	0x0	Any currently selected hart is halted. Note that this field is volatile and may be masked in IP-XACT.
9	allhalted	RO	0x0	All currently selected harts are halted. Note that this field is volatile and may be masked in IP-XACT.
10	anyrunning	RO	0x1	Any currently selected hart is running. Note that this field is volatile and may be masked in IP-XACT.
11	allrunning	RO	0x1	All currently selected harts are running. Note that this field is volatile and may be masked in IP-XACT.
12	anyunavail	RO	0x0	Any currently selected hart is not available (i.e., is powered down). DM supports it, but not currently used by EIC7700X MCPU cores Note that this field is volatile and may be masked in IP-XACT.
13	allunavail	RO	0x0	All currently selected harts are not available (i.e., is powered down). DM supports it, but not currently used by EIC7700X MCPU cores. Note that this field is volatile and may be masked in IP-XACT.
14	anynonexistent	RO	0x0	Any currently selected hart does not exist in the system. Note that this field is volatile and may be masked in IP-XACT.

Debug Module Status Register (dmstatus)				
DMI Address		0x11		
Bits	Field Name	Attr.	Rst.	Description
15	allnonexistent	RO	0x0	All currently selected harts do not exist in the system. Note that this field is volatile and may be masked in IP-XACT.
16	anyresumeack	RO	0x1	Any currently selected hart has resumed execution. Note that this field is volatile and may be masked in IP-XACT.
17	allresumeack	RO	0x1	All currently selected harts have resumed execution. Note that this field is volatile and may be masked in IP-XACT.
18	anyhavereset	RO	0x0	Any currently selected hart has been reset, but reset has not been acknowledged. Note that this field is volatile and may be masked in IP-XACT.
19	allhavereset	RO	0x0	All currently selected harts have been reset, but reset has not been acknowledged. Note that this field is volatile and may be masked in IP-XACT.
[21:20]	Reserved	RO	0x0	
22	impebreak	RO	0x0	1 if PROGBUF is followed by implicit EBREAK. Generally, 1 for E2 cores, 0 otherwise.
[31:23]	Reserved	RO	0x0	

- **Debug Module Control Register (dmcontrol)**

A debugger performs most hart controls through the dmcontrol register.

Table 3-152 Debug Module Control Register

Debug Module Control Register (dmcontrol)				
DMI Address		0x10		
Bits	Field Name	Attr.	Rst.	Description
0	dmactive	RW	0x0	0 disables the DM and sets DMI registers to their reset state, 1 puts the DM in operational mode. Drives dmactive output that could be used by a system power controller to maintain power to the DM while it is being used. When 1, dmcontrol should be read back until dmactive=1, which indicates that the Debug Module is fully operational. When 0, the DM TileLink clock is gated off to save power.
1	ndmreset	RW	0x0	Write 1 to reset system (assert ndreset output). Write 0 to operate normally.
2	clrresethaltreq	WO	0x0	Write 1 to clear the reset-halt-request bit
3	setresethaltreq	WO	0x0	When written to 1, the core will halt upon the next deassertion of its reset
[15:4]	Reserved	RO	0x0	
[25:16]	hartsel	RW	0x0	Selects the hart to operate on
26	hasel	RW	0x0	Selects hart(s) in the hart-array mask register (hawindow)
27	Reserved	RO	0x0	
28	ackhavereset	WO	0x0	Write 1 to acknowledge that a reset occurred on the selected hart
29	Reserved	RO	0x0	
30	resumereq	WO	0x0	Write 1 to request selected hart to resume, cleared to 0 automatically when hart resumes
31	haltreq	RW	0x0	Write 1 to request selected hart to halt. Generates debug interrupt to the core. Write 0 once halted has been set by the DM.

- **Hart Array Window Register (hawindow)**

This register contains a bitmap where bit 0 corresponds to hart 0, bit 1 to hart 1, etc. Any bits set in this register select the corresponding hart in addition to the hart selected by dmcontrol.hartsel.

- **Abstract Control and Status Register (abstractcs)**

Table 3-153 Abstract Control and Status Register

Abstract Control and Status Register (abstractcs)				
DMI Address		0x16		
Bits	Field Name	Attr.	Rst.	Description
[3:0]	datacount	RO	0x2	Number of DATA registers. 0x1 for 32-bit, 0x2 for 64-bit EIC7700X MCPU cores.
[7:4]	Reserved	RO	0x0	
[10:8]	cmderr	RW1C	0x0	Non-zero value indicates an abstract command error. Remains set until cleared by writing all ones. If set, no abstract commands are accepted. • 0x0 - No error • 0x1 - Busy. Abstract command or register was accessed while command was running. • 0x2 - Not supported. Abstract command type not supported by hardware was attempted. • 0x3 - Exception. An exception occurred during execution of an abstract command. • 0x4 - Halt/resume. Abstract command attempted while hart was running or unavailable. • 0x5 - Bus. Bus error occurred during abstract command. Not used by EIC7700X MCPU. • 0x7 - Other. Abstract command failed for another reason. Not used by EIC7700X MCPU. Note that this field is volatile and may be masked in IP-XACT.
11	Reserved	RO	0x0	
12	busy	RO	0x0	Reads as 1 while Abstract command is running, 0 if not. Note that this field is volatile and may be masked in IP-XACT.
[23:13]	Reserved	RO	0x0	
[28:24]	progbufsize	RO	0x10	Number of 32-bit words in PROGBUF. EIC7700X MCPU has 16 words.
[31:29]	Reserved	RO	0x0	

- **Abstract Command Register (command)**

Table 3-154 Abstract Command Register

Abstract Command Register (command)			
DMI Address		0x17	
Bits	Field Name	Attr.	Description
[15:0]	regno	RW	Select which register to read/write. EIC7700X MCPU only supports GPRs: 0x1000-0x101F.
16	write	RW	1=write register, 0=read register. Only done if transfer=1.
17	transfer	RW	1=do the register read/write, 0=don't.
18	postexec	RW	1=execute PROGBUF after the command, 0=don't.
19	aarpostincrement	RW	Not supported by EIC7700X MCPU.
[22:20]	aarsize	RW	0x2, 0x3, 0x4 select 32, 64, 128 bits, respectively.
23	Reserved	RO	
[31:24]	cmdtype	RW	0=Access Register is the only type supported by EIC7700X MCPU.

Note that this register is volatile and may be masked in IP-XACT.

- **Abstract Command Autoexec Register (abstractauto)**

Table 3-155 Abstract Command Autoexec Register

Abstract Command Autoexec Register (abstractauto)				
DMI Address		0x18		
Bits	Field Name	Attr.	Rst.	Description
[11:0]	autoexecdata	RW	0x0	Bitmap of DATA registers [11:0]. 1 indicates DATA access initiates command.
[15:12]	Reserved	RO	0x0	
[31:16]	autoexecprogbuf	RW	0x0	Bitmap of PROGBUF words [15:0]. 1 indicates PROGBUF access initiates command.

- **Debug Module Control and Status 2 Register (dmcs2)**

Table below describes the Debug Module Control and Status 2 Register dmcs2. If halt/resume groups are not implemented, then group will always read back as 0. The Debug Module exter- nal triggers may be allocated as needed between halt and resume groups.

Table 3-156 Debug Module Control and Status 2 Register

Debug Module Control and Status 2 Register (dmcs2)				
DMI Address		0x32		
Bits	Field Name	Attr.	Rst.	Description
0	hgselect	RW	0x0	0=operate on harts, 1=operate on external triggers.
1	hgwrite	WO	X	When written with 1, the selected harts or external trigger is assigned to group group.
[6:2]	group	RW	0x0	Specify the halt group or resume group number that the selected harts or external triggers will be assigned to. Note that this field is volatile and may be masked in IP-XACT.
[10:7]	exttrigger	RW	0x0	Select which external trigger to act upon if hgwrite and hgselect are written to 1 in the same write. Note that this field is volatile and may be masked in IP-XACT.
11	grouptype	RW	0x0	0=operate on Halt Group configuration, 1=operate on Resume Group configuration.
[31:12]	Reserved	RO	0x0	

- **Abstract Commands**

Abstract commands provide a debugger with a path to read and write processor state and are used for extracting and modifying processor state such as registers and memory. Register s0 is saved by the ROM and is available for use by the abstract command code. An abstract com- mand is started by the debugger writing to command. In command, the debugger selects whether to load/store a register, execute PROGBUF, or both. Only GPR register transfers are supported currently. Many aspects of Abstract Commands are optional in The RISC-V Debug Specifica- tion, Version 1.0 and are implemented as described below.

Table 3-157 Debug Abstract Commands

cmdtype	Feature	Support
Access Register	GPR registers	Access Register command, register number 0x1000 - 0x101F

cmdtype	Feature	Support
	CSR registers	Not supported. CSRs are accessed using the Program Buffer.
	FPU registers	Not supported. FPU registers are accessed using the Program Buffer.
	Autoexec	Both autoexecprogbuf and autoexecdata are supported.
	Post-increment	Not supported.
	Core Register Access	Not supported.
Quick Access		Not supported.
Access Memory		Not supported. Memory access is accomplished using the Program Buffer.

The use of abstract commands is outlined in the following example, describing how to read a word of target memory:

1. The debugger writes opcodes to PROGBUF to accomplish the desired function.

2. The debugger writes the desired memory address to DATA[0].

3. The debugger requests an abstract command specifying to load s0 from DATA[0], then execute PROGBUF. Writing to command while hart n is selected has the side effect of setting FLAGS[n].go. Writing to command also sets busy which is readable from the debugger, and indicates that an abstract command is in progress.

4. The ROM busy-wait loop being executed by hart n sees FLAGS[n].go set.

5. ROM code writes 0 to GOING which has the effect of clearing FLAGS[n].go.

6. ROM code jumps to WHERETO, then ABSTRACT which contains the opcode lw s0, 0(DATA) to load s0 from DATA[0]. Opcodes in ABSTRACT are constructed by DM hardware from com- mand. If command.transfer=0, no register transfer is done and instead ABSTRACT[0] reads as NOP.

7. If a register read/write is all that is needed, the debugger would set command.postexec to 0.

ABSTRACT[1] would then read as EBREAK.

8. If command.postexec=1, ABSTRACT[1] reads as NOP and execution falls through to PROGBUF which will have been previously written by the debugger with the opcodes lw s0, 0(s0), then sw s0, DATA(zero), then EBREAK.

9. EBREAK reenters ROM at address 0x800. ROM writes hartid to HALTED which has the side effect of clearing busy, telling the debugger that the abstract command is finished.

10. The debugger reads the result from DATA[0].

The autoexec feature of Abstract Commands is supported by EIC7700X MCPU hardware (and is used by OpenOCD for memory block read and write). Once an abstract command has been completed, the debugger can read or write a particular DATA or PROGBUF location to run the command again. For example, fast download can be accomplished by setting up PROGBUF for memory write, then repeatedly writing words to DATA[0]. Each write re-executes the register transfer and PROGBUF to store the word into memory. For a 32-bit block write, the abstract command would be set up like this:

ABSTRACT	regno=s1, write=1, transfer=1, postexec=1. DM constructs the instructions lw s1,0(DATA) // load s1 from debugger NOP // fall thru to PROGBUF
PROGBUF	sw s1, 0(s0) // store s1 to memory addi s0, s0, 4 // increment memory pointer ebreak // done

- **Multi-core Synchronization**

The DM is configured with one Halt Group that may be programmed to synchronize execution between harts, or between hart(s) and external logic, such as a cross-trigger matrix. The Halt Group is configured using the dmcs2 register.

- **Hart Array**

The Hart Array is an internal bitmap that selects a subset of the harts in a system. Debug opera- tions such as resume and halt are automatically applied to all the selected harts simultaneously.

To configure the Hart Array:

1. Set the hasel bit in the Debug Module dmcontrol register.

2. Set bits in the hawindow register corresponding to the harts to be selected. Bit 0 = hart 0, Bit 1 = hart 1, etc.

The Hart Array covers debug operations initiated by the debugger but does not cover the case when harts halt due to other causes, such as breakpoints. This is handled with a Halt Group.

- **Halt Groups and External Triggers**

Debug Module External Triggers consist of a pair of signals (request and acknowledgement) leading from the DM to external logic and another pair leading from external logic to the DM.

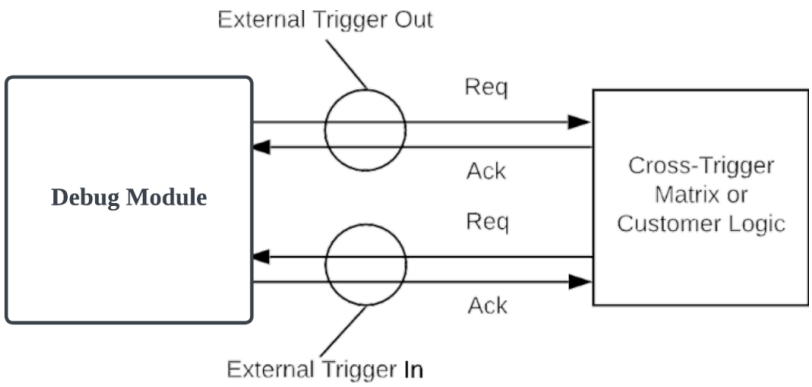


Figure 3-22 Debug External Triggers

When a system has more than one hart or is configured with any External Triggers, the Debug Module

is configured with one Halt Group and one Resume Group. Any subset of the harts and External Triggers present in the system may be set up to participate in the Halt Group and, inde- pendently, to participate in the Resume Group.

Setting up a halt group or resume group requires a series of writes to the dmcs2 register. For example:

```
dmcs2 <- 0x0006 // Add all selected harts to halt group 1
dmcs2 <- 0x0087 // Add external trigger 1 to halt group 1
dmcs2 <- 0x0806 // Add all selected harts to resume group 1
dmcs2 <- 0x0807 // Add external trigger 0 to resume group 1
```

At this point, the user can halt and resume the selected harts by writing the haltreq and resumereq bits. If any of the harts in the halt group halts either due to writing haltreq or due to hitting a breakpoint, hardware in the Debug Module will generate a haltreq to all the other harts in the group, and it will assert ExtTriggerOutReq[1]. If any of the harts in the resume group resumes, hardware will resume all the other harts in the group, and it will assert ExtTriggerOutReq[0]. Conversely, an external agent can assert ExtTriggerInReq[0] to cause all harts in the resume group to resume or assert ExtTriggerInReq[1] to cause all harts in the halt group to halt.

The trigger request is generally synchronized to the receiver's clock domain, then acknowl- edged by reflecting the synchronized request back to the corresponding acknowledge.

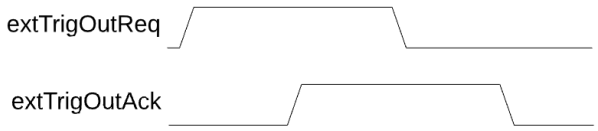


Figure 3-23 Debug External Trigger Handshake

When an external trigger is included in a Halt Group, any core in the group that halts for any reason causes the external trigger output to fire. Similarly, when an external trigger input that is configured into the Halt Group fires, it causes all the cores in the Halt Group to halt.

- **Using External Triggers**

The following section provides an example of how to use the Debug Module External Triggers. For example, this code sets up trigger 0 on a store to an address 0x12345678 with action “raise breakpoint exception”:

```
csrw 0x7a0, 0x00000000 // select trigger 0
csrw 0x7a2, 0x12345678 // program trigger address comparison
csrw 0x7a1, 0x200000000000005a // Action 0, MSU, Store
```

Debug external triggers connect to the Debug Module and can be programmed to send an event when the core halts (enters debug mode) or to cause the core to halt when an external trigger event is received.

> **Note**
>
> Debug external triggers cannot be generated by an M-mode trigger and cannot be programmed to cause an M-mode breakpoint.

- **System Bus Access**

System Bus Access (SBA) provides an alternative method to access memory. SBA operation conforms to The RISC-V Debug Specification, Version 1.0 and its description is not duplicated here. It implements a bus master that connects with the bus crossbar to allow access to the device's physical address space without involving a hart to perform accesses. SBA is controlled from the DMI using registers in the range 0x37 - 0x3F. By default, the maximum bus width sup- ported by SBA is 64. Comparing Program Buffer memory access and SBA:

Table 3-158 System Bus vs. Program Buffer Comparison

Program Buffer Memory Access	SBA Memory Access
Physical Address	Physical Address
Subject to Physical Memory Protection (PMP)	Not subject to PMP
Cache coherent	Cache coherent
Hart must be halted	Hart may be halted or running

- **Debug Module Operational Sequences**

The sections below describe the flow for entering into and exiting from debug mode. The user can halt and resume more than one hart at a time using the hart array mask.

- **Entering Debug Mode**

To use debug mode, the DM must be enabled by writing 0x0000_0001 to dmcontrol.

The debugger can request a halt by writing 0x8000_0001 to dmcontrol to set haltreq. This generates a debug interrupt to the core.

The core enters debug mode and jumps to the debug interrupt handler located at 0x800 and serviced from the DM.

ROM code at 0x800 writes hartid into the HALTED register which has the effect of setting the halted bit for this hart. Halted bits are readable from the debugger and generally will be continu- ally polled to check for breakpoints when a hart is running.

ROM code then busy-waits checking its hart-specific FLAGS register.

- **Exiting Debug Mode**

The debugger writes 1 to resumereq in the dmcontrol register to restart execution. This clears

resumeack and sets bit 1 of the FLAGS register for the selected hart.

The ROM busy-wait loop being executed by hart n sees FLAGS[n].resume set.

ROM code writes hartid to RESUMING, which has the effect of clearing FLAGS[n].resume, setting resumeack, and clearing halted for the hart.

ROM code then executes dret which returns to user code at the address currently in dpc. The debugger sees resumeack and knows the resume was successful.

- **Debug and Core Complex Power Gating**

The sections below describe the behavior of the Debug Module as it relates to core-complex clock gating.

- **Individual Core Debug Availability**

As a core prepares for power gating, it may execute a software sequence terminated with the WFI instruction. Retirement of the WFI instruction causes the wfi_from_tile_N signal to be asserted and also transitions the core debug status to "Unavailable" in the dmstatus register of the Debug Module in the uncore domain that remains powered. Cores that terminate instruction execution with WFI remain available for debug until the core-complex power is removed

- Core Complex Debug Availability

When any number of cores, but not the entire core-complex have been powered off, the Debug Module remains powered and can access powered cores and uncore state . When the core- complex is powered off, the Debug Module is also powered off, but the DebugTransportModule (DTM) remains powered to respond to JTAG transactions (if JTAG is used). Accesses to state within the core-complex receive an error, and accesses to state outside the core-complex are forwarded as normal.

The core-complex debug interface offers either APB or JTAG/cJTAG options. When the core- complex is being powered off and APB is chosen, isolation clamp values terminate all APB accesses. When JTAG/cJTAG is chosen, the core-complex power domain boundary is modified so the DebugTransportModule (DTM) remains powered by the ungated core-complex supply. This allows JTAG operations to be forwarded while the core-complex is powered off.

3.4.1.19 Trace System

- **Introduction**

Instruction Trace is a system that collects a history of processor execution, along with other events. The trace system is set up and controlled using a TileLink interface, typically connected to the Peripheral Crossbar. Core execution activity appears on the Trace Bundle and is com- pressed and formatted into trace messages which are transmitted via TileLink to a Trace Sink. The system includes a System Memory sink. Each core has its own Trace Encoder and, typically, all will connect to a single Trace Funnel, which aggregates trace data from multiple sources and sends it to a single destination.

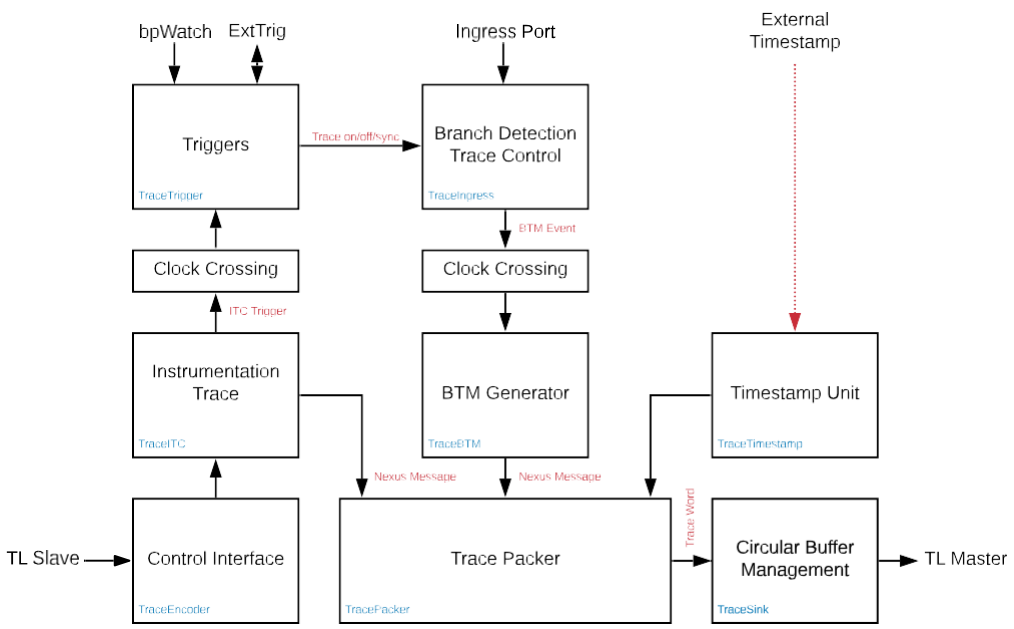


Figure 3-24 Trace Encoder Block Diagram

- **Trace Control Interface**

The Trace Encoder (TE) control interface is a TileLink slave node typically connected to the CBUS crossbar, so it appears in the system physical memory space. The control interface is used to set up and control a trace session, generate messages from instrumentation writes, retrieve collected trace history and control any trace system components directly connected to this Trace Encoder.

If the Debug Module is implemented with System Bus Access (SBA), a JTAG probe can control and monitor the trace system without requiring intervention by any of the cores.

Mapping the control interface into physical memory also allows one of the cores to manage a trace session—for example, collecting crash information in the field—or to modify trace collection parameters during execution. If desired, a Physical Memory Protection (PMP) range can be configured to restrict access to the trace system from core(s). If PMP is used, the ITC stimulus registers would normally be enabled for access.

The Trace Funnel (TF) is a variant of the Trace Encoder and shares many of the same regis- ters. The Trace Funnel occupies its own 4K block in the system memory space.

The EIC7700X MCPU includes multiple Trace Encoders. Each Trace Encoder feeds a single Trace Funnel.

Table 3-159 Trace Encoder Memory Map

Offset	TE Name	TF Name	Description
0x0000	teControl	tfControl	Trace Encoder/Funnel control register
0x0004	teImpl	tfImpl	Trace Encoder/Funnel implementation information
0x0010	teSinkBase	teSinkBase	Base address of circular trace buffer
0x0014	teSinkBaseHigh	teSinkBaseHigh	Bits [N:32] of the circular buffer address where N is equal to the most significant bit of the physical address.
0x0018	teSinkLimit	teSinkLimit	End address of circular trace buffer
0x001C	teSinkWP	teSinkWP	Current write location for trace data in circular buffer
0x0040	tsControl	tsControl	Timestamp control register
0x0044	tsLower	tsLower	Lower 32 bits of timestamp counter
0x0048	tsUpper	tsUpper	Upper bits of timestamp counter
0x0050	xtiControl	—	External Trigger Input control register
0x0054	xtoControl	—	External Trigger Output control register
0x0058	wpControl	—	Core watchpoint control register
0x0060	itcTraceEnable	—	ITC trace enable bits, 1 bit per virtual stimulus register
0x0064	itcTrigEnable	—	ITC trigger enable bits, 1 bit per virtual stimulus register
0x0080—0x00FF	itcStimulus	—	Instrumentation Trace stimulus 0 to 31

Offset	TE Name	TF Name	Description
0x0100–0x01FF	pcSample	—	Register interface to PC Sampling Unit, if present

- **Instruction Trace Encoder**

- **Ingress Port**

The Instruction Trace Encoder (ITE) monitors the activity of the core and determines whether and what type of branches are taken. ITE encodes a trace message for each taken branch or exception. Trace Messages conform to the IEEE-ISTO 5001-2012 (Nexus) Standard.

Table below shows the interface from the core to the Trace Encoder.

Table 3-160 Core to Trace Encoder Interface

Trace Spec Signal	TE Signal	Description
—	clock	Core clock
reset	reset	Core is in reset (does not reset TE)
iretire	valid	1 indicates an instruction has retired
iexception	exception	1 indicates an exception
interrupt	interrupt	1 indicates an interrupt
iaddr[63:0]	iaddr[63:0]	If iretire=1, the instruction address
instr[31:0]	insn[31:0]	If iretire=1, the instruction opcode
priv[2:0]	priv[2:0]	If iretire=1, the privilege mode: • 0x0 - User mode • 0x1 - Supervisor mode • 0x3 - Machine mode • 0x4-0x7 - Debug mode
—	bpwatch[n:0]	Core watchpoint actions, 4 bits per watchpoint: • valid: this watchpoint is matching in this cycle • action[2:0]: action field for this watchpoint
stall	teStall	Stall input to core

- **In-Circuit Trace Message**

In certain conditions, the Trace Encoder needs to report the address of the most recent retired instruction. The In-circuit Trace message (ICT) is used for this reporting. The address is either U-ADDR or F-ADDR and this is indicated by the TCODE (either ICT or ICT with Sync). When the message is ICT, the address is compressed and refers to the full address of the most recent message with an address and the timestamp is compressed normally. When the Sync form of ICT is used, the address is F-ADDR and, the timestamp is uncompressed. However, ICT with Sync cannot be used by the trace decoder as

a starting point because there may be previous ICNT history that cannot be reported with ICT.

Table 3-161 CKSRC Description

CKSRC	Description	Notes
8	External Trigger	Used when an external trigger with action=4 occurs at a time when no instructions are being retired or when a PTSYNC External Trigger message cannot be sent due to encoder resource limitation. Always uses ICT.
14	Watchpoint	Used to report a watchpoint with action=4 that occurs at a time when a PTSYNC Watchpoint message cannot be sent due to encoder resource limitation. The watchpoint is noted and sent as soon as possible. ICT is used for this reporting when there are no instructions being retired. Always uses ICT.
15	PC sample	Message sent when TE is in Periodic Sample mode (teInstruction=1). Normally uses ICT, but an ICT with Sync is sent at a rate determined by teSyncMaxBTM.

- **TE Trace Packer**

The Trace Ingress Port operates in the core clock domain. Minimal processing is done prior to crossing into the external_source_for_trace_com_clock domain, where BTMs are generated. If external_source_for_trace_com_clock and external_source_for_trace_N_clock are the same, then this async crossing is omitted.

The Trace Packer receives the BTM and packs its Nexus slices (see below) into words. Com- pleted words are sent to the TL master interface. If a message ends in the middle of a word and no additional messages are pending, then the Packer retains the partial word until another mes- sage is received. The Packer also receives formatted Nexus messages from the ITC unit and watchpoints. BTM has priority, followed by watchpoints, then ITC. The Packer may generate more than one trace word per cycle; these are stored in a shift register until the TileLink master can consume them. When TE is disabled (teEnable → 0), any partial word left in the trace packer is filled with idle slices and sent.

The Trace Packer adds timestamps to the formatted Nexus messages prior to saving in the shift register. Timestamps are added at this point so that the timestamp compression thread follows through all message types, whether BTM, ITC, or watchpoints.

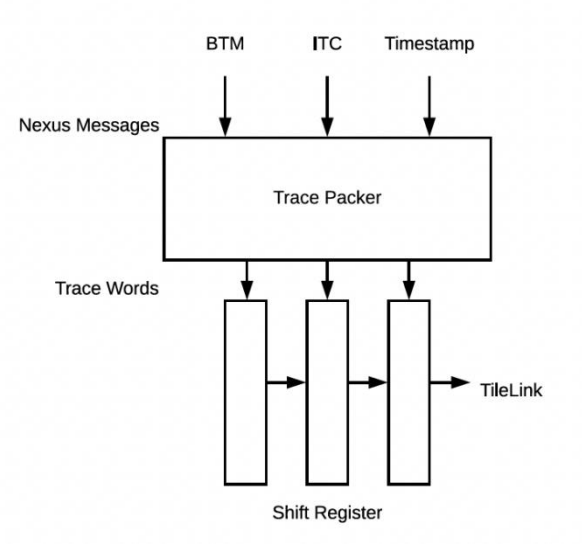


Figure 3-25 Trace Packer

- **Register Interface**

This section describes the functionality of the Trace Encoder registers.

Table 3-162 Trace Encoder Control Register

Trace Encoder Control Register (teControl)				
Address Offset		**0x0000**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	teActive	RW	0x0	Master enable for trace system. 0 resets the TE and clocks may be gated off. teActive must be set to 1 before any other registers (except pcControl), including other fields in teControl, are programmed.
1	teEnable	RW	0x0	1=TE enabled. Allows teTracing to turn all tracing on and off. Setting teEnable to 0 flushes any queued trace data to the designated sink.
2	teTracing	RW	0x0	1=Trace is being generated. Written from tool or controlled by triggers. Note that this field is volatile and may be masked in IP-XACT.
3	teEmpty	RO	0x1	Reads as 1 when all generated trace has been emitted. Note that this field is volatile and may be masked in IP-XACT.
Trace Encoder Control Register (teControl)				

[6:4]	teInstruction	WARL	0x0	Instruction trace generation parameters: • 0x0 - Don't generate BTM or Sync (no instruction trace) • 0x1 - Event trace mode • 0x2 - Reserved • 0x3 - Reserved • 0x4-0x5 - Reserved • 0x6 - Generate HTM and Sync without HTM optimizations • 0x7 - Generate HTM and Sync with full optimization
[8:7]	teInstrumentation	RW	0x0	ITC trace generation parameters for itcStimulus registers whose corresponding itcTraceEnable=1. • 0x0 - ITC trace disabled • 0x1 - Generate ITC messages for all itcStimulus registers • 0x2 - Generate Ownership message for itcStimulus[15, 31], disable other ITC trace • 0x3 - Generate Ownership message for itcStimulus[15, 31] and ITC messages for all other itcStimulus registers
[11:9]	Reserved	RO	0x0	
12	teStallOrOverflow	RW	0x0	Written to 1 by hardware when an overflow message is generated or when the TE requests a core stall. Debugger can write 0 to clear. Note that this field is volatile and may be masked in IP-XACT.
13	teStallEnable	RW	0x0	0=If TE cannot accept a message, an overflow is generated. 1=If TE cannot accept a message, the core is stalled until it can.
14	teStopOnWrap	RW	0x0	Disable trace (teEnable → 0) when circular buffer fills for the first time.
Trace Encoder Control Register (teControl)				
				Present only in systems with SRAM or System Memory sinks.

15	teInhibitSrc	RW	0x0	1=Disable SRC field in Nexus messages. Unless disabled, a SRC field of teImpl.nSrcBits is added to every Nexus message to indicate which TE generated each message.
[19:16]	teSyncMaxBTM	WARL	0x3	Maximum number of BTMs between periodic Sync messages. A Sync emitted for another reason will reset this timer. • 0x0-0xB - Generate Sync after 2^(teSyncMaxBTM + 5)^ BTM's. Range is 32 to 65536. Default is 256. • 0xC-0xE - Reserved • 0xF - Off. Do not generate a Sync message due to a count of BTMs.
[23:20]	teSyncMaxInst	WARL	0xA	Maximum I-CNT between BTMs. A BTM emitted for another reason will reset this timer. • 0x0-0x5 - Generate Sync when running I-CNT reaches 2^(teSyncMaxInst + 5)^ bytes. Range is 32 to 1024 bytes. Default is 1024. • 0x6-0xE - Reserved • 0xF - Off. Do not generate a Sync message due to instruction or cycle count.
[26:24]	teSliceFormat	RO	0x1	Nexus Slice Format • 0x0 - Reserved • 0x1 - 6 MDO + 2 MSEO • 0x2-0x7 - Reserved
27	teSinkError	RW1C	0x0	Hardware sets this to 1 when a trace sink TileLink transaction results in an error. This can happen with a System Memory sink when the address or size is invalid or with PIB, ATB, and Funnel sinks when the destination device is
Trace Encoder Control Register (teControl)				
				disabled. Write 1 to clear. Note that this field is volatile and may be masked in IP-XACT.

[31:28]	teSink	WARL	0x8	Which sink to send trace to. If there is only one sink type, this field may read as either 0 or that sink's type number: • 0x0 - Send trace to the one sink connected • 0x1-0x3 - Reserved • 0x5 - SRAM Sink • 0x5 - ATB Sink • 0x6 - PIB Sink • 0x7 - SBA Sink • 0x8 - Funnel Sink • 0x9-0xF - Reserved

Table 3-163 Trace Encoder Implementation Register

Trace Encoder Implementation Register (teImpl)				
Address Offset		**0x0004**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[3:0]	version	RO	0x0	TE Version (0 is this version)
4	hasSRAMSink	RO	0x0	1 if this TE has an on-chip SRAM sink. Size of SRAM may be determined by writing all 1s to teSinkWP, then reading the value back.
5	hasATBSink	RO	0x0	1 if this TE has an ATB sink.
6	hasPIBSink	RO	0x0	1 if this TE has an off-chip trace port via a Probe Interface Block.
7	hasSBASink	RO	0x0	1 if this TE has an on-chip bus master through the Front Port.
8	hasFunnelSink	RO	0x1	1 if this TE feeds into a Trace Funnel device.
[15:9]	Reserved	RO	0x0	
[17:16]	sinkBytes	RO	0x0	Width of this TE's trace sink. • 0x0 - 32-bit • 0x1 - 64-bit • 0x2 - 128-bit • 0x3 - 256-bit
[19:18]	crossingType	RO	preset	Type of clock crossing between Ingress and BTM Generator. • 0x0 - Synchronous

Trace Encoder Implementation Register (teImpl)				
Address Offset		0x0004		
Bits	Field Name	Attr.	Rst.	Description
				• 0x1 - Rational • 0x2 - Asynchronous
[23:20]	srcID	RO	preset	This TE's source ID. If nSrcBits>0 and SRC is not disabled by teInhibitSrc, then Nexus messages will all include a SRC field of nSrcBits bits. Messages from this TE will have this value in SRC.
[26:24]	nSrcBits	RO	preset	The number of bits in the Nexus SRC field, unless disabled by teInhibitSrc.
27	Reserved	RO	0x0	
[31:28]	hartid	RO	preset	The hartid of the hart this TE is connected to. Note that this field is volatile and may be masked in IP-XACT.

- **Instrumentation Trace Component (ITC)**

ITC appears as a block of 32 × 32-bit stimulus registers in the Trace Encoder control interface.

A bitmap enable register selectively enables any subset of ITC registers to cause trace or trig- ger events. If trace is enabled, any write to an ITC stimulus register generates a Data Acquisition Message (DQM) containing the data and byte offset of the ITC register that was written. Writes of 8, 16, or 32 bits are supported. The read value of a Stimulus register reflects the number of available write slots. Before writing to an ITC Stimulus register, software must first read a Stimulus register to obtain the number of available slots. Software may then write up to that number of items to ITC Stimulus before checking again.

ITC writes may also be configured to generate external trigger outputs for communicating events to other TEs or system logic.

- **ITC Register Interface**

This section describes the functionality of the Instrumentation Trace Component registers.

Table 3-164 Instrumentation Trace Enable Register

Instrumentation Trace Enable Register (itcTraceEnable)				
Address Offset		0x0060		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	itcTraceEnable	RW	0x0	1=write to corresponding stimulus register generates DQM or Ownership message if enabled by teInstrumentation

Table 3-165 Instrumentation Trigger Enable Register

Instrumentation Trigger Enable Register (itcTrigEnable)				
Address Offset		0x0064		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	itcTrigEnable	RW	0x0	1=write to corresponding stimulus register causes an External Trigger Out. The xtoEvent associated with a particular External Trigger Out must also be enabled to generate a Trig Out.

Table 3-166 Instrumentation Trace Stimulus Register

Instrumentation Trace Stimulus Registers (itcStimulus)				
Address Offset		0x0080 - 0x00FF		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	itcStimulus	RW	0x1	Writes generate Nexus Data Acquisition Message (DQM) with data value and byte offset if enabled by teInstrumentation and itcTraceEnable. Messages generated by writing to registers 0-15 (byte offset 0-63) do not include timestamps. Registers 16-31 include a timestamp if ITC timestamping is enabled by teInstrumentation. The byte offset generated by registers 16-31 will be 0-63. ITC supports full and partial word writes. To transmit a single byte through DQM, write the byte to offset 3 (bits [31:24]) in a register. To transmit 2 bytes, do a halfword write to offset 2 (bits [31:16]). The byte offset field in the DQM message will thereby indicate the size of the data value. Registers 15 and 31 may be set to generate Ownership trace message or DQM using teInstrumentation. A write to itcStimulus may also cause an External Trigger Out if enabled by xtoEvent and itcTrigEnable. Note that this field is volatile and may be masked in IP-XACT.

- **Using the ITC**

This section provides a code example how to use the ITC. This example assumes that the trace encoder has already been enabled.

Consider a configuration with a CLINT based at address 0x1820000 and a Trace Encoder at 0x4000. If the user would like to read mtime, a memory-mapped register in the CLINT and insert the value into the trace, you can use the following sequence:

```
Store 0x4060, 0xffffffff // enable trace for all ITC stimulus
Load  x1, 0x182bff8      // read low half of memory-mapped mtime register
Load  x2, 0x182bffc      // read high half of memory-mapped mtime register
Store 0x4080, x1         // send mtime_low to ITC
Store 0x4080, x2         // send mtime_high to ITC
Store 0x4060, 0x00000000 // disable trace for all ITC stimulus
```

- **Trace Encoder Timestamp Unit**

A Trace Encoder instance includes one of four types of timestamp unit: Internal, External, Internal with Core Clock, or Slave. The EIC7700X MCPU includes the following timestamp unit(s):

- Slave mode timestamp accepts a timestamp broadcast from a Trace Funnel configured with either Internal or External (not Core Clock). This one Trace Funnel is the active source that feeds all others.
- The timestamp counter or input is 40 bits.

The Timestamp Unit instantiated in the Trace Funnel is described in Section Trace Funnel Timestamp Unit.

- **Trace Encoder Timestamp Register Interface**

This section describes the functionality of the Timestamp Unit registers.

Table 3-167 Trace Encoder Timestamp Control Register

Trace Encoder Timestamp Control Register (tsControl)				
Address Offset		**0x0040**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	tsActive	RW	0x0	Master reset/enable for timestamp unit. tsActive must be set to 1 before any other fields in tsControl are programmed.
[3:1]	Reserved	RO	0x0	
[6:4]	tsType	RO	0x4	Type of Timestamp unit • 0x0 - None • 0x1 - External • 0x2 - Internal, bus clock • 0x3 - Internal, core clock • 0x4 - Slave • 0x5-0x7 - Reserved
[14:7]	Reserved	RO	0x0	

Trace Encoder Timestamp Control Register (tsControl)				
Address Offset		0x0040		
Bits	Field Name	Attr.	Rst.	Description
15	tsEnable	RW	0x0	Enable timestamps and add timestamp field to Sync messages

Table 3-168 Trace Encoder Timestamp Control Register

Trace Encoder Timestamp Control Register (tsControl)				
[17:16]	tsBranch	WARL	0x0	Add timestamp field to branch messages if tsEnable=1. • 0x0 - No timestamp on branch messages • 0x1 - Timestamp all indirect branch and exception messages • 0x2 - Reserved • 0x3 - Timestamp all branch, exception, PTCM, and Error messages
18	tsInstrumentation	RW	0x0	Add timestamp field to Instrumentation messages if tsEnable=1
19	tsOwnership	RW	0x0	Add timestamp field to Ownership messages if tsEnable=1
[23:20]	Reserved	RO	0x0	
[31:24]	tsWidth	RO	0x28	Width of timestamp in bits

Table 3-169 Trace Encoder Timestamp Lower Bits Register

Trace Encoder Timestamp Lower Bits Register (tsLower)				
Address Offset		0x0044		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	tsLower	RO	X	Lower 32 bits of timestamp counter. Note that this field is volatile and may be masked in IP-XACT.

Table 3-170 Trace Encoder Timestamp Upper Bits Register

Trace Encoder Timestamp Upper Bits Register (tsUpper)				
Address Offset		0x0048		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	tsUpper	RO	X	Upper bits of timestamp counter, zero-extended. Note that this field is volatile and may be masked in IP-XACT.

- **External Triggers**

The TE is configured with 3 external trigger inputs for controlling trace. These are in addition to the external triggers present in the Debug Module when Halt Groups are configured. A trigger input is a 2-signal pair consisting of a request from the system and an acknowledgment hand- shake from the TE.

The system has 3 external trigger outputs. A trigger output is a 2-signal pair consisting of a request from the TE and an acknowledgment handshake from the system.

This protocol is compatible with the Arm Cross Trigger Interface or may connect to another TE or other system logic, even in another clock domain.

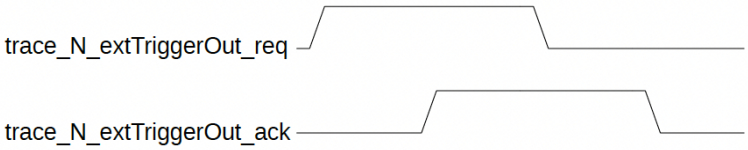


Figure 3-26 Trace Encoder External Trigger Handshake

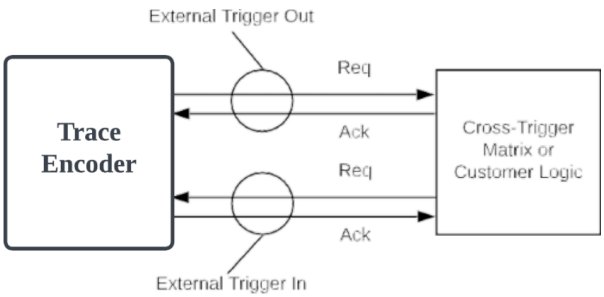


Figure 3-27 Trace Encoder External Triggers

trace_N_extTriggerIn_req is synchronized to external_source_for_trace_N_clock and is defined to have fired when a 0-to-1 transition occurs between two successive external_source_for_trace_N_clock cycles. The synchronized request is normally fed back to the system immediately on trace_N_extTriggerIn_ack. Note that if external triggers occur too often, some may be missed.

A trigger out may be generated by trace starting, trace stopping, or by a write to ITC.

In a multi-core system with one or more Trace Funnels, it is possible to configure the Funnels with

External Triggers. This internally routes a single set of off-chip External Triggers to all the Trace Encoders. Core Complex External Trigger Inputs are broadcast to all Trace Encoders. Each Core Complex External Trigger Output is the OR-reduction of the corresponding External Trigger Outputs from each Trace Encoder. No programming is needed in the Trace Funnel to perform this function.

- **Register Descriptions**

This section describes the functionality of the External Trigger registers.

Table 3-171 External Trigger Input Control Register

External Trigger Input Control Register (xtiControl)				
Address Offset		**0x0050**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[3:0]	xtiAction0	WARL	0x0	Select action to perform when external trigger input 0 fires. If external trigger input 0 does not exist, then its action is fixed at 0. • 0x0 - No action • 0x1 - Reserved • 0x2 - Start trace (teTracing → 1) • 0x3 - Stop trace (teTracing → 0) • 0x4 - Record Program Trace Sync message • 0x5-0xF - Reserved
[31:4]	xtiActionN	WARL	0x0	Select actions for external triggers 1 through 7. If an external trigger input does not exist, then its action is fixed at 0.

Table 3-172 External Trigger Output Register

External Trigger Output Control Register (xtoControl)				
Address Offset		0x0054		
Bits	Field Name	Attr.	Rst.	Description
[3:0]	xtoEvent0	RW	0x0	Bitmap to select which event(s) cause external trigger 0 output to fire. If external trigger output 0 does not exist, then all bits are fixed at 0. • 0x1 - Start trace (teTracing 0 → 1) • 0x2 - Stop trace (teTracing 1 → 0) • 0x4 - ITC write to location enabled by itcTrigEnable • 0x8 - Core watchpoint occurred with action=0x5
[31:4]	xtoEventN	RW	0x0	Select events for external trigger outputs 1 through 7. If an external trigger output does not exist, then its event bits are fixed at 0.

- **Using External Trigger Inputs**

This section describes how to use the external trigger inputs described above.

Each external trigger input is configured with a 4-bit field in xtiControl located at Trace Encoder address offset 0x50. The value in the field defines what action to take when an external trigger input fires (i.e., the req signal transitions from 0 to 1). For example, to turn trace on when external trigger 0 fires, you would write the value 0x2 to bits [3:0] of xtiControl. To turn trace off when external trigger 1 fires, write 0x3 to bits [7:4].

For the code example below, consider a design with a Trace Encoder located at 0x4000.

```
mww 0x4000 0x01530033 // Enable trace encoder with trace off
mww 0x4050 0x00000032 // Set xtiControl as described above
```

- **Using External Trigger Outputs**

External trigger outputs can be programmed to fire on any or all of 4 different conditions by writing to the 4-bit field of xtoControl at offset 0x54 corresponding to the trigger number.

The following code shows how to fire external trigger output 0 when trace starts and external trigger output 1 when trace stops. For the code example below, consider a design with a Trace Encoder located at 0x4000.

mww 0x4054 0x00000021 // xto0=trace start, xto1=trace stop

- **Code Examples**

This section provides code examples that utilize the External Trigger Outputs and Inputs. For the following examples, consider a configuration with a Trace Encoder at address 0x4000 in the memory

map. These examples assume that the Trace Encoder has already been enabled.

1. To trigger trace on an external incoming event:

Store 0x4050, 0x00000020 // Set Trigger input 1 action to 2 (start trace)

2. To generate a trigger event when trace is enabled:

Store 0x4054, 0x00001000 // Trigger out 3 responds to teTracing 0->1

- **Core Watchpoints**

Watchpoints are signals from the core that a breakpoint was hit, but the action associated with that breakpoint is a trace-related action. Action identifiers 2-5 are reserved for trace actions in the RISC-V Debug Spec, where breakpoints are defined. The desired action is written to the action field of the mcontrol CSR (0x7A1). Not all cores support trace actions; the debugger should read back mcontrol after setting one of these actions to verify that the option exists. Once synchronized, watchpoint actions take effect unconditionally, assuming the TE is enabled with teEnable.

The start and stop trace actions can operate in two modes, selected by wpControl.

• Edge mode: iretire & (wpAction[n]=2) sets teTracing to 1. iretire & (wpAction[m]=3) sets teTracing to 0.

• Range mode: iretire & (wpAction[n]=2) sets teTracing to 1. iretire & (wpAction[n]≠2) sets teTracing to 0. Range mode allows using only one address range watchpoint to enable tracing in a range of instruction addresses.

Table 3-173 Watchpoint Action Identifiers

wpAction	Effect
0	Breakpoint exception
1	Debug exception
2	Start trace (teTracing → 1)
3	Stop trace (teTracing → 0)
4	Record Program Trace Sync message
5	Generate an External Trigger Out

- **Register Descriptions**

This section describes the functionality of the Core Watchpoint registers.

Table 3-174 Core Watchpoint Control Register

Core Watchpoint Control Register (wpControl)				
Address Offset		0x0058		
Bits	Field Name	Attr.	Rst.	Description
0	wpRange0	RW	0x0	Select Edge (0) or Range (1) mode for core watchpoint 0
[N:1]	wpRangeN	RW	0x0	Select Edge (0) or Range (1) mode for core watchpoints 1 to N. Fixed at 0 for any watchpoints that don't exist.

- **Watchpoint Precedence**

Core watchpoints and External Trigger Inputs operate under a specific precedence when simultaneous conflicting actions occur.

• Core watchpoint with action=5 (generate trigger out) is always executed

• For trace start and stop actions, the precedence is: core watchpoints 0 through N, then external triggers 0 through N.

◦ The active watchpoint with the highest precedence that is programmed for trace start or stop is selected. All lower active watchpoints are ignored.

◦ If the selected watchpoint has trace start action and trace is currently off, then trace is turned on and a Trace-On message is sent.

◦ If the selected watchpoint has trace stop action and trace is currently on, then trace is turned off and a Trace-Off message is sent.

◦ If a watchpoint in Range mode is active and a lower precedence watchpoint or trigger with Edge action occurs, the Edge action is ignored.

◦ If a watchpoint in Range mode is active and a higher precedence Range watchpoint occurs, the lower one is disabled until it deactivates. A message is sent only if the action associated with the new Range watchpoint alters the state of tracing.

◦ If a watchpoint in Range mode is active and a higher precedence Edge watchpoint occurs, the Edge action is executed, and the range trigger is disabled until it deactivates. A message is sent only if the action associated with the Edge watchpoint alters the state of tracing.

◦ A disabled Range watchpoint must traverse to the inactive state (wpAction[n]≠2) before it can activate again.

• Core or external watchpoint with action=4 (record Sync message) is executed if trace is enabled (teTracing=1) and no message was produced by above rules. If more than one watchpoint occurs simultaneously, only one Sync message is generated. If the TE can't insert the Sync due to resource limitation, the Sync message request is held until the Sync can be generated. If another Sync request occurs during this time, the second request will be ignored.

- **PC Sampling Unit**

Since the Trace Encoder directly connects to the core's TracedInstruction output bundle, it is relatively easy to add a PC Sampling Unit that provides a read-only interface to the current execution address. In some hang scenarios, this information is useful to be able to pinpoint what the core was doing just prior to the hang. This interface can also be used for non-intrusive statistical profiling.

In systems with a watchdog timer that resets the core upon timeout, it is also useful to capture the address of the last retired instruction so that it can be read out as part of a crash analysis. The pcCapture registers capture a new sample on each occurrence of the core reset.

- **Register Descriptions**

This section describes the functionality of the PC Sampling Unit registers.

Table 3-175 PC Sample Control Register

PC Sample Control Register (pcControl)				
Address Offset		0x0100		
Bits	Field Name	Attr.	Rst.	Description
0	pcActive	RW	0x0	Master enable/reset for PC Sampling Unit. pcActive must be set to 1 before any other fields in pcControl are programmed. This bit is reset to 0 by the trace reset input and must be set to 0x1 in order to perform sampling. The clock to the PC Sampling Unit may be gated off when pcActive=0. Note that pcActive is not dependent on teActive, so the PC Sampling Unit can be enabled while the rest of the trace encoder is disabled.
[31:1]	Reserved	RO	0x0	

Table 3-176 PC Capture and Sample Implementation Register

PC Capture/Sample Implementation Register (pcImpl)				
Address Offset		0x0104		
Bits	Field Name	Attr.	Rst.	Description
[15:0]	pcCaptureFields	RO	preset	Bitmap indicating which core registers other than PC are present in the PC Capture register range. PC is always present. Capture registers are populated downward from 0x134 in the order listed (bit 0 to 15) with no gaps. Core registers in this list that are more than 32 bits occupy two capture register locations with the least-significant word at the higher address. • Bit 0 = MEPC • Bits [15:1] = Reserved

PC Capture/Sample Implementation Register (pcImpl)				
Address Offset		0x0104		
Bits	Field Name	Attr.	Rst.	Description
[31:16]	pcSampleFields	RO	preset	Bitmap indicating which core registers other than PC are present in the PC Sample register range. PC is always present. Sample registers are populated downward from 0x174 in the order listed (bit 0 to 15) with no gaps. Core registers in this list that are more than 32 bits occupy two sample register locations with the least-significant word at the higher address. • Bits [15:0] = Reserved

Table 3-177 PC Capture Register

PC Capture Register (pcCapture)				
Address Offset		0x013C		
Bits	Field Name	Attr.	Rst.	Description
0	pcCaptureValid	RO	0x0	The pcCapture/pcCaptureH registers contain a valid capture. This bit is reset to 0 when pcActive=0 and becomes 1 when there is an instruction retirement followed sometime later by a core reset. Note that this field is volatile and may be masked in IP-XACT.
[31:1]	pcCapture	RO	X	Address of the last instruction retired prior to the most recent core reset. Note that this field is volatile and may be masked in IP-XACT.

Table 3-178 PC Capture High Register

PC Capture High Register (pcCaptureH)				
Address Offset		0x0138		
Bits	Field Name	Attr.	Rst.	Description
[N:0]	pcCaptureH	RO	X	If the core address is more than 32 bits, this register reflects bits [N:32] of the address of the PC Capture register. Reads as zero in systems without PC Sampling or with an address width of 32 bits or less. Note that this field is volatile and may be masked in IP-XACT.

Table 3-179 PC Sample Register

PC Sample Register (pcSample)				
Address Offset		0x017C		
Bits	Field Name	Attr.	Rst.	Description
0	pcSampleValid	RO	0x0	The pcSample/pcSampleH registers contain a valid sample that has not yet been read. This bit is reset to 0 when pcActive=0 and becomes 1 when the core retires an instruction. This bit is also reset when pcSample is read. Note that this field is volatile and may be masked in IP-XACT.
[31:1]	pcSample	RO	X	Address of most recent retired instruction. Note that this field is volatile and may be masked in IP-XACT.

Table 3-180 PC Sample High Register

PC Sample High Register (pcSampleH)				
Address Offset		0x0178		
Bits	Field Name	Attr.	Rst.	Description
[N:0]	pcSampleH	RO	X	If the core address is more than 32 bits, this register reflects bits [N:32] of the address of the most recent retired instruction. Reads as zero in systems without PC Sampling or with an address width of 32 bits or less. Since a full PC sample covers more than one register, hardware is provided to latch the full sample when any one of the sample registers below pcSample is read. In this way, reading an address range starting with a lower address and ending with pcSample will provide the values of all the fields at the time the first register was read. There is space in the address map below pcSampleH for additional fields to be added in the future. Note that this field is volatile and may be masked in IP-XACT.

- **Trace Funnel**

The Trace Funnel (TF) combines messages from multiple sources into a single trace stream. The TF has the same options for sinks as a single Trace Encoder which are controlled by the same registers as described above. Trace words arriving via TileLink connections from each Trace Encoder are unpacked into full Nexus messages. Messages are chosen by a round-robin arbiter and sent to the Packer. From this point, the TF shares hardware blocks with the Trace Encoder.

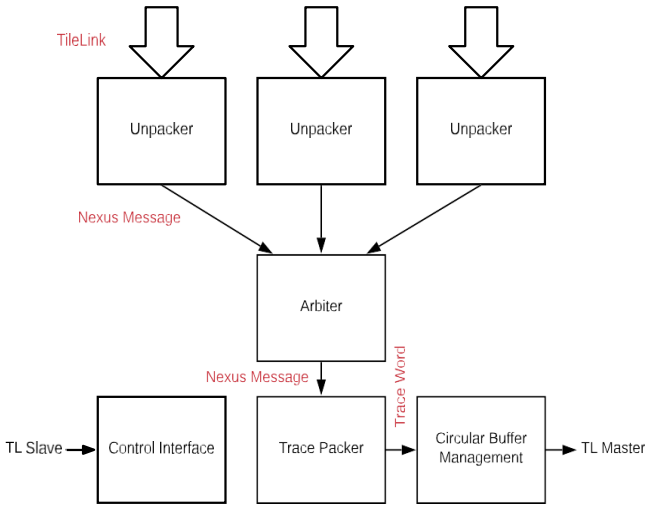


Figure 3-28 Trace Funnel Block Diagram

- **Register Descriptions**

This section describes the functionality of the Trace Funnel registers.

Trace Funnel Control Register

Table 3-181 Trace Funnel Control Register

Trace Funnel Control Register (tfControl)				
Address Offset		**0x0000**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	tfActive	RW	0x0	Master enable for trace funnel. 0 resets the TF and clocks may be gated off.
1	tfEnable	RW	0x0	1 indicates TF enabled. Messages arriving via TileLink will be accepted, combined, and sent to the designated sink. When the funnel is disabled, TileLink data will be accepted but dropped. Setting tfEnable to 0x0 will flush any partially filled trace words to the sink.
2	Reserved	RO	0x0	
3	tfEmpty	RO	0x1	Reads as 1 when all generated trace has been emitted. To stop trace, a debugger would typically write 0 to tfEnable, then read tfEmpty until it reads as 1 indicating all trace has been flushed. Note that this field is volatile and may be masked in IP-XACT.
[13:4]	Reserved	RO	0x0	
14	tfStopOnWrap	RW	0x0	Disable trace (tfEnable → 0x0) when circular buffer fills for the first time. Present only in systems with SRAM or System Memory sinks.
[26:15]	Reserved	RO	0x0	
27	tfSinkError	RW1C	0x0	Hardware sets this to 1 when a trace sink TileLink transaction results in an error. This can happen with System Memory sink when the address or size is invalid or with PIB, ATB, and Funnel sinks when the destination device is disabled. Write 1 to clear. Note that this field is volatile and may be masked in IP-XACT.
[31:28]	tfSink	WARL	0x7	• 0x0 - Send trace to the one sink

Trace Funnel Control Register (tfControl)				
Address Offset		0x0000		
Bits	Field Name	Attr.	Rst.	Description
				connected • 0x1-0x3 - Reserved • 0x4 - SRAM Sink • 0x5 - ATB Sink • 0x6 - PIB Sink

Trace Funnel Control Register

Table 3-182 Trace Funnel Control Register

Trace Funnel Control Register (tfControl)				
				• 0x7 - SBA Sink • 0x8 - Funnel Sink • 0x9-0xF - Reserved

Trace Funnel Implementation Register

Table 3-183 Trace Funnel Implementation Register

Trace Funnel Implementation Register (tfImpl)				
Address Offset		0x0004		
Bits	Field Name	Attr.	Rst.	Description
[3:0]	version	RW	0x0	Version (0 is this version).
4	hasSRAMSink	RO	0x0	1 indicates TF enabled. Messages arriving via TileLink will be accepted, combined, and sent to the designated sink.
5	hasATBSink	RO	0x0	1 if this TF has an ATB sink
6	hasPIBSink	RO	0x0	1 if this TF has an off-chip trace port via a Probe Interface Block
7	hasSBASink	RO	0x1	1 if this TF has an on-chip bus master through the Front Port
8	hasFunnelSink	RO	0x0	1 if this TF is a mid-level Funnel that feeds into another Funnel
[15:9]	Reserved	RO	0x0	

Trace Funnel Implementation Register (tfImpl)				
Address Offset		0x0004		
Bits	Field Name	Attr.	Rst.	Description
[17:16]	sinkBytes	RO	0x1	Width of this TF's trace sink. 0x0 – 32 bit, 0x1 - 64 bit, 0x2 - 128 bit, 0x3 - 256 bit.
[31:18]	Reserved	RO	0x0	

Trace Funnel Sink Base Register

Table 3-184 Trace Funnel Sink Base Register

Trace Funnel Sink Base Register (teSinkBase)				
Address Offset		0x0010		
Bits	Field Name	Attr.	Rst.	Description
[1:0]	Reserved	RO	0x0	
[N:2]	teSinkBase	RW	0x0	Base address of trace circular buffer relative to TF TileLink master node. Fixed to the natural boundary implied by width of trace sink.

Trace Funnel Sink Base High Bits Register

Table 3-185 Trace Funnel Sink Base High Bits Register

Trace Funnel Sink Base High Bits Register (teSinkBaseHigh)				
Address Offset		0x0014		
Bits	Field Name	Attr.	Rst.	Description
[N:0]	teSinkBaseH	RW	0x0	High order bits [N:32] of address of trace circular buffer in System Memory.

Trace Funnel Sink Limit Register

Table 3-186 Trace Funnel Sink Limit Register

Trace Funnel Sink Limit Register (teSinkLimit)				
Address Offset		0x0018		
Bits	Field Name	Attr.	Rst.	Description
[1:0]	Reserved	RO	0x0	

[N:2]	teSinkLimit	RW	0x0	Highest address of trace circular buffer relative to TF TileLink master node. Fixed to natural boundary.

Trace Funnel Sink Write Pointer Register

Table 3-187 Trace Funnel Sink Write Pointer Register

Trace Funnel Sink Write Pointer Register (teSinkWP)				
Address Offset		**0x001C**		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	teSinkWrapped	RW	0x0	Set to 1 by hardware when teSinkWP wraps. Note that this field is volatile and may be masked in IP-XACT.
1	Reserved	RO	0x0	
[N:2]	teSinkWP	RW	0x0	Address in trace sink where next trace message will be written. Fixed to natural boundary. When using a System Memory Sink and a trace word write occurs while teSinkWP=teSinkLimit, teSinkWP is set to teSinkBase. Note that this field is volatile and may be masked in IP-XACT.

- **Trace Funnel Timestamp Unit**

A Trace Funnel instance includes one of four types of timestamp unit: Internal, External, Internal with Core Clock, or Slave. The EIC7700X MCPU includes the following timestamp unit(s):

• Internal timestamp uses the external_source_for_trace_com_clock to increment the timestamp counter.

• Slave mode timestamp accepts a timestamp broadcast from a Trace Funnel configured with either Internal or External (not Core Clock). This one Trace Funnel is the active source that feeds all others.

• The timestamp counter or input is 40 bits.

An Internal Timestamp Unit includes a prescale divider, which can extend the range of a narrower timestamp and uses less power but has less resolution.

Since the Internal and External Timestamp clocks are system-dependent, it may be useful to measure the clock frequency for calibration. To do this, debug software can read the count value, wait a known interval, then read the count value again. Frequency = (count difference) / (time interval). Note when reading the count value that two reads are required to retrieve the complete value and that the counter continues to run between the two. Software can definitively determine the full count value by reading tsUpper, tsLower, then tsUpper again. If the two reads from tsUpper are the same, then count = {tsUpper, tsLower}. If different, then examine the MSB of tsLower; if 0, then use the second tsUpper, and if 1, use the first tsUpper.

- **Trace Funnel Timestamp Register Interface**

This section describes the functionality of the Timestamp Unit registers.

Trace Funnel Timestamp Control Register

Table 3-188 RAS Optimization Example

Trace Funnel Timestamp Control Register (tsControl)				
Address Offset		0x0040		
Bits	Field Name	Attr.	Rst.	Description
0	tsActive	RW	0x0	Master reset/enable for timestamp unit. tsActive must be set to 1 before any other fields in tsControl are programmed.
1	tsCount	RW	0x0	1 for counter runs, 0 for counter stopped
2	tsReset	WO	X	Write 1 to reset the timestamp counter
3	Reserved	RO	0x0	
[6:4]	tsType	RO	0x2	Type of Timestamp unit • 0x0 - None • 0x1 - External • 0x2 - Internal, bus clock • 0x3 - Internal, core clock • 0x4 - Slave • 0x5-0x7 - Reserved
7	Reserved	RO	0x0	
[9:8]	tsPrescale	RW	0x0	Prescale timestamp clock by 22n (1, 4, 16, 64)
[23:10]	Reserved	RO	0x0	
[31:24]	tsWidth	RO	0x28	Width of timestamp in bits

Trace Funnel Timestamp Lower Bits Register

Table 3-189 Trace Funnel Timestamp Lower Bits Register

Trace Funnel Timestamp Lower Bits Register (tsLower)				
Address Offset		0x0044		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	tsLower	RO	X	Lower 32 bits of timestamp counter. Note that this field is volatile and may be masked in IP-XACT.

Trace Funnel Timestamp Upper Bits Register

Table 3-190 Trace Funnel Timestamp Upper Bits Register

Trace Funnel Timestamp Upper Bits Register (tsUpper)				
Address Offset		0x0048		
Bits	Field Name	Attr.	Rst.	Description
[31:0]	tsUpper	RO	X	Upper bits of timestamp counter, zero-extended. Note that this field is volatile and may be masked in IP-XACT.

- **Nexus Messages**
- **Nexus Message Formats**

The following tables show the message formats generated by the Trace Encoder and are ordered with LSBs at the bottom. The TCODE in the LSBs indicates the message type.

Table 3-191 Direct Branch Message Description

Direct Branch Message		
Bit	Name	Description
var	TSTAMP	Timestamp value
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 3

Table 3-192 Indirect Branch Message Description

Indirect Branch Message		
Bit	Name	Description
var	TSTAMP	Timestamp value
var	U-ADDR	Compressed address of target of branch
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync
2	B-TYPE	Branch type, 0=indirect branch, 1=exception

n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 4

Table 3-193 Direct Branch with Sync Message Description

Direct Branch with Sync Message		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	F-ADDR	Full target address for a taken branch. High-order zeroes are not transmitted.
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync
4	SYNC	Reason for Sync
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 11

Table 3-194 Indirect Branch with Sync Message Description

Indirect Branch with Sync Message		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	F-ADDR	Full target address for a taken branch. High-order zeroes are not transmitted.
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync
2	B-TYPE	Branch type, 0=indirect branch, 1=exception
4	SYNC	Reason for Sync
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 12

Table 3-195 Program Trace Synchronization Message Description

Program Trace Synchronization Message		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	F-ADDR	Full target address of current PC. High-order zeroes are not transmitted.

Program Trace Synchronization Message		
Bit	**Name**	**Description**
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync. Valid only if SYNC[0]=0.
4	SYNC	Reason for Sync
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 9

Table 3-196 Error Message Description

Error Message		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	pad	Pad ETYPE to end of slice so timestamp begins on slice boundary
4	ETYPE	Error Type, 0=Queue Overrun
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 8

Table 3-197 Program Trace Correlation Message Description

Program Trace Correlation Message		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	I-CNT	Number of instruction units (half-words) executed since the last BTM or Sync.
2	CDF	Number of CDATA fields (always 0x0)
4	EVCODE	Reason for PTCM
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 33

Table 3-198 In-Circuit Trace Message Description (Reports PC)

In-Circuit Trace Message		
Bits	**Name**	**Description**
var	TSTAMP	Timestamp value
var	CKDATA	Address of most recent retired instruction, U-ADDR or F-ADDR

In-Circuit Trace Message		
Bits	**Name**	**Description**
2	CKDF	Number of CKDATA fields (always 0, meaning 1 CKDATA follows)
4	CKSRC	Reason for ICT. Possible values described in Table *CKSRC Description*
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 34 (compressed) or 35 (uncompressed)

Table 3-199 Data Acquisition Message (from ITC) Description

Data Acquisition Message (from ITC)		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	DQDATA	Write data value to ITC, zero extended.
var	IDTAG	Write data index (byte offset) in ITC, zero extended
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 7

Table 3-200 Ownership Trace Message (from ITC) Description

Ownership Trace Message (from ITC)		
Bit	**Name**	**Description**
var	TSTAMP	Timestamp value
var	PROCESS	Write data value to ITC, zero extended
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 2

- **Address Compression**

Address transmissions are compliant with the Nexus spec. Address fields begin with bit 1 since all execution addresses are on 2-byte boundaries. Addresses sent in U-ADDR compressed form are computed based on a reference address sent by or computed from the most recent preceding BTM or Sync message containing an address field. Starting with an F-ADDR, each U-ADDR modifies the reference address used for the next BTM. A U-ADDR is generated by XORing the full address with the reference address and sending the result starting with bit 1 and with high- order zeroes suppressed. The reverse process is used by software to recover the original full address.

Table 3-201 Address Compression Example

Full Address	Compression								New Reference
0x003FC04	-								0x003FC04
0x003F368	ref	0000	0000	0011	1111	1100	0000	0100	0x003F368
	F-ADDR	0000	0000	0011	1111	0011	0110	1000	
	XOR	0000	0000	0000	0000	1111	0110	1100	
	U-ADDR					1111	0110	110	
0x003E100	ref	0000	0000	0011	1111	0011	0110	1000	0x003E100
	F-ADDR	0000	0000	0011	1110	0001	0000	0000	
	XOR	0000	0000	0000	0001	0010	0110	1000	
	U-ADDR				1	0010	0110	100	

- **Timestamp Compression**

If timestamp recording is enabled, Sync messages all include a full timestamp value with upper zeroes suppressed. Other message types with timestamps use a compressed value using the same XOR algorithm as in address compression. To reconstruct the full timestamp, software begins at a Sync message and stores the TSTAMP value found there, zero-extended to the full timestamp width. Software extracts the compressed TSTAMP from each message thereafter and XORs it with the previous decompressed timestamp to obtain the full timestamp value associ- ated with each message.

Table 3-202 Timestamp Compression Description

Message	TSTAMP	Computed Timestamp
Sync	00 1111 1100 0001 1001	0x0000_0000FC19
BTM	11 1110	0x0000_0000FC27
BTM	0000 0100 0010	0x0000_0000FC65

- **Nexus Transmission Protocol**

The Nexus standard defines a messaging protocol using a number of Message Data Out (MDO) signals and two flag signals known as MSEO[1:0] (Message Start/End Out). A Nexus message is sent or stored in slices comprised of MDO and MSEO. The first slice of a message sends the LSBs of the message and is indicated by MSEO=0x0. A variable-length field in a message always ends on a slice boundary (zero extended as needed) and the last slice of a variable field is indicated by MSEO=0x1. The last slice of a message is indicated by MSEO=0x3. Most messages have more than one variable field. TSTAMP is an optional field and may be present in some messages and not in others. When not present, the variable field just preceding TSTAMP will end with MSEO=0x3.

The Trace Encoder uses a slice format consisting of 6 MDO bits and 2 MSEO bits. These slices are packed into words, LSB first, for storage in SRAM or sending to another destination. Following is an example of slice packing for an Indirect Branch Message with timestamp.

Table 3-203 Nexus Transmission Protocol Description

	MDO							MSEO		
Slice	**5**	**4**	**3**	**2**	**1**	**0**		**1**	**0**	**Notes**
	X	X	X	X	X	X		1	1	Idle or end of previous message
0	T5	T4	T3	T2	T1	T0		0	0	Start of message
1	I3	I2	I1	I0	B1	B0		0	0	Normal transfer
2	0	0	0	0	I5	I4		0	1	End of var field I-CNT zero-extended to slice boundary
3	0	A5	A4	A3	A2	A1		0	1	End of var field U-ADDR zero-extended to slice boundary
4	P5	P4	P3	P2	P1	P0		0	0	Normal transfer
5	0	0	P9	P8	P7	P6		1	1	End of message (implies end of var field)
Tx=TCODE, Bx=BTYPE, Ix=I-CNT, Ax=U-ADDR, Px=TSTAMP										

- **Nexus Branch History Trace Encoder**

The Trace Encoder (TE) is a hardware block that converts execution status signals from the core to formatted Nexus trace messages. There is one TE per core and the outputs from sev- eral encoders are combined in a Trace Funnel to produce the final trace message stream.

- **History Buffer**

A branch or jump is considered "inferable" if its destination can be determined solely by examin- ing the program image. Jumps whose destination depends on processor state such as a regis- ter value are said to be "uninferable".

Rather than reporting every branch, the History Trace Mode (HTM) Trace Encoder only reports the minimal information needed by the Trace Decoder to track execution history. Unconditional inferable jumps do not generate a trace message since the decoder can follow execution flow through this type of branch by disassembling it and computing the destination address.

Conditional branches are always inferable in the RISC-V architecture.

Instead of reporting them as direct branches as in Branch Trace Mode (BTM), conditional branches are gathered into a history buffer as they are executed.

A taken branch is represented by 1 and a not-taken branch by 0. An indirect jump or sync event will generate an Indirect Branch History Message (IBHM) that includes the target address of the jump, current contents of the history buffer (HIST), and an instruction unit count (I-CNT) that includes all of the instructions executed since the previous IBHM, including unconditional jumps and all the conditional branches represented by the history buffer.

HIST is a variable length field that includes a single stop bit with value 1. Since var fields are padded to the next slice boundary, the stop bit is needed in order for the decoder to determine where data ends and the pad begins. For example, a taken branch followed by a not-taken branch would result in a padded HIST field of 000110. Reading from the MSB (oldest instruc- tion), three 0s are the pad, then 1 for the stop bit, 1 indicates taken branch, and 0 indicates not- taken branch.

The TE has hardware limitations on the number of conditional branches that can be conveyed in the HIST field and on the number of instructions between IBHMs. If either of these limits is exceeded, the TE will transmit a Resource Full message. The message will indicate whether it applies to HIST or I-CNT and will include the HIST field or I-CNT up to that point.

When trace turns off, a Program Trace Correlation Message (PTCM) is sent. PTCM always includes an I-CNT, but in Branch History Trace, it may also include a HIST field. Per the Nexus spec, the CDF field of the PTCM indicates the number of variable CDATA fields in the message. When CDF=1, then CDATA contains a HIST field that indicates the conditional branch history up to the point where the trace turned off. If a PTCM is sent and there is no HIST to report, then the simpler form of PTCM is used (i.e., CDF=0).

- **Return-Address Stack Optimization**

Uninferable jumps are generally reported with an IBHM. However, in some cases, an uninfer- able jump target can be determined and uninferable jumps are not reported. The Trace Encoder will track the current depth of the return-address stack (RAS) of the core. The RAS is initialized when the Trace Encoder transmits a Sync message. Thereafter, when a call occurs, the count is incremented and when a return occurs, the count is decremented. If a return is executed and the count is non-zero, that return is considered inferable and, therefore, does not generate an IBHM. The RAS counter is "saturating"; that is, it sticks at a maximum value if too many calls are encountered and sticks at 0 if too many returns are encountered.

Consider the example below showing starting execution at 1000 and hitting a breakpoint at 1008:

Table 3-204 RAS Optimization Example

Addr	Instruction	BTM	HTM	Notes
1000	c.add	PTSync, FADDR=1000	PTSync, FADDR=1000	Exit debug mode
1002	jal x1,2000	Direct, ICNT=3	—	Inferable call (no HTM message)
2000	c.sub	—	—	
2002	c.beqz	Direct, ICNT=2	—	1 (taken branch) pushed to HIST
2010	c.jr x1	Indirect, ICNT=1, UADDR=6	—	Return inferable due to RAS
1006	c.add	—	—	
1008	—	PTCM, ICNT=1	PTCM, ICNT=7, HIST=1	Enter debug mode (breakpoint)

- **Repeat Branch Optimization**

Another relatively easy optimization involves reporting certain cases of HIST in a compressed format using a form of the Resource Full message. A long sequence of taken branches or a long sequence of not-taken branches are common in embedded software when polling hardware registers and would normally result in multiple Resource Full messages with HIST being either all 1s or all 0s. For these cases, a vendor-defined RCODE indicates that the RDATA field of the Resource Full message contains a count rather than a bitmap of conditional branch his- tory.

Example 1: Consider a polling loop such as while (*uart_status & 1) { }. Suppose the loop reads 1 (busy) 10,000 times before finally reading 0.

Table 3-3-205 Repeat Branch Optimization Example 1

Addr	Instruction	BTM	HTM
1000	lw x6, 0(x2)	—	—
1004	andi x6,x6,1	—	—
1008	bnez x6,1000	Direct, ICNT=6	HIST = (HIST << 1)

In BTM, this generates 10,000 direct branch messages occupying 20,000 bytes of trace buffer space. If the HIST buffer capacity of the HTM Encoder is 32 bits, this loop would generate 312 Resource Full messages, all indicating 32 taken branches, using 2184 bytes. With the Repeat Branch optimization, one Resource Full message with RCODE=9 is emitted, and the final not- taken branch is loaded into the HIST buffer to be reported the next time HIST is sent. This uses only 4 bytes of trace buffer space.

Example 2: A for-loop such as for (i = 0; i < 10000; i++) { buf[i] = 0 } typically compiles into an instruction sequence with a conditional branch at the top and an unconditional jump at the bottom. The conditional branch is repeatedly not taken until the loop exits.

Table 3-206 Repeat Branch Optimization Example 2

Addr	Instruction	BTM	HTM
1000	lui x6,0	—	—
1004	lui x7,10000	—	—
1008	bge x6,x7,101c	—	HIST = (HIST << 1) | 0
100c	add x28,x3,x6	—	—
1010	sw buf(x28),x0	—	—
1014	addi x6,x6,1	—	—
1018	jal x0,1008	Direct, ICNT=10	Inferable jump (no message)

Without Repeat Branch optimization, this loop would generate 312 Resource Full messages, all indicating 32 not-taken branches. With the Repeat Branch optimization, one Resource Full mes- sage with RCODE=8 is emitted, and the final taken branch is loaded into the HIST buffer to be reported the next time it is sent.

- **Sync Messages**

The Program Trace Sync message (PT Sync) does not have a HIST field, so in Branch History Trace, certain Sync events generate an IBHM+Sync (IBHSync) instead of a PT Sync. The F- ADDR field can be used to determine whether an IBHSync represents an indirect branch. If an uninferable jump and sync marker event occur simultaneously, only one IBHMSync will be emit- ted, and SYNC will indicate the marker reason and F-ADDR will point to the discontinuous jump target. PT Sync is still used for those Sync events that indicate trace is resuming after a gap, meaning there is no previous history to transmit.

Table 3-207 Sync Messages

SYNC	Description	Message	Notes
0	External trigger	IBHSync	Sent when External Trigger Input occurs with xtiAction=4

SYNC	Description	Message	Notes
1	Exit from reset	PTSync	PT Sync message sent at first iretire after reset
2	Periodic message counter	IBHSync	Include full address in every nth branch message
3	Exit from debug mode	PTSync	PT Sync Message sent at first iretire following a gap in recording
4	Sequential instruction counter	—	Not used in HTM (Resource Full is used instead)
5	Trace enable	PTSync	PT Sync message sent at first iretire following a gap in recording
6	Watchpoint	IBHSync	IBHM sent when core watchpoint occurs with action=4
7	Restart after FIFO overrun	PTSync	PT Sync message sent at first iretire following a gap in recording

- HTM Nexus Message Formats

Table 3-208 Indirect Branch History Message

Bits	Name	Description
var	TSTAMP	Timestamp value
var	HIST	Conditional branch history
var	U-ADDR	Compressed address of target of branch
var	I-CNT	Number of instruction units (halfwords) executed since the last BTM or Sync.
2	B-TYPE	Branch type, 0=indirect branch, 1=exception
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 28

Note

If there is no branch history to transmit, then the Indirect Branch message is emitted instead (TCODE=4, no HIST).

Table 3-209 Indirect Branch History with Sync Message

Bits	Name	Description
var	TSTAMP	Timestamp value
var	HIST	Conditional branch history

Bits	Name	Description
var	F-ADDR	Full target address for a taken branch. High-order zeroes are not transmitted.
var	I-CNT	Number of instruction units (halfwords) executed since the last BTM or Sync.
2	B-TYPE	Branch type, 0=indirect branch, 1=exception
4	SYNC	Reason for Sync
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 29

Note

If there is no branch history to transmit, then the Indirect Branch with Sync message is emitted instead (TCODE=12, no HIST).

Table 3-210 Resource Full Message

Bits	Name	Description
var	TSTAMP	Timestamp value
var	RDATA	I-CNT (RCODE=0), HIST (RCODE=1), or count (RCODE=8 or 9)
4	RCODE	Resource code (0=I-CNT, 1=HIST, 8=HIST_NOTTAKEN, 9=HIST_TAKEN)
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 27

Table 3-211 Program Trace Correlation Message (HTM format)

Bits	Name	Description
var	TSTAMP	Timestamp value
var	HIST	Conditional branch history (field present only if CDF=1)
var	I-CNT	Number of instruction units (halfwords) executed since the last BTM or Sync
2	CDF	Number of CDATA fields (0 or 1, 0 is backward-compatible with BTM)
4	EVCODE	Reason for PTCM
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 33

- **Event Trace**

Event Trace is an alternative mode for recording trace that does not account for every instruction but only records a trace message when certain events occur in the core. Used with time- stamps, this mode provides the basis for many powerful performance measurements of functions and interrupts.

- **Register Map**

One additional register is added to the Trace Encoder for Event Trace. evControl is used to select some of the events that will cause ICT trace records when teInstruction=1.

Table 3-212 Event Trace Control Register

Event Trace Control Register (evControl)				
Address Offset		0x000C		
Bits	Field Name	Attr.	Rst.	Description
0	evExtTrigger	WARL	0x0	1=Record ICT when an external trigger input with xtiAction=4 occurs.
1	evWatchpoint	WARL	0x0	1=Record ICT when a watchpoint with action=4 occurs.
2	evCall	WARL	0x0	1=Record ICT when a call, coroutine swap, or return instruction is executed.
3	evInterrupt	WARL	0x0	1=Record ICT with CKSRC=11 when an interrupt occurs, and ICT with CKSRC=9 when an exception return is executed.
4	evException	WARL	0x0	1=Record ICT with CKSRC=10 when an exception occurs, and ICT with CKSRC=9 when an exception return is executed.
5	evContext	WARL	0x0	1=Record ICT when mcontext or scontext changes.
[31:6]	Reserved			

Note

There is no control bit in `evControl` to disable periodic PC events. To turn off periodic PC, set `teSyncMaxInst` to 15.

- **Operational Description**

Event Trace is enabled by setting teInstruction=1 in teControl. In previous versions of the Trace Encoder, teInstruction=1 caused the TE to record the PC periodically at a rate deter- mined by teSyncMaxInst. With Event Trace, backward compatibility with that behavior is maintained, but additional event types are now possible and it is now possible to turn off the periodic PC recording by setting teSyncMaxInst to the maximum (15).

Variations of the In-Circuit Trace (ICT) Nexus message are used for Event Trace activity. Some variants of ICT are used in HTM or ITC trace as noted below. Should internal buffering overflow, the Error message is used (same as HTM trace). Instrumentation Trace (ITC) may be used with Event Trace, but HTM and Event Trace are mutually exclusive and controlled by the value writ- ten to teInstruction.

The ICT-With-Sync message is substituted for ICT when at least 2^(teSyncMaxBTM+5) ICTs have been emitted. The Sync form of the message includes an F-ADDR for the PC and an uncompressed timestamp. ICT-With-Sync message types that have more than one PC use F- ADDR for the first and U-ADDR for the second.

Address compression uses the same XOR algorithm as other Nexus messages. When there are two U-ADDR PCs in a message, they are applied in order:

- First full PC = previous PC ^ CKDATA0.
- Second full PC = First full PC ^ CKDATA1, which is then used as the previous PC for the next message.

> **Note**
>
> Only PC and PCdest are considered "addresses" for the purposes of address compression. Other data types that may be present in CKDATA0 or CKDATA1 are not compressed and do not affect the address compression thread.

No trace is recorded in debug mode. The Control message is sent when trace is enabled or dis- abled, when entering/leaving debug mode, and when entering or leaving reset state. Trace may be turned on or off from watchpoints, external triggers, and writes to teControl, whether initi- ated by code running on the core or by a remote debugger. When trace is enabled, core events that match the event types selected in evControl cause ICT messages.

Instrumented code running on the target may access TE registers to change recording parameters. Typically, such code should disable trace by clearing teTracing using a read-modify-write, set the new mode(s), then reenable trace. This operation could switch from normal HTM to Event Trace or vice-versa, or change HTM or Event Trace parameters.

Table 3-213 Event Trace Message Fields

Event	CKSRC	CKDF	CKDATA0	CKDATA1	Description
Ext Trigger	8	0	PC		Used in HTM
Ext Trigger + ID	8	1	PC	ID	ID indicates the index of the external trigger.
Watchpoint	14	0	PC		Used in HTM
Watchpoint + ID	14	1	PC	ID	ID indicates the trigger number. PC is normally the address of the instruction that causes the watchpoint. In multi-issue cores, this message may report the address of a later instruction retired in the same cycle
Inferable Call	9	0	PC		PC is the address of the Call instruction. Destination determined from program image.
Call / Return	9	1	PC	PCdest	PC is the address of the transfer instruction. PCdest is the destination. Instruction type is determined from program image and may be an uninferable call, function return, exception return, or coroutine swap.
Exception	10	1	PC	cause	PC is what is loaded into xepc, cause is what is loaded into the Exception Code field of xcause.

Event	CKSRC	CKDF	CKDATA0	CKDATA1	Description
Interrupt	11	1	PC	cause	PC is what is loaded into xepc, cause is what is loaded into the Exception Code field of xcause.
Context	13	1	PC	context	mcontext or scontext changed. New value of register is sent along with a 2-bit code in the LSBs indicating which register changed. * context = Concat(scontext, 2'd2) → scontext changed * context = Concat(mcontext, 2'd3) → mcontext changed
Periodic	15	0	PC		Backward-compatible. PC is the address of the last executed instruction.
Control	0	0	control		• 0 = no-event (ITC uses this to insert an uncompressed timestamp)
Event	**CKSRC**	**CKDF**	**CKDATA0**	**CKDATA1**	**Description**
Control + PC	0	1	PC	control	• 2 = trace-on. PC=instruction responsible for trace-on. • 3 = trace-off. PC=instruction responsible for trace-off. • 4 = exit-debug. PC=first instruction to execute after debug mode. • 5 = enter-debug. PC=address loaded into depc (has not executed). • 6 = exit-reset. PC=first instruction to execute after reset. • 7 = enter-reset. PC=last instruction to execute before reset.

Table 3-214 In-Circuit Trace Message

Bits	Name	Description
var	TSTAMP	Timestamp value
var	CKDATA	PC or other field(s), determined by CKSRC and CKDF
2	CKDF	0=One CKDATA field, 1=Two CKDATA fields, 2,3: reserved

Bits	Name	Description
4	CKSRC	Reason for ICT
n	SRC	Source of this message (width is teImpl.nSrcBits)
6	TCODE	Value = 34 (compressed) or 35 (uncompressed)

3.4.1.20 Custom Instructions and CSRs

This chapter describes some of the custom instructions and CSRs configured in the EIC7700X MCPU.

- **EIC7700X MCPU Custom Instructions**

These custom instructions use the SYSTEM instruction encoding space, which is the same as the custom CSR encoding space, but with funct3=0.

- **CFLUSH.D.L1**

• Implemented as state machine in L1 data cache, for cores with data caches.

• Only available in M-mode. Execution from modes other than M-mode raises an illegal- instruction exception.

• Opcode 0xFC000073, with optional rs1 field in bits [19:15].

• When rs1 = x0, CFLUSH.D.L1 writes back and invalidates all lines in the L1 data cache.

• When rs1 ≠ x0, CFLUSH.D.L1 writes back and invalidates the L1 data cache line contain- ing the virtual address in integer register rs1.

• If the effective privilege mode does not have write permissions to the address in rs1, then a store access or store page-fault exception is raised.

• If the address in rs1 is in an uncacheable region with write permissions, the instruction has no effect but raises no exceptions.

• If the address in rs1 is in an uncacheable region, cache line misaligned accesses may raise an access exception.

• Note that if the PMP scheme write-protects only part of a cache line, then using a value for rs1 in the write-protected region will cause an exception, whereas using a value for rs1 in the write-permitted region will write back the entire cache line.

- **CDISCARD.D.L1**

• Implemented as state machine in L1 data cache, for cores with data caches.

• Only available in M-mode. Execution from modes other than M-mode raises an illegal- instruction exception.

• Opcode 0xFC200073, with optional rs1 field in bits [19:15].

• When rs1 = x0, CDISCARD.D.L1 invalidates, but does not write back, all lines in the L1 data cache. Dirty data within the cache is lost.

• When rs1 ≠ x0, CDISCARD.D.L1 invalidates, but does not write back, the L1 data cache line containing the virtual address in integer register rs1. Dirty data within the cache line is lost.

• If the effective privilege mode does not have write permissions to the address in rs1, then a store access or store page-fault exception is raised.

• If the address in rs1 is in an uncacheable region with write permissions, the instruction has no effect but raises no exceptions.

• If the address in rs1 is in an uncacheable region, cache line misaligned accesses may raise an access exception.

• Note that if the PMP scheme write-protects only part of a cache line, then using a value for rs1 in the write-protected region will cause an exception, whereas using a value for rs1 in the write-permitted region will invalidate and discard the entire cache line.

- **CEASE**

• Privileged instruction only available in M-mode.

• Opcode 0x30500073.

• After retiring CEASE, hart will not retire another instruction until reset.

• Instigates power-down sequence, which will eventually raise the cease_from_tile_N sig- nal to the outside of the Core Complex, indicating that it is safe to power down.

• CEASE has no effect on System Bus Access.

• Debug haltreq will not work after a CEASE instruction has retired.

- **PAUSE**

• Opcode 0x0100000F, which is a FENCE instruction with predecessor set W and null succes- sor set. Therefore, PAUSE is a HINT instruction that executes as a no-op on all RISC-V implementations.

• This instruction may be used for more efficient idling in spin-wait loops.

• This instruction causes a stall of up to 32 cycles or until a cache eviction occurs, whichever comes first.

- **EIC7700X MCPU Custom CSRs**

These custom CSRs use the custom CSR encoding space. Refer to Appendix B for an exhaus- tive list of the custom CSRs available on the EIC7700X MCPU.

- **Branch Prediction Mode CSR**

This EIC7700X MCPU custom extension adds an M-mode CSR to control the current branch prediction mode, mbpm at CSR 0x7C0.

The EIC7700X MCPU's branch prediction system includes a Return Address Stack (RAS), a Branch Target Buffer (BTB), and a Branch History Table (BHT). While branch predictors are essential to achieve high performance in pipelined processors, they can also cause undesirable timing variability for hard real-time systems. The mbpm register provides a means to customize the branch predictor behavior to trade average performance for a more predictable execution time.

Table 3-215 Branch Prediction Mode CSR

Branch Prediction Mode CSR				
CSR	**0x7C0**			
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	static	RW	0x0	Branch-Direction Prediction
[7:1]	Reserved	RO	0x0	

The static bit determines the value returned by the BHT component of the branch prediction system. A zero value indicates dynamic direction prediction, and a non-zero value indicates sta- tic-taken direction prediction. The BTB is cleared on any write to static, and the RAS is unaf- fected by writes to static.

- **EIC7700X MCPU Feature Disable CSRs**

The EIC7700X MCPU custom M-mode Feature Disable CSRs are provided to enable or disable certain microarchitectural features. In the EIC7700X MCPU, CSRs 0x7C1 and 0x7C2 have been allocated for this purpose. These features are described in Table *EIC7700X MCPU Feature Disable 0 CSR* and Table *EIC7700X MCPU Feature Disable 1 CSR*.

A feature is fully enabled when the associated bit is zero. If a particular core does not support the disabling of a feature, the corresponding bit is hardwired to zero.

On reset, all implemented bits are set to 1, disabling all features. The bootloader is responsible for turning on all required features and can simply write zero to turn on the maximal set of fea- tures. EIC7700X MCPU's Freedom Metal bootloader handles turning on these features; when using a cus- tom bootloader, clearing the Feature Disable CSRs must be implemented.

Note that arbitrary toggling of the Feature Disable CSR bits is neither recommended nor sup- ported; they are only intended to be set from 1 to 0. A particular Feature Disable CSR bit is only to be used in a very limited number of situations, as detailed in the Example Usage entry in Table *EIC7700X MCPU Feature Disable CSR 0 Usage*.

Table 3-216 EIC7700X MCPU Feature Disable 0 CSR

EIC7700X MCPU Feature Disable 0 CSR				
CSR	**0x7C1**			
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	disableDCacheClockGate	RW	0x1	Disable data cache clock gating
1	disableICacheClockGate	RW	0x1	Disable instruction cache clock gating
2	disableCoreClockGate	RW	0x1	Disable core clock gating
3	disableSpeculativeICacheRefill	RW	0x1	Disable speculative instruction cache refill
4	disableIEXClockGate	RW	0x1	Disable integer pipeline clock gating
5	disableFEXClockGate	RW	0x1	Disable floating-point pipeline clock gating
6	disableVPUClockGate	RO	0x0	Disable VPU clock gating
7	disableTileClockGate	RW	0x1	Disable tile clock gating
8	Reserved	RO	0x0	
9	suppressCorruptOnGrantData	RW	0x1	Suppress corrupt signal on GrantData messages
10	ifuDisablePrefetchFillMissQ	RW	0x0	Disable allowing instruction prefetches to fill miss queue
11	ifuDisableMissQMulti	RW	0x0	Only allow a single outstanding instruction cache miss
12	ifuDisableICachePrefetch	RW	0x0	Disable instruction cache prefetcher

EIC7700X MCPU Feature Disable 0 CSR				
CSR	0x7C1			
Bits	Field Name	Attr.	Rst.	Description
13	ifuDisableRAS	RW	0x0	Disable Return-Address Stack
14	ifuDisableIJTP	RW	0x0	Disable Indirect-Jump Target Predictor
15	ifuDisableNLP	RW	0x0	Disable next-line predictor
16	ifuDisableIJTPHint	RW	0x0	Disable IJTP hint power optimization
17	ifuDisableSuperscalar	RW	0x0	Only decode up to one instruction per cycle
18	ifuDisableIQBypass	RW	0x0	Disable issue queue bypass
19	ifuDisableRASFwd	RW	0x0	Disable RAS forwarding to next-line predictor
20	ifuDisableICTagSkip	RW	0x0	Re-read instruction cache tag array on every fetch
21	Reserved	RO	0x0	
22	ifuDisableTaggedTables	RW	0x0	Disable tagged tables of branch direction predictor
23	ifuDisablePredSkip	RW	0x0	Disable prediction skip
24	ifuDisableNlpNoBrOptimization	RW	0x0	Disable BDP whenever the next-line predictor indicates the target of current fetch group has no conditional branch
[29:25]	Reserved	RO	0x0	
30	rduDisableMDPCountDecay	RW	0x0	Disable aging out MDP usefulness counters
31	rduDisableMDPNegTrainSLd	RW	0x0	Disable MDP training for unnecessarily synchronized single-set loads
32	rduDisableMDPNegTrain	RW	0x0	Disable MDP training for unnecessarily synchronized loads
33	rduDisableMDPHitTrain	RW	0x0	Disable MDP training for MDP-predicted dependencies
34	rduDisableMDP	RW	0x0	Disable training and lookup of memory dependence predictor (MDP)
35	rduDisableBrqEarlyUpd	RW	0x0	Disable updates of branch direction predictor before retirement
36	rduDisablePCMTOptimism	RW	0x0	Assume all instructions may require PCMT allocaiton. Potential slight performance impact

EIC7700X MCPU Feature Disable 0 CSR				
CSR	0x7C1			
Bits	Field Name	Attr.	Rst.	Description
37	rduDisableMemGrouping	RW	0x0	All memory-type instructions will start and end an ROB group
38	rduDisableFPGrouping	RW	0x0	All floating-point instructions will start and end an ROB group
39	rduDisableBranchGrouping	RW	0x0	All branch and jump instructions will start and end an ROB group
40	rduDisableROBGrouping	RW	0x0	Disable all ROB grouping
41	rduDisableCrossIssue	RW	0x0	Disable cross-issue feature
42	rduDisableMoveElim	RW	0x0	Disable move elimination
43	rduForceSingleStep	RW	0x0	Force single-stepping behavior
44	rduDisableCmov	RW	0x0	Disable conditional move conversion
45	rduDisableFusion	RW	0x0	Disable instruction fusion
[49:46]	Reserved	RO	0x0	
50	iexDisableEarlyIssQRelease	RW	0x0	Disable early issue queue release for integer pipeline
51	iexDisableEarlyDiv	RW	0x0	Disable early issue of new divide while current divide execution is finishing for integer pipeline
[56:52]	Reserved	RO	0x0	
57	fexDisableEarlyIssQRelease	RW	0x0	Disable early issue queue release for floating-point pipeline
[63:58]	Reserved	RO	0x0	

Table 3-217 EIC7700X MCPU Feature Disable CSR 0 Usage

Feature Disable CSR 0 Usage	
Bit	Description / Usage
3	Disable speculative instruction cache refill Example Usage A particular integration might require that execution from the System Port range be disallowed. Startup code would first configure PMP to prevent execution from the System Port range, followed by clearing bit 3 of the Feature Disable CSR 0. This would enable speculative instruction cache refill accesses, without allowing those to access the System Port range because PMP would prohibit such accesses.

Feature Disable CSR 0 Usage	
Bit	**Description / Usage**
9	Suppress corrupt signal on GrantData messages Example Usage 1 When running in debug mode on configurations having both ECC and a BEU, setting bit 9 of the Feature Disable CSR 0 will suppress debug mode errors. Example Usage 2 Startup code could scrub errors present in RAMs at power-on, followed by clearing bit 9 of the Feature Disable CSR 0 to allow normal operation.

Table 3-218 EIC7700X MCPU Feature Disable 1 CSR

EIC7700X MCPU Feature Disable 1 CSR				
CSR	**0x7C2**			
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	disableTileLinkCorruptOrDeniedErrors	RW	0x0	Disable reporting L2 corrupt or denied errors
1	disableL1dcDataArrayUncorrectableErrors	RW	0x0	Disable reporting L1 data chace data array uncorrectable ECC errors
2	disableL1dcDataArrayCorrectableErrors	RW	0x0	Disable reporting l1 data cache data array correctable ECC errors
3	disableL1dcTagArrayUncorrectableErrors	RW	0x0	Disable reporting L1 data cache tag array uncorrectable ECC errors
4	disableL1dcTagArrayCorrectableErrors	RW	0x0	Disable reporting L1 data cache tag array correctable ECC errors
5	forceSFenceAllAsid	RW	0x0	Force SFENCE to target all ASIDs
6	forceSFenceAllAddr	RW	0x0	Force SFENCE to target all addresses
7	forceTLReleaseWhenEvictClean	RW	0x0	Force Noisy Evict to send Release message from any Valid coherence permission state
8	forceTLReleaseInvalidate	RW	0x0	Force Evict Release to always downgrade coherence permission to Invalidate

EIC7700X MCPU Feature Disable 1 CSR				
CSR	0x7C2			
Bits	Field Name	Attr.	Rst.	Description
9	forceTLProbeInvalidate	RW	0x0	Force Snoop Probe to always downgrade coherence permission to Invalidate
10	Reserved	RO	0x0	
11	disableTLAcquirePerm	RW	0x0	Disable requesting L2 AcquirePerm.toT
12	Reserved	RO	0x0	
13	disablePipeliningMMIO	RW	0x0	Disable pipelining multiple MMIO reads/writes
14	forceFenceFullMatch	RW	0x0	Force all FENCEs upgrade to RWIO, and all AMO/LR/SC to SC=AQ+RL
15	disableMshrAgeTimers	RW	0x0	Disable MSHR Cleaning timer and Keeper threshold
16	forceL2ReadWaitForCommit	RW	0x0	Disable speculative L2 accesses
17	disableL2FastRequest	RW	0x0	Disable L2 request fast-path
18	Reserved	RO	0x0	
19	disableMshrReuse	RW	0x0	Disable MSHR resuse by multiple accesses
20	forceMshrFullDependency	RW	0x0	Sequential Consistency Memory Ordering
21	disableFastStldfSideBySide	RW	0x0	Disable Store-to-Load Forwarding from adjacent stage (LSTM+SDFP to LSTM+LDR)
22	disableFastStldfBackToBack	RW	0x0	Disable Store-to-Load Forwarding from stage ahead (LSTO+SDO to LSTM+LDR)
23	disableFastStldf	RW	0x0	Disable Store-to-Load Forwarding to LSTM+LDR
24	disableStldf	RW	0x0	Disable Store-to-Load Forwarding
25	disableStoreQWriteGathering	RW	0x0	Disable store queue write gathering
26	disableLoadDataReturnForPotentialRaWHazard	RW	0x0	Disable load data return for potential read-after-write hazard

EIC7700X MCPU Feature Disable 1 CSR				
CSR	0x7C2			
Bits	Field Name	Attr.	Rst.	Description
27	disableLoadDataReturnForPotentialRaRHazard	RW	0x0	Disable load data return for potential read-after-read hazard
28	disableLoadDataReturnForRaRHazard	RW	0x0	Disable load data return for known read-after-read hazard
29	disableFastLoadQRequestDataReturn	RW	0x0	Disable load queue request data return fast-path
30	disableFastL2DataReturn	RW	0x0	Disable L2 data return fast-path
31	disableFastNonSpecDataReturn	RW	0x0	Force non-speculative accesses (AMO/LR/SC) to return data from the load queue
32	disableFastLoadDataReturn	RW	0x0	Force loads to return data from the load queue
33	disableFastLoadResolve	RW	0x0	Force loads to resolve from the load queue
34	forceLoadResolveWaitDataReturn	RW	0x0	Force loads to wait to resolve until returning data, to enable precise error reporting on loads
35	forceHazardBetweenAllLoadsAndStores	RW	0x0	Force hazards to conservatively match between all loads and stores
36	disablePtwFlushWait	RW	0x0	Disable PTW waiting after flushing newer instructions
37	disableFastLoadReturnPredictor	RW	0x0	Disable fast load return predictor
[39:38]	Reserved	RO	0x0	
40	disableFastLoadForward	RW	0x0	Disable fast load forwarding from LDF stage
41	Reserved	RO	0x0	
42	forceSelfReplayToFlushNewer	RW	0x0	Force self-replay operation to flush newer operations
43	forceSelfReplayToFlushSelf	RW	0x0	Force self-replay operation to flush itself and newer operations
44	forceAmoToL2Cache	RW	0x0	Force AMOs to the L2 cache

EIC7700X MCPU Feature Disable 1 CSR				
CSR	0x7C2			
Bits	Field Name	Attr.	Rst.	Description
45	forceL1DataCacheDirectMapped	RW	0x0	Force L1 data cache allocation to way 0
46	forceNoAllocateL1DataCache	RW	0x0	Force non-allocating L1 data cache, except for AMO/LR/SC
47	sdIssForceWaitAddrNonReplay	RW	0x0	Force store data issue to wait until store address will not replay
48	Reserved	RO	0x0	
49	lsIssForceStackLoadsAfterStores	RW	0x0	Force loads from the stack to wait before issue for all older stores to the stack
50	lsIssForceInOrderWaitNonReplay	RW	0x0	Force in-order operations to wait before issuing until all older memory operations will not replay
51	Reserved	RO	0x0	
52	lsIssForceInOrderWaitNonSpec	RW	0x0	Force in-order operations to wait before issuing until all older operations are commited and retired
53	lsIssForceAllInOrder	RW	0x0	Force all memory operations to issue in-order
54	forceSingleIssueToLSU	RW	0x0	Force all memory operations to single issue
55	storeL1AllocVec	RW	0x0	L1 allocate for vector store
56	storeL1AllocFloat	RW	0x0	L1 allocate for floating-point store
57	storeL1AllocInt	RW	0x0	L1 allocate for scalar integer store
58	storeL1AllocStack	RW	0x0	L1 allocate for stores to the stack
[63:59]	Reserved	RO	0x0	

- **Hardware Prefetcher CSRs**

The Hardware Prefetcher CSRs, at 0x7C3 and 0x7C4, are used to configure the Hardware Prefetcher.

- **Power Dial CSR**

The Power Dial CSR, at 0x7C8, provides a method of scaling down dynamic power in a core in order to limit maximum power without frequency changes.

3.4.2 Co-processor

3.4.2.1 Overview

EIC7700X has two co-processors, both located in the aon subsystem, one for security boot(SCPU) and one for low Power control (LPCPU). The two Co-processors have same features, they are 32-bit RISC-V processors which is compatible with all applicable RISC-V standards.

Please get in touch with ESWIN to request more detailed technical information.

<ibu_techsupport@eswincomputing.com>

3.5 Interrupt System

Table 3-219 RISC-V mcpu interrupt list

Interrupt source	Interrupt ID	Interrupt name	polarity
DDR0	5	ddr0_pmp0_intr	High level
	6	ddr0_pmp1_intr	High level
	7	ddr0_pmp2_intr	High level
	8	ddr0_pmp3_intr	High level
	9	ddr0_pmp4_intr	High level
	10	ddr0_ecc_intr	High level
	11	ddr0_derate_intr	High level
	12	ddr0_poison_intr	High level
	13	ddr0_phy_intr	High level
GPU	14	gpu_misc_intr	High level
	15	gpu_core_intr0	High level
NPU	16	npu_intr	High level
	17	npu_tbu_ras_intr	High level
	18	npu_tbu_pmu_intr	High level
Video IN	19	isp0_mi_intr	High level
	20	isp0_fe_intr	High level
	21	isp0_intr	High level
	22	isp1_mi_intr	High level
	23	isp1_fe_intr	High level
	24	isp1_intr	High level
	25	dwe_vse_intr	High level
	26	dwe_intr	High level
	27	dwe_fe_intr	High level
	28	vi_cmn_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	29	mipi_csi0_intr	High level
	30	mipi_csi1_intr	High level
	31	mipi_csi2_intr	High level
	32	mipi_csi3_intr	High level
	33	mipi_csi4_intr	High level
	34	mipi_csi5_intr	High level
	35	vicap0_u84_intr	High level
	36	vicap1_u84_intr	High level
	37	vicap2_u84_intr	High level
	38	vicap3_u84_intr	High level
	39	vicap4_u84_intr	High level
	40	vicap5_u84_intr	High level
DSP	41	dspt_intr0	High level
	42	dspt_intr1	High level
	43	dspt_intr2	High level
	44	dspt_intr3	High level
	45	dspt_PFatalErr0	High level
	46	dspt_PFatalErr1	High level
	47	dspt_PFatalErr2	High level
	48	dspt_PFatalErr3	High level
	49	dspt_g2d0_intr	High level
	50	dspt_g2d1_intr	High level
	51	dspt_g2d0_abn_intr	High level
	52	dspt_g2d1_abn_intr	High level
	43~56	Reserved	High level
DMA	57	hsp_dma0_u84_intr	High level
SATA	58	sata_intrq	High level
	59	sata_msi_req	High level
	60	sata_pme_req	High level
GMAC	61	eth0_sbd_intr	High level
	62	eth0_sbd_perch_tx_intr0	High level
	63	eth0_sbd_perch_tx_intr1	High level
	64	eth0_sbd_perch_tx_intr2	High level
	65	eth0_sbd_perch_tx_intr3	High level
	66	eth0_sbd_perch_rx_intr0	High level
	67	eth0_sbd_perch_rx_intr1	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	68	eth0_sbd_perch_rx_intr2	High level
	69	eth0_sbd_perch_rx_intr3	High level
	70	eth1_sbd_intr	High level
	71	eth1_sbd_perch_tx_intr0	High level
	72	eth1_sbd_perch_tx_intr1	High level
	73	eth1_sbd_perch_tx_intr2	High level
	74	eth1_sbd_perch_tx_intr3	High level
	75	eth1_sbd_perch_rx_intr0	High level
	76	eth1_sbd_perch_rx_intr1	High level
	77	eth1_sbd_perch_rx_intr2	High level
	78	eth1_sbd_perch_rx_intr3	High level
EMMC	79	emmc0_intr	High level
	80	emmc0_wakeup_intr	High level
SDIO	81	sd0_intr	High level
	82	sd0_wakeup_intr	High level
	83	sd1_intr	High level
	84	sd1_wakeup_intr	High level
USB	85	usb0_intr	High level
	86	usb1_intr	High level
WatchDog	87	lsp_wdt0_intr	High level
	88	lsp_wdt1_intr	High level
	89	lsp_wdt2_intr	High level
	90	lsp_wdt3_intr	High level
SPI	91	lsp_ssi0_intr	High level
	92	lsp_ssi1_intr	High level
PWM	93	lsp_timer_intr0	High level
	94	lsp_timer_intr1	High level
	95	lsp_timer_intr2	High level
	96~99	Reserved	High level
UART	100	lsp_uart0_intr	High level
	101	lsp_uart1_intr	High level
	102	lsp_uart2_intr	High level
	103	lsp_uart3_intr	High level
	104	lsp_uart4_intr	High level
I2C	105	lsp_i2c0_intr	High level
	106	lsp_i2c1_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	107	lsp_i2c2_intr	High level
	108	lsp_i2c3_intr	High level
	109	lsp_i2c4_intr	High level
	110	lsp_i2c5_intr	High level
	111	lsp_i2c6_intr	High level
	112	lsp_i2c7_intr	High level
	113	lsp_i2c8_intr	High level
	114	lsp_i2c9_intr	High level
	115	Reserved	High level
Mailbox	116	u84_mbox0_intr	High level
	117	u84_mbox1_intr	High level
	118	u84_mbox2_intr	High level
	119	u84_mbox3_intr	High level
	120	u84_mbox4_intr	High level
	121	u84_mbox5_intr	High level
	122	u84_mbox6_intr	High level
	123	u84_mbox7_intr	High level
	124	u84_mbox8_intr	High level
	125	u84_mbox9_intr	High level
	126	u84_mbox10_intr	High level
	127	u84_mbox11_intr	High level
	128	u84_mbox12_intr	High level
	129	u84_mbox13_intr	High level
	130	u84_mbox14_intr	High level
	131	u84_mbox15_intr	High level
PCIe	132	pcie_dtim_err_tresp_oas_int	High level
	133	pcie_dtim_err_tresp_uid_int	High level
	134	pcie_dtim_err_icpl_uc_int	High level
	135	pcie_dtim_err_ireq_opertn_int	High level
	136	pcie_dtim_err_ireq_to_int	High level
	137	pcie_dtim_int	High level
	138	pcie_dtim_ireq_timeout_int	High level
	139~159	Reserved	High level
	160	pcie_hp_pme	High level
	161	pcie_hp_int	High level
	162	pcie_hp_msi	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	163	pcie_edma_int0	High level
	164	pcie_edma_int1	High level
	165	pcie_edma_int2	High level
	166	pcie_edma_int3	High level
	167	pcie_edma_int4	High level
	168	pcie_edma_int5	High level
	169	pcie_edma_int6	High level
	170	pcie_edma_int7	High level
	171	pcie_edma_int8	High level
	172	pcie_edma_int9	High level
	173	pcie_edma_int10	High level
	174	pcie_edma_int11	High level
	175	pcie_edma_int12	High level
	176	pcie_edma_int13	High level
	177	pcie_edma_int14	High level
	178	pcie_edma_int15	High level
	179	pcie_assert_inta	High level
	180	pcie_assert_intb	High level
	181	pcie_assert_intc	High level
	182	pcie_assert_intd	High level
	183	pcie_deassert_inta	High level
	184	pcie_deassert_intb	High level
	185	pcie_deassert_intc	High level
	186	pcie_deassert_intd	High level
	187	pcie_usp_eq_redo_executed_int	High level
	188~219	Reserved	High level
	220	pcie_msi_ctrl_int	High level
Video codec	221	vd_tbu_ras_intr	High level
	222	vd_tbu_pmu_intr	High level
	223	jd_tbu_ras_intr	High level
	224	jd_tbu_pmu_intr	High level
	225	ve_tbu_ras_intr	High level
	226	ve_tbu_pmu_intr	High level
	227	je_tbu_ras_intr	High level
	228	je_tbu_pmu_intr	High level
	229	ve_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	230	ve_intr_norm	High level
	231	ve_intr_abn	High level
	232	je_intr	High level
	233	je_intr_norm	High level
	234	je_intr_abn	High level
	235	Reserved	High level
	236	vd_intr_vcd	High level
	237	jd_intr	High level
OSD	238	osd_intr	High level
	239~270	Reserved	High level
I2S	271	vo_i2s0_intr	High level
	272	vo_i2s1_intr	High level
	273	vo_i2s2_intr	High level
HDMI	274	hdmi_intr	High level
	275	hdcp_intr	High level
MIPI_DSI	276	dsi_intr	High level
MMU	277	tbu0_ras_intr	High level
	278	tbu0_pmu_intr	High level
	279	tbu2_ras_intr	High level
	280	tbu2_pmu_intr	High level
	281	tbu3_ras_intr	High level
	282	tbu3_pmu_intr	High level
	283	tbu4_ras_intr	High level
	284	tbu4_pmu_intr	High level
	285	Reserved	High level
	286	Reserved	High level
D2D	287	d2d_ctrl_int	High level
	288	d2d_ctrl_int_2nd	High level
TOP	289	aon_dma_intr	High level
	290	aon_i2c0_intr	High level
	291	aon_i2c1_intr	High level
	292	rtc_intr	High level
	293	aon_spic_intr	High level
DDR1	294	ddr1_pmp0_intr	High level
	295	ddr1_pmp1_intr	High level
	296	ddr1_pmp2_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	297	ddr1_pmp3_intr	High level
	298	ddr1_pmp4_intr	High level
	299	ddr1_ecc_intr	High level
	300	ddr1_derate_intr	High level
	301	ddr1_poison_intr	High level
	302	ddr1_phy_intr	High level
GPIO	303	aon_u84_gpio_intr0	High level
	304	aon_u84_gpio_intr1	High level
	305	aon_u84_gpio_intr2	High level
	306	aon_u84_gpio_intr3	High level
	307	aon_u84_gpio_intr4	High level
	308	aon_u84_gpio_intr5	High level
	309	aon_u84_gpio_intr6	High level
	310	aon_u84_gpio_intr7	High level
	311	clmm_u84_gpio_intr0	High level
	312	clmm_u84_gpio_intr1	High level
	313	clmm_u84_gpio_intr2	High level
	314	clmm_u84_gpio_intr3	High level
	315	clmm_u84_gpio_intr4	High level
	316	clmm_u84_gpio_intr5	High level
	317	clmm_u84_gpio_intr6	High level
	318	clmm_u84_gpio_intr7	High level
	319	clmm_u84_gpio_intr8	High level
	320	clmm_u84_gpio_intr9	High level
	321	clmm_u84_gpio_intr10	High level
	322	clmm_u84_gpio_intr11	High level
	323	clmm_u84_gpio_intr12	High level
	324	clmm_u84_gpio_intr13	High level
	325	clmm_u84_gpio_intr14	High level
	326	clmm_u84_gpio_intr15	High level
	327	clmm_u84_gpio_intr16	High level
	328	clmm_u84_gpio_intr17	High level
	329	clmm_u84_gpio_intr18	High level
	330	clmm_u84_gpio_intr19	High level
	331	clmm_u84_gpio_intr20	High level
	332	clmm_u84_gpio_intr21	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	333	clmm_u84_gpio_intr22	High level
	334	clmm_u84_gpio_intr23	High level
NPU	335	npu_u84_llc0_c_intr	High level
	236	npu_u84_llc0_event_intr	High level
	237	npu_u84_llc0_uc_intr	High level
	238	npu_u84_llc1_c_intr	High level
	239	npu_u84_llc1_event_intr	High level
	240	npu_u84_llc1_uc_intr	High level
	341~342	Reserved	High level
Video In	343	shutter_u84_intr	High level
	344	Reserved	High level
TOP	345	aon_u84_timers_intr0	High level
	346	aon_u84_timers_intr1	High level
	347	aon_u84_timers_intr2	High level
	348	aon_u84_timers_intr3	High level
PVT	349	lsp_u84_pvt_intr0	High level
DDR0	350	lpddr_u84_pvt_intr	High level
	351~353	Reserved	High level
FAN_CTRL	354	lsp_u84_fanctrl_intr	High level
MMU	355	tcu_u84_event_q_irpt_s	High level
	356	tcu_u84_event_q_irpt_ns	High level
	357	tcu_u84_pri_q_irpt_ns	High level
	358	tcu_u84_cmd_sync_irpt_ns	High level
	359	tcu_u84_cmd_sync_irpt_s	High level
	360	tcu_u84_global_irpt_ns	High level
	361	tcu_u84_global_irpt_s	High level
	362	tcu_u84_ras_irpt	High level
	363	tcu_u84_pmu_irpt	High level
	359~367	Reserved	High level
Video in	368	dvp2axi_u84_intr0	High level
	369	dvp2axi_u84_intr1	High level
	370	dvp2axi_u84_intr2	High level
	371	dvp2axi_u84_intr3	High level
	372	dvp2axi_u84_intr4	High level
	373	dvp2axi_u84_intr5	High level
	374	dvp2axi_u84_intr6	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	375	dvp2axi_u84_intr7	High level
PCIe	376	pciet_u84_hot_reset_int	High level
	377	pciet_u84_flr_int	High level
VO	378	vo_u84_cec_intr	High level
DSP	379	dsp0_tbu0_u84_ras_intr	High level
	380	dsp0_tbu0_u84_pmu_intr	High level
	381	dsp1_tbu0_u84_ras_intr	High level
	382	dsp1_tbu0_u84_pmu_intr	High level
	383	Reserved	
	384	Reserved	
	385	Reserved	
	386	Reserved	
NPU	387	npu_u84_nvdla_intr	High level
PCIe	388	pciet_u84_radm_fatal_err_int	High level
	389	pciet_u84_radm_nonfatal_err_int	High level
	390	pciet_u84_radm_correctable_err_int	High level
	391	Reserved	High level
NOC	392	cnoc_vo_timeout	High level
	393	cnoc_vi_timeout	High level
	394	cnoc_vc_timeout	High level
	395	cnoc_tcu_timeout	High level
	396	cnoc_pciet_x_timeout	High level
	397	cnoc_pciet_p_timeout	High level
	398	cnoc_npu_timeout	High level
	399	cnoc_mcput_d2d_timeout	High level
	400	cnoc_lsp_apb6_timeout	High level
	401	cnoc_lsp_apb4_timeout	High level
	402	cnoc_lsp_apb3_timeout	High level
	403	cnoc_lsp_apb2_timeout	High level
	404	cnoc_hsp_timeout	High level
	405	cnoc_gpu_timeout	High level
	406	cnoc_dspt_timeout	High level
	407	cnoc_ddrt1_phy_timeout	High level
	408	cnoc_ddrt1_ctrl_timeout	High level
	409	cnoc_ddrt0_phy_timeout	High level
	410	cnoc_ddrt0_ctrl_timeout	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	411	cnoc_aon_timeout	High level
	412	clmm_timeout	High level
	413	rnoc_ddrt1_p4_timeout	High level
	414	rnoc_ddrt0_p4_timeout	High level
	415	mnoc_ddr1_p3_timeout	High level
	416	mnoc_ddr0_p3_timeout	High level
	417	snoc_pciet_timeout	High level
	418	snoc_npu_timeout	High level
	419	snoc_dspt_timeout	High level
	420	snoc_ddrt1_p2_timeout	High level
	421	snoc_ddrt1_p1_timeout	High level
	422	snoc_ddrt0_p2_timeout	High level
	423	snoc_ddrt0_p1_timeout	High level
	424	snoc_aon_timeout	High level
	425	lnoc_ddrt1_p0_timeout	High level
	426	lnoc_ddrt0_p0_timeout	High level
PCIe	427	pcie_wake_n_in	High level
NOC	428	sys_trans2_statalarm	High level
	429	sys_trans1_statalarm	High level
	430	sys_trans0_statalarm	High level
	431	sys_observer_fault	High level
	432	Reserved	High level
	433	ddrt1_p2_reqpkt_statalarm	High level
	434	Reserved	High level
	435	ddrt1_p1_reqpkt_statalarm	High level
	436	Reserved	High level
	437	ddrt0_p2_reqpkt_statalarm	High level
	438	Reserved	High level
	439	ddrt0_p1_reqpkt_statalarm	High level
	440	llc_trans_statalarm	High level
	441	llc_observer_fault	High level
	442	Reserved	High level
	443	ddrt1_p0_reqpkt_statalarm	High level
	444	Reserved	High level
	445	ddrt0_p0_reqpkt_statalarm	High level
	446	cfg_observer_fault	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	447	realtime_trans_statalarm	High level
	448	realtime_observer_fault	High level
	449	Reserved	High level
	450	ddrt1_p4_reqpkt_statalarm	High level
	451	Reserved	High level
	452	ddrt0_p4_reqpkt_statalarm	High level
	453	media_trans_statalarm	High level
	454	media_observer_fault	High level
	455	Reserved	High level
	456	ddrt1_p3_reqpkt_statalarm	High level
	457	Reserved	High level
	458	ddrt0_p3_reqpkt_statalarm	High level

Note: Interrupt ID 0~4 of mcpu corresponds are reserved.

Table 3-220 SCPU interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
TOP	0	trng_edu_intr	High level
	1	trng_intr	High level
	2	pka_intr	High level
	3	spacc_intr	High level
Maibox	4	e21_0_mbox0_intr	High level
	5	e21_0_mbox1_intr	High level
	6	e21_0_mbox2_intr	High level
	7	e21_0_mbox3_intr	High level
	8	e21_0_mbox4_intr	High level
	9	e21_0_mbox5_intr	High level
	10	e21_0_mbox6_intr	High level
	11	e21_0_mbox7_intr	High level
	12	e21_0_mbox8_intr	High level
	13	e21_0_mbox9_intr	High level
	14	e21_0_mbox10_intr	High level
	15	e21_0_mbox11_intr	High level
	16	e21_0_mbox12_intr	High level
	17	e21_0_mbox13_intr	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	18	e21_0_mbox14_intr	High level
	19	e21_0_mbox15_intr	High level
DDR1	20	ddr1_pmp0_intr	High level
	21	ddr1_pmp1_intr	High level
	22	ddr1_pmp2_intr	High level
	23	ddr1_pmp3_intr	High level
	24	ddr1_pmp4_intr	High level
DDR0	25	ddr0_pmp0_intr	High level
	26	ddr0_pmp1_intr	High level
	27	ddr0_pmp2_intr	High level
	28	ddr0_pmp3_intr	High level
	29	ddr0_pmp4_intr	High level
TOP	30	bootspi_int	High level
Timer	31	aon_timer0_intr_flag[0]	High level
	32	aon_timer0_intr_flag[1]	High level
	33	aon_timer0_intr_flag[2]	High level
	34	aon_timer0_intr_flag[3]	High level
	35	aon_timer0_intr_flag[4]	High level
	36	aon_timer0_intr_flag[5]	High level
	37	aon_timer0_intr_flag[6]	High level
	38	aon_timer0_intr_flag[7]	High level
	39	aon_timer1_intr_flag[0]	High level
	40	aon_timer1_intr_flag[1]	High level
	41	aon_timer1_intr_flag[2]	High level
	42	aon_timer1_intr_flag[3]	High level
	43	aon_timer1_intr_flag[4]	High level
	44	aon_timer1_intr_flag[5]	High level
	45	aon_timer1_intr_flag[6]	High level
	46	aon_timer1_intr_flag[7]	High level
	47	aon_timer2_intr_flag[0]	High level
	48	aon_timer2_intr_flag[1]	High level
	49	aon_timer2_intr_flag[2]	High level
	50	aon_timer2_intr_flag[3]	High level
	51	aon_timer2_intr_flag[4]	High level
	52	aon_timer2_intr_flag[5]	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	53	aon_timer2_intr_flag[6]	High level
	54	aon_timer2_intr_flag[7]	High level
	55	aon_timer3_intr_flag[0]	High level
	56	aon_timer3_intr_flag[1]	High level
	57	aon_timer3_intr_flag[2]	High level
	58	aon_timer3_intr_flag[3]	High level
	59	aon_timer3_intr_flag[4]	High level
	60	aon_timer3_intr_flag[5]	High level
	61	aon_timer3_intr_flag[6]	High level
	62	aon_timer3_intr_flag[7]	High level
TOP	63	otp_int	High level
	64	hblock_accvio	High level
	65	edu_vtrng_irq	High level
	66	scpu_rtc_intr	High level

Table 3-221 LPCPU interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
TOP	0	bootspi_int	High level
	1	rtc_int	High level
	2	aon_i2c1_intr	High level
	3	aon_i2c0_intr	High level
	4	aon_dma_intr	High level
Maibox	5	lpcpu_mbox0_intr	High level
	6	lpcpu_mbox1_intr	High level
	7	lpcpu_mbox2_intr	High level
	8	lpcpu_mbox3_intr	High level
	9	lpcpu_mbox4_intr	High level
	10	lpcpu_mbox5_intr	High level
	11	lpcpu_mbox6_intr	High level
	12	lpcpu_mbox7_intr	High level
	13	lpcpu_mbox8_intr	High level
	14	lpcpu_mbox9_intr	High level
	15	lpcpu_mbox10_intr	High level
	16	lpcpu_mbox11_intr	High level
	17	lpcpu_mbox12_intr	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	18	lpcpu_mbox13_intr	High level
	19	lpcpu_mbox14_intr	High level
	20	lpcpu_mbox15_intr	High level
	21	Reserved	High level
Timer	22	aon_timer0_intr_flag[0]	High level
	23	aon_timer0_intr_flag[1]	High level
	24	aon_timer0_intr_flag[2]	High level
	25	aon_timer0_intr_flag[3]	High level
	26	aon_timer0_intr_flag[4]	High level
	27	aon_timer0_intr_flag[5]	High level
	28	aon_timer0_intr_flag[6]	High level
	29	aon_timer0_intr_flag[7]	High level
	30	aon_timer1_intr_flag[0]	High level
	31	aon_timer1_intr_flag[1]	High level
	32	aon_timer1_intr_flag[2]	High level
	33	aon_timer1_intr_flag[3]	High level
	34	aon_timer1_intr_flag[4]	High level
	35	aon_timer1_intr_flag[5]	High level
	36	aon_timer1_intr_flag[6]	High level
	37	aon_timer1_intr_flag[7]	High level
	38	aon_timer2_intr_flag[0]	High level
	39	aon_timer2_intr_flag[1]	High level
	40	aon_timer2_intr_flag[2]	High level
	41	aon_timer2_intr_flag[3]	High level
	42	aon_timer2_intr_flag[4]	High level
	43	aon_timer2_intr_flag[5]	High level
	44	aon_timer2_intr_flag[6]	High level
	45	aon_timer2_intr_flag[7]	High level
	46	aon_timer3_intr_flag[0]	High level
	47	aon_timer3_intr_flag[1]	High level
	48	aon_timer3_intr_flag[2]	High level
	49	aon_timer3_intr_flag[3]	High level
	50	aon_timer3_intr_flag[4]	High level
	51	aon_timer3_intr_flag[5]	High level
	52	aon_timer3_intr_flag[6]	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	53	aon_timer3_intr_flag[7]	High level
GPIO	54	lpcpu_gpio_intr0	High level
	55	lpcpu_gpio_intr1	High level
	56	lpcpu_gpio_intr2	High level
	57	lpcpu_gpio_intr3	High level
	58	lpcpu_gpio_intr4	High level
	59	lpcpu_gpio_intr5	High level
	60	lpcpu_gpio_intr6	High level
	61	lpcpu_gpio_intr7	High level

Table 3-222 DSP0 interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
MAILBOX	0	dsp_0_mbox0_intr	High level
	1	dsp_0_mbox1_intr	High level
	2	dsp_0_mbox2_intr	High level
	3	dsp_0_mbox3_intr	High level
	4	dsp_0_mbox4_intr	High level
	5	dsp_0_mbox5_intr	High level
	6	dsp_0_mbox6_intr	High level
	7	dsp_0_mbox7_intr	High level
	8	dsp_0_mbox8_intr	High level
	9	dsp_0_mbox9_intr	High level
	10	dsp_0_mbox10_intr	High level
	11	dsp_0_mbox11_intr	High level
	12	dsp_0_mbox12_intr	High level
	13	dsp_0_mbox13_intr	High level
	14	dsp_0_mbox14_intr	High level
	15	dsp_0_mbox15_intr	High level

Table 3-223 DSP1 interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
MAILBOX	0	dsp_1_mbox0_intr	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	1	dsp_1_mbox1_intr	High level
	2	dsp_1_mbox2_intr	High level
	3	dsp_1_mbox3_intr	High level
	4	dsp_1_mbox4_intr	High level
	5	dsp_1_mbox5_intr	High level
	6	dsp_1_mbox6_intr	High level
	7	dsp_1_mbox7_intr	High level
	8	dsp_1_mbox8_intr	High level
	9	dsp_1_mbox9_intr	High level
	10	dsp_1_mbox10_intr	High level
	11	dsp_1_mbox11_intr	High level
	12	dsp_1_mbox12_intr	High level
	13	dsp_1_mbox13_intr	High level
	14	dsp_1_mbox14_intr	High level
	15	dsp_1_mbox15_intr	High level

3.6 System address mapping

The total physical address space of EIC7700X system is 2T. The following is the address space mapping diagram, from left to right, which is the global addr map, system space, and config space. The internal SRAM and TIM in the system are mapped in the low address space within 4G and the high address space above 4G. The Mem space is divided into consistent space and inconsistent space. The memory port address segment corresponds to the consistent address space, and the system port1 address segment corresponds to the inconsistent address space.

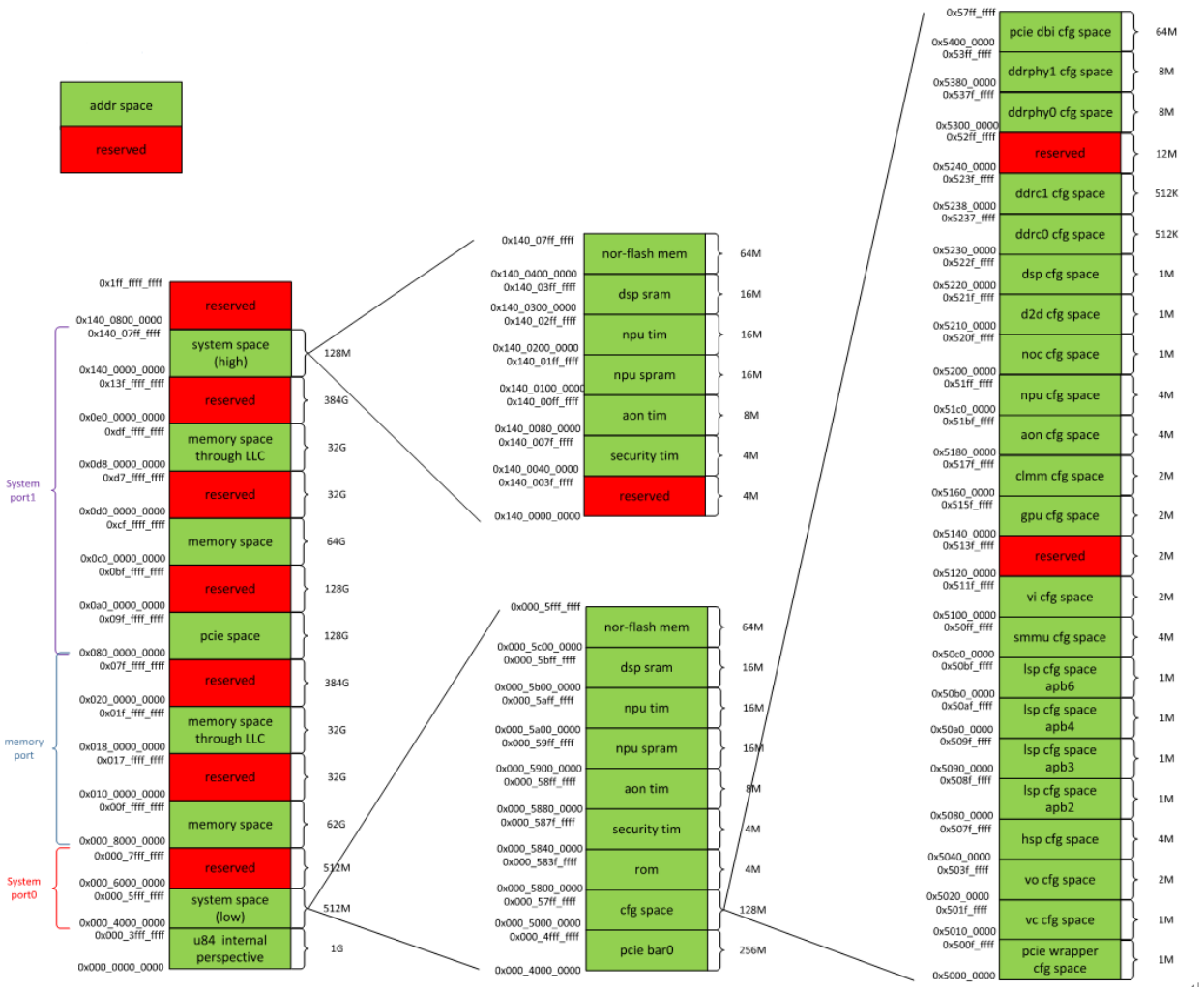


Figure 3-29 Address mapping

3.7 Interconnect

3.7.1 Overview

The EIC7700X system interconnect enables data interaction between various subsystems and DDR, SRAM, and Config Space.

The following are the main features supported by EIC7700X NoC：

- Supporting the standard amba protocol as the interface protocol between noc and generic ip. The related amba protocols are axi4, AXI4-Lite, ahb3, apb3, and apb4;
- Supporting the NSP protocol as an interface protocol for cascading NOCs;
- Use topology structures such as cross bar and distributed interconnection nodes;
- Supporting data networks to achieve interconnection between IPs, with each IP accessing memory, peripherals, register spaces, etc. through data networks;
- Supports monitoring networks, with monitoring points mainly including three types: transaction probe point, packet probe point, and error probe point. Through various monitoring points, latency, bandwidth, and error information are monitored;
- Support for a service network, which enables read and write access to the internal register

space of the Noc through the service network;

- Support qos generator, including three modes: fixed, limiter, regulator;
- The Noc internal arbiter supports two levels of arbitration. The first level of arbitration is based on urgency, which can be converted through socket QoS maps or obtained through QoS generators. The second level of arbitration is based on age information;
- Support watch dog, monitor the response time of slave, report timeout interrupt to the timeout host, and generate response err to ensure the integrity of the response;
- Support for the sideband manager, which allows the user to query the idle status of each interface in the noc by querying the sideband manager;
- Support default error target. Accessing the error address issued by the host will be routed to the corresponding default error target, and the default err target will return the response err;
- Little-Endian；

3.7.2 Block diagram

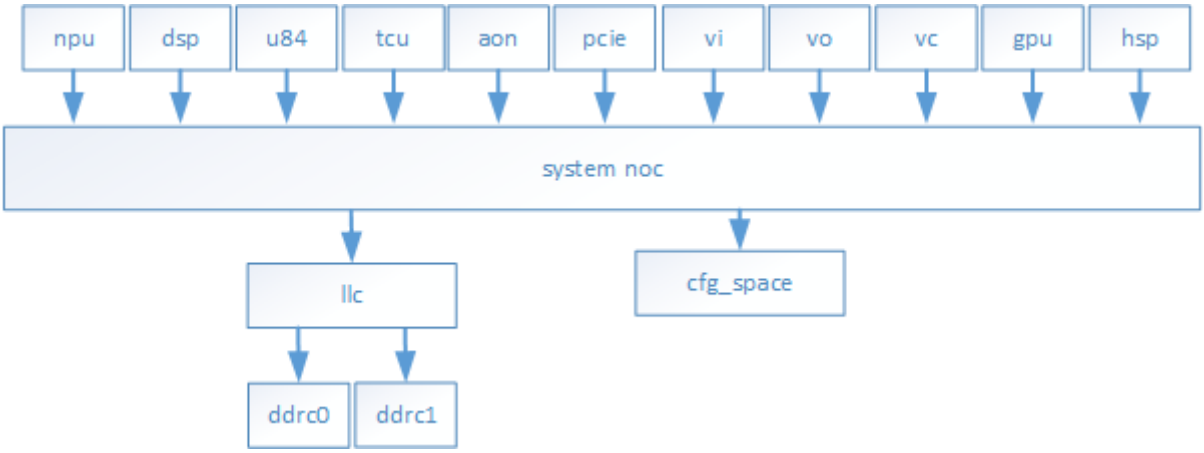


Figure 3-30 NoC interconnect diagram

3.7.3 Function Description

3.7.3.1 network interface unit

NoC interface converts between standard AMBA interface protocol and NoC's custom NSP protocol through the NIU module (Network Interface Unit), which also performs packet assembly and disassembly functions.

The internal sideband manager of the NoC records the idle status of each NIU's bus. The system can cooperate to query the corresponding bus interface status, enabling subsystem-specific reset.

The target NIU of the NoC's non-cascaded interface implements the watch dog function. The watch dog uses the input ref clk as the counter enable, and the EIC7700X sets the timeout period to 4k ref clk cycles (sampled at both rising and falling edges, effectively 2k ref clk cycles). If a pending request times out without a response, it is determined that the slave has failed, and no response will be returned. At this time, the NoC reports a timeout interrupt to the CPU and returns a response error to the host, ensuring that the failure of a single slave does not affect the overall system. The default ref clk is 24M with a 1-division factor. The division factor can be configured, with a total of 16 bits. By default, the timeout period is 85us with the division factor set to its maximum, resulting in a timeout period of 5.5s. The watch dog function has a functional enable register that is disabled by default.

The NoC sets default error targets for each init flow. If an abnormal address is issued, such as an

address that is not mapped, it will be routed by the NoC to the corresponding default error target, which will return a response error from that target. All default error targets operate in the always-on clock domain.

The following table shows the settings for all NIU interfaces in the NoC, along with their corresponding sideband managers, watch dogs, and default error targets:

llc_noc				
init/target	**niu**	**sideband**	**watch dog**	**default error target**
initiator	npu_lnoc_llc0	sense in[0]	\	lnoc_ddrt0_p0
	npu_lnoc_llc1	sense in[1]	\	lnoc_ddrt0_p0
	snoc_lnoc	\	\	lnoc_service
target	lnoc_ddrt0_p0	sense in[2]	Y	Y
	lnoc_ddrt1_p0	sense in[3]	Y	\
	lnoc_service	\	\	Y

sys_noc				
init/target	**niu**	**sideband**	**watch dog**	**default error target**
initiator	aon_snoc_sp0	sense in[0]	\	snoc_cnoc
	dspt_snoc	sense in[1]	\	snoc_cnoc
	jtag_snoc	sense in[2]	\	snoc_cnoc
	mcput_snoc_d2d	sense in[3]	\	snoc_cnoc
	mcput_snoc_mp	sense in[4]	\	snoc_npu
	mcput_snoc_sp0	sense in[5]	\	snoc_cnoc
	mcput_snoc_sp1	sense in[6]	\	snoc_aon
	npu_snoc_sp0	sense in[7]	\	snoc_cnoc
	npu_snoc_sp1	sense in[8]	\	snoc_aon
	pciet_snoc_p	sense in[9]	\	snoc_cnoc
	spislv_tbu3_snoc	sense in[10]	\	snoc_cnoc
	tbu4_snoc	sense in[11]	\	snoc_npu
	tcu_snoc	sense in[12]	\	snoc_npu
	rnoc_snoc	\	\	snoc_npu
	mnoc_snoc	\	\	snoc_npu
target	snoc_aon	sense in[13]	Y	Y
	snoc_ddrt0_p1	sense in[14]	Y	\
	snoc_ddrt0_p2	sense in[15]	Y	\
	snoc_ddrt1_p1	sense in[16]	Y	\
	snoc_ddrt1_p2	sense in[17]	Y	\
	snoc_dspt	sense in[18]	Y	\

sys_noc				
init/target	niu	sideband	watch dog	default error target
	snoc_mcput_d2d	sense in[19]	\	\
	snoc_npu	sense in[20]	Y	Y
	snoc_pciet	sense in[21]	Y	\
	snoc_cnoc	\	\	Y
	snoc_lnoc	\	\	\
	snoc_mnoc	\	\	\
	snoc_rnoc	\	\	\
	snoc_service	\	\	\

media_noc				
init/target	niu	sideband	watch dog	default error target
initiator	gpu_mnoc	sense in[0]	\	mnoc_snoc
	snoc_mnoc	\	\	mnoc_service
	tbu2_mnoc	sense in[1]	\	mnoc_snoc
	vc_mnoc	sense in[2]	\	mnoc_snoc
target	mnoc_ddrt0_p3	sense in[3]	Y	\
	mnoc_ddrt0_p3	sense in[4]	Y	\
	mnoc_service	\	\	Y
	mnoc_snoc	\	\	Y

realtime_noc				
init/target	niu	sideband	watch dog	default error target
initiator	snoc_rnoc	\	\	rnoc_service
	tbu0_rnoc	sense in[0]	\	rnoc_snoc
	vo_rnoc	sense in[1]	\	rnoc_snoc
target	rnoc_ddrt0_p4	sense in[2]	Y	\
	rnoc_ddrt1_p4	sense in[3]	Y	\
	rnoc_service	\	\	Y
	rnoc_snoc	\	\	Y

realtime_noc				
init/target	niu	sideband	watch dog	default error target
initiator	snoc_rnoc	\	\	rnoc_service
	tbu0_rnoc	sense in[0]	\	rnoc_snoc
	vo_rnoc	sense in[1]	\	rnoc_snoc

realtime_noc				
init/target	niu	sideband	watch dog	default error target
target	rnoc_ddrt0_p4	sense in[2]	Y	\
	rnoc_ddrt1_p4	sense in[3]	Y	\
	rnoc_service	\	\	Y
	rnoc_snoc	\	\	Y

cfg_noc				
init/target	niu	sideband	watch dog	default error target
initiator	snoc_cnoc	\	\	cnoc_aon
target	clmm	sense in[0]	Y	\
	cnoc_aon	sense in[1]	Y	Y
	cnoc_ddrt0_ctrl	sense in[2]	Y	\
	cnoc_ddrt0_phy	sense in[3]	Y	\
	cnoc_ddrt1_ctrl	sense in[4]	Y	\
	cnoc_ddrt1_phy	sense in[5]	Y	\
	cnoc_dspt	sense in[6]	Y	\
	cnoc_gpu	sense in[7]	Y	\
	cnoc_hsp	sense in[8]	Y	\
	cnoc_lsp_apb2	sense in[9]	Y	\
	cnoc_lsp_apb3	sense in[10]	Y	\
	cnoc_lsp_apb4	sense in[11]	Y	\
	cnoc_lsp_apb6	sense in[12]	Y	\
	cnoc_mcput_d2d	sense in[13]	Y	\
	cnoc_npu	sense in[14]	Y	\
	cnoc_pciet_p	sense in[15]	Y	\
	cnoc_pciet_x	sense in[16]	Y	\
	cnoc_service	\	\	\
	cnoc_tcu	sense in[17]	Y	\
	cnoc_vc	sense in[18]	Y	\
	cnoc_vi	sense in[19]	Y	\
	cnoc_vo	sense in[20]	Y	\

3.7.3.2 qos generator

EIC7700X NoC supports socket QoS mode and also supports regulation through QoS generator in NoC niu. EIC7700X NoC QoS generator has three modes:

- Fixed: fixed priority, which can configure the read and write priority of the corresponding interface. It is suitable for latency-sensitive types (in the case of socket QoS, it can be directly used with socket QoS). The internal priority range of EIC7700X NoC only supports 0-7 (in

ascending order, indicating higher priority). Therefore, the input QoS needs to be mapped to the urgency field actually used by the NoC internally. EIC7700X uses qos[3:1] as the converted NoC urgency value. The input original value is used for the output of the NoC QoS. The corresponding configuration registers p0 are write priority and p1 are read priority;

- Limiter: It can limit the bandwidth of the initiator by configuration. If the bandwidth exceeds the expected value, the NoC init niu will directly generate a backpressure signal. It is suitable for greedy types of hosts to avoid affecting the performance of other hosts due to their high bandwidth requirements. The limiter mode directly acts on the req payload. It does not distinguish between read and write, limiting the bandwidth to the total read and write bandwidth. The following are the relevant configuration registers for the limiter mode:

Table 3-224 Configure registers of limiter mode

Register name	Byte offset	Field	Bits	Bit offset	Description
Priority	0x08	P0	3	0	Priority of writes
		P1	3	8	Priority of reads
Mode	0x0C	Mode	2	0	0 = Fixed, 1 = Limiter, 2 = Bypass
Bandwidth	0x10	Bandwidth	if(useExtRef) $13 + \log_2(2\times wData/8)$ else $9 + \log_2(2\times wData/8)$	0	Throughput threshold in $1/256^{th}$-byte-per-cycle units
Saturation	0x14	Saturation	10	0	Maximum unregulated burst
ExtControl	0x18	SocketQosEn	1	0	Initiator drives QoS

Configuration description:
Extcontrol.SocketQosEn: Interface qos use enable, EIC7700X Noc supports the simultaneous use of socket qos and qos generator. The maximum value of the two is used simultaneously. However, it is recommended that socket qos and qos generator be used mutually exclusive;
Priority.P0: Write priority configuration, priority range 0-7;
Priority.P1: Read priority configuration, priority range 0-7;
Mode:limiter mode is configured as 1;
Bandwidth: bandwidth limit threshold, reg_val = (bandwidth/frequency)*256;
Saturation: burst tolerant value. The noc will ensure that the payload within the burst tolerant value is delivered at full speed. It is recommended to set the value according to the max burst size of the scenario. reg_val=saturation/16;
The Limiter mode consumes the expected bandwidth within the saturation limit time window. Once the bandwidth is consumed, the next payload of saturation size is allowed to continue to be delivered. The following figure is a configuration example:

- 32 byte burst, F=400MHz,BW=200MB/s, Saturation=64 byte(T=0.32us)
- Bandwidth register: 128, Saturation register: 4

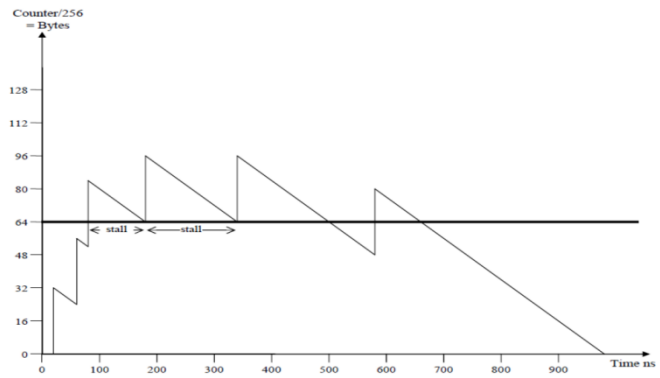


Figure 3-31 Bandwidth Limiter Behavior Example

- Regulator: This mode ensures that the host can obtain the desired bandwidth level in the system. Before the counter value reaches the payload threshold (saturation/2), the NoC will automatically adjust the priority of the interface to the configured priority1 value (high priority) and deliver urgent packets; if the threshold is exceeded, the NoC will adjust the priority of the interface to the configured priority0 value (low priority). This type is suitable for real-time type hosts. The following are the relevant configuration registers for the regulator mode:

Table 3-225 Configure registers of regulator mode

Register name	Byte offset	Field	Bits	Bit offset	Description
Priority	0x08	P0	3	0	High throughput priority level (or priority of writes)
		P1	3	8	Low throughput priority level (or priority of reads)
Mode	0x0C	Mode	2	0	0 = Fixed 1 = Limiter 2 = Bypass 3 = Regulator
Bandwidth	0x10	Bandwidth	if(useExtRef) $13 + \log_2(2\times wData/8)$ else $9 + \log_2(2\times wData/8)$	0	Throughput threshold in 1/256th-byte-per-cycle units
Saturation	0x14	Saturation	10	0	Maximum unregulated burst
ExtControl	0x18	SocketQosEn	1	0	Initiator and regulator drive QoS

Compared to the limiter mode, the register settings for the regulator mode are basically the same, with some differences in the usage of Priority.P0, Priority.P1, and Saturation:

Priority.P0: If the counter-measured traffic exceeds the threshold, the NoC adjusts the corresponding packet urgency domain to the value of Priority.P0 (low priority, with consistent read/write urgency values).

Priority.P1: If the counter-measured traffic is below the threshold, the NoC adjusts the corresponding packet urgency domain to the value of Priority.P1 (high priority, with consistent read/write urgency values).

Mode: The regulator mode is configured as 3.

Bandwidth: Expected bandwidth, reg_val = (bandwidth/frequency)*256.

Saturation: The NoC uses saturation/2 as the threshold. If the remaining payload is below saturation/2, it is delivered with high priority; if it exceeds the threshold, it is delivered with low priority. It is recommended to set the saturation value according to the internal buffer capacity of the real-time type host. reg_val=saturation/16.

The following figure shows an example configuration of the regulator mode:

- 1Kbyte burst, F=500MHz, BW=2GB/s, Saturation=4Kbytes(T=2.048us.)
- Bandwidth register: 1024 , Saturation register: 256

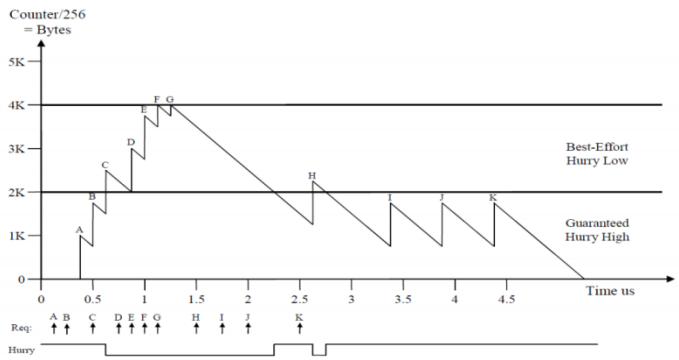


Figure 3-32 Bandwidth Regulator Behavior Example

In the regulator mode, the host must be able to obtain the expected bandwidth within the saturation/2 time window. However, if the payload required exceeds this time window, the request can continue to be delivered since it is not urgent. At this time, the NoC can serve other high-priority transactions first.

The following table records the deployment of qos generator on each NoC, and other hosts that have not deployed qos generator are divided into two categories. One is the init flow that does not require the use of qos, and the other is the latency-sensitive type of host that does not require bandwidth regulation and can directly use socket qos to configure high priority.

Table 3-226 qos generator of NoCs

noc	init niu
sys_noc	dspt_snoc
	npu_snoc_sp1
	spislv_tbu3_snoc
media_noc	gpu_mnoc
	tbu2_mnoc
	vc_mnoc
realtime_noc	tbu0_rnoc
	vo_rnoc

The system's noc internal arbiter uses two levels of arbitration, the first level based on urgency and the second level based on age information.

3.7.3.3 latency

EIC7700X NoC has latency monitoring requirements for some hosts. For these hosts, transaction

probe points are set in their init niu to monitor the distribution of latency over a period of time. The typical configuration registers for latency statistics are described below (for full register details, refer to the register description section):

- Threshold: The hardware implementation sets four bins per transaction probe, and each bin can be set to a statistical latency range. The four bins correspond to three Threshold* registers. For example, configuring the values of these three registers to 100, 200, and 500, respectively, indicates that the latency statistical ranges for the four bins are bin0:0-100, bin1:100-200, bin2:200-500, and bin3:500+ (unit: cycle);
- StatFilter: niu implements a statistical filter internally, which can filter address windows, opcode, user information, etc., achieving accurate statistics through filtering;
- StatPeriod: This register sets the time for statistical measurement. The value of this register is log2(stat_period_cycles). Within the statistical measurement time, latency statistics are collected for each pending request. Finally, the collected latency values are compared to determine which bin they belong to, and the corresponding bin counter is incremented by 1. The allocation of statistic counters is designed to be consistent with the ots capability of the EIC7700X . The EIC7700X sets the bit width of the statistic counter to 10 bits. If the latency exceeds 1023, it will be capped at 1023;
- NTenureLine*: Several transpoints share a transprobe function module, which involves the allocation of statistical resources, such as the allocation of statistic counters. On the hardware, the statistic counters are bound in groups of four, which means that the statistic counters allocated to a certain probe point must be a multiple of four. NNTenureLine is the register that completes the allocation of statistic counters. If three trans probe points share a trans probe, then there will be two registers, and the configuration values are assumed to be 8 and 16, respectively. This means that trans_probe_point_0 allocates 84 statistic counters, trans_probe_point_1 allocates 816 statistic counters, and trans_probe_point_2 allocates the remaining statistic counters;
- StatAlarm*: Configurable whether to enable reporting of statistical interrupt. Events that trigger interrupt reporting can be configured, such as detecting a statistical value that exceeds statmax or detecting a statistical value that is less than statmin. The interrupt status needs to be cleared through register writing;

The following table records the deployment points of the EIC7700X transaction probe point:

Table 3-227 transaction probe point

noc	trans probe point	trans probe
llc_noc	npu_lnoc_llc0	lnoc.trans_probe
	npu_lnoc_llc1	lnoc.trans_probe
sys_noc	dspt_snoc	snoc.trans_probe0
	npu_snoc_sp1	snoc.trans_probe0
	mcput_snoc_mp	snoc.trans_probe1
	mcput_snoc_sp1	snoc.trans_probe1
	tcu_snoc	snoc.trans_probe1
	spislv_tbu3_snoc	snoc.trans_probe2
	tbu4_snoc	snoc.trans_probe2
media_noc	gpu_mnoc	mnoc.trans_probe
	tbu2_mnoc	mnoc.trans_probe
	vc_mnoc	mnoc.trans_probe

noc	trans probe point	trans probe
realtime_noc	tbu0_rnoc	rnoc.trans_probe
	vo_rnoc	rnoc_trans_probe

3.7.3.4 bandwith

EIC7700X has set packet probe points for all five DDDR ports (5x2chan) to monitor the actual bus bandwidth of the corresponding DDDR interface. The typical configuration register for bandwidth statistics is described below (see the register description section for a full description of all registers):

- Filter*: You can achieve accurate statistics by configuring Filter*. Common filtering configurations include window address range, opcode, route id, opcode, urgency, user information, and so on;
- StatPeriod: This register sets the time for statistical measurement. The value of this register is log2(stat_period_cycles). During the statistical measurement period, the counter will be incremented based on the size of the received packet. Once the statistical measurement is complete, the final count value will be written to the Couters_*_Val register. By reading this register, you can calculate the bandwidth based on StatPeriod and the corresponding bus frequency;
- Counter_*_Src: Each packet probe is set up with two bandwidth counters, each with a 32-bit bit width, which can be used in series in the form of a chain. When calculating bandwidth, Counter_0_Src is configured to use the Bytes mode, and Counter_0_Src is configured to use the CHAIN mode;
- StatAlarm*: The usage method of this register is the same as for the transaction probe;

The following figure shows an example of a bandwidth statistics configuration process, where the interrupt event is StatMax:

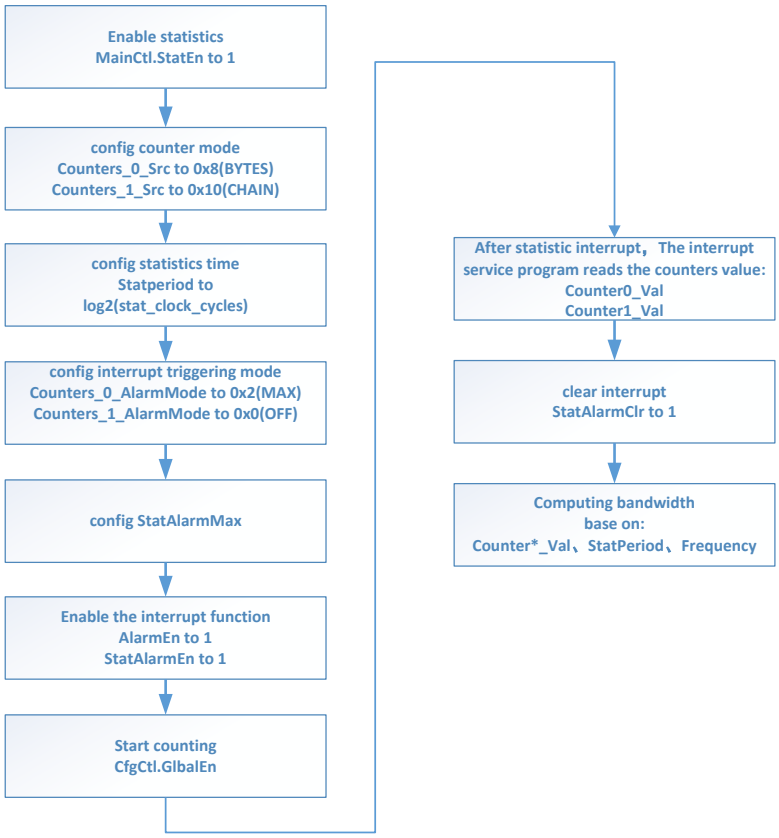


Figure 3-33 Flow of bandwith count

3.7.3.5 error logger

EIC7700X NoC sets error probe points on some link paths returned by RSPS, which can monitor the following error messages:

Table 3-228 error probe point information

code	value	source	type
SLV	0	target	error information generated within the slave;
DEC	1	iinti niu	An address decoding error occurs. An address request that cannot be accessed is sent, or an address segment is not mapped on the sysytem addr map.
UNS	2	target niu	A request type not supported by noc is sent. For example, a noc interface does not support fixed packets, but receives fixed requests.
TMO	6	target niu	If the slave does not respond due to timeout, the noc target niu watch dog will generate a timeout error
RSV	7	none	reserved

The following table records the deployment of error transaction:

Table 3-229 error transaction information

noc	trans probe point	obersever
llc_noc	ddrt0_p0_err_point	lnoc.observer
	ddrt1_p0_err_point	

noc	trans probe point	obersever
sys_noc	aon_err_point	snoc.observer
	ddrt0_p1_err_point	
	ddrt0_p2_err_point	
	ddrt1_p1_err_point	
	ddrt1_p2_err_point	
	dspt_err_point	
	npu_err_point	
	pciet_err_point	
media_noc	ddrt0_p3_err_point	mnoc.observer
	ddrt1_p3_err_point	
realtime_noc	ddrt0_p4_err_point	rnoc.observer
	ddrt1_p4_err_point	
cfg_noc	cnoc_err_point	cnoc.observer

3.8 DMA Controller

3.8.1 Overview

EIC7700X contains two DMA controllers for data interaction between memory and memory, and between memory and peripherals. After receiving the corresponding request, DMA starts the controller according to the system's configuration of the channel, sends the address and control signal to the memory or peripherals, and reports the transmission status to the system through the interrupt.

DMA0 supports the following features:

- Supports 12 programmable channels;channel1/2 fifo depth 256, used for m2m data transfer; channel3~12 fifo depth 64, used for m2p and p2m data transfer
- Programmable channel priority
- Programmable channel burst length
- Includes 16 group DMA request, interact with 8 peripherals
- Programmable multi-block transfer(llinked list/auto reload/shadow register/contiguous address)
- Supports one 32bit Master Interface
- Supports 8bit/16bit/32bit data bus transfer
- Supports channel locking

DMA0 request connection is shown in the following table.

Table 3-230 DMA0 Req

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
0	uart1_dma_rx	uart1_dma_rx_single	valid when scu_lsp_uart0_dma_sel=0
1	uart1_dma_tx	uart1_dma_tx_single	valid when scu_lsp_uart0_dma_sel=0
2	i2c1_dma_rx	i2c1_dma_rx_single	valid when scu_lsp_i2c1_dma_sel=0
3	i2c1_dma_tx	i2c1_dma_tx_single	valid when scu_lsp_i2c1_dma_sel=0

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
4	i2c0_dma_rx	i2c0_dma_rx_single	valid when scu_lsp_i2c0_dma_sel=0
5	i2c0_dma_tx	i2c0_dma_tx_single	valid when scu_lsp_i2c0_dma_sel=0
6	ssi2_dma_rx	ssi2_dma_rx_single	valid when scu_lsp_ssi1_dma_sel=0
7	ssi2_dma_tx	ssi2_dma_tx_single	valid when scu_lsp_ssi1_dma_sel=0
8	ssi1_dma_rx	ssi1_dma_rx_single	valid when scu_lsp_ssi0_dma_sel=0
9	ssi1_dma_tx	ssi1_dma_tx_single	valid when scu_lsp_ssi0_dma_sel=0
10	i2s2_dma_rx	i2s2_dma_rx_single	valid when scu_vo_i2s2_dma_sel=0
11	i2s2_dma_tx	i2s2_dma_tx_single	valid when scu_vo_i2s2_dma_sel=0
12	i2s1_dma_rx	i2s1_dma_rx_single	valid when scu_vo_i2s1_dma_sel=0
13	i2s1_dma_tx	i2s1_dma_tx_single	valid when scu_vo_i2s1_dma_sel=0
14	i2s0_dma_rx	i2s0_dma_rx_single	valid when scu_vo_i2s0_dma_sel=0
15	i2s0_dma_tx	i2s0_dma_tx_single	valid when scu_vo_i2s0_dma_sel=0

DMA1 supports the following features:

- Supports 16 programmable channels;channel1/2 fifo depth 256, used for m2m data transfer; channel3~16 fifo depth 64, used for m2p and p2m data transfer
- Programmable channel priority
- Programmable channel burst length
- Includes 45 DMA request, interact with 23 peripherals
- Programmable multi-block transfer(llinked list/auto reload/shadow register/contiguous address)
- Supports two 64bit Master Interface
- Supports 8bit/16bit/32/64bit data bus transfer
- Supports channel locking

DMA1 request connection is shown in the following table.

Table 3-231 DMA1 req

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
0	i2s2_dma_rx	i2s2_dma_rx_single	valid when scu_vo_i2s2_dma_sel=1
1	i2s2_dma_tx	i2s2_dma_tx_single	valid when scu_vo_i2s2_dma_sel=1
2	i2s1_dma_rx	i2s1_dma_rx_single	valid when scu_vo_i2s1_dma_sel=1
3	i2s1_dma_tx	i2s1_dma_tx_single	valid when scu_vo_i2s1_dma_sel=1
4	i2s0_dma_rx	i2s0_dma_rx_single	valid when scu_vo_i2s0_dma_sel=1
5	i2s0_dma_tx	i2s0_dma_tx_single	valid when scu_vo_i2s0_dma_sel=1
6	i2c9_dma_rx	i2c9_dma_rx_single	always valid

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
7	i2c9_dma_tx	i2c9_dma_tx_single	always valid
8	i2c8_dma_rx	i2c8_dma_rx_single	always valid
9	i2c8_dma_tx	i2c8_dma_tx_single	always valid
10	i2c7_dma_rx	i2c7_dma_rx_single	always valid
11	i2c7_dma_tx	i2c7_dma_tx_single	always valid
12	i2c6_dma_rx	i2c6_dma_rx_single	always valid
13	i2c6_dma_tx	i2c6_dma_tx_single	always valid
14	i2c5_dma_rx	i2c5_dma_rx_single	always valid
15	i2c5_dma_tx	i2c5_dma_tx_single	always valid
16	i2c4_dma_rx	i2c4_dma_rx_single	always valid
17	i2c4_dma_tx	i2c4_dma_tx_single	always valid
18	i2c3_dma_rx	i2c3_dma_rx_single	always valid
19	i2c3_dma_tx	i2c3_dma_tx_single	always valid
20	i2c2_dma_rx	i2c2_dma_rx_single	always valid
21	i2c2_dma_tx	i2c2_dma_tx_single	always valid
22	i2c1_dma_rx	i2c1_dma_rx_single	valid when scu_lsp_i2c1_dma_sel=1
23	i2c1_dma_tx	i2c1_dma_tx_single	valid when scu_lsp_i2c1_dma_sel=1
24	i2c0_dma_rx	i2c0_dma_rx_single	valid when scu_lsp_i2c0_dma_sel=1
25	i2c0_dma_tx	i2c0_dma_tx_single	valid when scu_lsp_i2c0_dma_sel=1
26	uart4_dma_rx	uart4_dma_rx_single	always valid
27	uart4_dma_tx	uart4_dma_tx_single	always valid
28	uart3_dma_rx	uart3_dma_rx_single	always valid
29	uart3_dma_tx	uart3_dma_tx_single	always valid
30	uart2_dma_rx	uart2_dma_rx_single	always valid
31	uart2_dma_tx	uart2_dma_tx_single	always valid
32	uart1_dma_rx	uart1_dma_rx_single	valid when scu_lsp_uart0_dma_sel=1
33	uart1_dma_tx	uart1_dma_tx_single	valid when scu_lsp_uart0_dma_sel=1
34	uart0_dma_rx	uart0_dma_rx_single	always valid
35	uart0_dma_tx	uart0_dma_tx_single	always valid
36	ssi1_dma_rx	ssi1_dma_rx_single	valid when scu_lsp_ssi1_dma_sel=1

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
37	ssi1_dma_tx	ssi1_dma_tx_single	valid when scu_lsp_ssi1_dma_sel=1
38	ssi0_dma_rx	ssi0_dma_rx_single	valid when scu_lsp_ssi0_dma_sel=1
39	ssi0_dma_tx	ssi0_dma_tx_single	valid when scu_lsp_ssi0_dma_sel=1
40	-	bootspi_dma_req	always valid
41	aon_i2c0_dma_rx	aon_i2c0_dma_rx_single	always valid
42	aon_i2c0_dma_tx	aon_i2c0_dma_tx_single	always valid
43	aon_i2c1_dma_rx	aon_i2c1_dma_rx_single	always valid
44	aon_i2c1_dma_tx	aon_i2c1_dma_tx_single	always valid

3.8.2 Block Diagram

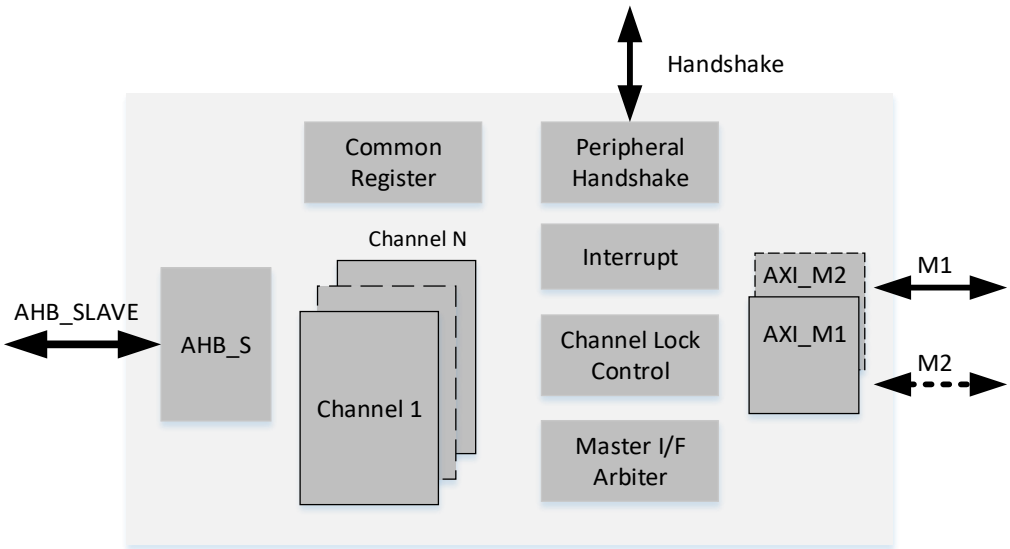


Figure 3-34 DMA Diagram

DMA Controller interact with system through axi bus, main blocks are shown as the following:

- Channel Contains the specific implementation of each channel to complete the data transmission of the system configuration
- Interrupt block generate interrupt according to the transmission status and inform CPU to process
- Peripheral Handshake handles the request of peripherals
- Channel Lock allows a channel to be locked to the arbiter
- Arbiter handles requests of all channels according to the configuration

3.8.3 Function Description

- Clock Gating
  Close DMA clock through system register
- Soft Reset
  Assert DMA reset through system register
- Flow for linked-list-based multi-block data transfer

1.Software reads the DMAC channel enable register (DMAC_ChEnReg) to select an available (unused) channel.

2. Software programs the CHx_CFG register with appropriate values for the DMA transfer.The SRC_MLTBLK_TYPE and/or DST_MLTBLK_TYPE bits must be set to 2'b11.

3. Software programs the base address of the first linked list item and the master interface on which the linked list item is available in the CHx_LLP register.

4. Software creates one or more linked list items in system memory. Software can create the entire linked list item in advance or dynamically extend the linked list using the CHx_CTL.ShadowReg_Or_LLI_Valid and CHx_CTL.LLI_Last fields of the LLI.

5. Software enables the channel by writing 1 to the appropriate bit location in DMAC_ChEnReg register.

6.dmac initiates the DMA block transfer operation based on the settings for the block transfer.The block transfer might start immediately or after the hardware or software handshaking request,depending on the settings of the TT_FC field in the CHx_CFG register. dmac copies the linked list contents to the registers used for executing the DMA block transfer (that is, the CHx_SAR and/or CHx_DAR, CHx_BLOCK_TS, and CHx_CTL registers) and initiates the DMA block transfer.

7.During the linked list fetch phase:(a) if the current block is the final block in the transfer and completes the DMA transfer operation at the end of current block transfer (b)if there are one or more blocks to be transferred and goes to step 6 (c)if CHx_CTL.ShadowReg_Or_LLI_Valid bit of the fetched LLI as 0, dmac might generate ShadowReg_Or_LLI_Invalid_ERR Interrupt. dmac waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid LLI availability, before attempting another LLI read operation.

3.8.4 Register Description

The following registers without special explanation are applicable to the two DMA controllers, but the access to the base address is different. Due to different features lead to register details are different in the specific location has been added explanation.

3.8.4.1 Register list

Common Register

Register	Offset	Description
DMAC_CFGREG	0x10	gloabal config
DMAC_CHENREG2	0x18	channel enable
DMAC_CHSUSPREG	0x20	channel suspend
DMAC_CHABORTREG	0x28	channel abort
DMAC_INTSTATUSREG2	0x30	combined inter status
DMAC_INTCLEARREG	0x38	common reg space inter clear
DMAC_INTSTATUS_ENABLEREG	0x40	common reg space inter enable
DMAC_INTSIGNAL_ENABLEREG	0x48	common reg space inter signal enable
DMAC_INTSTATUSREG	0x50	common reg space inter status
DMAC_RESETREG	0x58	software reset

Channel Register

Register	Offset	Description
CHx_SAR	0x100+(x-1)*0x100	source addr
CHx_DAR	0x108+(x-1)*0x100	destination addr
CHx_BLOCK_TS	0x110+(x-1)*0x100	block transfer size
CHx_CTL	0x118+(x-1)*0x100	control reg
CHx_CFG2	0x120+(x-1)*0x100	config reg
CHx_LLP	0x128+(x-1)*0x100	link-list pointer
CHx_STATUSREG	0x130+(x-1)*0x100	status reg
CHx_SWHSSRCREG	0x138+(x-1)*0x100	software handshake source reg
CHx_SWHSDSTREG	0x140+(x-1)*0x100	software handshake destination reg
CHx_BLK_TFR_RESUMEREQREG	0x148+(x-1)*0x100	block transfer resume request reg
CHx_AXI_IDREG	0x150+(x-1)*0x100	AXI ID reg
CHx_SSTAT	0x160+(x-1)*0x100	source status (only valid in channel1 and 2)
CHx_DSTAT	0x168+(x-1)*0x100	destination status (only valid in channel1 and 2)
CHx_SSTATAR	0x170+(x-1)*0x100	source status fetch addr (only valid in channel1 and 2)
CHx_DSTATAR	0x178+(x-1)*0x100	destination status fetch addr (only valid in channel1 and 2)
CHx_INTSTATUS_ENABLEREG	0x180+(x-1)*0x100	interrupt status enable
CHx_INTSTATUS	0x188+(x-1)*0x100	interrupt status
CHx_INTSIGNAL_ENABLEREG	0x190+(x-1)*0x100	interrupt signal enable
CHx_INTCLEARREG	0x198+(x-1)*0x100	interrupt status clear

3.8.4.2 Register detail

DMAC_CFGREG

- **Name:** dmac Global Configuration Register
- **Description:** This register is used to enable the dmac, which must be done before any channel activity can begin. This register also contains global interrupt enable bit.
- **Size:** 64 bits
- Offset: 0x10

Bits	Name	Memory Access	Description
63:2	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
1	INT_EN	R/W	This bit is used to globally enable the interrupt generation. • 0: dmac Interrupts are disabled • 1: dmac Interrupt logic is enabled.

Bits	Name	Memory Access	Description
			Value After Reset: 0x0
0	DMAC_EN	R/W	This bit is used to enable the axi_dmac. • 0: dmac disabled • 1: dmac enabled **NOTE**: If this bit DMAC_EN bit is cleared while any channel is still active, then this bit still returns 1 to indicate that there are channels still active until dmac hardware has terminated all activity on all channels, at which point this bit returns zero (0). **Value After Reset:** 0x0

DMAC_CHENREG2(DMA0)

- **Name:** dmac Channel Enable Register 2
- **Description:** This is dmac Channel Enable Register. If software wants to set up a new channel, it can read this register to find out which channels are currently inactive and then enable an inactive channel with the required priority. All bits of this register are cleared to 0 when the dmac Global Enable bit (DMAC_CfgReg.DMAC_EN) is 0. When DMAC_CfgReg.DMAC_EN is 0, a write to the DMAC_ChEnReg register is ignored and a read always reads back 0. The channel enable bit, DMAC_ChEnReg.CH_EN, is written only if the corresponding channel write enable bit, DMAC_ChEnReg.CH_EN_WE, is asserted on the same slave interface write transfer. For example, writing hex 0000_XXXX_0001_XXX1 writes a 1 into DMAC_ChEnReg [0], while DMAC_ChEnReg [15:1] and DMAC_ChEnReg [47:32] remains unchanged.Writing hex 0000_XXXX_0000_XXXX leaves DMAC_ChEnReg [15:0] and DMAC_ChEnReg [47:32] unchanged.
- **Size:** 64 bits
- Offset: 0x18

Bits	Name	Memory Access	Description
63:28	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
27	CH12_EN_WE	W	DW_axi_dmac Channel-12 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH12_EN): Enable Write to CH12_EN bit • 0x0 (DISABLE_WR_CH12_EN): Disable Write to respective CH12_EN bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_EN_WE	W	DW_axi_dmac Channel-11 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH11_EN): Enable Write to CH11_EN bit

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_WR_CH11_EN): Disable Write to respective CH11_EN bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_EN_WE	W	DW_axi_dmac Channel-10 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH10_EN): Enable Write to CH10_EN bit • 0x0 (DISABLE_WR_CH10_EN): Disable Write to respective CH10_EN bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_EN_WE	W	DW_axi_dmac Channel-9 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH9_EN): Enable Write to CH9_EN bit • 0x0 (DISABLE_WR_CH9_EN): Disable Write to respective CH9_EN bit **Value After Reset:** 0x0 **Volatile:** true
23	CH8_EN_WE	W	DW_axi_dmac Channel-8 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH8_EN): Enable Write to CH8_EN bit • 0x0 (DISABLE_WR_CH8_EN): Disable Write to respective CH8_EN bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_EN_WE	W	DW_axi_dmac Channel-7 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH7_EN): Enable Write to CH7_EN bit • 0x0 (DISABLE_WR_CH7_EN): Disable Write to respective CH7_EN bit **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
21	CH6_EN_WE	W	DW_axi_dmac Channel-6 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH6_EN): Enable Write to CH6_EN bit • 0x0 (DISABLE_WR_CH6_EN): Disable Write to respective CH6_EN bit **Value After Reset:** 0x0 **Volatile:** true
20	CH5_EN_WE	W	DW_axi_dmac Channel-5 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH5_EN): Enable Write to CH5_EN bit • 0x0 (DISABLE_WR_CH5_EN): Disable Write to respective CH5_EN bit **Value After Reset:** 0x0 **Volatile:** true
19	CH4_EN_WE	W	DW_axi_dmac Channel-4 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH4_EN): Enable Write to CH4_EN bit • 0x0 (DISABLE_WR_CH4_EN): Disable Write to respective CH4_EN bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_EN_WE	W	DW_axi_dmac Channel-3 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH3_EN): Enable Write to CH3_EN bit • 0x0 (DISABLE_WR_CH3_EN): Disable Write to respective CH3_EN bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_EN_WE	W	DW_axi_dmac Channel-2 Enable Write Enable bit. Read back value of this register bit is always '0'.

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_WR_CH2_EN): Enable Write to CH2_EN bit • 0x0 (DISABLE_WR_CH2_EN): Disable Write to respective CH2_EN bit **Value After Reset:** 0x0 **Volatile:** true
16	CH1_EN_WE	W	DW_axi_dmac Channel-1 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH1_EN): Enable Write to CH1_EN bit • 0x0 (DISABLE_WR_CH1_EN): Disable Write to respective CH1_EN bit **Value After Reset:** 0x0 **Volatile:** true
15:12	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
11	CH12_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-12. • 0: DW_axi_dmac Channel-12 is disabled • 1: DW_axi_dmac Channel-12 is enabled The bit 'DMAC_ChEnReg.CH12_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH12): DW_axi_dmac: Channel-12 is enabled • 0x0 (DISABLE_CH12): DW_axi_dmac: Channel-12 is disabled **Value After Reset:** 0x0 **Volatile:** true
10	CH11_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-11. • 0: DW_axi_dmac Channel-11 is disabled • 1: DW_axi_dmac Channel-11 is enabled The bit 'DMAC_ChEnReg.CH11_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ENABLE_CH11): DW_axi_dmac: Channel-11 is enabled • 0x0 (DISABLE_CH11): DW_axi_dmac: Channel-11 is disabled **Value After Reset:** 0x0 **Volatile:** true
9	CH10_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-10. • 0: DW_axi_dmac Channel-10 is disabled • 1: DW_axi_dmac Channel-10 is enabled The bit 'DMAC_ChEnReg.CH10_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH10): DW_axi_dmac: Channel-10 is enabled • 0x0 (DISABLE_CH10): DW_axi_dmac: Channel-10 is disabled **Value After Reset:** 0x0 **Volatile:** true
8	CH9_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-9. • 0: DW_axi_dmac Channel-9 is disabled • 1: DW_axi_dmac Channel-9 is enabled The bit 'DMAC_ChEnReg.CH9_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH9): DW_axi_dmac: Channel-9 is enabled • 0x0 (DISABLE_CH9): DW_axi_dmac: Channel-9 is disabled **Value After Reset:** 0x0 **Volatile:** true
7	CH8_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-8. • 0: DW_axi_dmac Channel-8 is disabled • 1: DW_axi_dmac Channel-8 is enabled The bit 'DMAC_ChEnReg.CH8_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH8): DW_axi_dmac: Channel-8 is enabled

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_CH8): DW_axi_dmac: Channel-8 is disabled **Value After Reset:** 0x0 **Volatile:** true
6	CH7_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-7. • 0: DW_axi_dmac Channel-7 is disabled • 1: DW_axi_dmac Channel-7 is enabled The bit 'DMAC_ChEnReg.CH7_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH7): DW_axi_dmac: Channel-7 is enabled • 0x0 (DISABLE_CH7): DW_axi_dmac: Channel-7 is disabled **Value After Reset:** 0x0 **Volatile:** true
5	CH6_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-6. • 0: DW_axi_dmac Channel-6 is disabled • 1: DW_axi_dmac Channel-6 is enabled The bit 'DMAC_ChEnReg.CH6_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH6): DW_axi_dmac: Channel-6 is enabled • 0x0 (DISABLE_CH6): DW_axi_dmac: Channel-6 is disabled **Value After Reset:** 0x0 **Volatile:** true
4	CH5_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-5. • 0: DW_axi_dmac Channel-5 is disabled • 1: DW_axi_dmac Channel-5 is enabled The bit 'DMAC_ChEnReg.CH5_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH5): DW_axi_dmac: Channel-5 is enabled • 0x0 (DISABLE_CH5): DW_axi_dmac: Channel-5 is disabled

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
3	CH4_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-4. • 0: DW_axi_dmac Channel-4 is disabled • 1: DW_axi_dmac Channel-4 is enabled The bit 'DMAC_ChEnReg.CH4_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH4): DW_axi_dmac: Channel-4 is enabled • 0x0 (DISABLE_CH4): DW_axi_dmac: Channel-4 is disabled **Value After Reset:** 0x0 **Volatile:** true
2	CH3_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-3. • 0: DW_axi_dmac Channel-3 is disabled • 1: DW_axi_dmac Channel-3 is enabled The bit 'DMAC_ChEnReg.CH3_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH3): DW_axi_dmac: Channel-3 is enabled • 0x0 (DISABLE_CH3): DW_axi_dmac: Channel-3 is disabled **Value After Reset:** 0x0 **Volatile:** true
1	CH2_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-2. • 0: DW_axi_dmac Channel-2 is disabled • 1: DW_axi_dmac Channel-2 is enabled The bit 'DMAC_ChEnReg.CH2_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH2): DW_axi_dmac: Channel-2 is enabled • 0x0 (DISABLE_CH2): DW_axi_dmac: Channel-2 is disabled **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
0	CH1_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-1. • 0: DW_axi_dmac Channel-1 is disabled • 1: DW_axi_dmac Channel-1 is enabled The bit 'DMAC_ChEnReg.CH1_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH1): DW_axi_dmac: Channel-1 is enabled • 0x0 (DISABLE_CH1): DW_axi_dmac: Channel-1 is disabled **Value After Reset:** 0x0 **Volatile:** true

DMAC_CHENENREG2 (DMA1)

Bits	Name	Memory Access	Description
63:32	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
31	CH16_EN_WE	W	DW_axi_dmac Channel-16 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH16_EN): Enable Write to CH16_EN bit • 0x0 (DISABLE_WR_CH16_EN): Disable Write to respective CH16_EN bit **Value After Reset:** 0x0 **Volatile:** true
30	CH15_EN_WE	W	DW_axi_dmac Channel-15 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH15_EN): Enable Write to CH15_EN bit • 0x0 (DISABLE_WR_CH15_EN): Disable Write to respective CH15_EN bit **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
29	CH14_EN_WE	W	DW_axi_dmac Channel-14 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH14_EN): Enable Write to CH14_EN bit • 0x0 (DISABLE_WR_CH14_EN): Disable Write to respective CH14_EN bit **Value After Reset:** 0x0 **Volatile:** true
28	CH13_EN_WE	W	DW_axi_dmac Channel-13 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH13_EN): Enable Write to CH13_EN bit • 0x0 (DISABLE_WR_CH13_EN): Disable Write to respective CH13_EN bit **Value After Reset:** 0x0 **Volatile:** true
27	CH12_EN_WE	W	DW_axi_dmac Channel-12 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH12_EN): Enable Write to CH12_EN bit • 0x0 (DISABLE_WR_CH12_EN): Disable Write to respective CH12_EN bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_EN_WE	W	DW_axi_dmac Channel-11 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH11_EN): Enable Write to CH11_EN bit • 0x0 (DISABLE_WR_CH11_EN): Disable Write to respective CH11_EN bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_EN_WE	W	DW_axi_dmac Channel-10 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ENABLE_WR_CH10_EN): Enable Write to CH10_EN bit • 0x0 (DISABLE_WR_CH10_EN): Disable Write to respective CH10_EN bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_EN_WE	W	DW_axi_dmac Channel-9 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH9_EN): Enable Write to CH9_EN bit • 0x0 (DISABLE_WR_CH9_EN): Disable Write to respective CH9_EN bit **Value After Reset:** 0x0 **Volatile:** true
23	CH8_EN_WE	W	DW_axi_dmac Channel-8 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH8_EN): Enable Write to CH8_EN bit • 0x0 (DISABLE_WR_CH8_EN): Disable Write to respective CH8_EN bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_EN_WE	W	DW_axi_dmac Channel-7 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH7_EN): Enable Write to CH7_EN bit • 0x0 (DISABLE_WR_CH7_EN): Disable Write to respective CH7_EN bit **Value After Reset:** 0x0 **Volatile:** true
21	CH6_EN_WE	W	DW_axi_dmac Channel-6 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH6_EN): Enable Write to CH6_EN bit • 0x0 (DISABLE_WR_CH6_EN): Disable Write to respective CH6_EN bit

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
20	CH5_EN_WE	W	DW_axi_dmac Channel-5 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH5_EN): Enable Write to CH5_EN bit • 0x0 (DISABLE_WR_CH5_EN): Disable Write to respective CH5_EN bit **Value After Reset:** 0x0 **Volatile:** true
19	CH4_EN_WE	W	DW_axi_dmac Channel-4 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH4_EN): Enable Write to CH4_EN bit • 0x0 (DISABLE_WR_CH4_EN): Disable Write to respective CH4_EN bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_EN_WE	W	DW_axi_dmac Channel-3 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH3_EN): Enable Write to CH3_EN bit • 0x0 (DISABLE_WR_CH3_EN): Disable Write to respective CH3_EN bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_EN_WE	W	DW_axi_dmac Channel-2 Enable Write Enable bit. Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH2_EN): Enable Write to CH2_EN bit • 0x0 (DISABLE_WR_CH2_EN): Disable Write to respective CH2_EN bit **Value After Reset:** 0x0 **Volatile:** true
16	CH1_EN_WE	W	DW_axi_dmac Channel-1 Enable Write Enable bit.

Bits	Name	Memory Access	Description
			Read back value of this register bit is always '0'. **Values:** • 0x1 (ENABLE_WR_CH1_EN): Enable Write to CH1_EN bit • 0x0 (DISABLE_WR_CH1_EN): Disable Write to respective CH1_EN bit **Value After Reset:** 0x0 **Volatile:** true
15	CH16_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-16. • 0: DW_axi_dmac Channel-16 is disabled • 1: DW_axi_dmac Channel-16 is enabled The bit 'DMAC_ChEnReg.CH16_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH16): DW_axi_dmac: Channel-16 is enabled • 0x0 (DISABLE_CH16): DW_axi_dmac: Channel-16 is disabled **Value After Reset:** 0x0 **Volatile:** true
14	CH15_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-15. • 0: DW_axi_dmac Channel-15 is disabled • 1: DW_axi_dmac Channel-15 is enabled The bit 'DMAC_ChEnReg.CH15_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH15): DW_axi_dmac: Channel-15 is enabled • 0x0 (DISABLE_CH15): DW_axi_dmac: Channel-15 is disabled **Value After Reset:** 0x0 **Volatile:** true
13	CH14_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-14. • 0: DW_axi_dmac Channel-14 is disabled • 1: DW_axi_dmac Channel-14 is enabled The bit 'DMAC_ChEnReg.CH14_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_CH14): DW_axi_dmac: Channel-14 is enabled • 0x0 (DISABLE_CH14): DW_axi_dmac: Channel-14 is disabled **Value After Reset:** 0x0 **Volatile:** true
12	CH13_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-13. • 0: DW_axi_dmac Channel-13 is disabled • 1: DW_axi_dmac Channel-13 is enabled The bit 'DMAC_ChEnReg.CH13_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH13): DW_axi_dmac: Channel-13 is enabled • 0x0 (DISABLE_CH13): DW_axi_dmac: Channel-13 is disabled **Value After Reset:** 0x0 **Volatile:** true
11	CH12_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-12. • 0: DW_axi_dmac Channel-12 is disabled • 1: DW_axi_dmac Channel-12 is enabled The bit 'DMAC_ChEnReg.CH12_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH12): DW_axi_dmac: Channel-12 is enabled • 0x0 (DISABLE_CH12): DW_axi_dmac: Channel-12 is disabled **Value After Reset:** 0x0 **Volatile:** true
10	CH11_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-11. • 0: DW_axi_dmac Channel-11 is disabled • 1: DW_axi_dmac Channel-11 is enabled The bit 'DMAC_ChEnReg.CH11_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ENABLE_CH11): DW_axi_dmac: Channel-11 is enabled • 0x0 (DISABLE_CH11): DW_axi_dmac: Channel-11 is disabled **Value After Reset:** 0x0 **Volatile:** true
9	CH10_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-10. • 0: DW_axi_dmac Channel-10 is disabled • 1: DW_axi_dmac Channel-10 is enabled The bit 'DMAC_ChEnReg.CH10_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH10): DW_axi_dmac: Channel-10 is enabled • 0x0 (DISABLE_CH10): DW_axi_dmac: Channel-10 is disabled **Value After Reset:** 0x0 **Volatile:** true
8	CH9_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-9. • 0: DW_axi_dmac Channel-9 is disabled • 1: DW_axi_dmac Channel-9 is enabled The bit 'DMAC_ChEnReg.CH9_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH9): DW_axi_dmac: Channel-9 is enabled • 0x0 (DISABLE_CH9): DW_axi_dmac: Channel-9 is disabled **Value After Reset:** 0x0 **Volatile:** true
7	CH8_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-8. • 0: DW_axi_dmac Channel-8 is disabled • 1: DW_axi_dmac Channel-8 is enabled The bit 'DMAC_ChEnReg.CH8_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH8): DW_axi_dmac: Channel-8 is enabled

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_CH8): DW_axi_dmac: Channel-8 is disabled **Value After Reset:** 0x0 **Volatile:** true
6	CH7_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-7. • 0: DW_axi_dmac Channel-7 is disabled • 1: DW_axi_dmac Channel-7 is enabled The bit 'DMAC_ChEnReg.CH7_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH7): DW_axi_dmac: Channel-7 is enabled • 0x0 (DISABLE_CH7): DW_axi_dmac: Channel-7 is disabled **Value After Reset:** 0x0 **Volatile:** true
5	CH6_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-6. • 0: DW_axi_dmac Channel-6 is disabled • 1: DW_axi_dmac Channel-6 is enabled The bit 'DMAC_ChEnReg.CH6_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH6): DW_axi_dmac: Channel-6 is enabled • 0x0 (DISABLE_CH6): DW_axi_dmac: Channel-6 is disabled **Value After Reset:** 0x0 **Volatile:** true
4	CH5_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-5. • 0: DW_axi_dmac Channel-5 is disabled • 1: DW_axi_dmac Channel-5 is enabled The bit 'DMAC_ChEnReg.CH5_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH5): DW_axi_dmac: Channel-5 is enabled • 0x0 (DISABLE_CH5): DW_axi_dmac: Channel-5 is disabled

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
3	CH4_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-4. • 0: DW_axi_dmac Channel-4 is disabled • 1: DW_axi_dmac Channel-4 is enabled The bit 'DMAC_ChEnReg.CH4_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH4): DW_axi_dmac: Channel-4 is enabled • 0x0 (DISABLE_CH4): DW_axi_dmac: Channel-4 is disabled **Value After Reset:** 0x0 **Volatile:** true
2	CH3_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-3. • 0: DW_axi_dmac Channel-3 is disabled • 1: DW_axi_dmac Channel-3 is enabled The bit 'DMAC_ChEnReg.CH3_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH3): DW_axi_dmac: Channel-3 is enabled • 0x0 (DISABLE_CH3): DW_axi_dmac: Channel-3 is disabled **Value After Reset:** 0x0 **Volatile:** true
1	CH2_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-2. • 0: DW_axi_dmac Channel-2 is disabled • 1: DW_axi_dmac Channel-2 is enabled The bit 'DMAC_ChEnReg.CH2_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH2): DW_axi_dmac: Channel-2 is enabled • 0x0 (DISABLE_CH2): DW_axi_dmac: Channel-2 is disabled **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
0	CH1_EN	R/W	This bit is used to enable the DW_axi_dmac Channel-1. • 0: DW_axi_dmac Channel-1 is disabled • 1: DW_axi_dmac Channel-1 is enabled The bit 'DMAC_ChEnReg.CH1_EN' is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Values:** • 0x1 (ENABLE_CH1): DW_axi_dmac: Channel-1 is enabled • 0x0 (DISABLE_CH1): DW_axi_dmac: Channel-1 is disabled **Value After Reset:** 0x0 **Volatile:** true

DMAC_CHSUSPREG(DMA0)

- **Name:** dmac Channel Suspend Register
- **Description:** This is dmac Channel Suspend Register. The channel suspend bit, DMAC_ChSuspReg.CH_SUSP, is written only if the corresponding channel write enable bit, DMAC_ChSuspReg.CH_SUSP_WE, is asserted on the same slave interface write transfer. For example, writing hex 0000_XXXX_0001_XXX1 writes a 1 into DMAC_ChSuspReg [0], while DMAC_ChSuspReg [15:1] and DMAC_ChSuspReg [47:32] remains unchanged. Writing hex 0000_XXXX_0000_XXXX leaves DMAC_ChSuspReg [15:0] and DMAC_ChSuspReg [47:32] unchanged.
- **Size:** 64 bits
- Offset: 0x20

Bits	Name	Memory Access	Description
63:28	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
27	CH12_SUSP_WE	W	This bit is used as a write enable to the Channel-12 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH12_SUSP): Enable Write to respective CH12_SUSP bit • 0x0 (DISABLE_WR_CH12_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_SUSP_WE	W	This bit is used as a write enable to the Channel-11 Suspend bit. The read back

Bits	Name	Memory Access	Description
			value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH11_SUSP): Enable Write to respective CH11_SUSP bit • 0x0 (DISABLE_WR_CH11_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_SUSP_WE	W	This bit is used as a write enable to the Channel-10 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH10_SUSP): Enable Write to respective CH10_SUSP bit • 0x0 (DISABLE_WR_CH10_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_SUSP_WE	W	This bit is used as a write enable to the Channel-9 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH9_SUSP): Enable Write to respective CH9_SUSP bit • 0x0 (DISABLE_WR_CH9_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
23	CH8_SUSP_WE	W	This bit is used as a write enable to the Channel-8 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH8_SUSP): Enable Write to respective CH8_SUSP bit • 0x0 (DISABLE_WR_CH8_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_SUSP_WE	W	This bit is used as a write enable to the Channel-7 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH7_SUSP): Enable Write to respective CH7_SUSP bit • 0x0 (DISABLE_WR_CH7_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
21	CH6_SUSP_WE	W	This bit is used as a write enable to the Channel-6 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH6_SUSP): Enable Write to respective CH6_SUSP bit • 0x0 (DISABLE_WR_CH6_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
20	CH5_SUSP_WE	W	This bit is used as a write enable to the Channel-5 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH5_SUSP): Enable Write to respective CH5_SUSP bit • 0x0 (DISABLE_WR_CH5_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
19	CH4_SUSP_WE	W	This bit is used as a write enable to the Channel-4 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH4_SUSP): Enable Write to respective CH4_SUSP bit • 0x0 (DISABLE_WR_CH4_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_SUSP_WE	W	This bit is used as a write enable to the Channel-3 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH3_SUSP): Enable Write to respective CH3_SUSP bit • 0x0 (DISABLE_WR_CH3_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_SUSP_WE	W	This bit is used as a write enable to the Channel-2 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH2_SUSP): Enable Write to respective CH2_SUSP bit • 0x0 (DISABLE_WR_CH2_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
16	CH1_SUSP_WE	W	This bit is used as a write enable to the Channel-1 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH1_SUSP): Enable Write to respective CH1_SUSP bit • 0x0 (DISABLE_WR_CH1_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
15:12	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
11	CH12_SUSP	R/W	Channel-12 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH12_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH12_SUSP bit to 1 and polls CH12_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH12_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH12_SUSP bit to 0, after DW_axi_dmac sets CH12_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
10	CH11_SUSP	R/W	Channel-11 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH11_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH11_SUSP bit to 1 and polls CH11_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH11_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH11_SUSP bit to 0, after DW_axi_dmac sets CH11_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled.

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
9	CH10_SUSP	R/W	Channel-10 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH10_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH10_SUSP bit to 1 and polls CH10_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH10_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH10_SUSP bit to 0, after DW_axi_dmac sets CH10_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Volatile:** true
8	CH9_SUSP	R/W	Channel-9 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH9_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH9_SUSP bit to 1 and polls CH9_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH9_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH9_SUSP bit to 0, after DW_axi_dmac sets CH9_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
7	CH8_SUSP	R/W	Channel-8 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH8_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH8_SUSP bit to 1 and polls CH8_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH8_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend.

Bits	Name	Memory Access	Description
			Software can clear CH8_SUSP bit to 0, after DW_axi_dmac sets CH8_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
6	CH7_SUSP	R/W	Channel-7 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH7_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH7_SUSP bit to 1 and polls CH7_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH7_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH7_SUSP bit to 0, after DW_axi_dmac sets CH7_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
5	CH6_SUSP	R/W	Channel-6 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH6_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH6_SUSP bit to 1 and polls CH6_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH6_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH6_SUSP bit to 0, after DW_axi_dmac sets CH6_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
4	CH5_SUSP	R/W	Channel-5 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH5_Status.CH_SUSPENDED to cleanly disable the channel without losing any

Bits	Name	Memory Access	Description
			data. In this case, software first sets CH5_SUSP bit to 1 and polls CH5_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH5_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH5_SUSP bit to 0, after DW_axi_dmac sets CH5_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
3	CH4_SUSP	R/W	Channel-4 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH4_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH4_SUSP bit to 1 and polls CH4_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH4_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH4_SUSP bit to 0, after DW_axi_dmac sets CH4_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
2	CH3_SUSP	R/W	Channel-3 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH3_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH3_SUSP bit to 1 and polls CH3_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH3_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH3_SUSP bit to 0, after DW_axi_dmac sets CH3_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
1	CH2_SUSP	R/W	Channel-2 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH2_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH2_SUSP bit to 1 and polls CH2_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH2_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH2_SUSP bit to 0, after DW_axi_dmac sets CH2_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true
0	CH1_SUSP	R/W	Channel-1 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH1_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH1_SUSP bit to 1 and polls CH1_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH1_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH1_SUSP bit to 0, after DW_axi_dmac sets CH1_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Value After Reset:** 0x0 **Volatile:** true

DMAC_CHSUSPREG (DMA1)

Bits	Name	Memory Access	Description
63:32	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
31	CH16_SUSP_WE	W	This bit is used as a write enable to the Channel-16 Suspend bit. The read back value of this register bit is always 0. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ENABLE_WR_CH16_SUSP): Enable Write to respective CH16_SUSP bit • 0x0 (DISABLE_WR_CH16_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
30	CH15_SUSP_WE	W	This bit is used as a write enable to the Channel-15 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH15_SUSP): Enable Write to respective CH15_SUSP bit • 0x0 (DISABLE_WR_CH15_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
29	CH14_SUSP_WE	W	This bit is used as a write enable to the Channel-14 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH14_SUSP): Enable Write to respective CH14_SUSP bit • 0x0 (DISABLE_WR_CH14_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
28	CH13_SUSP_WE	W	This bit is used as a write enable to the Channel-13 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH13_SUSP): Enable Write to respective CH13_SUSP bit • 0x0 (DISABLE_WR_CH13_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
27	CH12_SUSP_WE	W	This bit is used as a write enable to the Channel-12 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH12_SUSP): Enable Write to respective CH12_SUSP bit • 0x0 (DISABLE_WR_CH12_SUSP): Disable Write to CH${ch_num}_SUSP

Bits	Name	Memory Access	Description
			bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_SUSP_WE	W	This bit is used as a write enable to the Channel-11 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH11_SUSP): Enable Write to respective CH11_SUSP bit • 0x0 (DISABLE_WR_CH11_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_SUSP_WE	W	This bit is used as a write enable to the Channel-10 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH10_SUSP): Enable Write to respective CH10_SUSP bit • 0x0 (DISABLE_WR_CH10_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_SUSP_WE	W	This bit is used as a write enable to the Channel-9 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH9_SUSP): Enable Write to respective CH9_SUSP bit • 0x0 (DISABLE_WR_CH9_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
23	CH8_SUSP_WE	W	This bit is used as a write enable to the Channel-8 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH8_SUSP): Enable Write to respective CH8_SUSP bit • 0x0 (DISABLE_WR_CH8_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_SUSP_WE	W	This bit is used as a write enable to the Channel-7 Suspend bit. The read back value of this register bit is always 0.

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_WR_CH7_SUSP): Enable Write to respective CH7_SUSP bit • 0x0 (DISABLE_WR_CH7_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
21	CH6_SUSP_WE	W	This bit is used as a write enable to the Channel-6 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH6_SUSP): Enable Write to respective CH6_SUSP bit • 0x0 (DISABLE_WR_CH6_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
20	CH5_SUSP_WE	W	This bit is used as a write enable to the Channel-5 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH5_SUSP): Enable Write to respective CH5_SUSP bit • 0x0 (DISABLE_WR_CH5_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
19	CH4_SUSP_WE	W	This bit is used as a write enable to the Channel-4 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH4_SUSP): Enable Write to respective CH4_SUSP bit • 0x0 (DISABLE_WR_CH4_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_SUSP_WE	W	This bit is used as a write enable to the Channel-3 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH3_SUSP): Enable Write to respective CH3_SUSP bit • 0x0 (DISABLE_WR_CH3_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_SUSP_WE	W	This bit is used as a write enable to the Channel-2 Suspend bit. The read back value of this register bit is always 0.

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_WR_CH2_SUSP): Enable Write to respective CH2_SUSP bit • 0x0 (DISABLE_WR_CH2_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
16	CH1_SUSP_WE	W	This bit is used as a write enable to the Channel-1 Suspend bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH1_SUSP): Enable Write to respective CH1_SUSP bit • 0x0 (DISABLE_WR_CH1_SUSP): Disable Write to CH${ch_num}_SUSP bit **Value After Reset:** 0x0 **Volatile:** true
15	CH16_SUSP	R/W	Channel-16 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH16_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH16_SUSP bit to 1 and polls CH16_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH16_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH16_SUSP bit to 0, after DW_axi_dmac sets CH16_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH16_SUSP): Request to Suspended Channel-16 • 0x0 (DISABLE_CH16_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
14	CH15_SUSP	R/W	Channel-15 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH15_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH15_SUSP bit to 1 and polls CH15_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH15_EN bit to 0 to disable the channel.

Bits	Name	Memory Access	Description
			• 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH15_SUSP bit to 0, after DW_axi_dmac sets CH15_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH15_SUSP): Request to Suspended Channel-15 • 0x0 (DISABLE_CH15_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
13	CH14_SUSP	R/W	Channel-14 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH14_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH14_SUSP bit to 1 and polls CH14_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH14_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH14_SUSP bit to 0, after DW_axi_dmac sets CH14_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH14_SUSP): Request to Suspended Channel-14 • 0x0 (DISABLE_CH14_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
12	CH13_SUSP	R/W	Channel-13 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH13_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH13_SUSP bit to 1 and polls CH13_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH13_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH13_SUSP bit to 0, after DW_axi_dmac sets

Bits	Name	Memory Access	Description
			CH13_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH13_SUSP): Request to Suspended Channel-13 • 0x0 (DISABLE_CH13_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
11	CH12_SUSP	R/W	Channel-12 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH12_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH12_SUSP bit to 1 and polls CH12_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH12_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH12_SUSP bit to 0, after DW_axi_dmac sets CH12_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH12_SUSP): Request to Suspended Channel-12 • 0x0 (DISABLE_CH12_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
10	CH11_SUSP	R/W	Channel-11 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH11_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH11_SUSP bit to 1 and polls CH11_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH11_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH11_SUSP bit to 0, after DW_axi_dmac sets CH11_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ENABLE_CH11_SUSP): Request to Suspended Channel-11 • 0x0 (DISABLE_CH11_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
9	CH10_SUSP	R/W	Channel-10 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH10_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH10_SUSP bit to 1 and polls CH10_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH10_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH10_SUSP bit to 0, after DW_axi_dmac sets CH10_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH10_SUSP): Request to Suspended Channel-10 • 0x0 (DISABLE_CH10_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
8	CH9_SUSP	R/W	Channel-9 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH9_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH9_SUSP bit to 1 and polls CH9_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH9_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH9_SUSP bit to 0, after DW_axi_dmac sets CH9_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH9_SUSP): Request to Suspended Channel-9 • 0x0 (DISABLE_CH9_SUSP): No Channel Suspend Request **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
7	CH8_SUSP	R/W	Channel-8 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH8_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH8_SUSP bit to 1 and polls CH8_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH8_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH8_SUSP bit to 0, after DW_axi_dmac sets CH8_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH8_SUSP): Request to Suspended Channel-8 • 0x0 (DISABLE_CH8_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
6	CH7_SUSP	R/W	Channel-7 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH7_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH7_SUSP bit to 1 and polls CH7_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH7_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH7_SUSP bit to 0, after DW_axi_dmac sets CH7_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH7_SUSP): Request to Suspended Channel-7 • 0x0 (DISABLE_CH7_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
5	CH6_SUSP	R/W	Channel-6 Suspend Request.

Bits	Name	Memory Access	Description
			Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH6_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH6_SUSP bit to 1 and polls CH6_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH6_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH6_SUSP bit to 0, after DW_axi_dmac sets CH6_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH6_SUSP): Request to Suspended Channel-6 • 0x0 (DISABLE_CH6_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
4	CH5_SUSP	R/W	Channel-5 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH5_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH5_SUSP bit to 1 and polls CH5_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH5_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH5_SUSP bit to 0, after DW_axi_dmac sets CH5_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH5_SUSP): Request to Suspended Channel-5 • 0x0 (DISABLE_CH5_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
3	CH4_SUSP	R/W	Channel-4 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with

Bits	Name	Memory Access	Description
			CH4_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH4_SUSP bit to 1 and polls CH4_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH4_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH4_SUSP bit to 0, after DW_axi_dmac sets CH4_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH4_SUSP): Request to Suspended Channel-4 • 0x0 (DISABLE_CH4_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
2	CH3_SUSP	R/W	Channel-3 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH3_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH3_SUSP bit to 1 and polls CH3_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH3_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH3_SUSP bit to 0, after DW_axi_dmac sets CH3_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH3_SUSP): Request to Suspended Channel-3 • 0x0 (DISABLE_CH3_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
1	CH2_SUSP	R/W	Channel-2 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH2_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH2_SUSP bit to 1 and polls CH2_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH2_EN bit to 0 to disable the channel.

Bits	Name	Memory Access	Description
			• 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH2_SUSP bit to 0, after DW_axi_dmac sets CH2_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH2_SUSP): Request to Suspended Channel-2 • 0x0 (DISABLE_CH2_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true
0	CH1_SUSP	R/W	Channel-1 Suspend Request. Software sets this bit to 1 to request channel suspend. If this bit is set to 1, DW_axi_dmac suspends all DMA data transfers from the source gracefully until this bit is cleared. There is no guarantee that the current dma transaction will complete. This bit can also be used in conjunction with CH1_Status.CH_SUSPENDED to cleanly disable the channel without losing any data. In this case, software first sets CH1_SUSP bit to 1 and polls CH1_Status.CH_SUSPENDED till it is set to 1. Software can then clear CH1_EN bit to 0 to disable the channel. • 0: No Channel Suspend Request. • 1: Request for Channel Suspend. Software can clear CH1_SUSP bit to 0, after DW_axi_dmac sets CH1_Status.CH_SUSPENDED bit to 1, to exit the channel suspend mode. **Note**: CH_SUSP is cleared when channel is disabled. **Values:** • 0x1 (ENABLE_CH1_SUSP): Request to Suspended Channel-1 • 0x0 (DISABLE_CH1_SUSP): No Channel Suspend Request **Value After Reset:** 0x0 **Volatile:** true

DMAC_CHABORTREG(DMA0)

- **Name:** dmac Channel Abort Register
- **Description:** This is dmac Channel Abort Register. The channel abort bit, DMAC_ChAbortReg.CH_ABORT, is written only if the corresponding channel write enable bit, DMAC_ChAbortReg.CH_ABORT_WE, is asserted on the same slave interface write transfer.
- **Size:** 64 bits
- Offset: 0x28

Bits	Name	Memory Access	Description
63:28	RSVD	R	Reserved bits - Read Only

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
27	CH12_ABORT_WE	R	This bit is used to write enable the Channel-12 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH12_ABORT): Enable Write to CH12_ABORT bit • 0x0 (DISABLE_WR_CH12_ABORT): Disable Write to CH12_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_ABORT_WE	R	This bit is used to write enable the Channel-11 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH11_ABORT): Enable Write to CH11_ABORT bit • 0x0 (DISABLE_WR_CH11_ABORT): Disable Write to CH11_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_ABORT_WE	R	This bit is used to write enable the Channel-10 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH10_ABORT): Enable Write to CH10_ABORT bit • 0x0 (DISABLE_WR_CH10_ABORT): Disable Write to CH10_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_ABORT_WE	R	This bit is used to write enable the Channel-9 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH9_ABORT): Enable Write to CH9_ABORT bit • 0x0 (DISABLE_WR_CH9_ABORT): Disable Write to CH9_ABORT bit **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
23	CH8_ABORT_WE	R	This bit is used to write enable the Channel-8 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH8_ABORT): Enable Write to CH8_ABORT bit • 0x0 (DISABLE_WR_CH8_ABORT): Disable Write to CH8_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_ABORT_WE	R	This bit is used to write enable the Channel-7 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH7_ABORT): Enable Write to CH7_ABORT bit • 0x0 (DISABLE_WR_CH7_ABORT): Disable Write to CH7_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
21	CH6_ABORT_WE	R	This bit is used to write enable the Channel-6 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH6_ABORT): Enable Write to CH6_ABORT bit • 0x0 (DISABLE_WR_CH6_ABORT): Disable Write to CH6_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
20	CH5_ABORT_WE	R	This bit is used to write enable the Channel-5 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH5_ABORT): Enable Write to CH5_ABORT bit • 0x0 (DISABLE_WR_CH5_ABORT): Disable Write to CH5_ABORT bit **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
19	CH4_ABORT_WE	R	This bit is used to write enable the Channel-4 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH4_ABORT): Enable Write to CH4_ABORT bit • 0x0 (DISABLE_WR_CH4_ABORT): Disable Write to CH4_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_ABORT_WE	R	This bit is used to write enable the Channel-3 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH3_ABORT): Enable Write to CH3_ABORT bit • 0x0 (DISABLE_WR_CH3_ABORT): Disable Write to CH3_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_ABORT_WE	R	This bit is used to write enable the Channel-2 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH2_ABORT): Enable Write to CH2_ABORT bit • 0x0 (DISABLE_WR_CH2_ABORT): Disable Write to CH2_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
16	CH1_ABORT_WE	R	This bit is used to write enable the Channel-1 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH1_ABORT): Enable Write to CH1_ABORT bit • 0x0 (DISABLE_WR_CH1_ABORT): Disable Write to CH1_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
15:12	RSVD	R	CH_ABORT Reserved bits - Read Only

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
11	CH12_ABORT	R	Channel-12 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. dmac clears this bit to 0 once the channel is aborted (when it sets CH12_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH12_ABORT): Request for Channel-12 Abort • 0x0 (DISABLE_CH12_ABORT): No Request for Channel-12 Abort **Value After Reset:** 0x0 **Volatile:** true
10	CH11_ABORT	R	Channel-11 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH11_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH11_ABORT): Request for Channel-11 Abort • 0x0 (DISABLE_CH11_ABORT): No Request for Channel-11 Abort

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
9	CH10_ABORT	R	Channel-10 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH10_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH10_ABORT): Request for Channel-10 Abort • 0x0 (DISABLE_CH10_ABORT): No Request for Channel-10 Abort **Value After Reset:** 0x0 **Volatile:** true
8	CH9_ABORT	R	Channel-9 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH9_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH9_ABORT): Request for Channel-9 Abort • 0x0 (DISABLE_CH9_ABORT): No Request for Channel-9 Abort **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
7	CH8_ABORT	R	Channel-8 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH8_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH8_ABORT): Request for Channel-8 Abort • 0x0 (DISABLE_CH8_ABORT): No Request for Channel-8 Abort **Value After Reset:** 0x0 **Volatile:** true
6	CH7_ABORT	R	Channel-7 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH7_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH7_ABORT): Request for Channel-7 Abort • 0x0 (DISABLE_CH7_ABORT): No Request for Channel-7 Abort **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
5	CH6_ABORT	R	Channel-6 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH6_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH6_ABORT): Request for Channel-6 Abort • 0x0 (DISABLE_CH6_ABORT): No Request for Channel-6 Abort **Value After Reset:** 0x0 **Volatile:** true
4	CH5_ABORT	R	Channel-5 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH5_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH5_ABORT): Request for Channel-5 Abort • 0x0 (DISABLE_CH5_ABORT): No Request for Channel-5 Abort **Value After Reset:** 0x0 **Volatile:** true
3	CH4_ABORT	R	Channel-4 Abort Request.

Bits	Name	Memory Access	Description
			Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH4_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH4_ABORT): Request for Channel-4 Abort • 0x0 (DISABLE_CH4_ABORT): No Request for Channel-4 Abort **Value After Reset:** 0x0 **Volatile:** true
2	CH3_ABORT	R	Channel-3 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH3_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH3_ABORT): Request for Channel-3 Abort • 0x0 (DISABLE_CH3_ABORT): No Request for Channel-3 Abort **Value After Reset:** 0x0 **Volatile:** true
1	CH2_ABORT	R	Channel-2 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the

Bits	Name	Memory Access	Description
			channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH2_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH2_ABORT): Request for Channel-2 Abort • 0x0 (DISABLE_CH2_ABORT): No Request for Channel-2 Abort **Value After Reset:** 0x0 **Volatile:** true
0	CH1_ABORT	R	Channel-1 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH1_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH1_ABORT): Request for Channel-1 Abort • 0x0 (DISABLE_CH1_ABORT): No Request for Channel-1 Abort **Value After Reset:** 0x0 **Volatile:** true

DMAC_CHABORTREG (DMA1)

Bits	Name	Memory Access	Description
63:32	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
31	CH16_ABORT_WE	W	This bit is used to write enable the Channel-16 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH16_ABORT): Enable Write to CH16_ABORT bit • 0x0 (DISABLE_WR_CH16_ABORT): Disable Write to CH16_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
30	CH15_ABORT_WE	W	This bit is used to write enable the Channel-15 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH15_ABORT): Enable Write to CH15_ABORT bit • 0x0 (DISABLE_WR_CH15_ABORT): Disable Write to CH15_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
29	CH14_ABORT_WE	W	This bit is used to write enable the Channel-14 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH14_ABORT): Enable Write to CH14_ABORT bit • 0x0 (DISABLE_WR_CH14_ABORT): Disable Write to CH14_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
28	CH13_ABORT_WE	W	This bit is used to write enable the Channel-13 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH13_ABORT): Enable Write to CH13_ABORT bit

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_WR_CH13_ABORT): Disable Write to CH13_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
27	CH12_ABORT_WE	W	This bit is used to write enable the Channel-12 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH12_ABORT): Enable Write to CH12_ABORT bit • 0x0 (DISABLE_WR_CH12_ABORT): Disable Write to CH12_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
26	CH11_ABORT_WE	W	This bit is used to write enable the Channel-11 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH11_ABORT): Enable Write to CH11_ABORT bit • 0x0 (DISABLE_WR_CH11_ABORT): Disable Write to CH11_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
25	CH10_ABORT_WE	W	This bit is used to write enable the Channel-10 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH10_ABORT): Enable Write to CH10_ABORT bit • 0x0 (DISABLE_WR_CH10_ABORT): Disable Write to CH10_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
24	CH9_ABORT_WE	W	This bit is used to write enable the Channel-9 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH9_ABORT): Enable Write to CH9_ABORT bit • 0x0 (DISABLE_WR_CH9_ABORT): Disable Write to

Bits	Name	Memory Access	Description
			CH9_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
23	CH8_ABORT_WE	W	This bit is used to write enable the Channel-8 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH8_ABORT): Enable Write to CH8_ABORT bit • 0x0 (DISABLE_WR_CH8_ABORT): Disable Write to CH8_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
22	CH7_ABORT_WE	W	This bit is used to write enable the Channel-7 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH7_ABORT): Enable Write to CH7_ABORT bit • 0x0 (DISABLE_WR_CH7_ABORT): Disable Write to CH7_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
21	CH6_ABORT_WE	W	This bit is used to write enable the Channel-6 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH6_ABORT): Enable Write to CH6_ABORT bit • 0x0 (DISABLE_WR_CH6_ABORT): Disable Write to CH6_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
20	CH5_ABORT_WE	W	This bit is used to write enable the Channel-5 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH5_ABORT): Enable Write to CH5_ABORT bit • 0x0 (DISABLE_WR_CH5_ABORT): Disable Write to CH5_ABORT bit

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
19	CH4_ABORT_WE	W	This bit is used to write enable the Channel-4 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH4_ABORT): Enable Write to CH4_ABORT bit • 0x0 (DISABLE_WR_CH4_ABORT): Disable Write to CH4_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
18	CH3_ABORT_WE	W	This bit is used to write enable the Channel-3 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH3_ABORT): Enable Write to CH3_ABORT bit • 0x0 (DISABLE_WR_CH3_ABORT): Disable Write to CH3_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
17	CH2_ABORT_WE	W	This bit is used to write enable the Channel-2 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH2_ABORT): Enable Write to CH2_ABORT bit • 0x0 (DISABLE_WR_CH2_ABORT): Disable Write to CH2_ABORT bit **Value After Reset:** 0x0 **Volatile:** true
16	CH1_ABORT_WE	W	This bit is used to write enable the Channel-1 Abort bit. The read back value of this register bit is always 0. **Values:** • 0x1 (ENABLE_WR_CH1_ABORT): Enable Write to CH1_ABORT bit • 0x0 (DISABLE_WR_CH1_ABORT): Disable Write to CH1_ABORT bit **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
15	CH16_ABORT	R/W	Channel-16 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH16_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH16_ABORT): Request for Channel-16 Abort • 0x0 (DISABLE_CH16_ABORT): No Request for Channel-16 Abort **Value After Reset:** 0x0 **Volatile:** true
14	CH15_ABORT	R/W	Channel-15 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH15_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH15_ABORT): Request for Channel-15 Abort • 0x0 (DISABLE_CH15_ABORT): No Request for Channel-15 Abort **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
13	CH14_ABORT	R/W	Channel-14 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH14_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH14_ABORT): Request for Channel-14 Abort • 0x0 (DISABLE_CH14_ABORT): No Request for Channel-14 Abort **Value After Reset:** 0x0 **Volatile:** true
12	CH13_ABORT	R/W	Channel-13 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH13_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH13_ABORT): Request for Channel-13 Abort • 0x0 (DISABLE_CH13_ABORT): No Request for Channel-13 Abort **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
11	CH12_ABORT	R/W	Channel-12 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH12_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH12_ABORT): Request for Channel-12 Abort • 0x0 (DISABLE_CH12_ABORT): No Request for Channel-12 Abort **Value After Reset:** 0x0 **Volatile:** true
10	CH11_ABORT	R/W	Channel-11 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH11_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH11_ABORT): Request for Channel-11 Abort • 0x0 (DISABLE_CH11_ABORT): No Request for Channel-11 Abort **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
9	CH10_ABORT	R/W	Channel-10 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH10_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH10_ABORT): Request for Channel-10 Abort • 0x0 (DISABLE_CH10_ABORT): No Request for Channel-10 Abort **Value After Reset:** 0x0 **Volatile:** true
8	CH9_ABORT	R/W	Channel-9 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH9_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH9_ABORT): Request for Channel-9 Abort • 0x0 (DISABLE_CH9_ABORT): No Request for Channel-9 Abort **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
7	CH8_ABORT	R/W	Channel-8 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH8_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH8_ABORT): Request for Channel-8 Abort • 0x0 (DISABLE_CH8_ABORT): No Request for Channel-8 Abort **Value After Reset:** 0x0 **Volatile:** true
6	CH7_ABORT	R/W	Channel-7 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH7_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH7_ABORT): Request for Channel-7 Abort • 0x0 (DISABLE_CH7_ABORT): No Request for Channel-7 Abort **Value After Reset:** 0x0 **Volatile:** true
5	CH6_ABORT	R/W	Channel-6 Abort Request.

Bits	Name	Memory Access	Description
			Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH6_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH6_ABORT): Request for Channel-6 Abort • 0x0 (DISABLE_CH6_ABORT): No Request for Channel-6 Abort **Value After Reset:** 0x0 **Volatile:** true
4	CH5_ABORT	R/W	Channel-5 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH5_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH5_ABORT): Request for Channel-5 Abort • 0x0 (DISABLE_CH5_ABORT): No Request for Channel-5 Abort **Value After Reset:** 0x0 **Volatile:** true
3	CH4_ABORT	R/W	Channel-4 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the

Bits	Name	Memory Access	Description
			channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH4_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH4_ABORT): Request for Channel-4 Abort • 0x0 (DISABLE_CH4_ABORT): No Request for Channel-4 Abort **Value After Reset:** 0x0 **Volatile:** true
2	CH3_ABORT	R/W	Channel-3 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH3_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH3_ABORT): Request for Channel-3 Abort • 0x0 (DISABLE_CH3_ABORT): No Request for Channel-3 Abort **Value After Reset:** 0x0 **Volatile:** true
1	CH2_ABORT	R/W	Channel-2 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and

Bits	Name	Memory Access	Description
			should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH2_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH2_ABORT): Request for Channel-2 Abort • 0x0 (DISABLE_CH2_ABORT): No Request for Channel-2 Abort **Value After Reset:** 0x0 **Volatile:** true
0	CH1_ABORT	R/W	Channel-1 Abort Request. Software sets this bit to 1 to request channel abort. If this bit is set to 1, DW_axi_dmac disables the channel immediately. Aborting the channel might result in AXI Protocol violation as DW_axi_dmac does not make sure that all AXI transfers initiated on the master interface are completed.Aborting the channel is not recommended and should be used only in situations where a particular channel hangs due to no response from the corresponding AXI slave interface and software wants to disable the channel without resetting the entire DW_axi_dmac. It is recommended to try channel disabling first and then only opt for channel aborting. • 0: No Channel Abort Request. • 1: Request for Channel Abort. DW_axi_dmac clears this bit to 0 once the channel is aborted (when it sets CH1_Status.CH_ABORTED bit to 1). **Values:** • 0x1 (ENABLE_CH1_ABORT): Request for Channel-1 Abort • 0x0 (DISABLE_CH1_ABORT): No Request for Channel-1 Abort **Value After Reset:** 0x0 **Volatile:** true

DMAC_INTSTATUSREG2(DMA0)

- **Name:** dmac Combined Interrupt Status Register 2
- **Description:** DMAC Interrupt Status Register captures the combined channel interrupt for each channel and Combined common register block interrupt.
- **Size:** 64 bits
- Offset: 0x30

Bits	Name	Memory Access	Description
63:33	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
32	CommonReg_IntStat	R	Common Register Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Common Register Interrupt is Active • 0x0 (INACTIVE): Common Register Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
31:12	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
11	CH12_IntStat	R	Channel 12 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
10	CH11_IntStat	R	Channel 11 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
9	CH10_IntStat	R	Channel 10 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
8	CH9_IntStat	R	Channel 9 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active

Bits	Name	Memory Access	Description
			• 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
7	CH8_IntStat	R	Channel 8 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
6	CH7_IntStat	R	Channel 7 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
5	CH6_IntStat	R	Channel 6 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
4	CH5_IntStat	R	Channel 5 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
3	CH4_IntStat	R	Channel 4 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
2	CH3_IntStat	R	Channel 3 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
1	CH2_IntStat	R	Channel 2 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
0	CH1_IntStat	R	Channel 1 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true

DMAC_INTSTATUSREG2 (DMA1)

Bits	Name	Memory Access	Description
63:33	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
32	CommonReg_IntStat	R	Common Register Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Common Register Interrupt is Active • 0x0 (INACTIVE): Common Register Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
31:16	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
15	CH16_IntStat	R	Channel 16 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
14	CH15_IntStat	R	Channel 15 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
13	CH14_IntStat	R	Channel 14 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
12	CH13_IntStat	R	Channel 13 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
11	CH12_IntStat	R	Channel 12 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
10	CH11_IntStat	R	Channel 11 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active

Bits	Name	Memory Access	Description
			• 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
9	CH10_IntStat	R	Channel 10 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
8	CH9_IntStat	R	Channel 9 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
7	CH8_IntStat	R	Channel 8 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
6	CH7_IntStat	R	Channel 7 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
5	CH6_IntStat	R	Channel 6 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
4	CH5_IntStat	R	Channel 5 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
3	CH4_IntStat	R	Channel 4 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
2	CH3_IntStat	R	Channel 3 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
1	CH2_IntStat	R	Channel 2 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true
0	CH1_IntStat	R	Channel 1 Interrupt Status Bit. **Values:** • 0x1 (ACTIVE): Channel ${ch_num} Interrupt is Active • 0x0 (INACTIVE): Channel ${ch_num} Interrupt is Inactive **Value After Reset:** 0x0 **Volatile:** true

DMAC_COMMONREG_INTCLEARREG

- **Name:** dmac Common Register Space Interrupt Clear Register
- **Description:** Writing 1 to specific field clears the corresponding field in DMAC Common register Interrupt Status Register (DMAC_CommonReg_IntStatusReg).

- **Size:** 64 bits
- Offset: 0x38

Bits	Name	Memory Access	Description
63:21	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
20	Clear_MXIF2_BCH_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 2 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF2_BCH_EccPROT_UnCorrERR): Clear the MXIF2_BCH_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
19	Clear_MXIF2_BCH_EccPROT_CorrERR_IntStat	W	AXI Master Interface 2 Write Response Channel ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF2_BCH_EccPROT_CorrERR): Clear the MXIF2_BCH_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
18	Clear_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (CLEAR_MXIF2_RCH1_EccPROT_UnCorrERR): Clear the MXIF2_RCH1_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
17	Clear_MXIF2_RCH1_EccPROT_CorrERR_IntStat	W	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF2_RCH1_EccPROT_CorrERR): Clear the MXIF2_RCH1_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
16	Clear_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 2 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF2_RCH0_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF2_RCH0_EccPROT_UnCorrERR): Clear the MXIF2_RCH0_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
15	Clear_MXIF2_RCH0_EccPROT_CorrERR_IntStat	W	AXI Master Interface 2 Read Channel (Data) ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel

Bits	Name	Memory Access	Description
			interrupt status bit (MXIF2_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF2_RCH0_EccPROT_CorrERR): Clear the MXIF2_RCH0_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
14	Clear_MXIF1_BCH_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 1 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_BCH_EccPROT_UnCorrERR): Clear the MXIF1_BCH_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
13	Clear_MXIF1_BCH_EccPROT_CorrERR_IntStat	W	AXI Master Interface 1 Write Response Channel ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_BCH_EccPROT_CorrERR): Clear the MXIF1_BCH_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
12	Clear_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_RCH1_EccPROT_UnCorrERR): Clear the MXIF1_RCH1_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
11	Clear_MXIF1_RCH1_EccPROT_CorrERR_IntStat	W	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_RCH1_EccPROT_CorrERR): Clear the MXIF1_RCH1_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
10	Clear_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat	W	AXI Master Interface 1 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_RCH0_EccPROT_UnCorrERR): Clear the MXIF1_RCH0_EccPROT_UnCorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg

Bits	Name	Memory Access	Description
			• 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
9	Clear_MXIF1_RCH0_EccPROT_CorrERR_IntStat	W	AXI Master Interface 1 Read Channel (Data) ECC Protection related Correctable Error Interrupt Clear bit. This bit is used to clear the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (CLEAR_MXIF1_RCH0_EccPROT_CorrERR): Clear the MXIF1_RCH0_EccPROT_CorrERR_IntStat interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
8	Clear_SLVIF_UndefinedReg_DEC_ERR_IntStat	W	Slave Interface Undefined register Decode Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit(SLVIF_UndefinedReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (CLEAR_SLVIF_UndefinedReg_DEC_ERR): Clear the SLVIF_UndefinedReg_DEC_ERR interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
7	Clear_SLVIF_CommonReg_WRPARITY_ERR_IntStat	W	Slave Interface Common Register Write Parity Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit(SLVIF_CommonReg_WRPARITY_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (CLEAR_SLVIF_CommonReg_WRPARITY_ERR): Clear the SLVIF_CommonReg_WRPARITY_ERR

Bits	Name	Memory Access	Description
			interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
6:4	RSVD	W	Reserved bits - Read Only **Value After Reset:** 0x0
3	Clear_SLVIF_CommonReg_WrOnHold_ERR_IntStat	W	Slave Interface Common Register Write On Hold Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit(SLVIF_CommonReg_WrOnHold_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (CLEAR_SLVIF_CommonReg_WrOnHold_ERR): Clear the SLVIF_CommonReg_WrOnHold_ERR interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
2	Clear_SLVIF_CommonReg_RD2WO_ERR_IntStat	W	Slave Interface Common Register Read to Write only Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit(SLVIF_CommonReg_RD2WO_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (CLEAR_SLVIF_CommonReg_RD2WO_ERR): Clear the SLVIF_CommonReg_RD2WO_ERR interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
1	Clear_SLVIF_CommonReg_WR2RO_ERR_IntStat	W	Slave Interface Common Register Write to Read only Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit(SLVIF_CommonReg_WR2RO_ERR_IntStat in DMAC_CommonReg_IntStatusReg.

Bits	Name	Memory Access	Description
			Values: • 0x1 (CLEAR_SLVIF_CommonReg_WR2RO_ERR): Clear the SLVIF_CommonReg_WR2RO_ERR interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
0	Clear_SLVIF_CommonReg_DEC_ERR_IntStat	W	Slave Interface Common Register Decode Error Interrupt clear Bit. This bit is used to clear the corresponding channel interrupt status bit (SLVIF_CommonReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (CLEAR_SLVIF_CommonReg_DEC_ERR): Clear the SLVIF_CommonReg_DEC_ERR interrupt in the interrupt register DMAC_CommonReg_IntStatusReg • 0x0 (No_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

DMAC_COMMONREG_INTSTATUS_ENABLEREG

- **Name:** dmac Common Register Space Interrupt Enable Register
- **Description:** Writing 1 to specific field enables the corresponding interrupt status generation in DMAC Common register Interrupt Status Register (DMAC_CommonReg_IntStatusReg).
- **Size:** 64 bits
- Offset: 0x40

Bits	Name	Memory Access	Description
63:21	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x7fffffffffff
20	Enable_MXIF2_BCH_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 2 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1

Bits	Name	Memory Access	Description
			(ENABLE_MXIF2_BCH_EccPROT_UnCorrERR_IntStat): MXIF2_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_BCH_EccPROT_UnCorrERR_IntStat): MXIF2_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
19	Enable_MXIF2_BCH_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Write Response Channel ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF2_BCH_EccPROT_CorrERR_IntStat): MXIF2_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_BCH_EccPROT_CorrERR_IntStat): MXIF2_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
18	Enable_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat): MXIF2_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat):

Bits	Name	Memory Access	Description
			MXIF2_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
17	Enable_MXIF2_RCH1_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF2_RCH1_EccPROT_CorrERR_IntStat): MXIF2_RCH1_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_RCH1_EccPROT_CorrERR_IntStat): MXIF2_RCH1_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
16	Enable_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH0_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat): MXIF2_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat): MXIF2_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
15	Enable_MXIF2_RCH0_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Data) ECC Protection related Correctable Error Interrupt Status bit.

Bits	Name	Memory Access	Description
			This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF2_RCH0_EccPROT_CorrERR_IntStat): MXIF2_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF2_RCH0_EccPROT_CorrERR_IntStat): MXIF2_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
14	Enable_MXIF1_BCH_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_BCH_EccPROT_UnCorrERR_IntStat): MXIF1_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_BCH_EccPROT_UnCorrERR_IntStat): MXIF1_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
13	Enable_MXIF1_BCH_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Write Response Channel ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_BCH_EccPROT_CorrERR_In

Bits	Name	Memory Access	Description
			tStat): MXIF1_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_BCH_EccPROT_CorrERR_IntStat): MXIF1_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
12	Enable_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat): MXIF1_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat): MXIF1_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
11	Enable_MXIF1_RCH1_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_RCH1_EccPROT_CorrERR_IntStat): MXIF1_RCH1_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_RCH1_EccPROT_CorrERR_IntStat): MXIF1_RCH1_EccPROT_CorrERR_IntStat bit

Bits	Name	Memory Access	Description
			in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
10	Enable_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat): MXIF1_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat): MXIF1_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
9	Enable_MXIF1_RCH0_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Data) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg). **Values:** • 0x1 (ENABLE_MXIF1_RCH0_EccPROT_CorrERR_IntStat): MXIF1_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_MXIF1_RCH0_EccPROT_CorrERR_IntStat): MXIF1_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
8	Enable_SLVIF_UndefinedReg_DEC_ERR_IntStat	R/W	Slave Interface Undefined register Decode Error Interrupt Status enable Bit. This bit is used to enable the corresponding channel interrupt status bit

Bits	Name	Memory Access	Description
			(SLVIF_UndefinedReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_UndefinedReg_DEC_ERR): SLVIF_UndefinedReg_DEC_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_UndefinedReg_DEC_ERR): SLVIF_UndefinedReg_DEC_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
7	Enable_SLVIF_CommonReg_WRPARITY_ERR_IntStat	R	Slave Interface Common Register Write Parity Error Interrupt Status Enable Bit. This bit is used to enable the corresponding channel interrupt status bit (SLVIF_CommonReg_WRPARITY_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WRPARITY_ERR): SLVIF_CommonReg_WRPARITY_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_CommonReg_WRPARITY_ERR): SLVIF_CommonReg_WRPARITY_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
6:4	RSVD_DMAC_COMMONREG_INTSTATUS_ENABLEREG_6to4	R	DMAC Common Register Interrupt Status Enable Register (bits 6to4) Reserved bits - Read Only **Value After Reset:** 0x7
3	Enable_SLVIF_CommonReg_WrOnHold_ERR_IntStat	R/W	Slave Interface Common Register Write On Hold Error Interrupt Status Enable Bit. This bit is used to enable the corresponding channel interrupt status bit (SLVIF_CommonReg_WrOnHold_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WrOnHold_ERR): SLVIF_CommonReg_WrOnHold_ERR_IntStat

Bits	Name	Memory Access	Description
			bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_CommonReg_WrOnHold_ERR): SLVIF_CommonReg_WrOnHold_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
2	Enable_SLVIF_CommonReg_RD2WO_ERR_IntStat	R/W	Slave Interface Common Register Read to Write only Error Interrupt Status Enable Bit. This bit is used to enable the corresponding channel interrupt status bit (SLVIF_CommonReg_RD2WO_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_RD2WO_ERR): SLVIF_CommonReg_RD2WO_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_CommonReg_RD2WO_ERR): SLVIF_CommonReg_RD2WO_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
1	Enable_SLVIF_CommonReg_WR2RO_ERR_IntStat	R/W	Slave Interface Common Register Write to Read only Error Interrupt Status Enable Bit. This bit is used to enable the corresponding channel interrupt status bit (SLVIF_CommonReg_WR2RO_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WR2RO_ERR): SLVIF_CommonReg_WR2RO_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_CommonReg_WR2RO_ERR): SLVIF_CommonReg_WR2RO_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1
0	Enable_SLVIF_CommonReg_DEC_ERR_IntStat	R/W	Slave Interface Common Register Decode Error Interrupt Status Enable Bit. This bit is used to enable the corresponding channel

Bits	Name	Memory Access	Description
			interrupt status bit (SLVIF_CommonReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_DEC_ERR): SLVIF_CommonReg_DEC_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled • 0x0 (DISABLE_SLVIF_CommonReg_DEC_ERR): SLVIF_CommonReg_DEC_ERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Disabled **Value After Reset:** 0x1

DMAC_COMMONREG_INTSIGNAL_ENABLEREG

- **Name:** dmac Common Register Space Interrupt Signal Enable Register
- **Description:** Writing 1 to specific field will propagate the corresponding interrupt status in DMAC Common register Interrupt Status Register (DMAC_CommonReg_IntStatusReg) to generate an port level interrupt.
- **Size:** 64 bits
- Offset: 0x48

Bits	Name	Memory Access	Description
63:21	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x7fffffffffff
20	Enable_MXIF2_BCH_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 2 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_BCH_EccPROT_UnCorrERR_IntStat): MXIF2_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF2_BCH_EccPROT_UnCorrERR_IntStat): MXIF2_BCH_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level

Bits	Name	Memory Access	Description
			Value After Reset: 0x1
19	Enable_MXIF2_BCH_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 2 Write Response Channel ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_BCH_EccPROT_CorrERR_IntStat): MXIF2_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF2_BCH_EccPROT_CorrERR_IntStat): MXIF2_BCH_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
18	Enable_MXIF2_RCH1_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat): MXIF2_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF2_RCH1_EccPROT_UnCorrERR_IntStat): MXIF2_RCH1_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
17	Enable_MXIF2_RCH1_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port

Bits	Name	Memory Access	Description
			level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_RCH1_EccPROT_CorrERR_IntStat): MXIF2_RCH1_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF2_RCH1_EccPROT_CorrERR_IntStat): MXIF2_RCH1_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
16	Enable_MXIF2_RCH0_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 2 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH0_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat): MXIF2_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF2_RCH0_EccPROT_UnCorrERR_IntStat): MXIF2_RCH0_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
15	Enable_MXIF2_RCH0_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 2 Read Channel (Data) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF2_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF2_RCH0_EccPROT_CorrERR_IntStat): MXIF2_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled

Bits	Name	Memory Access	Description
			at port level • 0x0 (DISABLE_MXIF2_RCH0_EccPROT_CorrERR_IntStat): MXIF2_RCH0_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
14	Enable_MXIF1_BCH_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 1 Write Response Channel ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_BCH_EccPROT_UnCorrERR_IntStat): MXIF1_BCH_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_BCH_EccPROT_UnCorrERR_IntStat): MXIF1_BCH_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
13	Enable_MXIF1_BCH_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 1 Write Response Channel ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_BCH_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_BCH_EccPROT_CorrERR_IntStat): MXIF1_BCH_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_BCH_EccPROT_CorrERR_IntStat): MXIF1_BCH_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1

Bits	Name	Memory Access	Description
12	Enable_MXIF1_RCH1_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_UnCorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat): MXIF1_RCH1_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_RCH1_EccPROT_UnCorrERR_IntStat): MXIF1_RCH1_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
11	Enable_MXIF1_RCH1_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH1_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_RCH1_EccPROT_CorrERR_IntStat): MXIF1_RCH1_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_RCH1_EccPROT_CorrERR_IntStat): MXIF1_RCH1_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
10	Enable_MXIF1_RCH0_EccPROT_UnCorrERR_IntSignal	R	AXI Master Interface 1 Read Channel (Data) ECC Protection related Uncorrectable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_UnCorrERR_IntStat in

Bits	Name	Memory Access	Description
			DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat): MXIF1_RCH0_EccPROT_UnCorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_RCH0_EccPROT_UnCorrERR_IntStat): MXIF1_RCH0_EccPROT_UnCorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
9	Enable_MXIF1_RCH0_EccPROT_CorrERR_IntSignal	R	AXI Master Interface 1 Read Channel (Data) ECC Protection related Correctable Error Interrupt Status bit. This bit is used to enable the corresponding channel interrupt status bit (MXIF1_RCH0_EccPROT_CorrERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_MXIF1_RCH0_EccPROT_CorrERR_IntStat): MXIF1_RCH0_EccPROT_CorrERR_IntStat bit in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_MXIF1_RCH0_EccPROT_CorrERR_IntStat): MXIF1_RCH0_EccPROT_CorrERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
8	Enable_SLVIF_UndefinedReg_DEC_ERR_IntSignal	R/W	Slave Interface Undefined register Decode Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit(SLVIF_UndefinedReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_UndefinedReg_DEC_ERR_IntSignal): SLVIF_UndefinedReg_DEC_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_SLVIF_UndefinedReg_DEC_ERR_IntSignal): SLVIF_UndefinedReg_DEC_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
7	Enable_SLVIF_CommonReg_WRPARITY_ERR_IntSignal	R	Slave Interface Write Parity Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit(SLVIF_CommonReg_WRPARITY_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WRPARITY_ERR_IntSignal): SLVIF_CommonReg_WRPARITY_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_SLVIF_CommonReg_WRPARITY_ERR_IntSignal): SLVIF_CommonReg_WRPARITY_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
6:4	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x7
3	Enable_SLVIF_CommonReg_WrOnHold_ERR_IntSignal	R/W	Slave Interface Common Register Write On Hold Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit(SLVIF_CommonReg_WrOnHold_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WrOnHold_ERR_IntSignal): SLVIF_CommonReg_WrOnHold_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_SLVIF_CommonReg_WrOnHold_ERR_IntSignal): SLVIF_CommonReg_WrOnHold_ERR_IntStat

Bits	Name	Memory Access	Description
			signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
2	Enable_SLVIF_CommonReg_RD2WO_ERR_IntSignal	R/W	Slave Interface Common Register Read to Write only Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit (SLVIF_CommonReg_RD2WO_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_RD2WO_ERR_IntSignal): SLVIF_CommonReg_RD2WO_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_SLVIF_CommonReg_RD2WO_ERR_IntSignal): SLVIF_CommonReg_RD2WO_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
1	Enable_SLVIF_CommonReg_WR2RO_ERR_IntSignal	R/W	Slave Interface Common Register Write to Read only Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit (SLVIF_CommonReg_WR2RO_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_WR2RO_ERR_IntSignal): SLVIF_CommonReg_WR2RO_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_SLVIF_CommonReg_WR2RO_ERR_IntSignal): SLVIF_CommonReg_WR2RO_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1
0	Enable_SLVIF_CommonReg_DEC_ERR_IntSignal	R/W	Slave Interface Common Register Decode Error Interrupt Signal Enable Bit. This bit is used to enable the propagation of corresponding channel interrupt status bit

Bits	Name	Memory Access	Description
			(SLVIF_CommonReg_DEC_ERR_IntStat in DMAC_CommonReg_IntStatusReg) to generate a port level interrupt. **Values:** • 0x1 (ENABLE_SLVIF_CommonReg_DEC_ERR_IntSignal): SLVIF_CommonReg_DEC_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Enabled at port level • 0x0 (DISABLE_SLVIF_CommonReg_DEC_ERR_IntSignal): SLVIF_CommonReg_DEC_ERR_IntStat signal in DMAC_CommonReg_IntStatusReg is Disabled at port level **Value After Reset:** 0x1

DMAC_COMMONREG_INTSTATUSREG

- **Name:** dmac Common Register Space Interrupt Status Register
- **Description:** This Register captures Slave interface access errors.
- **Size:** 64 bits
- Offset: 0x50

Bits	Name	Memory Access	Description
63:21	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
20	MXIF2_BCH_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 2 Write Response Channel ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Write Response payload. • 0: No AXI Master 2 Write Response Channel Uncorrectable Error. • 1: AXI Master 2 Write Response Channel Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
19	MXIF2_BCH_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Write Response Channel ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on

Bits	Name	Memory Access	Description
			the AXI Channel Write Response payload. • 0: No AXI Master 2 Write Response Channel Correctable Error. • 1: AXI Master 2 Write Response Channel Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
18	MXIF2_RCH1_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Read Control payload. • 0: No AXI Master 2 Read Channel Control signals related Uncorrectable Error. • 1: AXI Master 2 Read Channel Control signals related Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
17	MXIF2_RCH1_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Other Control signals) ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on the AXI Channel Read Control payload. • 0: No AXI Master 2 Read Channel Control signlas related Correctable Error. • 1: AXI Master 2 Read Channel Control signals related Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
16	MXIF2_RCH0_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Data) ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Read Data payload. • 0: No AXI Master 2 Read Channel Data related Uncorrectable Error. • 1: AXI Master 2 Read Channel Data related Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
15	MXIF2_RCH0_EccPROT_CorrERR_IntStat	R	AXI Master Interface 2 Read Channel (Data) ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on the AXI Channel Read Data payload. • 0: No AXI Master 2 Read Channel Data related Correctable Error. • 1: AXI Master 2 Read Channel Data related Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
14	MXIF1_BCH_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Write Response Channel ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Write Response payload. • 0: No AXI Master 1 Write Response Channel Uncorrectable Error. • 1: AXI Master 1 Write Response Channel Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register

Bits	Name	Memory Access	Description
			is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
13	MXIF1_BCH_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Write Response Channel ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on the AXI Channel Write Response payload. • 0: No AXI Master 1 Write Response Channel Correctable Error. • 1: AXI Master 1 Write Response Channel Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
12	MXIF1_RCH1_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Read Control payload. • 0: No AXI Master 1 Read Channel Control signals related Uncorrectable Error. • 1: AXI Master 1 Read Channel Control signals related Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
11	MXIF1_RCH1_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Other Control signals) ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on the AXI Channel Read Control payload. • 0: No AXI Master 1 Read Channel Control signlas related Correctable Error.

Bits	Name	Memory Access	Description
			• 1: AXI Master 1 Read Channel Control signals related Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
10	MXIF1_RCH0_EccPROT_UnCorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Data) ECC Protection Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the AXI Channel Read Data payload. • 0: No AXI Master 1 Read Channel Data related Uncorrectable Error. • 1: AXI Master 1 Read Channel Data related Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
9	MXIF1_RCH0_EccPROT_CorrERR_IntStat	R	AXI Master Interface 1 Read Channel (Data) ECC Protection Correctable Error Interrupt Status bit. This error occurs if ECC Correctable error is detected on the AXI Channel Read Data payload. • 0: No AXI Master 1 Read Channel Data related Correctable Error. • 1: AXI Master 1 Read Channel Data related Correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_COMMONREG_INTSTATUS_ENABLEREG register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG. **Value After Reset:** 0x0 **Volatile:** true
8	SLVIF_UndefinedReg_DEC_ERR_IntStat	R	Slave Interface Undefined register Decode Error Interrupt Signal Enable Bit. Decode Error generated by DW_axi_dmac during register

Bits	Name	Memory Access	Description
			access. This error occurs if the register access is to undefined address range (>0x8FF if 8 channels are configured, >0x4FF if 4 channels are configured etc.) resulting in error response by DW_axi_dmac slave interface. • 0: No Slave Interface Decode Errors. • 1: Slave Interface Decode Error detected. Error Interrupt Status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in DMAC_COMMONREG_INTCLEARREG on enabling the channel (required when the interrupt is not enabled). **Value After Reset:** 0x0 **Volatile:** true
7	SLVIF_CommonReg_WRPARITY_ERR_IntStat	R	Slave Interface Common Register Write Parity Error Interrupt Status Bit. This error occurs if write operation with data and parity is performed to a register in the common register space, but the locally computed parity bits not matching the received parity bits. • 0: No Common Register Space Write Parity Error. • 1: Common Register Space Write Parity Error detected. Error Interrupt Status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in DMAC_COMMONREG_INTCLEARREG (required when the interrupt is not enabled. **Value After Reset:** 0x0 **Volatile:** true
6:4	RSVD	R	DMAC Common Register Interrupt Status Register (bits 6to4) Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
3	SLVIF_CommonReg_WrOnHold_ERR_IntStat	R	Slave Interface Common Register Write On Hold Error Interrupt Status Bit. This error occurs if an illegal write operation is performed on a common register; this happens if a write operation is performed on a common register except DMAC_RESETREG with DMAC_RST field set to 1 when DW_axi_dmac is in Hold mode. • 0: No Slave Interface Common Register Write On

Bits	Name	Memory Access	Description
			Hold Errors. • 1: Slave Interface Common Register Write On Hold Error detected. Error Interrupt Status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in DMAC_COMMONREG_INTCLEARREG on enabling the channel (required when the interrupt is not enabled). **Value After Reset:** 0x0 **Volatile:** true
2	SLVIF_CommonReg_RD2WO_ERR_IntStat	R	Slave Interface Common Register Read to Write only Error Interrupt Status bit. This error occurs if Read operation is performed to a Write Only register in the common register space (0x000 to 0x0FF). • 0: No Slave Interface Read to Write Only Errors. • 1: Slave Interface Read to Write Only Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in DMAC_COMMONREG_INTCLEARREG on enabling the channel (required when the interrupt is not enabled). **Value After Reset:** 0x0 **Volatile:** true
1	SLVIF_CommonReg_WR2RO_ERR_IntStat	R	Slave Interface Common Register Write to Read Only Error Interrupt Status bit. This error occurs if write operation is performed to a Read Only register in the common register space (0x000 to 0x0FF). • 0: No Slave Interface Write to Read Only Errors. • 1: Slave Interface Write to Read Only Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in DMAC_COMMONREG_INTCLEARREG on enabling the channel (required when the interrupt is not enabled). **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
0	SLVIF_CommonReg_DEC_ERR_IntStat	R	Slave Interface Common Register Decode Error Interrupt Status Bit. Decode Error generated by DW_axi_dmac during register access. This error occurs if the register access is to an invalid address in the common register space (0x000 to 0x0FF) resulting in error response by DW_axi_dmac slave interface. • 0: No Slave Interface Decode Errors. • 1: Slave Interface Decode Error detected. The Error Interrupt status is generated if the corresponding Status Enable bit in DMAC_CommonReg_IntStatus_Enable register bit is set to 1. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in DMAC_COMMONREG_INTCLEARREG on enabling the channel (required when the interrupt is not enabled). **Value After Reset:** 0x0 **Volatile:** true

DMAC_RESETREG

- **Name:** dmac Software Reset Register
- **Description:** This register is used to initiate the Software Reset to dmac.
- **Size:** 64 bits
- Offset: 0x58

Bits	Name	Memory Access	Description
63:1	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
0	DMAC_RST	R/W	DMAC Reset Request bit Software writes 1 to this bit to reset the axi_dmac and polls this bit to see it as 0. axi_dmac resets all the modules except the slave bus interface module and clears this bit to 0. **NOTE**: Software is not allowed to write 0 to this bit. **Value After Reset:** 0x0 **Volatile:** true

CHx_SAR

- **Name:** dmac Channel x Source Address Register
- **Description:** The starting source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current AXI transfer.
- **Size:** 64 bits
- Offset: 0x100

Bits	Name	Memory Access	Description
63:0	SAR	R/W	Current Source Address of DMA transfer. Updated after each source transfer. The SINC fields in the CHx_CTL register determines whether the address increments or is left unchanged on every source transfer throughout the block transfer. **Value After Reset:** 0x0 **Volatile:** true

CHx_DAR

- **Name:** dmac Channel x Destination Address Register
- **Description:** The starting destination address is programmed by the software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current AXI transfer.
- **Size:** 64 bits
- Offset: 0x108

Bits	Name	Memory Access	Description
63:0	DAR	R/W	Current Destination Address of DMA transfer. Updated after each destination transfer. The DINC fields in the CHx_CTL register determines whether the address increments or is left unchanged on every destination transfer throughout the block transfer. **Value After Reset:** 0x0 **Volatile:** true

CHx_BLOCK_TS

- **Name:** dmac Channel x Block Transfer Size Register
- **Description:** When dmac is the flow controller, the DMAC uses this register before the channel is enabled for block-size.
- **Size:** 64 bits
- Offset: 0x110

Bits	Name	Memory Access	Description
63:22	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
21:0	BLOCK_TS	R/W	Block Transfer Size. The number programmed into BLOCK_TS field indicates the total number of data of width CHx_CTL.SRC_TR_WIDTH to be transferred in a DMA block transfer. Block Transfer Size = BLOCK_TS+1 When the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of who is the flow controller. When the source or destination peripheral is assigned as the flow controller, the value before the transfer starts saturates at DMAX_CHx_MAX_BLK_SIZE, but the actual block size can

Bits	Name	Memory Access	Description
			be greater. **Value After Reset:** 0x0 **Volatile:** true

CHx_CTL

- **Name:** dmac Channel x Control Register
- **Description:** This register contains fields that control the DMA transfer. This register should be programmed prior to enabling the channel except for LLI-based multi-block transfer. When LLI-based multi-block transfer is enabled, the CHx_CTL register is loaded from the corresponding location of the LLI and it can be varied on a block-by-block basis within a DMA transfer. The software is not allowed to directly update this register through dmac slave interface. Any write to this register during LLI based multi-block transfer is ignored.
- **Size:** 64 bits
- Offset: 0x118

Bits	Name	Memory Access	Description
63	SHADOWREG_OR_LLI_VALID	R/W	Shadow Register content/Linked List Item valid. Indicates whether the content of shadow register or the linked list item fetched from the memory is valid. • 0: Shadow Register content/LLI is invalid. • 1: Last Shadow Register/LLI is valid. **LLI based multi-block transfer**: The CHx_CTL register is loaded from the LLI. Hence, the software is not allowed to directly update this register through the DW_axi_dmac slave interface. This field can be used to dynamically extend the LLI by the software. On noticing this bit as 0, DW_axi_dmac discards the LLI and generates the ShadowReg_Or_LII_Invalid_ERR Interrupt if the corresponding channel error interrupt mask bit is set to 0. In the case of LLI pre-fetching, the ShadowReg_Or_LLI_Invalid_ERR interrupt is not generated even if the ShadowReg_Or_LLI_Valid bit is seen to be 0 for the pre-fetched LLI. In this case, DW_axi_dmac attempts the LLI fetch operation again after completing the current block transfer and generates the ShadowReg_Or_LII_Invalid_ERR interrupt only if ShadowReg_Or_LII_Valid bit is still seen to be 0. This error condition causes the DW_axi_dmac to halt the corresponding channel gracefully. DW_axi_dmac waits until software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid LLI availability before attempting another LLI read operation. This bit is cleared to 0 and written back to the corresponding LLI location after block transfer completion when LLI write-back option is enabled. Hence, for LLI-based multi-block transfers, the software might manipulate/redefine any descriptor with the ShadowReg_Or_LII_Valid bit set to 0 if LLI write-back option is enabled. **Shadow Reg based multi-block transfer**: On noticing this bit as 0 during shadow register fetch phase, DW_axi_dmac discards the Shadow Register contents and generates ShadowReg_Or_LLI_Invlid_ERR Interrupt. In this case, the software has to write (any value) to CHx_BLK_TFR_ResumeReqReg after updating the shadow registers and after setting

Bits	Name	Memory Access	Description
			ShadowReg_Or_LLI_Valid bit to 1 to indicate to DW_axi_dmac that shadow register contents are valid and the next block transfer can be resumed. DW_axi_dmac clears this bit to 0 after copying the shadow register contents. Software can reprogram the shadow registers only if ShadowReg_Or_LLI_Valid bit is 0. Software needs to read this register in block completion interrupt service routine (if interrupt is enabled)/continuously poll this register (if interrupt is not enabled) to make sure that this bit is 0 before updating the shadow registers. If shadow-register-based multi-block transfer is enabled and software attempts to write to the shadow register when ShadowReg_Or_LLI_Valid bit is 1, DW_axi_dmac generates SLVIF_ShadowReg_WrOnValid_ERR interrupt. **Values:** • 0x1 (VALID): Indicates shadowreg/LLI content is Valid • 0x0 (INVALID): Indicates shadowreg/LLI content is Invalid **Value After Reset:** 0x0 **Volatile:** true
62	SHADOWREG_OR_LLI_LAST	R/W	Last Shadow Register/Linked List Item. Indicates whether shadow register content or the linked list item fetched from the memory is the last one or not. • 0: Not last Shadow Register/LLI • 1: Last Shadow Register/LLI **LLI based multi-block transfer**: DW_axi_dmac uses this bit to decide if another LLI fetch is needed in the current DMA transfer. • If this bit is 0, DW_axi_dmac fetches the next LLI from the address pointed out by LLP field in the current LLI. • If this bit is 1, DW_axi_dmac understands that current block is the final block in the dma transfer and ends the dma transfer once the AMBA transfer corresponding to the current block completes. **Shadow Reg based multi-block transfer**: DW_axi_dmac uses this bit to decide if another Shadow Register fetch is needed in the current DMA transfer. • If this bit is 0, DW_axi_dmac understands that there are one or more blocks to be transferred in the current block and hence one or more shadow register set contents will be valid and needs to be fetched. • If this bit is 1, DW_axi_dmac understands that current block is the final block in the dma transfer and ends the dma transfer once the AMBA transfer corresponding to the current block completes. **Values:** • 0x1 (LAST_ITEM): Indicates shadowreg/LLI content is the last one

Bits	Name	Memory Access	Description
			• 0x0 (NOT_LAST_ITEM): Indicates shadowreg/LLI content is not the last one **Value After Reset:** 0x0 **Volatile:** true
61:59	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
58	IOC_BlkTfr	R/W	Interrupt On completion of Block Transfer This bit is used to control the block transfer completion interrupt generation on a block by block basis for shadow register or linked list based multi-block transfers. Writing 1 to this register field enables CHx_IntStatusReg.BLOCK_TFR_DONE_IntStat field if this interrupt generation is enabled in CHx_IntStatus_EnableReg register and the external interrupt output is is asserted if this interrupt generation is enabled in CHx_IntSignal_EnableReg register. **Note**: If a linked-list or shadow-register-based multi-block transfer is not used for both source and destination (for instance if source and destination use contiguous address or auto-reload-based multi-block transfer), the value of this field cannot be modified per block. Additionally, the value programmed before the channel is enabled is used for all the blocks in the DMA transfer. **Values:** • 0x1 (Enable_BLKTFR_INTR): Enables CHx_IntStatusReg.BLOCK_TFR_DONE_IntStat field • 0x0 (DISABLE_BLKTFR_INTR): Disables CHx_IntStatusReg.BLOCK_TFR_DONE_IntStat field **Value After Reset:** 0x0
57	DST_STAT_EN	R/W (for dma0/1 channel 1 and 2) R(for all other channels)	Destination Status Enable Enable the logic to fetch status from destination peripheral of channel x pointed to by the content of CHx_DSTATAR register and stores it in CHx_DSTAT register. This value is written back to the CHx_DSTAT location of linked list at end of each block transfer if DMAX_CHx_LLI_WB_EN is set to 1 and if linked list based multi-block transfer is used by either source or destination peripheral. **Values:** • 0x1 (Enable_STAT_FETCH): Enables status fetch for Destination and store the value in CH1_DSTAT register • 0x0 (NO_STAT_FETCH): No status fetch for Destination device **Value After Reset:** 0x0
56	SRC_STAT_EN	R/W (for dma0/1 channel 1 and 2) R(for all other	Source Status Enable Enable the logic to fetch status from source peripheral of channel x pointed to by the content of CHx_SSTATAR register and stores it in CHx_SSTAT register. This value is written back to the CHx_SSTAT location of linked list at end of each block transfer if

Bits	Name	Memory Access	Description
		channels)	DMAX_CHx_LLI_WB_EN is set to 1 and if linked list based multi-block transfer is used by either source or destination peripheral. **Values:** • 0x1 (Enable_STAT_FETCH): Enables status fetch for Source and store the value in CH1_SSTAT register • 0x0 (NO_STAT_FETCH): No status fetch for Source device **Value After Reset:** 0x0
55:48	AWLEN	R/W	Destination Burst Length AXI Burst length used for destination data transfer. The specified burst length is used for destination data transfer till the extent possible; remaining transfers use maximum possible value that is less than or equal to 16. The maximum value of AWLEN is limited by 16. **Note**: The AWLEN setting may not be honored towards end-to-block transfers, the end of a transaction (only applicable to non-memory peripharals), and during 4K boundary crossings. **Value After Reset:** 0x0
47	AWLEN_EN	R/W	Destination Burst Length Enable If this bit is set to 1, DW_axi_dmac uses the value of CHx_CTL.AWLEN as AXI Burst length for destination data transfer till the extent possible; remaining transfers use maximum possible burst length. If this bit is set to 0, DW_axi_dmac uses any possible value which is less than or equal to 16 as AXI Burst length for destination data transfer. **Values:** • 0x1 (Enable): AXI Burst Length is CH1_CTL.AWLEN (till the extent possible) for Destination data transfers • 0x0 (Disable): AXI Burst Length is any possible value <= 16 for Destination data transfers **Value After Reset:** 0x0
46:39	ARLEN	R/W	Source Burst Length AXI Burst length used for source data transfer. The specified burst length is used for source data transfer till the extent possible; remaining transfers use maximum possible value that is less than or equal to16. The maximum value of ARLEN is limited by 16. **Value After Reset:** 0x0
38	ARLEN_EN	R/W	Source Burst Length Enable If this bit is set to 1, DW_axi_dmac uses the value of CHx_CTL.ARLEN as AXI Burst length for source data transfer till the extent possible; remaining transfers use maximum possible burst length. If this bit is set to 0, DW_axi_dmac uses any possible value that is less

Bits	Name	Memory Access	Description
			than or equal to 16 as AXI Burst length for source data transfer. **Values:** • 0x1 (Enable): AXI Burst Length is CH1_CTL.ARLEN (till the extent possible) for Source data transfers • 0x0 (Disable): AXI Burst Length is any possible value <= 16 for Source data transfers **Value After Reset:** 0x0
37:35	AW_PROT	R/W	AXI 'aw_prot' signal.Not use **Value After Reset:** 0x0
34:32	AR_PROT	R/W	AXI 'ar_prot' signal.Not use **Value After Reset:** 0x0
31	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
30	NonPosted_LastWrite_En	R/W	Non Posted Last Write Enable This bit decides whether posted writes can be used throughout the block transfer. • 0: Posted writes may be used throughout the block transfer. • 1: Posted writes may be used till the end of the block (inside a block) and the last write in the block must be non-posted. This is to synchronize block completion interrupt generation to the last write data reaching the end memory/peripheral. **Values:** • 0x1 (Enable): Last write in the block must be non-posted • 0x0 (Disable): Posted writes may be used throughout the block transfer **Value After Reset:** 0x0
29:26	AW_CACHE	R/W	AXI 'aw_cache' signal **Value After Reset:** 0x0
25:22	AR_CACHE	R/W	AXI 'ar_cache' signal **Value After Reset:** 0x0
21:18	DST_MSIZE	R/W	Destination Burst Transaction Length. Number of data items, each of width CHx_CTL.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from the corresponding hardware or software handshaking interface. **Note**: This Value is not related to the AXI awlen signal. **Values:**

Bits	Name	Memory Access	Description
			• 0x0 (DATA_ITEM_1): 1 Data Item read from Destination in the burst transaction • 0x1 (DATA_ITEMS_4): 4 Data Item read from Destination in the burst transaction • 0x2 (DATA_ITEMS_8): 8 Data Item read from Destination in the burst transaction • 0x3 (DATA_ITEMS_16): 16 Data Item read from Destination in the burst transaction • 0x4 (DATA_ITEMS_32): 32 Data Item read from Destination in the burst transaction • 0x5 (DATA_ITEMS_64): 64 Data Item read from Destination in the burst transaction • 0x6 (DATA_ITEMS_128): 128 Data Item read from Destination in the burst transaction • 0x7 (DATA_ITEMS_256): 256 Data Item read from Destination in the burst transaction • 0x8 (DATA_ITEMS_512): 512 Data Item read from Destination in the burst transaction • 0x9 (DATA_ITEMS_1024): 1024 Data Item read from Destination in the burst transaction **Value After Reset:** 0x0
17:14	SRC_MSIZE	R/W	Source Burst Transaction Length. Number of data items, each of width CHx_CTL.SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from the corresponding hardware or software handshaking interface. The maximum value of DST_MSIZE is limited by 64. **Note**: This Value is not related to the AXI arlen signal. **Values:** • 0x0 (DATA_ITEM_1): 1 Data Item read from Source in the burst transaction • 0x1 (DATA_ITEMS_4): 4 Data Item read from Source in the burst transaction • 0x2 (DATA_ITEMS_8): 8 Data Item read from Source in the burst transaction • 0x3 (DATA_ITEMS_16): 16 Data Item read from Source in the burst transaction • 0x4 (DATA_ITEMS_32): 32 Data Item read from Source in the burst transaction • 0x5 (DATA_ITEMS_64): 64 Data Item read from Source in the burst transaction • 0x6 (DATA_ITEMS_128): 128 Data Item read from Source in the burst transaction

Bits	Name	Memory Access	Description
			• 0x7 (DATA_ITEMS_256): 256 Data Item read from Source in the burst transaction • 0x8 (DATA_ITEMS_512): 512 Data Item read from Source in the burst transaction • 0x9 (DATA_ITEMS_1024): 1024 Data Item read from Source in the burst transaction **Value After Reset:** 0x0
13:11	DST_TR_WIDTH	R/W	Destination Transfer Width. Mapped to AXI bus awsize, this value must be less than or equal to 32. **Values:** • 0x0 (BITS_8): Destination Transfer Width is 8 bits • 0x1 (BITS_16): Destination Transfer Width is 16 bits • 0x2 (BITS_32): Destination Transfer Width is 32 bits • 0x3 (BITS_64): Destination Transfer Width is 64 bits **Value After Reset:** 0x2
10:8	SRC_TR_WIDTH	R/W	Source Transfer Width. Mapped to AXI bus arsize, this value must be less than or equal to 32. **Values:** • 0x0 (BITS_8): Source Transfer Width is 8 bits • 0x1 (BITS_16): Source Transfer Width is 16 bits • 0x2 (BITS_32): Source Transfer Width is 32 bits • 0x3 (BITS_64): Source Transfer Width is 64 bits **Value After Reset:** 0x2
7	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
6	DINC	R/W	Destination Address Increment. Indicates whether to increment the destination address on every destination transfer. If the device is writing data from a source peripheral FIFO with a fixed address, then set this field to 'No change'. • 0: Increment • 1: No Change **NOTE**: Increment aligns the address to the next CHx_CTL.DST_TR_WIDTH boundary. **Values:** • 0x0 (INCREMENTAL): Destination address incremented on every source transfer

Bits	Name	Memory Access	Description
			• 0x1 (FIXED): Destination address is fixed **Value After Reset:** 0x0
5	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
4	SINC	R/W	Source Address Increment. Indicates whether to increment the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to 'No change'. • 0: Increment • 1: No Change **NOTE**: Increment aligns the address to the next CHx_CTL.SRC_TR_WIDTH boundary. **Values:** • 0x0 (INCREMENTAL): Source address incremented on every source transfer • 0x1 (FIXED): Source address is fixed **Value After Reset:** 0x0
3	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
2	DMS	R/W (for dma1) R(for dma0)	Destination Master Select. Identifies the Master Interface layer from which the destination device (peripheral or memory) is accessed. • 0: AXI master 1 • 1: AXI Master 2 **Values:** • 0x1 (MASTER2_INTF): Destination device on Master-2 interface layer • 0x0 (MASTER1_INTF): Destination device on Master-1 interface layer **Value After Reset:** 0x0
1	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
0	SMS	R/W (for dma1) R(for dma0)	Source Master Select. Identifies the Master Interface layer from which the source device (peripheral or memory) is accessed. • 0: AXI master 1

Bits	Name	Memory Access	Description
			• 1: AXI Master 2 **Values:** • 0x1 (MASTER2_INTF): Source device on Master-2 interface layer • 0x0 (MASTER1_INTF): Source device on Master-1 interface layer **Value After Reset:** 0x0

CHx_CFG2

- **Name:** dmac Channel x Configuration Register 2
- **Description:** This register contains fields that configure the DMA transfer. This register should be programmed prior to enabling the channel. Bits [63:32] of the channel configuration register remains fixed for all blocks of a multi-block transfer and can be programmed only when channel is disabled. Bits [3:0] of the channel configuration register can be programmed even when channel is enabled. Software clears these bits to end the multi-block transfers. For Contiguous-Address and Auto-Reloading-based multi-block transfers (if neither source nor destination peripheral uses Shadow-Register or Linked-List-based multi-block transfers), if the corresponding multi-block type selection bits namely CHx_CFG.SRC_MLTBLK_TYPE and/or CHx_CFG.DST_MLTBLK_TYPE bits are seen to be 2'b00 at the end of a block transfer, the DW_axi_dmac understands that the previous block was the final block in the transfer and completes the DMA transfer operation.
- **Size:** 64 bits
- Offset: 0x120

Bits	Name	Memory Access	Description
63	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x0 **Volatile:** true
62:59	DST_OSR_LMT	R/W	Destination Outstanding Request Limit • Maximum outstanding request supported is 16. • Source Outstanding Request Limit = DST_OSR_LMT + 1 **Value After Reset:** 0x0 **Volatile:** true
58:55	SRC_OSR_LMT	R/W	Source Outstanding Request Limit • Maximum outstanding request supported is 16. • Source Outstanding Request Limit = SRC_OSR_LMT + 1 **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
54:53	LOCK_CH_L	R	Channel Lock Level This bit indicates the duration over which CHx_CFG.LOCK_CH bit applies. • 00: Over complete DMA transfer • 01: Over DMA block transfer • 1x: Reserved LOCK_EN=0, the read-back value is always 0. **Values:** • 0x0 (DMA_transfer_CH_LOCK): Duration of the Channel locking is for the entire DMA transfer • 0x1 (BLOCK_TRANFER_CH_LOCK): Duration of the Channel locking is for the current block transfer **Value After Reset:** 0x0 **Volatile:** true
52	LOCK_CH	R	Channel Lock bit When the channel is granted control of the master bus interface and if the CHx_CFG.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CHx_CFG.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CHx_CFG.LOCK_CH_L. LOCK_EN=0, the read-back value is always 0. Locking the channel locks AXI Read Address, Write Address and Write Data channels on the corresponding master interface. **Note**: Channel locking feature is supported only for memory-to-memory transfer at Block Transfer and DMA Transfer levels. Hardware does not check for the validity of channel locking setting, hence the software must take care of enabling the channel locking only for memory-to-memory transfers at Block Transfer or DMA Transfer levels. Illegal programming of channel locking might result in unpredictable behavior. **Values:** • 0x0 (NO_CHANNEL_LOCK): Channel is not locked during the transfers • 0x0 (CHANNEL_LOCK): Channel is locked and granted exclusive access to the Master Bus Interface **Value After Reset:** 0x0 **Volatile:** true
51:47	CH_PRIOR	R/W	Channel Priority A priority of 11 is the highest priority, and 0 is the lowest. This field must be programmed within the following range:

Bits	Name	Memory Access	Description
			0: 11 A programmed value outside this range will cause erroneous behavior. **Value After Reset:** 0xb **Volatile:** true
46:39	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
38	DST_HWHS_POL	R	Destination Hardware Handshaking Interface Polarity. • 0: ACTIVE HIGH • 1: ACTIVE LOW **Values:** • 0x0 (ACTIVE_HIGH): Polarity of the Handshaking Interface used for the Destination peripheral is Active High • 0x1 (ACTIVE_LOW): Polarity of the Handshaking Interface used for the Destination peripheral is Active Low **Value After Reset:** 0x0 **Volatile:** true
37	SRC_HWHS_POL	R	Source Hardware Handshaking Interface Polarity. • 0: ACTIVE HIGH • 1: ACTIVE LOW **Values:** • 0x0 (ACTIVE_HIGH): Polarity of the Handshaking Interface used for the Source peripheral is Active High • 0x1 (ACTIVE_LOW): Polarity of the Handshaking Interface used for the Source peripheral is Active Low **Value After Reset:** 0x0 **Volatile:** true
36	HS_SEL_DST	R/W	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces (hardware or software) is active for destination requests on this channel. • 0: Hardware handshaking interface. Software-initiated transaction requests are ignored. • 1: Software handshaking interface. Hardware-initiated transaction requests are ignored.

Bits	Name	Memory Access	Description
			If the destination peripheral is memory, then this bit is ignored. **Values:** • 0x0 (HARDWARE_HS): Hardware Handshaking Interface is used for the Destination peripheral • 0x1 (SOFTWARE_HS): Software Handshaking Interface is used for the Destination peripheral **Value After Reset:** 0x1 **Volatile:** true
35	HS_SEL_SRC	R/W	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces (hardware or software) is active for source requests on this channel. • 0: Hardware handshaking interface. Software-initiated transaction requests are ignored. • 1: Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored. **Values:** • 0x0 (HARDWARE_HS): Hardware Handshaking Interface is used for the Source peripheral • 0x1 (SOFTWARE_HS): Software Handshaking Interface is used for the Source peripheral **Value After Reset:** 0x1 **Volatile:** true
34:32	TT_FC	R/W	Transfer Type and Flow Control. The following transfer types are supported. • Memory to Memory • Memory to Peripheral • Peripheral to Memory • Peripheral to Peripheral Flow Control can be assigned to the DW_axi_dmac, the source peripheral, or hte destination peripheral. **Values:** • 0x0 (MEM_TO_MEM_DMAC): Transfer Type is memory to memory and Flow Controller is DW_axi_dmac • 0x1 (MEM_TO_PER_DMAC): Transfer Type is memory to peripheral and Flow Controller is DW_axi_dmac • 0x2 (PER_TO_MEM_DMAC): Transfer Type is peripheral to memory and Flow Controller is DW_axi_dmac

Bits	Name	Memory Access	Description
			• 0x3 (PER_TO_PER_DMAC): Transfer Type is peripheral to peripheral and Flow Controller is DW_axi_dmac • 0x4 (PER_TO_MEM_SRC): Transfer Type is peripheral to Memory and Flow Controller is Source peripheral • 0x5 (PER_TO_PER_SRC): Transfer Type is peripheral to peripheral and Flow Controller is Source peripheral • 0x6 (MEM_TO_PER_DST): Transfer Type is memory to peripheral and Flow Controller is Destination peripheral • 0x7 (PER_TO_PER_DST): Transfer Type is peripheral to peripheral and Flow Controller is Destination peripheral **Value After Reset:** 0x3 **Volatile:** true
31:15	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
14:11	DST_PER	R/W	Assigns a hardware handshaking interface (0 - 15) to the destination of Channelx if the CHx_CFG.HS_SEL_DST field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DW_axi_dmac operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. This field does not exist if the configuration parameter DMAX_NUM_HS_IF is set to 0. x = 11 if DMAC_NUM_HS_IF is 1 x = ceil(log2(DMAC_NUM_HS_IF)) + 10 if DMAC_NUM_HS_IF is greater than 1 Bits 16: (x+1) do not exist and return 0 on a read. **Value After Reset:** 0x0 **Volatile:** true
10:8	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
7:4	SRC_PER	R/W	Assigns a hardware handshaking interface (0 - 15) to the source of Channelx if the CHx_CFG.HS_SEL_SRC field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DW_axi_dmac operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. This field does not exist if the configuration parameter

Bits	Name	Memory Access	Description
			DMAX_NUM_HS_IF is set to 0. x = 4 if DMAC_NUM_HS_IF is 1 x = ceil(log2(DMAC_NUM_HS_IF) + 3 if DMAC_NUM_HS_IF is greater than 1. Bits 9: (x+1) do not exist and return 0 on a read. **Value After Reset:** 0x0 **Volatile:** true
3:2	DST_MULTBLK_TYPE	R/W	Destination Multi Block Transfer Type. These bits define the type of multi-block transfer used for destination peripheral. • 00: Contiguous • 01: Reload • 10: Shadow Register • 11: Linked List If the type selected is Contiguous, the CHx_DAR register is loaded with the value of the end source address of previous block + 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Reload, the CHx_DAR register is reloaded from the initial value of DAR at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Shadow Register, the CHx_DAR register is loaded from the content of its shadow register if CHx_CTL.ShadowReg_Or_LLI_Valid bit is set to 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Linked List, the CHx_DAR register is loaded from the Linked List if CTL.ShadowReg_Or_LLI_Valid bit is set to 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. CHx_CTL and CHx_BLOCK_TS registers are loaded from their initial values or from the contents of their shadow registers (if CHx_CTL.ShadowReg_Or_LLI_Valid bit is set to 1) or from the linked list (if CTL.ShadowReg_Or_LLI_Valid bit is set to 1) at the end of every block for multi-block transfers based on the multi-block transfer type programmed for source and destination peripherals. Contiguous transfer on both source and destination peripheral is not a valid multi-block transfer configuration. This field does not exist if the configuration parameter DMAX_CHx_MULTI_BLK_EN is not selected; in that case, the read-back value is always 0. **Values:** • 0x0 (CONTINGUOUS): Contiguous Multiblock Type used for Destination Transfer

Bits	Name	Memory Access	Description
			• 0x1 (RELOAD): Reload Multiblock Type used for Destination Transfer • 0x2 (SHADOW_REGISTER): Shadow Register based Multiblock Type used for Destination Transfer • 0x3 (LINKED_LIST): Linked List based Multiblock Type used for Destination Transfer **Value After Reset:** 0x0 **Volatile:** true
1:0	SRC_MULTBLK_TYPE	R/W	Source Multi Block Transfer Type. These bits define the type of multi-block transfer used for source peripheral. • 00: Contiguous • 01: Reload • 10: Shadow Register • 11: Linked List If the type selected is Contiguous, the CHx_SAR register is loaded with the value of the end source address of previous block + 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Reload, the CHx_SAR register is reloaded from the initial value of SAR at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Shadow Register, the CHx_SAR register is loaded from the content of its shadow register if CHx_CTL.ShadowReg_Or_LLI_Valid bit is set to 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. If the type selected is Linked List, the CHx_SAR register is loaded from the Linked List if CTL.ShadowReg_Or_LLI_Valid bit is set to 1 at the end of every block for multi-block transfers. A new block transfer is then initiated. CHx_CTL and CHx_BLOCK_TS registers are loaded from their initial values or from the contents of their shadow registers (if CHx_CTL.ShadowReg_Or_LLI_Valid bit is set to 1) or from the linked list (if CTL.ShadowReg_Or_LLI_Valid bit is set to 1) at the end of every block for multi-block transfers based on the multi-block transfer type programmed for source and destination peripherals. Contiguous transfer on both source and destination peripheral is not a valid multi-block transfer configuration. This field does not exist if the configuration parameter DMAX_CHx_MULTI_BLK_EN is not selected; in that case, the read-back value is always 0. **Values:** • 0x0 (CONTINGUOUS): Contiguous Multiblock Type

Bits	Name	Memory Access	Description
			used for Source Transfer • 0x1 (RELOAD): Reload Multiblock Type used for Source Transfer • 0x2 (SHADOW_REGISTER): Shadow Register based Multiblock Type used for Source Transfer • 0x3 (LINKED_LIST): Linked List based Multiblock Type used for Source Transfer **Value After Reset:** 0x0 **Volatile:** true

CHx_LLP

- **Name:** dmac Channel x Linked List Pointer Register
- **Description:** This is the Linked List Pointer register. This register must be programmed to point to the first Linked List Item (LLI) in memory prior to enabling the channel if linked-list-based block chaining is enabled. This register is updated with new value of linked list pointer during the LLI update stage of dma transfer.
- **Size:** 64 bits
- Offset: 0x128

Bits	Name	Memory Access	Description
63:6	LOC	R/W	Starting Address Memory of LLI block Starting Address In Memory of next LLI if block chaining is enabled. The six LSBs of the starting address are not stored because the address is assumed to be aligned to a 64-byte boundary. LLI access always uses the burst size (arsize/awsize) that is same as the data bus width and cannot be changed or programmed to anything other than this. Burst length (awlen/arlen) is chosen based on the data bus width so that the access does not cross one complete LLI structure of 64 bytes. DW_axi_dmac will fetch the entire LLI (40 bytes) in one AXI burst if the burst length is not limited by other settings. **Value After Reset:** 0x0 **Volatile:** true
5:1	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
0	LMS	R/W (for dma1) R (for dma0)	LLI master Select This bit identifies the AXI layer/interface where the memory device that stores the next linked list item resides. • 0: AXI Master 1 • 1: AXI Master 2 the read-back value is always the 0. **Values:**

Bits	Name	Memory Access	Description
			• 0x0 (MASTER1_INTF): next Linked List item resides on AXI Master1 interface • 0x1 (MASTER2_INTF): next Linked List item resides on AXI Master2 interface **Value After Reset:** 0x0 **Volatile:** true

CHx_STATUSREG

- **Name:** dmac Channel x Status Register
- **Description:** Channelx Status Register contains fields that indicate the status of DMA transfers for Channelx.
- **Size:** 64 bits
- Offset: 0x130

Bits	Name	Memory Access	Description
63:47	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
46:32	DATA_LEFT_IN_FIFO	R	Data Left in FIFO. This bit indicates the total number of data left in DW_axi_dmac channel FIFO after completing the current block transfer. The width of the data in channel FIFO is equal to CHx_CTL.SRC_TR_WIDTH. For normal block transfer completion without errors, Data_Left_In_FIFO = 0. If any error occurs during the dma transfer, the block transfer might be terminated early and in such a case, Data_Left_In_FIFO indicates the data remaining in channel FIFO which could not be transferred to destination peripheral. This field is cleared to zero on enabling the channel. **Note**: If CHx_CTL.DST_TR_WIDTH > CHx_CTL.SRC_TR_WIDTH, there may be residual data left in the FIFO which is not enough to form one CHx_CTL.SRC_TR_WIDTH of data and Data_Left_In_FIFO will return 0 in this case. **Value After Reset:** 0x0 **Volatile:** true
31:22	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
21:0	CMPLTD_BLK_TFR_SIZE	R	Completed Block Transfer Size.

Bits	Name	Memory Access	Description
			This bit indicates the total number of data of width CHx_CTL.SRC_TR_WIDTH transferred for the previous block transfer. For normal block transfer completion without any errors, this value will be equal to the value programmed in BLOCK_TS field of CHx_BLOCK_TS register. If any error occurs during the dma transfer, the block transfer might be terminated early and in such a case, this value indicates the actual data transferred without error in the current block. This field is cleared to zero on enabling the channel. **Value After Reset:** 0x0 **Volatile:** true

CHx_SWHSSRCREG

- **Name:** dmac Channel x Software Handshake Source Register
- **Description:** Channelx Software handshake Source Register.
- **Size:** 64 bits
- Offset: 0x138

Bits	Name	Memory Access	Description
63:6	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
5	SWHS_LST_SRC_WE	W	Write Enable bit for Software Handshake Last Request for Channel Source. **Values:** • 0x1 (ENABLE_SWHS_LAST_SRC): Enables write to the SWHS_LAST_SRC bit • 0x0 (DISABLE_SWHS_LAST_SRC): Disables write to the SWHS_LAST_SRC bit **Value After Reset:** 0x0 **Volatile:** true
4	SWHS_LST_SRC	R/W	Software Handshake Last Request for Channel Source. This bit is used to request LAST dma source data transfer if software handshaking method is selected for the source of the corresponding channel. This bit is ignored if software handshaking is not enabled for the source of the Channelx or if the source of Channelx is not the flow controller. CHx_SWHSSrcReg.SWHS_Req_Src bit must be set to 1 for DW_axi_dmac to treat it as a valid software handshaking request. If CHx_SWHSSrcReg.SWHS_SglReq_Src is set to 1, the LAST request is

Bits	Name	Memory Access	Description
			for SINGLE dma transaction (AXI burst length = 1), else the request is treated as a BURST transaction request. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSSrcReg.SWHS_Ack_Src bit and sees it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_Lst_Src bit is written only if the corresponding write enable bit, SWHS_Lst_Src_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSSrcReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_LAST_SRC): Source peripheral indication to dmac that the current transfer is the last transfer • 0x0 (INACTIVE_SWHS_LAST_SRC): Source peripheral indication that the curent transfer is not the last transfer **Value After Reset:** 0x0 **Volatile:** true
3	SWHS_SGLREQ_SRC_WE	W	Write Enable bit for Software Handshake Single Request for Channel Source. **Values:** • 0x1 (ENABLE_SWHS_SGLREQ_SRC): Enables write to the SWHS_SGLREQ_SRC bit • 0x0 (DISABLE_SWHS_SGLREQ_SRC): Disables write to the SWHS_SGLREQ_SRC bit **Value After Reset:** 0x0 **Volatile:** true
2	SWHS_SGLREQ_SRC	R/W	Software Handshake Single Request for Channel Source. This bit is used to request SINGLE (AXI burst length = 1) dma source data transfer if software handshaking method is selected for the source of the corresponding channel. This bit is ignored if software handshaking is not enabled for the source of the Channelx. The functionality of this field depends on whether the peripheral is the flow controller. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSSrcReg.SWHS_Ack_Src bit and sees it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_SglReq_Src bit is written only if the corresponding write enable bit, SWHS_SglReq_Src_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSSrcReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_SGLREQ_SRC): Source peripheral request for a single dma transfer

Bits	Name	Memory Access	Description
			• 0x0 (INACTIVE_SWHS_SGLREQ_SRC): Source peripheral is not requesting for a single transfer **Value After Reset:** 0x0 **Volatile:** true
1	SWHS_REQ_SRC_WE	W	Write Enable bit for Software Handshake Request for Channel Source. **Note**: This bit always returns 0 on a read back. **Values:** • 0x1 (ENABLE_SWHS_REQ_SRC): Enables write to the SWHS_REQ_SRC bit • 0x0 (DISABLE_SWHS_REQ_SRC): Disables write to the SWHS_REQ_SRC bit **Value After Reset:** 0x0 **Volatile:** true
0	SWHS_REQ_SRC	R/W	Software Handshake Request for Channel Source. This bit is used to request dma source data transfer if software handshaking method is selected for the source of the corresponding channel. This bit is ignored if software handshaking is not enabled for the source of the Channelx. The functionality of this field depends on whether the peripheral is the flow controller or not. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSSrcReg.SWHS_Ack_Src bit and sees it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_Req_Src bit is written only if the corresponding write enable bit, SWHS_Req_Src_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSSrcReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_REQ_SRC): Source peripheral request for a dma transfer • 0x0 (INACTIVE_SWHS_REQ_SRC): Source peripheral is not requesting for a burst transfer **Value After Reset:** 0x0 **Volatile:** true

CHx_SWHSDSTREG

- **Name:** dmac Channel x Software Handshake Destination Register
- **Description:** Channelx Software handshake Destination Register.
- **Size:** 64 bits
- Offset: 0x140

Bits	Name	Memory Access	Description
63:6	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
5	SWHS_LST_DST_WE	W	Write Enable bit for Software Handshake Last Request for Channel Destination. **Note**: This bit always returns 0 on a read back. **Values:** • 0x1 (ENABLE_SWHS_LAST_DST): Enables write to the SWHS_LAST_DST bit • 0x0 (DISABLE_SWHS_LAST_DST): Disables write to the SWHS_LAST_DST bit **Value After Reset:** 0x0 **Volatile:** true
4	SWHS_LST_DST	R/W	Software Handshake Last Request for Channel Destination. This bit is used to request LAST dma destination data transfer if software handshaking method is selected for the destination of the corresponding channel. This bit is ignored if software handshaking is not enabled for the destination of the Channelx or if the destination of Channelx is not the flow controller. CHx_SWHSDstReg.SWHS_Req_Dst bit must be set to 1 for DW_axi_dmac to treat it as a valid software handshaking request. If CHx_SWHSDstReg.SWHS_SglReq_Dst is set to 1, the LAST request is for SINGLE dma transaction (AXI burst length = 1), else the request is treated as a BURST transaction request. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSDstReg.SWHS_Ack_Dst bit and sets it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_Lst_Src bit is written only if the corresponding write enable bit, SWHS_Lst_Src_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSDstReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_LAST_DST): Destination peripheral indication to dmac that the current transfer is the last transfer • 0x0 (INACTIVE_SWHS_LAST_DST): Destination peripheral indication that the current transfer is not the last transfer **Value After Reset:** 0x0 **Volatile:** true
3	SWHS_SGLREQ_DST_WE	W	Write Enable bit for Software Handshake Single Request for Channel Destination. **Note**: This bit always returns 0 on a read block.

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_SWHS_SGLREQ_DST): Enables write to the SWHS_SGLREQ_DST bit • 0x0 (DISABLE_SWHS_SGLREQ_DST): Disables write to the SWHS_SGLREQ_DST bit **Value After Reset:** 0x0 **Volatile:** true
2	SWHS_SGLREQ_DST	R/W	Software Handshake Single Request for Channel Destination. This bit is used to request SINGLE (AXI burst length = 1) dma destination data transfer if software handshaking method is selected for the destination of the corresponding channel. This bit is ignored if software handshaking is not enabled for the destination of the Channelx. The functionality of this field depends on whether the peripheral is the flow controller. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSDstReg.SWHS_Ack_Dst bit and sees it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_SglReq_Dst bit is written only if the corresponding write enable bit, SWHS_SglReq_Dst_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSDstReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_SGLREQ_DST): Destination peripheral request for a single dma transfer • 0x0 (INACTIVE_SWHS_SGLREQ_DST): Destination peripheral is not requesting for a single transfer **Value After Reset:** 0x0 **Volatile:** true
1	SWHS_REQ_DST_WE	W	Write Enable bit for Software Handshake Request for Channel Destination. **Note**: This bit always returns 0 on a read block. **Values:** • 0x1 (ENABLE_SWHS_REQ_DST): Enables write to the SWHS_REQ_DST bit • 0x0 (DISABLE_SWHS_REQ_DST): Disables write to the SWHS_REQ_DST bit **Value After Reset:** 0x0 **Volatile:** true
0	SWHS_REQ_DST	R/W	Software Handshake Request for Channel Destination.

Bits	Name	Memory Access	Description
			This bit is used to request dma destination data transfer if software handshaking method is selected for the destination of the corresponding channel. This bit is ignored if software handshaking is not enabled for the source of the Channelx. The functionality of this field depends on whether the peripheral is the flow controller. DW_axi_dmac clears this bit to 0 once software reads CHx_SWHSDstReg.SWHS_Ack_Dst bit and sees it as 1. Software can only set this bit to 1; it is not allowed to clear this bit to 0; only DW_axi_dmac can clear this bit. **Note**: SWHS_Req_Dst bit is written only if the corresponding write enable bit, SWHS_Req_Dst_WE is asserted on the same register write operation and if the Channelx is enabled in the DMAC_ChEnReg register. This allows software to set a bit in the CHx_SWHSDstReg register without performing a read-modified write operation. **Values:** • 0x1 (ACTIVE_SWHS_REQ_DST): Destination peripheral request for a dma transfer • 0x0 (INACTIVE_SWHS_REQ_DST): Destination peripheral is not requesting for a burst transfer **Value After Reset:** 0x0 **Volatile:** true

CHx_BLK_TFR_RESUMEREQREG

- **Name:** dmac Channel x Block Transfer Resume Request Register
- **Description:** Channelx Block Transfer Resume Request Register. This register is used during Linked List or Shadow Register based multi-block transfer.
- For Linked-List-based multi-block transfer, ShadowReg_Or_LLI_Valid bit in LLI.CHx_CTL indicates whether the linked list item fetched from the memory is valid (0: LLI is invalid, 1: LLI is valid). On noticing this bit as 0, DW_axi_dmac discards the LLI and generates ShadowReg_Or_LLI_Invalid_ERR Interrupt if the corresponding channel error interrupt mask bit is set to 0. This error condition causes the DW_axi_dmac to halt the corresponding channel gracefully. DW_axi_dmac waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid LLI availability, before attempting another LLI read operation.
- For Shadow-Register-based multi-block transfer, ShadowReg_Or_LLI_Valid bit in CHx_CTL register indicates whether the shadow register contents are valid (0: Shadow Register contents are invalid, 1: Shadow Register contents are valid). On noticing this bit as 0 during shadow register fetch phase, DW_axi_dmac discards the Shadow Register contents and generates ShadowReg_Or_LLI_Invalid_ERR Interrupt. DW_axi_dmac waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid shadow register availability, before attempting another shadow register fetch operation and continue the next block transfer.
- **Size:** 64 bits
- Offset: 0x148

Bits	Name	Memory Access	Description
63:1	RSVD	W	Reserved bits - Read Only **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
0	BLK_TFR_RESUMEREQ	W	Block Transfer Resume Request during Linked-List or Shadow-Register-based multi-block transfer. **Values:** • 0x0 (INACTIVE_BLK_TFR_RESUMEREQ): No request to resume the block transfer • 0x1 (ACTIVE_BLK_TFR_RESUMEREQ): Request for resuming the block transfer **Value After Reset:** 0x0

CHx_AXI_IDREG

- **Name:** dmac Channel x AXI ID Register
- **Description:** Channelx AXI ID Register. This register is allowed to be updated only when the channel is disabled, which means that it remains fixed for the entire DMA transfer.
- **Note:** The presence of this register is determined by the DMAC_M_ID_WIDTH and DMAC_NUM_CHANNELS configuration parameters.
- DMA0: DMAC_M_ID_WIDTH=8, DMAC_NUM_CHANNELS=12
- DMA1: DMAC_M_ID_WIDTH=6, DMAC_NUM_CHANNELS=16
- If LLI is enabled for any of the channel, then the register is present only when:
- DMAX_M_ID_WIDTH - (log2(DMAC_NUM_CHANNELS) +1) > 0
- Otherwise:
- DMAX_M_ID_WIDTH - log2(DMAC_NUM_CHANNELS) > 0
- **Size:** 64 bits
- Offset: 0x150

Bits	Name	Memory Access	Description
63:19	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
18:16	AXI_WRITE_ID_SUFFIX	R/W	AXI Write ID Suffix. These bits form part of the AWID output of AXI3/AXI4 master interface. IDW = DMAX_M_ID_WIDTH L2NC = log2(DMAX_NUM_CHANNELS) The upper L2NC+1 bits of awidN is derived from the channel number which is currently accessing the master interface. This varies for LLI fetch and source data transfer. For source data transfer, awidN for channel1 4'b0000, awidN for channel8 4'b0111 and so on. For LLI fetch access, awidN for channel1 4'b1000, awidN for channel8 4'b1111 and so on. Lower bits are same as the value programmed in CHx_AXI_IDReg.AXI_Write_ID_Suffix filed. **Value After Reset:** 0x0
15:3	RSVD	R	Reserved bits - Read Only

Bits	Name	Memory Access	Description
			Value After Reset: 0x0
2:0	AXI_READ_ID_SUFFIX	R/W	AXI Read ID Suffix These bits form part of the ARID output of AXI3/AXI4 master interface. IDW = DMAX_M_ID_WIDTH L2NC = log2(DMAX_NUM_CHANNELS) The upper L2NC+1 bits of aridN is derived from the channel number which is currently accessing the master interface. This varies for LLI fetch and source data transfer. For source data transfer, aridN for channel1 4'b0000, aridN for channel8 4'b0111 and so on. For LLI fetch access, aridN for channel1 4'b1000, aridN for channel8 4'b1111 and so on. Lower bits are same as the value programmed in CHx_AXI_IDReg.AXI_Read_ID_Suffix filed. **Value After Reset:** 0x0

CHx_SSTAT

- **Name:** dmac Channel x Source Status Register
- Only valid in Channel 1 and 2
- **Description:** Channelx Source Status Register. After each block transfer completes, hardware can retrieve the source status information from the address pointed to by the contents of the CHx_SSTATAR register. This status information is then stored in the CHx_SSTAT register and written out to the CHx_SSTAT register location of the LLI before the start of the next block. Source status write-back to the CHx_SSTAT register location of the LLI is performed only if DMAX_CHx_LLI_WB_EN = 1 and linked-list-based multi-block transfer is enabled for either source or destination peripheral of the channel. This register does not exist if DMAC_CHx_SRC_STAT_EN is set to False; in this case, the read-back value is always 0.
- **Size:** 64 bits
- Offset: 0x160

Bits	Name	Memory Access	Description
63:32	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
31:0	SSTAT	R	Source Status Source status information retrieved by hardware from the address pointed to by the contents of the CHx_SSTATAR register. Source peripheral should update the source status information, if any, at the location pointed to by CHx_SSTATAR to utilize this feature.This status is not related to any internal status of DW_axi_dmac. This status is not related to any internal status of DW_axi_dmac. **Value After Reset:** 0x0

CHx_DSTAT

- **Name:** dmac Channel x Destination Status Register

- Only valid in Channel 1 and 2
- **Description:** Channelx Destination Status Register. After each block transfer completes, hardware can retrieve the destination status information from the address pointed to by the contents of the CHx_DSTATAR register. This status information is then stored in the CHx_DSTAT register and written out to the CHx_DSTAT register location of the LLI before the start of the next block. Destination status write-back to the CHx_DSTAT register location of the LLI is performed only if DMAX_CHx_LLI_WB_EN = 1 and linked list based multi-block transfer is enabled for either source or destination peripheral of the channel.
- **Size:** 64 bits
- Offset: 0x168

Bits	Name	Memory Access	Description
63:32	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0
31:0	DSTAT	R	Destination Status Destination status information retrieved by hardware from the address pointed to by the contents of the CHx_DSTATAR register. Destination peripheral should update the destination status information, if any, at the location pointed to by CHx_DSTATAR to utilize this feature. This status is not related to any internal status of DW_axi_dmac. **Value After Reset:** 0x0

CHx_SSTATAR

- **Name:** DW_axi_dmac Channel x Source Status Fetch Register
- Only valid in Channel 1 and 2
- **Description:** Channelx Source Status Fetch Register. After completion of each block transfer, hardware can retrieve the source status information from the user-defined address to which the contents of the CHx_SSTATAR register point. You can select any location in system memory that provides a 64-bit value to indicate the status of the source transfer. This register does not exist if DMAC_CHx_SRC_STAT_EN is set to False; in this case, the read-back value is always 0.
- **Size:** 64 bits
- Offset: 0x170

Bits	Name	Memory Access	Description
63:0	SSTATAR	R/W	Source Status Fetch Address Pointer from where hardware can fetch the source status information, which is registered in the CHx_SSTAT register and written out to the CHx_SSTAT register location of the LLI before the start of the next block if DMAX_CHx_LLI_WB_EN = 1 and linked list based multi-block transfer is enabled for either source or destination peripheral of the channel. Source peripheral should update the source status information, if any, at the location pointed to by CHx_SSTATAR to utilize this feature. This status is not related to any internal status of DW_axi_dmac. **Value After Reset:** 0x0

CHx_DSTATAR

- **Name:** DW_axi_dmac Channel x Destination Status Fetch Register
- Only valid in Channel 1 and 2
- **Description:** Channelx Destination Status Fetch Register. After completion of each block transfer,

hardware can retrieve the destination status information from the user-defined address to which the contents of the CHx_DSTATAR register points. You can select any location in system memory that would provide a 64-bit value to indicate the status of the destination transfer. This register does not exist if DMAC_CHx_SRC_STAT_EN is set to False; in this case, the read-back value is always 0.

- **Size:** 64 bits
- Offset: 0x178

Bits	Name	Memory Access	Description
63:0	DSTATAR	R/W	Destination Status Fetch Address Pointer from where hardware can fetch the Destination status information, which is registered in the CHx_DSTAT register and written out to the CHx_DSTAT register location of the LLI before the start of the next block if DMAX_CHx_LLI_WB_EN = 1 and linked list based multi-block transfer is enabled for either source or destination peripheral of the channel. Destination peripheral should update the destination status information, if any, at the location pointed to by CHx_DSTATAR to utilize this feature. This status is not related to any internal status of DW_axi_dmac. **Value After Reset:** 0x0

CHx_INTSTATUS_ENABLEREG

- **Name:** dmac Channel x Interrupt Status Enable Register
- **Description:** Writing 1 to specific field enables the corresponding interrupt status generation in Channelx Interrupt Status Register(CH1_IntStatusReg).
- **Size:** 64 bits
- Offset: 0x180

Bits	Name	Memory Access	Description
63:36	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0xffffffff
35	Enable_ECC_PROT_UIDMem_UnCorrERR_IntStat	R	Channel x Unique ID Memory Interface ECC Protection Uncorrectable Error Interrupt enable. • 0: Disable the generation of Channel x UID Memory Interface Uncorrectable Error Interrupt in CHx_INTSTATUSREG. • 1: Enable the generation Channel x UID Memory Interface Uncorrectable Error Interrupt in CHx_INTSTATUSREG. **Value After Reset:** 0x1
34	Enable_ECC_PROT_UIDMem_CorrERR_IntStat	R	Channel x Unique ID Memory Interface ECC Protection Correctable Error Interrupt enable. • 0: Disable the generation of Channel x UID Memory Interface Correctable Error Interrupt in CHx_INTSTATUSREG. • 1: Enable the generation Channel x UID Memory Interface Correctable Error Interrupt in CHx_INTSTATUSREG.

Bits	Name	Memory Access	Description
			Value After Reset: 0x1
33	Enable_ECC_PROT_CHMem_UnCorrERR_IntStat	R	Channel x Channel Memory Interface ECC Protection Uncorrectable Error Interrupt enable. • 0: Disable the generation of Channel x Channel Memory Interface Uncorrectable Error Interrupt in CHx_INTSTATUSREG. • 1: Enable the generation Channel x Channel Memory Interface Uncorrectable Error Interrupt in CHx_INTSTATUSREG. **Value After Reset:** 0x1
32	Enable_ECC_PROT_CHMem_CorrERR_IntStat	R	Channel x Channel Memory Interface ECC Protection Correctable Error Interrupt enable. • 0: Disable the generation of Channel x Channel Memory Interface Correctable Error Interrupt in CHx_INTSTATUSREG. • 1: Enable the generation of Channel x Channel Memory Interface Correctable Error Interrupt in CHx_INTSTATUSREG. **Value After Reset:** 0x1
31	Enable_CH_ABORTED_IntStat	R/W	Channel Aborted Status Enable. • 0: Disable the generation of Channel Aborted Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Channel Aborted Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
30	Enable_CH_DISABLED_IntStat	R/W	Channel Disabled Status Enable. • 0: Disable the generation of Channel Disabled Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Channel Disabled Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
29	Enable_CH_SUSPENDED_IntStat	R/W	Channel Suspended Status Enable. • 0: Disable the generation of Channel Suspended Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Channel Suspended Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
28	Enable_CH_SRC_SUSPENDED_IntStat	R/W	Channel Source Suspended Status Enable. • 0: Disable the generation of Channel Source Suspended Interrupt in CHx_INTSTATUSREG

Bits	Name	Memory Access	Description
			• 1: Enable the generation of Channel Source Suspended Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
27	Enable_CH_LOCK_CLEARED_IntStat	R/W	Channel Lock Cleared Status Enable. • 0: Disable the generation of Channel LOCK CLEARED Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Channel LOCK CLEARED Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
26	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
25	Enable_SLVIF_WRPARITY_ERR_IntStat	R	Slave Interface Write Parity Error Enable. • 0: Disable the generation of Slave Interface Write Parity Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Write Parity Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
24:22	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x7
21	Enable_SLVIF_WRONHOLD_ERR_IntStat	R/W	Slave Interface Write On Hold Error Status Enable. • 0: Disable the generation of Slave Interface Write On Hold Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Write On Hold Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
20	Enable_SLVIF_SHADOWREG_WRON_VALID_ERR_IntStat	R/W	Shadow Register Write On Valid Error Status Enable. • 0: Disable the generation of Shadow Register Write On Valid Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Shadow register Write On Valid Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
19	Enable_SLVIF_WRONCHEN_ERR_IntStat	R/W	Slave Interface Write On Channel Enabled Error Status Enable.

Bits	Name	Memory Access	Description
			• 0: Disable the generation of Slave Interface Write On Channel enabled Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Write On Channel enabled Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
18	Enable_SLVIF_RD2RWO_ERR_IntStat	R/W	Slave Interface Read to write Only Error Status Enable. • 0: Disable the generation of Slave Interface Read to Write only Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Read to Write Only Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
17	Enable_SLVIF_WR2RO_ERR_IntStat	R/W	Slave Interface Write to Read Only Error Status Enable. • 0: Disable the generation of Slave Interface Write to Read only Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Write to Read Only Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
16	Enable_SLVIF_DEC_ERR_IntStat	R/W	Slave Interface Decode Error Status Enable. • 0: Disable the generation of Slave Interface Decode Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Decode Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
15	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
14	Enable_SLVIF_MULTIBLKTYPE_ERR_Int Stat	R/W	Slave Interface Multi Block type Error Status Enable. • 0: Disable the generation of Slave Interface Multi Block type Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Slave Interface Multi Block type Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
13	Enable_SHADOWREG_OR_LLI_INVALID _ERR_IntStat	R/W	Shadow register or LLI Invalid Error Status Enable.

Bits	Name	Memory Access	Description
			• 0: Disable the generation of Shadow Register or LLI Invalid Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Shadow Register or LLI Invalid Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
12	Enable_LLI_WR_SLV_ERR_IntStat	R/W	LLI WRITE Slave Error Status Enable. • 0: Disable the generation of LLI WRITE Slave Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of LLI WRITE Slave Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
11	Enable_LLI_RD_SLV_ERR_IntStat	R/W	LLI Read Slave Error Status Enable. • 0: Disable the generation of LLI Read Slave Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of LLI Read Slave Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
10	Enable_LLI_WR_DEC_ERR_IntStat	R/W	LLI WRITE Decode Error Status Enable. • 0: Disable the generation of LLI WRITE Decode Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of LLI WRITE Decode Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
9	Enable_LLI_RD_DEC_ERR_IntStat	R/W	LLI Read Decode Error Status Enable. • 0: Disable the generation of LLI Read Decode Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of LLI Read Decode Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
8	Enable_DST_SLV_ERR_IntStat	R/W	Destination Slave Error Status Enable. • 0: Disable the generation of Destination Slave Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Destination Slave Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
7	Enable_SRC_SLV_ERR_IntStat	R/W	Source Slave Error Status Enable.

Bits	Name	Memory Access	Description
			• 0: Disable the generation of Source Slave Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Source Slave Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
6	Enable_DST_DEC_ERR_IntStat	R/W	Destination Decode Error Status Enable. • 0: Disable the generation of Destination Decode Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Destination Decode Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
5	Enable_SRC_DEC_ERR_IntStat	R/W	Source Decode Error Status Enable. • 0: Disable the generation of Source Decode Error Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Source Decode Error Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
4	Enable_DST_TRANSCOMP_IntStat	R/W	Destination Transaction Completed Status Enable. • 0: Disable the generation of Destination Transaction complete Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Destination Transaction complete Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
3	Enable_SRC_TRANSCOMP_IntStat	R/W	Source Transaction Completed Status Enable. • 0: Disable the generation of Source Transaction Complete Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Source Transaction Complete Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1
2	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
1	Enable_DMA_TFR_DONE_IntStat	R/W	DMA Transfer Done Interrupt Status Enable. • 0: Disable the generation of DMA Transfer Done Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of DMA Transfer Done Interrupt in CHx_INTSTATUSREG

Bits	Name	Memory Access	Description
			Value After Reset: 0x1
0	Enable_BLOCK_TFR_DONE_IntStat	R/W	Block Transfer Done Interrupt Status Enable. • 0: Disable the generation of Block Transfer Done Interrupt in CHx_INTSTATUSREG • 1: Enable the generation of Block Transfer Done Interrupt in CHx_INTSTATUSREG **Value After Reset:** 0x1

CHx_INTSTATUS

- **Name:** dmac Channel x Interrupt Status Register
- **Description:** Channelx Interrupt Status Register captures the Channelx specific interrupts
- **Size:** 64 bits
- Offset: 0x188

Bits	Name	Memory Access	Description
63:36	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
35	ECC_PROT_UIDMem_UnCorrERR_IntStat	R	Channel x UID Memory Interface ECC Protection related Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the UID Memory Interface data. • 0: No Channel x UID Memory Interface Uncorrectable Error. • 1: Channel x UID Memory Interface Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in CHx_INTSTATUS_ENABLEReg register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in CHx_IntClearReg. **Values:** • 0x1 (ACTIVE_EccPROT_UIDMem_UnCorrERR): Channel x UID Memory Interface Uncorrectable Error detected • 0x0 (INACTIVE_EccPROT_UIDMem_UnCorrERR): Channel x UID Memory Interface Uncorrectable Error not detected **Value After Reset:** 0x0 **Volatile:** true
34	ECC_PROT_UIDMem_CorrERR_IntStat	R	Channel x UID Memory Interface ECC Protection related Correctable Error Interrupt Status bit.

Bits	Name	Memory Access	Description
			This error occurs if ECC correctable error is detected on the UID Memory Interface data. • 0: No Channel x UID Memory Interface correctable Error. • 1: Channel x UID Memory Interface correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in CHx_INTSTATUS_ENABLEReg register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in CHx_IntClearReg. **Values:** • 0x1 (ACTIVE_EccPROT_UIDMem_CorrERR): Channel x UID Memory Interface correctable Error detected • 0x0 (INACTIVE_EccPROT_UIDMem_CorrERR): Channel x UID Memory Interface correctable Error not detected **Value After Reset:** 0x0 **Volatile:** true
33	ECC_PROT_CHMem_UnCorrERR_IntStat	R	Channel x FIFO Memory Interface ECC Protection related Uncorrectable Error Interrupt Status bit. This error occurs if ECC Uncorrectable error is detected on the FIFO Memory Interface data. • 0: No Channel x FIFO Memory Interface Uncorrectable Error. • 1: Channel x FIFO Memory Interface Uncorrectable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in CHx_INTSTATUS_ENABLEReg register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in CHx_IntClearReg. **Values:** • 0x1 (ACTIVE_EccPROT_CHMem_UnCorrERR): Channel x FIFO Memory Interface Uncorrectable Error detected • 0x0 (INACTIVE_EccPROT_CHMem_UnCorrERR): Channel x FIFO Memory Interface Uncorrectable Error not detected **Value After Reset:** 0x0 **Volatile:** true
32	ECC_PROT_CHMem_CorrERR_IntStat	R	Channel x FIFO Memory Interface ECC Protection related Correctable Error Interrupt Status bit. This error occurs if ECC correctable error is detected on

Bits	Name	Memory Access	Description
			the FIFO Memory Interface data. • 0: No Channel x FIFO Memory Interface correctable Error. • 1: Channel x FIFO Memory Interface correctable Error detected. Error Interrupt status is generated if the corresponding Status Enable bit in CHx_INTSTATUS_ENABLEReg register is set to 1. This bit is cleared to 0 on writing 1 to the corresponding interrupt clear bit in CHx_IntClearReg. **Values:** • 0x1 (ACTIVE_EccPROT_CHMem_CorrERR): Channel x FIFO Memory Interface correctable Error detected • 0x0 (INACTIVE_EccPROT_CHMem_CorrERR): Channel x FIFO Memory Interface correctable Error not detected **Value After Reset:** 0x0 **Volatile:** true
31	CH_ABORTED_IntStat	R	Channel Aborted. This indicates to the software that the corresponding channel in DW_axi_dmac is aborted. • 0: Channel is not aborted • 1: Channel is aborted Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_CH_ABORTED): Channel is aborted • 0x0 (INACTIVE_CH_ABORTED): Channel is not aborted **Value After Reset:** 0x0 **Volatile:** true
30	CH_DISABLED_IntStat	R	Channel Disabled. This indicates to the software that the corresponding channel in DW_axi_dmac is disabled. • 0: Channel is not disabled. • 1: Channel is disabled. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding

Bits	Name	Memory Access	Description
			channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_CH_DISABLED): Channel is disabled • 0x0 (INACTIVE_CH_DISABLED): Channel is not disabled **Value After Reset:** 0x0 **Volatile:** true
29	CH_SUSPENDED_IntStat	R	Channel Suspended. This indicates to the software that the corresponding channel in DW_axi_dmac is suspended. • 0: Channel is not suspended. • 1: Channel is suspended. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_CH_SUSPENDED): Channel is suspended • 0x0 (INACTIVE_CH_SUSPENDED): Channel is not suspended **Value After Reset:** 0x0 **Volatile:** true
28	CH_SRC_SUSPENDED_IntStat	R	Channel Source Suspended. This indicates to the software that the corresponding channel source data transfer in DW_axi_dmac is suspended. • 0: Channel source is not suspended • 1: Channel Source is suspended. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_CH_SRC_SUSPENDED): Channel Source is suspended • 0x0 (INACTIVE_CH_SRC_SUSPENDED): Channel source is not suspended **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
27	CH_LOCK_CLEARED_IntStat	R	Channel Lock Cleared. This indicates to the software that the locking of the corresponding channel in DW_axi_dmac is cleared. • 0: Channel locking is not cleared. • 1: Channel locking is cleared. Channel locking is cleared by DW_axi_dmac during the following situations: • Channel locking is cleared and the channel locking settings in CHx_CFG register is reset if DW_axi_dmac disables the channel upon request from software. • Channel locking is cleared and the channel locking settings in CHx_CFG register is reset if DW_axi_dmac disables the channel upon receiving error response on the master interface. This bit is cleared to 0 on enabling the channel. **Values:** • 0x1 (ACTIVE_CH_LOCK_CLEARED): Channel Locking is cleared • 0x0 (INACTIVE_CH_LOCK_CLEARED): Channel locking is not cleared, if present. **Value After Reset:** 0x0 **Volatile:** true
26	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x0 **Volatile:** true
25	SLVIF_WRPARITY_ERR_IntStat	R	Slave Interface Write Parity Error. This error occurs if a Write operation is performed on a channel register; But the Write data fails the even/odd parity check. • 0: No Slave Interface Write Parity Errors. • 1: Slave Interface Write Parity Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_WRPARITY_ERR): Slave Interface Write Parity Error detected • 0x0 (INACTIVE_SLVIF_WRPARITY_ERR): No

Bits	Name	Memory Access	Description
			Slave Interface Write Parity Errors **Value After Reset:** 0x0 **Volatile:** true
24:22	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true
21	SLVIF_WRONHOLD_ERR_IntStat	R	Slave Interface Write On Hold Error. This error occurs if an illegal write operation is performed on a register; this happens if a write operation is performed on a channel register when DW_axi_dmac is in Hold mode. • 0: No Slave Interface Write On Hold Errors. • 1: Slave Interface Write On Hold Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_WRONHOLD_ERR): Slave Interface Write On Hold Error detected • 0x0 (INACTIVE_SLVIF_WRONHOLD_ERR): No Slave Interface Write On Hold Errors **Value After Reset:** 0x0 **Volatile:** true
20	SLVIF_SHADOWREG_WRON_VALID_ERR_IntStat	R	Shadow Register Write On Valid Error. This error occurs if shadow register based multi-block transfer is enabled and software tries to write to the shadow register when CHx_CTL.ShadowReg_Or_LLI_Valid bit is 1. • 0: No Slave Interface Shadow Register Write On Valid Errors. • 1: Slave Interface Shadow Register Write On Valid Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_SHADOWREG_WRON_VALID_ERR): Slave Interface Shadow Register Write On Valid Error detected

Bits	Name	Memory Access	Description
			• 0x0 (INACTIVE_SLVIF_SHADOWREG_WRON_VALID_ERR): No Slave Interface Shadow Register Write On Valid Errors **Value After Reset:** 0x0 **Volatile:** true
19	SLVIF_WRONCHEN_ERR_IntStat	R	Slave Interface Write On Channel Enabled Error. This error occurs if an illegal write operation is performed on a register; this happens if a write operation is performed on a register when the channel is enabled and if it is not allowed for the corresponding register as per the DW_axi_dmac specification. • 0: No Slave Interface Write On Channel Enabled Errors. • 1: Slave Interface Write On Channel Enabled Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_WRONCHEN_ERR): Slave Interface Write On Channel Enabled Error detected • 0x0 (INACTIVE_SLVIF_WRONCHEN_ERR): No Slave Interface Write On Channel Enabled Errors **Value After Reset:** 0x0 **Volatile:** true
18	SLVIF_RD2RWO_ERR_IntStat	R	Slave Interface Read to write Only Error. This error occurs if read operation is performed to a Write Only register. • 0: No Slave Interface Read to Write Only Errors. • 1: Slave Interface Read to Write Only Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_RD2RWO_ERR): Slave Interface Read to Write Only Error detected • 0x0 (INACTIVE_SLVIF_RD2RWO_ERR): No Slave Interface Read to Write Only Errors

Bits	Name	Memory Access	Description
			Value After Reset: 0x0 **Volatile:** true
17	SLVIF_WR2RO_ERR_IntStat	R	Slave Interface Write to Read Only Error. This error occurs if write operation is performed to a Read Only register. • 0: No Slave Interface Write to Read Only Errors. • 1: Slave Interface Write to Read Only Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_WR2RO_ERR): Slave Interface Write to Read Only Error detected • 0x0 (INACTIVE_SLVIF_WR2RO_ERR): No Slave Interface Write to Read Only Errors **Value After Reset:** 0x0 **Volatile:** true
16	SLVIF_DEC_ERR_IntStat	R	Slave Interface Decode Error. Decode Error generated by DW_axi_dmac during register access. This error occurs if the register access is to invalid address in Channelx register space resulting in error response by DW_axi_dmac slave interface. • 0: No Slave Interface Decode errors. • 1: Slave Interface Decode Error detected. Error Interrupt is generated if the corresponding bit in CHxINTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_DEC_ERR): Slave Interface Decode Error detected • 0x0 (INACTIVE_SLVIF_DEC_ERR): No Slave Interface Decode errors **Value After Reset:** 0x0 **Volatile:** true
15	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x0 **Volatile:** true

Bits	Name	Memory Access	Description
14	SLVIF_MULTIBLKTYPE_ERR_IntStat	R	Slave Interface Multi Block type Error. This error occurs if multi-block transfer type programmed in CHx_CFG register (SRC_MLTBLK_TYPE and DST_MLTBLK_TYPE) is invalid. This error condition causes the DW_axi_dmac to halt the corresponding channel gracefully; Error Interrupt is generated if the corresponding channel error interrupt mask bit is set to 0 and the channel waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid multi-block transfer type availability. • 0: No Multi-block transfer type Errors. • 1: Multi-block transfer type Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SLVIF_MULTIBLKTYPE_ERR): Multi-block transfer type Error detected • 0x0 (INACTIVE_SLVIF_MULTIBLKTYPE_ERR): No Multi-block transfer type Errors **Value After Reset:** 0x0 **Volatile:** true
13	SHADOWREG_OR_LLI_INVALID_ERR_IntStat	R	Shadow register or LLI Invalid Error. This error occurs if CHx_CTL.ShadowReg_Or_LLI_Valid bit is seen to be 0 during DW_axi_dmac Shadow Register / LLI fetch phase. This error condition causes the DW_axi_dmac to halt the corresponding channel gracefully; Error Interrupt is generated if the corresponding channel error interrupt mask bit is set to 0 and the channel waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid Shadow Register availability. In the case of LLI pre-fetching, ShadowReg_Or_LLI_Invalid_ERR Interrupt is not generated even if ShadowReg_Or_LLI_Valid bit is seen to be 0 for the pre-fetched LLI. In this case, DW_axi_dmac re-attempts the LLI fetch operation after completing the current block transfer and generates ShadowReg_Or_LLI_Invalid_ERR Interrupt only if ShadowReg_Or_LLI_Valid bit is still seen to be 0. • 0: No Shadow Register / LLI Invalid errors. • 1: Shadow Register / LLI Invalid error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ACTIVE_SHADOWREG_OR_LLI_INVALID_ERR): Shadow Register / LLI Invalid error detected • 0x0 (INACTIVE_SHADOWREG_OR_LLI_INVALID_ERR): No Shadow Register / LLI Invalid errors **Value After Reset:** 0x0 **Volatile:** true
12	LLI_WR_SLV_ERR_IntStat	R	LLI WRITE Slave Error. Slave Error detected by Master Interface during LLI write-back operation. This error occurs if the slave interface on which LLI resides issues a Slave Error. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN1 bit which received the error is set to 0. • 0: No LLI write Slave Errors. • 1: LLI Write SLAVE Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_LLI_WR_SLV): LLI Write SLAVE Error detected • 0x0 (INACTIVE_LLI_WR_SLV): No LLI write Slave Errors **Value After Reset:** 0x0 **Volatile:** true
11	LLI_RD_SLV_ERR_IntStat	R	LLI Read Slave Error. Slave Error detected by Master Interface during LLI read operation. This error occurs if the slave interface on which LLI resides issues a Slave Error. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN1 bit which received the error is set to 0. • 0: No LLI Read Slave Errors. • 1: LLI read Slave Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:**

Bits	Name	Memory Access	Description
			• 0x1 (ACTIVE_LLI_RD_SLV_ERR): LLI read Slave Error detected • 0x0 (INACTIVE_LLI_RD_SLV_ERR): No LLI Read Slave Errors **Value After Reset:** 0x0 **Volatile:** true
10	LLI_WR_DEC_ERR_IntStat	R	LLI WRITE Decode Error. Decode Error detected by Master Interface during LLI write-back operation. This error occurs if the access is to invalid address and a Decode Error is returned from interconnect/slave. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN1 bit which received the error is set to 0. • 0: NO LLI Write Decode Errors. • 1: LLI write Decode Error detected. Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_LLI_WR_DEC_ERR): LLI write Decode Error detected • 0x0 (INACTIVE_LLI_WR_DEC_ERR): NO LLI Write Decode Errors **Value After Reset:** 0x0 **Volatile:** true
9	LLI_RD_DEC_ERR_IntStat	R	LLI Read Decode Error. Decode Error detected by Master Interface during LLI read operation. This error occurs if the access is to invalid address and a Decode Error is returned from interconnect/slave. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN1 bit which received the error is set to 0. • 0: NO LLI Read Decode Errors. • 1: LLI Read Decode Error detected Error Interrupt is generated if the corresponding bit in CHx_INTSTATUS_ENABLEReg is enabled. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_LLI_RD_DEC_ERR): LLI Read

Bits	Name	Memory Access	Description
			Decode Error detected • 0x0 (INACTIVE_LLI_RD_DEC_ERR): NO LLI Read Decode Errors **Value After Reset:** 0x0 **Volatile:** true
8	DST_SLV_ERR_IntStat	R	Destination Slave Error. Slave Error detected by Master Interface during destination data transfer. This error occurs if the slave interface to which the data is written issues a Slave Error. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN bit corresponding to the channel which received the error is set to 0. • 0: No Destination Slave Errors • 1: Destination Slave Errors Detected This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_DST_SLV_ERR): Destination Slave Errors Detected • 0x0 (INACTIVE_DST_SLV_ERR): No Destination Slave Errors **Value After Reset:** 0x0 **Volatile:** true
7	SRC_SLV_ERR_IntStat	R	Source Slave Error. Slave Error detected by Master Interface during source data transfer. This error occurs if the slave interface from which the data is read issues a Slave Error. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN bit corresponding to the channel which received the error is set to 0. • 0: No Source Slave Errors • 1: Source Slave Error Detected This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SRC_SLV_ERR): Source Slave Error Detected • 0x0 (INACTIVE_SRC_SLV_ERR): No Source Slave Errors **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
6	DST_DEC_ERR_IntStat	R	Destination Decode Error. Decode Error detected by Master Interface during destination data transfer. This error occurs if the access is to invalid address and a Decode Error is returned from interconnect/slave. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN bit corresponding to the channel which received the error is set to 0. • 0: No destination Decode Errors. • 1: Destination Decode Error Detected This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_DST_DEC_ERR): Destination Decode Error Detected • 0x0 (INACTIVE_DST_DEC_ERR): No destination Decode Errors. **Value After Reset:** 0x0 **Volatile:** true
5	SRC_DEC_ERR_IntStat	R	Source Decode Error. Decode Error detected by Master Interface during source data transfer. This error occurs if the access is to invalid address and a Decode Error is returned from interconnect/slave. This error condition causes the DW_axi_dmac to disable the corresponding channel gracefully; the DMAC_ChEnReg.CH_EN bit corresponding to the channel which received the error is set to 0. • 0: No Source Decode Errors. • 1: Source Decode Error detected. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (ACTIVE_SRC_DEC_ERR): Source Decode Error detected • 0x0 (INACTIVE_SRC_DEC_ERR): No Source Decode Errors **Value After Reset:** 0x0 **Volatile:** true
4	DST_TRANSCOMP_IntStat	R	Destination Transaction Completed.

Bits	Name	Memory Access	Description
			This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register or on enabling the channel (needed when interrupt is not enabled. **Values:** • 0x1 (ACTIVE_DST_TRANSCOMP): Destination transaction is complete • 0x0 (INACTIVE_DST_TRANSCOMP): Destination transaction is not complete **Value After Reset:** 0x0 **Volatile:** true
3	SRC_TRANSCOMP_IntStat	R	Source Transaction Completed. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register or on enabling the channel (needed when interrupt is not enabled. **Values:** • 0x1 (ACTIVE_SRC_TRANSCOMP): Source transaction is complete • 0x0 (INACTIVE_SRC_TRANSCOMP): Source transation is not complete **Value After Reset:** 0x0 **Volatile:** true
2	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x0 **Volatile:** true
1	DMA_TFR_DONE_IntStat	R	DMA Transfer Done. This indicates to the software that the DW_axi_dmac has completed the requested DMA transfer. The DW_axi_dmac sets this bit to 1 along with setting CHx_INTSTATUS.BLOCK_TFR_DONE bit to 1 when the last block transfer is completed. • 0: DMA Transfer not completed. • 1: DMA Transfer Completed This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (DMA_TFR_COMPLETED): DMA Transfer completed • 0x0 (DMA_TFR_NOT_COMPLETE): DMA

Bits	Name	Memory Access	Description
			Transfer not complete **Value After Reset:** 0x0 **Volatile:** true
0	BLOCK_TFR_DONE_IntStat	R	Block Transfer Done. This indicates to the software that the DW_axi_dmac has completed the requested block transfer. The DW_axi_dmac sets this bit to 1 when the transfer is successfully completed. • 0: Block Transfer not completed. • 1: Block Transfer completed. This bit is cleared to 0 on writing 1 to the corresponding channel interrupt clear bit in CHx_IntClearReg register. **Values:** • 0x1 (BLOCK_TFR_COMPLETED): Block Transfer completed • 0x0 (BLOCK_TFR_NOT_COMPLETE): Block Transfer not complete **Value After Reset:** 0x0 **Volatile:** true

CHx_INTSIGNAL_ENABLEREG

- **Name:** dmac Channel x Interrupt Signal Enable Register
- **Description:** This register contains fields that are used to enable the generation of port level interrupt at the channel level.
- **Size:** 64 bits
- Offset: 0x190

Bits	Name	Memory Access	Description
63:36	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0xffffffff
35	Enable_ECC_PROT_UIDMem_UnCorrERR_IntSignal	R	Channel x Unique ID Memory Interface ECC Protection Uncorrectable Error Interrupt Signal enable. • 0: Disable the propagation of Channel x UID Memory Interface Uncorrectable Error to generate a port level interrupt. • 1: Enable the propagation of Channel x UID Memory Interface Uncorrectable Error to generate a port level interrupt. **Values:** • 0x1 (ENABLE_ECC_UIDMEM_UNCORRERR): Enable the propagation of UID Memory ECC

Bits	Name	Memory Access	Description
			Uncorrectable error Interrupt to port level interrupt • 0x0 (DISABLE_ECC_UIDMEM_UNCORRERR): Disable the propagation of UID Memory ECC Uncorrectable error Interrupt to port level interrupt **Value After Reset:** 0x1
34	Enable_ECC_PROT_UIDMem_CorrERR_IntSignal	R	Channel x Unique ID Memory Interface ECC Protection Correctable Error Interrupt Signal enable. • 0: Disable the propagation of Channel x UID Memory Interface Correctable Error to generate a port level interrupt. • 1: Enable the propagation of Channel x UID Memory Interface Correctable Error to generate a port level interrupt. **Values:** • 0x1 (ENABLE_ECC_UIDMEM_CORRERR): Enable the propagation of UID Memory ECC Correctable error Interrupt to port level interrupt • 0x0 (DISABLE_ECC_UIDMEM_CORRERR): Disable the propagation of UID Memory ECC Correctable error Interrupt to port level interrupt **Value After Reset:** 0x1
33	Enable_ECC_PROT_CHMem_UnCorrERR_IntSignal	R	Channel x Channel Memory Interface ECC Protection Uncorrectable Error Interrupt Signal enable. • 0: Disable the propagation of Channel x Channel Memory Interface Uncorrectable Error to generate a port level interrupt. • 1: Enable the propagation of Channel x Channel Memory Interface Uncorrectable Error to generate a port level interrupt. **Values:** • 0x1 (ENABLE_ECC_CHMEM_UNCORRERR): Enable the propagation of Channel Memory ECC Uncorrectable error Interrupt to port level interrupt • 0x0 (DISABLE_ECC_CHMEM_UNCORRERR): Disable the propagation of Channel Memory ECC Uncorrectable error Interrupt to port level interrupt **Value After Reset:** 0x1
32	Enable_ECC_PROT_CHMem_CorrERR_IntSignal	R	Channel x Channel Memory Interface ECC Protection Correctable Error Interrupt Signal enable.

Bits	Name	Memory Access	Description
			• 0: Disable the propagation of Channel x Channel Memory Interface Correctable Error to generate a port level interrupt. • 1: Enable the propagation of Channel x Channel Memory Interface Correctable Error to generate a port level interrupt. **Values:** • 0x1 (ENABLE_ECC_CHMEM_CORRERR): Enable the propagation of Channel Memory ECC Correctable error Interrupt to port level interrupt • 0x0 (DISABLE_ECC_CHMEM_CORRERR): Disable the propagation of Channel Memory ECC Correctable error Interrupt to port level interrupt **Value After Reset:** 0x1
31	Enable_CH_ABORTED_IntSignal	R/W	Channel Aborted Signal Enable. • 0: Disable the propagation of Channel Aborted Interrupt to generate a port level interrupt • 1: Enable the propagation of Channel Aborted Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_CH_ABORTED_IntSignal): Enable the propagation of Channel Aborted Interrupt to generate a port level interrupt • 0x0 (DISABLE_CH_ABORTED_IntSignal): Disable the propagation of Channel Aborted Interrupt to generate a port level interrupt **Value After Reset:** 0x1
30	Enable_CH_DISABLED_IntSignal	R/W	Channel Disabled Signal Enable. • 0: Disable the propagation of Channel Disabled Interrupt to generate a port level interrupt • 1: Enable the propagation of Channel Disabled Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_CH_DISABLED_IntSignal): Enable the propagation of Channel Disabled Interrupt to generate a port level interrupt • 0x0 (DISABLE_CH_DISABLED_IntSignal): Disable the propagation of Channel Disabled Interrupt to generate a port level interrupt **Value After Reset:** 0x1
29	Enable_CH_SUSPENDED_IntSignal	R/W	Channel Suspended Signal Enable.

Bits	Name	Memory Access	Description
			• 0: Disable the propagation of Channel Suspended Interrupt to generate a port level interrupt • 1: Enable the propagation of Channel Suspended Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_CH_SUSPENDED_IntSignal): Enable the propagation of Channel Suspended Interrupt to generate a port level interrupt • 0x0 (DISABLE_CH_SUSPENDED_IntSignal): Disable the propagation of Channel Suspended Interrupt to generate a port level interrupt **Value After Reset:** 0x1
28	Enable_CH_SRC_SUSPENDED_IntSignal	R/W	Channel Source Suspended Signal Enable. • 0: Disable the propagation of Channel Source Suspended Interrupt to generate a port level interrupt • 1: Enable the propagation of Channel Source Suspended Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_CH_SRC_SUSPENDED_IntSignal): Enable the propagation of Channel Source Suspended Interrupt to generate a port level interrupt • 0x0 (DISABLE_CH_SRC_SUSPENDED_IntSignal): Disable the propagation of Channel Source Suspended Interrupt to generate a port level interrupt **Value After Reset:** 0x1
27	Enable_CH_LOCK_CLEARED_IntSignal	R/W	Channel Lock Cleared Signal Enable. • 0: Disable the propagation of Channel Lock Cleared Interrupt to generate a port level interrupt • 1: Enable the propagation of Channel Lock Cleared Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_CH_LOCK_CLEARED_IntSignal): Enable the propagation of Channel Lock Cleared Interrupt to generate a port level interrupt

Bits	Name	Memory Access	Description
			• 0x0 (DISABLE_CH_LOCK_CLEARED_IntSignal): Disable the propagation of Channel Lock Cleared Interrupt to generate a port level interrupt **Value After Reset:** 0x1
26	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
25	Enable_SLVIF_WRPARITY_ERR_IntSignal	R	Slave Interface Write Parity Error Signal Enable. • 0: Disable the propagation of Slave Interface Write Parity Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Write Parity Error Interrupt to generate a port level interrupt **Values:** • 0x1 (Enable_SLVIF_WRPARITY_ERR_IntSignal): Enable the propagation of Slave Interface Write Parity Error Interrupt to generate a port level interrupt • 0x0 (Disable_SLVIF_WRPARITY_ERR_IntSignal): Disable the propagation of Slave Interface Write Parity Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
24:22	RSVD	R	Reserved bits - Read Only **Value After Reset:** 0x7
21	Enable_SLVIF_WRONHOLD_ERR_IntSignal	R/W	Slave Interface Write On Hold Error Signal Enable. • 0: Disable the propagation of Slave Interface Write On Hold Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Write On Hold Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_WRONHOLD_ERR_IntSignal): Enable the propagation of Slave Interface Write On Hold Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_WRONHOLD_ERR_IntSignal): Disable the propagation of Slave Interface

Bits	Name	Memory Access	Description
			Write On Hold Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
20	Enable_SLVIF_SHADOWREG_WRON_VALID_ERR_IntSignal	R/W	Shadow Register Write On Valid Error Signal Enable. • 0: Disable the propagation of Shadow Register Write On Valid Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Shadow register Write On Valid Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_SHADOWREG_WRON_VALID_ERR_IntSignal): Enable the propagation of Shadow register Write On Valid Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_SHADOWREG_WRON_VALID_ERR_IntSignal): Disable the propagation of Shadow Register Write On Valid Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
19	Enable_SLVIF_WRONCHEN_ERR_IntSignal	R/W	Slave Interface Write On Channel Enabled Error Signal Enable. • 0: Disable the propagation of Slave Interface Write On Channel enabled Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Write On Channel enabled Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_WRONCHEN_ERR_IntSignal): Enable the propagation of Slave Interface Write On Channel enabled Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_WRONCHEN_ERR_IntSignal): Disable the propagation of Slave Interface Write On Channel enabled Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
18	Enable_SLVIF_RD2RWO_ERR_IntSignal	R/W	Slave Interface Read to write Only Error Signal Enable. • 0: Disable the propagation of Slave Interface Read to Write only Error Interrupt to generate a port level interrupt

Bits	Name	Memory Access	Description
			• 1: Enable the propagation of Slave Interface Read to Write Only Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_RD2RWO_ERR_IntSignal): Enable the propagation of Slave Interface Read to Write Only Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_RD2RWO_ERR_IntSignal): Disable the propagation of Slave Interface Read to Write only Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
17	Enable_SLVIF_WR2RO_ERR_IntSignal	R/W	Slave Interface Write to Read Only Error Signal Enable. • 0: Disable the propagation of Slave Interface Write to Read only Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Write to Read Only Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_WR2RO_ERR_IntSignal): Enable the propagation of Slave Interface Write to Read Only Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_WR2RO_ERR_IntSignal): Disable the propagation of Slave Interface Write to Read only Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
16	Enable_SLVIF_DEC_ERR_IntSignal	R/W	Slave Interface Decode Error Signal Enable. • 0: Disable the propagation of Slave Interface Decode Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Decode Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_DEC_ERR_IntSignal): Enable the propagation of Slave Interface Decode Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_DEC_ERR_IntSignal): Disable the propagation of Slave Interface Decode Error Interrupt to generate a port level

Bits	Name	Memory Access	Description
			interrupt **Value After Reset:** 0x1
15	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
14	Enable_SLVIF_MULTIBLKTYPE_ERR_IntSignal	R/W	Slave Interface Multi Block type Error Signal Enable. • 0: Disable the propagation of Slave Interface Multi Block type Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Slave Interface Multi Block type Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SLVIF_MULTIBLKTYPE_ERR_IntSignal): Enable the propagation of Slave Interface Multi Block type Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SLVIF_MULTIBLKTYPE_ERR_IntSignal): Disable the propagation of Slave Interface Multi Block type Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
13	Enable_SHADOWREG_OR_LLI_INVALID_ERR_IntSignal	R/W	Shadow register or LLI Invalid Error Signal Enable. • 0: Disable the propagation of Shadow Register or LLI Invalid Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Shadow Register or LLI Invalid Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SHADOWREG_OR_LLI_INVALID_ERR_IntSignal): Enable the propagation of Shadow Register or LLI Invalid Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SHADOWREG_OR_LLI_INVALID_ERR_IntSignal): Disable the propagation of Shadow Register or LLI Invalid Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
12	Enable_LLI_WR_SLV_ERR_IntSignal	R/W	LLI WRITE Slave Error Signal Enable. • 0: Disable the propagation of LLI WRITE Slave

Bits	Name	Memory Access	Description
			Error Interrupt to generate a port level interrupt • 1: Enable the propagation of LLI WRITE Slave Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_LLI_WR_SLV_ERR_IntSignal): Enable the propagation of LLI WRITE Slave Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_LLI_WR_SLV_ERR_IntSignal): Disable the propagation of LLI WRITE Slave Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
11	Enable_LLI_RD_SLV_ERR_IntSignal	R/W	LLI Read Slave Error Signal Enable. • 0: Disable the propagation of LLI Read Slave Error Interrupt to generate a port level interrupt • 1: Enable the propagation of LLI Read Slave Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_LLI_RD_SLV_ERR_IntSignal): Enable the propagation of LLI Read Slave Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_LLI_RD_SLV_ERR_IntSignal): Disable the propagation of LLI Read Slave Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
10	Enable_LLI_WR_DEC_ERR_IntSignal	R/W	LLI WRITE Decode Error Signal Enable. • 0: Disable the propagation of LLI WRITE Decode Error Interrupt to generate a port level interrupt • 1: Enable the propagation of LLI WRITE Decode Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_LLI_WR_DEC_ERR_IntSignal): Enable the propagation of LLI WRITE Decode Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_LLI_WR_DEC_ERR_IntSignal): Disable the propagation of LLI WRITE Decode Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
9	Enable_LLI_RD_DEC_ERR_IntSignal	R/W	LLI Read Decode Error Signal Enable. • 0: Disable the propagation of LLI Read Decode Error Interrupt to generate a port level interrupt • 1: Enable the propagation of LLI Read Decode

Bits	Name	Memory Access	Description
			Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_LLI_RD_DEC_ERR_IntSignal): Enable the propagation of LLI Read Decode Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_LLI_RD_DEC_ERR_IntSignal): Disable the propagation of LLI Read Decode Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
8	Enable_DST_SLV_ERR_IntSignal	R/W	Destination Slave Error Signal Enable. • 0: Disable the propagation of Destination Slave Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Destination Slave Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_DST_SLV_ERR_IntSignal): Enable the propagation of Destination Slave Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_DST_SLV_ERR_IntSignal): Disable the propagation of Destination Slave Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
7	Enable_SRC_SLV_ERR_IntSignal	R/W	Source Slave Error Signal Enable. • 0: Disable the propagation of Source Slave Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Source Slave Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SRC_SLV_ERR_IntSignal): Enable the propagation of Source Slave Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SRC_SLV_ERR_IntSignal): Disable the propagation of Source Slave Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
6	Enable_DST_DEC_ERR_IntSignal	R/W	Destination Decode Error Signal Enable. • 0: Disable the propagation of Destination Decode Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Destination Decode Error Interrupt to generate a port level interrupt

Bits	Name	Memory Access	Description
			Values: • 0x1 (ENABLE_DST_DEC_ERR_IntSignal): Enable the propagation of Destination Decode Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_DST_DEC_ERR_IntSignal): Disable the propagation of Destination Decode Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
5	Enable_SRC_DEC_ERR_IntSignal	R/W	Source Decode Error Signal Enable. • 0: Disable the propagation of Source Decode Error Interrupt to generate a port level interrupt • 1: Enable the propagation of Source Decode Error Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SRC_DEC_ERR_IntSignal): Enable the propagation of Source Decode Error Interrupt to generate a port level interrupt • 0x0 (DISABLE_SRC_DEC_ERR_IntSignal): Disable the propagation of Source Decode Error Interrupt to generate a port level interrupt **Value After Reset:** 0x1
4	Enable_DST_TRANSCOMP_IntSignal	R/W	Destination Transaction Completed Signal Enable. • 0: Disable the propagation of Destination Transaction complete Interrupt to generate a port level interrupt • 1: Enable the propagation of Destination Transaction complete Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_DST_TRANSCOMP_IntSignal): Enable the propagation of Destination Transaction complete Interrupt to generate a port level interrupt • 0x0 (DISABLE_DST_TRANSCOMP_IntSignal): Disable the propagation of Destination Transaction complete Interrupt to generate a port level interrupt **Value After Reset:** 0x1
3	Enable_SRC_TRANSCOMP_IntSignal	R/W	Source Transaction Completed Signal Enable. • 0: Disable the propagation of Source Transaction Complete Interrupt to generate a port level interrupt • 1: Enable the propagation of Source

Bits	Name	Memory Access	Description
			Transaction Complete Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_SRC_TRANSCOMP_IntSignal): Enable the propagation of Source Transaction Complete Interrupt to generate a port level interrupt • 0x0 (DISABLE_SRC_TRANSCOMP_IntSignal): Disable the propagation of Source Transaction Complete Interrupt to generate a port level interrupt **Value After Reset:** 0x1
2	RSVD	R	Reserved bit - Read Only **Value After Reset:** 0x1
1	Enable_DMA_TFR_DONE_IntSignal	R/W	DMA Transfer Done Interrupt Signal Enable. • 0: Disable the propagation of DMA Transfer Done Interrupt to generate a port level interrupt • 1: Enable the propagation of DMA Transfer Done Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_DMA_TFR_DONE_IntSignal): Enable the propagation of DMA Transfer Done Interrupt to generate a port level interrupt • 0x0 (DISABLE_DMA_TFR_DONE_IntSignal): Disable the propagation of DMA Transfer Done Interrupt to generate a port level interrupt **Value After Reset:** 0x1
0	Enable_BLOCK_TFR_DONE_IntSignal	R/W	Block Transfer Done Interrupt Signal Enable. • 0: Disable the propagation of Block Transfer Done Interrupt to generate a port level interrupt • 1: Enable the propagation of Block Transfer Done Interrupt to generate a port level interrupt **Values:** • 0x1 (ENABLE_BLOCK_TFR_DONE_IntSignal): Enable the propagation of Block Transfer Done Interrupt to generate a port level interrupt • 0x0 (DISABLE_BLOCK_TFR_DONE_IntSignal): Disable the propagation of Block Transfer Done Interrupt to generate a port level interrupt **Value After Reset:** 0x1

CHx_INTCLEARREG

- **Name:** dmac Channel x Interrupt Status Clear Register

- **Description:** Writing 1 to specific field will clear the corresponding field in Channelx Interrupt Status Register(CHx_IntStatusReg).
- **Size:** 64 bits
- Offset: 0x198

Bits	Name	Memory Access	Description
63:36	RSVD	W	Reserved bit - Read Only **Value After Reset:** 0x0
35	Clear_ECC_PROT_UIDMem_UnCorrERR_IntStat	W	ECC Protection Uncorrectable UID Memory Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (Clear_ECC_UIDMEM_CORRERR_IntStat): Clear the ECC_PROT_UIDMem_UnCorrERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
34	Clear_ECC_PROT_UIDMem_CorrERR_IntStat	W	ECC Protection Correctable UID Memory Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_ECC_UIDMEM_CORRERR_IntStat): Clear the ECC_PROT_UIDMem_CorrERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
33	Clear_ECC_PROT_CHMem_UnCorrERR_IntStat	W	ECC Protection Uncorrectable Channel Memory Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_ECC_CHMEM_CORRERR_IntStat): Clear the ECC_PROT_CHMem_UnCorrERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
32	Clear_ECC_PROT_CHMem_CorrERR_Int	W	ECC Protection Correctable Channel Memory Interrupt

Bits	Name	Memory Access	Description
	Stat		Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_ECC_CHMEM_CORRERR_IntStat): Clear the ECC_PROT_CHMem_CorrERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
31	Clear_CH_ABORTED_IntStat	W	Channel Aborted Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_CH_ABORTED): Clear the CH_ABORTED interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
30	Clear_CH_DISABLED_IntStat	W	Channel Disabled Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_CH_DISABLED): Clear the CH_DISABLED interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
29	Clear_CH_SUSPENDED_IntStat	W	Channel Suspended Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_CH_SUSPENDED): Clear the CH_SUSPENDED interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
28	Clear_CH_SRC_SUSPENDED_IntStat	W	Channel Source Suspended Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_CH_SRC_SUSPENDED): Clear the CH_SRC_SUSPENDED interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
27	Clear_CH_LOCK_CLEARED_IntStat	W	Channel Lock Cleared Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_CH_LOCK_CLEARED): Clear the CH_LOCK_CLEARED interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
26	RSVD	W	Reserved bit - Read Only **Value After Reset:** 0x0
25	Clear_SLVIF_WRPARITY_ERR_IntStat	W	Slave Interface Write Parity Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_WRPARITY_ERR): Clear the SLVIF_WRPARITY_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
24:22	RSVD	W	Reserved bit - Read Only **Value After Reset:** 0x0
21	Clear_SLVIF_WRONHOLD_ERR_IntStat	W	Slave Interface Write On Hold Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_WRONHOLD_ERR): Clear the

Bits	Name	Memory Access	Description
			SLVIF_WRONHOLD_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
20	Clear_SLVIF_SHADOWREG_WRON_VALID_ERR_IntStat	W	Shadow Register Write On Valid Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_SHADOWREG_WRON_VALID_ERR): Clear the SLVIF_SHADOWREG_WRON_VALID_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
19	Clear_SLVIF_WRONCHEN_ERR_IntStat	W	Slave Interface Write On Channel Enabled Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_WRONCHEN_ERR): Clear the SLVIF_WRONCHEN_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
18	Clear_SLVIF_RD2RWO_ERR_IntStat	W	Slave Interface Read to write Only Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_RD2RWO_ERR): Clear the SLVIF_RD2RWO_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
17	Clear_SLVIF_WR2RO_ERR_IntStat	W	Slave Interface Write to Read Only Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG.

Bits	Name	Memory Access	Description
			Values: • 0x1 (CLEAR_SLVIF_WR2RO_ERR): Clear the SLVIF_WR2RO_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
16	Clear_SLVIF_DEC_ERR_IntStat	W	Slave Interface Decode Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_DEC_ERR): Clear the SLVIF_DEC_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
15	RSVD	W	Reserved bit - Read Only **Value After Reset:** 0x0
14	Clear_SLVIF_MULTIBLKTYPE_ERR_IntStat	W	Slave Interface Multi Block type Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SLVIF_MULTIBLKTYPE_ERR): Clear the SLVIF_MULTIBLKTYPE_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
13	Clear_SHADOWREG_OR_LLI_INVALID_ERR_IntStat	W	Shadow register or LLI Invalid Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SHADOWREG_OR_LLI_INVALID_ERR): Clear the SHADOWREG_OR_LLI_INVALID_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
12	Clear_LLI_WR_SLV_ERR_IntStat	W	LLI WRITE Slave Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_LLI_WR_SLV_ERR): Clear the LLI_WR_SLV_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
11	Clear_LLI_RD_SLV_ERR_IntStat	W	LLI Read Slave Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_LLI_RD_SLV_ERR): Clear the LLI_RD_SLV_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
10	Clear_LLI_WR_DEC_ERR_IntStat	W	LLI WRITE Decode Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_LLI_WR_DEC_ERR): Clear the LLI_WR_DEC_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
9	Clear_LLI_RD_DEC_ERR_IntStat	W	LLI Read Decode Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_LLI_RD_DEC_ERR): Clear the LLI_RD_DEC_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
8	Clear_DST_SLV_ERR_IntStat	W	Destination Slave Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_DST_SLV_ERR): Clear the DST_SLV_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
7	Clear_SRC_SLV_ERR_IntStat	W	Source Slave Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SRC_SLV_ERR): Clear the SRC_SLV_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
6	Clear_DST_DEC_ERR_IntStat	W	Destination Decode Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_DST_DEC_ERR): Clear the DST_DEC_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
5	Clear_SRC_DEC_ERR_IntStat	W	Source Decode Error Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SRC_DEC_ERR): Clear the SRC_DEC_ERR interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
4	Clear_DST_TRANSCOMP_IntStat	W	Destination Transaction Completed Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_DST_TRANSCOMP): Clear the DST_TRANSCOMP interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
3	Clear_SRC_TRANSCOMP_IntStat	W	Source Transaction Completed Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_SRC_TRANSCOMP): Clear the SRC_TRANSCOMP interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
2	RSVD	W	Reserved bit - Read Only **Value After Reset:** 0x0
1	Clear_DMA_TFR_DONE_IntStat	W	DMA Transfer Done Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CHx_INTSTATUSREG. **Values:** • 0x1 (CLEAR_DMA_TFR_DONE): Clear the DMA_TFR_DONE interrupt in the Interrupt Status Register(CH1_IntStatusReg). • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0
0	Clear_BLOCK_TFR_DONE_IntStat	W	Block Transfer Done Interrupt Clear Bit. This bit is used to clear the corresponding channel interrupt status bit in CH1_INTSTATUSREG **Values:** • 0x1 (CLEAR_BLOCK_TFR_DONE): Clear the interrupt in the Interrupt Status Register(CHx_IntStatusReg). Writing a 1 to this register field clears the corresponding bit in

Bits	Name	Memory Access	Description
			the CHx_IntStatusReg register. • 0x0 (NO_ACTION): Inactive signal. No action taken. **Value After Reset:** 0x0

3.9 SMMU

3.9.1 Overview

This is a System-level Memory Management Unit (SMMU) that translates an input address to an output address. The SMMU can perform:

- support ARM SMMU v3.1
- Support configurable stream id, each AXI masters can have its dedicated stream id to have independent address space, and they can also have same stream id to share page table
- support 4KB, 16KB, 64KB translation granule size
- Adderss space covered by each entry in each level of page table, depending on translation granule size.
- Support configurable TBU priority
- support configurable AXI qos, higher qos for read page table is recommend.
- support global and per stream id bypass, address of bypassed transaction will never translated.
- Support prefetch: no prefetch, forward, backward.

3.9.2 Block diagram

AXI masters send AXI request to SMMU, then SMMU translate virtual address to physical address. If the needed page table is found in TLB cache, TBU translate the address with page table in the TLB cache. Otherwise, TBU send a page table request to tcu and translate with page table get from TCU. TCU checks its cache just like the TBU, but request page table from DDR while cache missed.

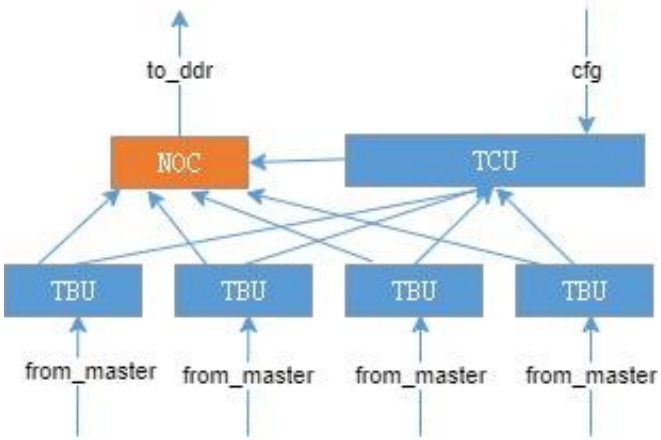


Figure 3-35 block diagram

3.9.3 Firmware flow

SMMU initialization:

1. allocate base address for command queue
2. allocate base address for event queue
3. configure stream table
4. initialize command queue
5. initialize event queue

6. Invalidate TLB and cache
7. configure context descriptor
8. configure stream table entry

Enable smmu：

1. set smmu enable bit in register
2. polling smmu status, until smmu enable request accepted

Disable TBU：

1. send power down request to the TBU to be disabled
2. polling power down status, until power down request accepted

Note: registers for disabling TBU are located in low power register of SOC not in smmu. Diable TBU should be performed before reset TBU, and release power down request before reset relase.

3.9.4 register

address mapping

address	Sub module name
0x50c0_0000~0x50c3_fffc	TCU
0x50c4_0000~0x50c5_fffc	Video in TBU
0x50c6_0000~0x50c7_fffc	Video decoder TBU
0x50c8_0000~0x50c9_fffc	Video encoder TBU
0x50ca_0000~0x50cb_fffc	JPEG encoder TBU
0x50cc_0000~0x50cd_fffc	JPEG decoder TBU
0x50ce_0000~0x50cf_fffc	High speed interface TBU
0x50d0_0000~0x50d1_fffc	PCIe TBU
0x50d2_0000~0x50d3_fffc	Misc TBU
0x50d4_0000~0x50d5_fffc	NPU TBU
0x50d8_0000~0x50d9_fffc	DSP0 TBU
0x50da_0000~0x50db_fffc	DSP1 TBU
0x50dc_0000~0x50dd_fffc	DSP2 TBU
0x50de_0000~0x50df_fffc	DSP3 TBU

SMMUv3 architectural register

Register	Name	Description
SMMU_S_IDR0- SMMU_S_IDR3	SMMU Secure feature Identification Registers	Provides information about the Secure features that the SMMU implementation supports.
SMMU_S_CR0	Secure global Control Register0	Provides global configuration of the Secure SMMU.

Register	Name	Description
SMMU_S_CR0ACK	Secure global Control Register 0	update Acknowledge Provides acknowledgement of completion of updates to SMMU_S_CR0.
SMMU_S_CR1 SMMU_S_CR2	Secure global Control Registers	Provides the controls for Secure table and queue access attributes.
SMMU_S_INIT	Secure Initialization control register	Provides a control to invalidate all Secure SMMU caching on system initialization.
SMMU_S_GBPA	Secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Secure streams when the MMU is disabled.
SMMU_S_IRQ_CTRL	Secure Interrupt Control register	Contains enables for SMMU interrupts.
SMMU_S_IRQ_CTRLACK	Secure Interrupt Control register	update Acknowledge Provides acknowledgement of the completion of updates to SMMU_S_IRQ_CTRL.
SMMU_S_GERROR	Secure Global Error status register	Provides information on Secure global programming interface errors.
SMMU_S_GERRORN	Secure Global Error Acknowledgement register	Contains the acknowledgement fields for SMMU_S_GERROR errors.
SMMU_S_GERROR_IRQ_CFG0 - SMMU_S_GERROR_IRQ_CFG2	Secure Global Error IRQ Configuration register	Contains the Secure MSI address configuration for the GERROR IRQ.
SMMU_S_STRTAB_BASE	Secure Stream Table Base address register	Contains the base address and attributes for the Secure Stream table.
SMMU_S_STRTAB_BASE_CFG	Secure Stream Table Base Configuration register	Contains configuration fields for the Secure Stream table.
SMMU_S_CMDQ_BASE	Secure Command queue Base address register	Contains the base address and attributes for the Secure Command queue.
SMMU_S_CMDQ_PROD	Secure Command queue Producer index register	Contains the Secure Command queue index for writes by the producer.
SMMU_S_CMDQ_CONS	Secure Command queue Consumer index register	Contains the Secure Command queue index for reads by the consumer.
SMMU_S_EVENTQ_BASE	Secure Event queue Base address register	Contains the base address and attributes for the Secure

Register	Name	Description
		Event queue.
SMMU_S_EVENTQ_PROD	Secure Event queue Producer index register	Contains the Secure Event queue index for writes by the producer.
SMMU_S_EVENTQ_CONS	Secure Event queue Consumer index register	Contains the Secure Event queue index for reads by the consumer.
SMMU_S_EVENTQ_IRQ_CFG0 - SMMU_S_EVENTQ_IRQ_CFG2	Secure Event queue IRQ Configuration registers	Contains the MSI address configuration for the Secure Event queue IRQ.
SMMU_IDR0 - SMMU_IDR3 SMMU_IDR5	SMMU feature Identification Registers	Provides information about the features that the SMMU implementation supports.
SMMU_IIDR	Implementation Identification Register	Provides implementer, part, and revision information for the SMMU implementation.
SMMU_AIDR	Architecture Identification Register	Identifies the SMMU architecture version to which the implementation conforms.
SMMU_CR0	Non-secure global Control Register 0	Provides the controls for the global configuration of the Non-secure SMMU.
SMMU_CR0ACK	Non-secure global Control Register 0	update Acknowledge register Provides acknowledgement of completion of updates to SMMU_CR0.
SMMU_CR1	Non-secure global Control Register 1	Provides the controls for Non-secure table and queue access attributes.
SMMU_CR2	Non-secure global Control Register 2	Provides the controls for the configuration of the global Non-secure features.
SMMU_GBPA	Non-secure Global Bypass Attribute register	Controls the global bypass attributes that are used for transactions from Non-secure streams when the MMU is disabled.
SMMU_IRQ_CTRL	Non-secure Interrupt Control register	Provides IRQ enable flags for edge-triggered wired outputs, if implemented, and MSI writes, if implemented.
SMMU_IRQ_CTRLACK	Non-secure Interrupt Control register	update Acknowledge register Provides acknowledgement of the completion of updates to SMMU_IRQ_CTRL.
SMMU_GERROR	Non-secure Global Error status register	Provides information about Non-secure global

Register	Name	Description
		programming interface errors.
SMMU_GERRORN	Non-secure Global Error acknowledgement register	Contains the acknowledgement fields for SMMU_GERROR errors.
SMMU_GERROR_IRQ_CFG0	Non-secure Global Error IRQ Configuration register 0	Contains the MSI address configuration for the GERROR IRQ.
SMMU_GERROR_IRQ_CFG1	Non-secure Global Error IRQ Configuration register 1	Contains the MSI payload configuration for the GERROR IRQ.
SMMU_GERROR_IRQ_CFG2	Non-secure Global Error IRQ Configuration register 2	Contains the MSI attribute configuration for the GERROR IRQ.
SMMU_STRTAB_BASE	Non-secure Stream Table Base address register	Contains the base address and attributes for the Nonsecure Stream table.
SMMU_STRTAB_BASE_CFG	Non-secure Stream Table Configuration register	Contains configuration fields for the Non-secure Stream table.
SMMU_CMDQ_BASE	Non-secure Command queue Base address register	Contains the base address and attributes for the Non-secure Command queue.
SMMU_CMDQ_PROD	Non-secure Command queue Producer index register	Contains the Non-secure Command queue index for writes by the producer.
SMMU_CMDQ_CONS	Non-secure Command queue Consumer index register	Contains the Non-secure Command queue index for reads by the consumer.
SMMU_EVENTQ_BASE	Non-secure Event queue Base address register	Contains the base address and attributes for the Non-secure Event queue.
SMMU_EVENTQ_PROD	Non-secure Event queue Producer index register	Contains the Non-secure Event queue index for writes by the producer.
SMMU_EVENTQ_CONS	Non-secure Event queue Consumer index register	Contains the Non-secure Event queue index for reads by the consumer.
SMMU_EVENTQ_IRQ_CFG0	Non-secure Event queue IRQ Configuration register 0	Contains the MSI address configuration for the Event queue IRQ.
SMMU_EVENTQ_IRQ_CFG1	Non-secure Event queue IRQ Configuration register 1	Contains the MSI payload configuration for the Event queue IRQ.
SMMU_EVENTQ_IRQ_CFG2	Non-secure Event queue IRQ Configuration register 2	Contains the MSI attribute configuration for the Event queue IRQ.

Register	Name	Description
SMMU_PRIQ_BASE	Non-secure PRI queue Base address register	Contains the base address and attributes for the Non-secure PRI queue.
SMMU_PRIQ_PROD	Non-secure PRI queue Producer index register	Contains the Non-secure PRI queue index for writes by the producer.
SMMU_PRIQ_CONS	Non-secure PRI queue Consumer index register	Contains the Non-secure PRI queue index for reads by the consumer.
SMMU_PRIQ_IRQ_CFG0	Non-secure PRI queue IRQ Configuration register 0	Contains the MSI address configuration for the PRI queue IRQ.
SMMU_PRIQ_IRQ_CFG1	Non-secure PRI queue IRQ Configuration register 1	Contains the MSI payload configuration for the PRI queue IRQ.
SMMU_PRIQ_IRQ_CFG2	Non-secure PRI queue IRQ Configuration register 2	Contains the MSI attribute configuration for the PRI queue IRQ.

SMMUv3 PMCG

Register	Name	Description
SMMU_PMCG_EVCNTR0 - SMMU_PMCG_EVCNTR3	SMMU PMCG Event Counter registers	Contains the values of the event counters.
SMMU_PMCG_EVTYPER0 - SMMU_PMCG_EVTYPER3	SMMU PMCG Event Type configuration registers	Configures the events that the corresponding counter counts.
SMMU_PMCG_SVR0 - SMMU_PMCG_SVR3	SMMU PMCG Shadow Value Registers	Contains the shadow value of the corresponding event counter.
SMMU_PMCG_SMR0	SMMU PMCG Stream Match filter Register	Configures the stream match filter for the corresponding event counter.
SMMU_PMCG_CNTENSET0	SMMU PMCG Counter Enable Set register	Provides the set mechanism for the counter enables.
SMMU_PMCG_CNTENCLR0	SMMU PMCG Counter Enable Clear register	Provides the clear mechanism for the counter enables.
SMMU_PMCG_INTENSET0	SMMU PMCG Interrupt contribution Enable Set register	Provides the set mechanism for the counter interrupt contribution enables.
SMMU_PMCG_INTENCLR0	SMMU PMCG Interrupt contribution Enable Clear register	Provides the clear mechanism for the counter interrupt enables.
SMMU_PMCG_OVSCLR0	SMMU PMCG Overflow Status Clear register	Provides the clear mechanism for the overflow

Register	Name	Description
		status bits and provides read access to the overflow status bit values.
SMMU_PMCG_OVSSET0	SMMU PMCG Overflow Status Set register	Provides the set mechanism for the overflow status bits and provides read access to the overflow status bit values.
SMMU_PMCG_CAPR	SMMU PMCG Counter shadow value Capture Register	Controls the counter shadow value capture mechanism.
SMMU_PMCG_SCR	SMMU PMCG Secure Control Register	Secure Control Register.
SMMU_PMCG_CFGR	SMMU PMCG Configuration information Register	Provides information about the PMCG implementation.
SMMU_PMCG_CR	SMMU PMCG Control Register	Contains the Performance Monitor control flags.
SMMU_PMCG_CEID0 - SMMU_PMCG_CEID1	SMMU PMCG Common Event ID registers	Contains the lower and upper 64 bits of the Common Event identification bitmap.
SMMU_PMCG_IRQ_CTRL	SMMU PMCG IRQ enable register	Contains the Performance Monitors IRQ enable.
SMMU_PMCG_IRQ_CTRLACK	SMMU PMCG IRQ enable Acknowledge register	Provides acknowledgement of the completion of updates to SMMU_PMCG_IRQ_CTRL.
SMMU_PMCG_AIDR	SMMU PMCG Architecture Identification Register	Provides the Performance Monitor Architecture Identification.
SMMU_PMCG_ID_REGS	ID registers	IMPLEMENTATION DEFINED.
SMMU_PMCG_PMAUTHSTATUS	PMU Authentication Status register	Performance Monitor authentication status.
SMMU_PMCG_PMDEVARCH	PMU Device Architecture register	Performance Monitor architecture identifier.
SMMU_PMCG_PMDEVTYPE	PMU Device Type register	Performance Monitor device type.

TCU ID

Register	Offset	Bits	Value	Description
SMMU_PIDR4	0x00FD0	[7:4]	0x0	4KB region count.
		[3:0]	0x4	JEP106 continuation code for Arm.
SMMU_PIDR5	0x00FD4	[7:0]	0x00	Reserved.
SMMU_PIDR6	0x00FD8	[7:0]	0x00	Reserved.
SMMU_PIDR7	0x00FDC	[7:0]	0x00	Reserved.

Register	Offset	Bits	Value	Description
SMMU_PIDR0	0x00FE0	[7:0]	0x84	Part number[7:0].
SMMU_PIDR1	0x00FE4	[7:4]	0xB	JEP106 ID code[3:0] for Arm.
		[3:0]	0x4	Part number[11:8].
SMMU_PIDR2	0x00FE8	[7:4]	0x2	MMU-600 major revision. The value 0x2 indicates major product revision r2.
		[3]	0b1	The component uses a manufacturer identity code that JEDEC allocates, according to the JEP106 specification.
		[2:0]	0b011	JEP106 ID code[6:4] for Arm.
SMMU_PIDR3	0x00FEC	[7:4]	MAX[0x2,ecorevnum]	MMU-600 minor revision. The value 0x2 indicates minor product revision p2.
		[3:0]	0x0	CMOD. This field is not used.
SMMU_CIDR0	0x00FF0	[7:0]	0x0D	Preamble.
SMMU_CIDR1	0x00FF4	[7:0]	0xF0	
SMMU_CIDR2	0x00FF8	[7:0]	0x05	
SMMU_CIDR3	0x00FFC	[7:0]	0xB1	

TBU ID

Same as register of TCU ID。

TCU PMU

Register	Offset	Bits	Value	Description
SMMU_PMCG_PMAUTHSTATUS	0x02FB8	[7:0]	0x00	No authentication interface is implemented.
SMMU_PMCG_PIDR4	0x02FD0	[7:4]	0x0	4KB region count.
		[3:0]	0x4	JEP106 continuation code for Arm.
SMMU_PMCG_PIDR5	0x02FD4	[7:0]	0x00	Reserved.
SMMU_PMCG_PIDR6	0x02FD8	[7:0]	0x00	Reserved.
SMMU_PMCG_PIDR7	0x02FDC	[7:0]	0x00	Reserved.
SMMU_PMCG_PIDR0	0x02FE0	[7:0]	0x83	Part number[7:0].
SMMU_PMCG_PIDR1	0x02FE4	[7:4]	0xB	JEP106 ID code[3:0] for Arm.
		[3:0]	0x4	Part number[11:8].

Register	Offset	Bits	Value	Description
SMMU_PMCG_PIDR2	0x02FE8	[7:4]	0x2	MMU-600 major revision. The value 0x2 indicates major product revision r2.
		[3]	0b1	The component uses a manufacturer identity code that JEDEC allocates, according to the JEP106 specification.
		[2:0]	0b011	JEP106 ID code[6:4] for Arm.
SMMU_PMCG_PIDR3	0x02FEC	[7:4]	MAX[0x2,ecorevnum]	MMU-600 minor revision. The value 0x2 indicates minor product revision p2.
		[3:0]	0x0	CMOD. This field is not used.
SMMU_PMCG_CIDR0	0x02FF0	[7:0]	0x0D	Preamble.
SMMU_PMCG_CIDR1	0x02FF4	[7:0]	0x90	
SMMU_PMCG_CIDR2	0x02FF8	[7:0]	0x05	
SMMU_PMCG_CIDR3	0x02FFC	[7:0]	0xB1	

TBU PMU

Same as register of TCU PMU。

TCU implement define registers

Register	Offset	Bits	Name	Value	Description
TCU_CTRL	0x08E00	[31:20]	AUX[31:20]	0x000	Leave each of these bits as 0.
		[19]	ASID_VMID_HASH	0x0	0 VMID and input address are used for walk cache indexing. Transactions with the same VMID and input address, but different ASID, use the same walk cache index. Walk cache utilization is poor if such transactions are common. 1 ASID, VMID, and input address are used for walk cache indexing. Transactions with the same VMID and input address, but different ASID, use different walk cache indexes. This improves walk cache utilization if different ASIDs are used for the same input address and VMID, but invalidation performance is worse for invalidations that

Register	Offset	Bits	Name	Value	Description
					do not provide an ASID because the whole cache must be walked instead of invalidating based on a specific index.
		[18:16]	AUX[18:16]	0x0	Leave each of these bits as 0.
		[15]	WCS2L3_DIS	0x0	Walk cache disable: 0 Stage 2 level 3 walk cache is enabled. 1 Stage 2 level 3 walk cache is disabled.
		[14]	WCS2L2_DIS	0x0	Walk cache disable: 0 Stage 2 level 2 walk cache is enabled. 1 Stage 2 level 2 walk cache is disabled.
		[13]	WCS2L1_DIS	0x0	Walk cache disable: 0 Stage 2 level 1 walk cache is enabled. 1 Stage 2 level 1 walk cache is disabled
		[12]	WCS2L0_DIS	0x0	Walk cache disable: 0 Stage 2 level 0 walk cache is enabled. 1 Stage 2 level 0 walk cache is disabled.
		[11]	WCS1L3_DIS	0x0	Walk cache disable: 0 Stage 1 level 3 walk cache is enabled. 1 Stage 1 level 3 walk cache is disabled.
		[10]	WCS1L2_DIS	0x0	Walk cache disable: 0 Stage 1 level 2 walk cache is enabled. 1 Stage 1 level 2 walk cache is disabled.
		[9]	WCS1L1_DIS	0x0	Walk cache disable: 0 Stage 1 level 1 walk cache is enabled. 1 Stage 1 level 1 walk cache is disabled.
		[8]	WCS1L0_DIS	0x0	Walk cache disable: 0 Stage 1 level 0 walk cache is enabled. 1 Stage 1 level 0 walk cache is disabled.
		[7:0]	AUX[7:0]	0x00	Leave each of these bits as 0.
TCU_QOS	0x08E04	[31:28]		0x0	Reserved.
		[27:24]	QOS_DVMSYNC	0x0	The AxQOS value that is used for DVM Sync Completion messages.
		[23:20]	QOS_MSI	0x0	The AxQOS value that is used for MSIs.

Register	Offset	Bits	Name	Value	Description
		[19:16]	QOS_QUEUE	0x0	The AxQOS value that is used for queue accesses.
		[15:12]	QOS_PTW3	0x0	The AxQOS value that is used for translation table walks for translations where TCU_NODE_CTRLn.PRIORITY = 3 for the requesting node.
		[11:8]	QOS_PTW2	0x0	The AxQOS value that is used for translation table walks for translations where TCU_NODE_CTRLn.PRIORITY = 2 for the requesting node.
		[7:4]	QOS_PTW1	0x0	The AxQOS value that is used for translation table walks for translations where TCU_NODE_CTRLn.PRIORITY = 1 for the requesting node.
		[3:0]	QOS_PTW0	0x0	The AxQOS value that is used for translation table walks for translations where TCU_NODE_CTRLn.PRIORITY = 0 for the requesting node.
TCU_CFG	0x08E08	[31:16]		0x0000	Reserved.
		[15:4]	XLATE_SLOTS	0x000	The number of translation slots that are available for sharing between all nodes. The reset value of this field is TCUCFG_XLATE_SLOTS.
		[3:0]		0x0	Reserved.
TCU_STATUS	0x08E10	[31:16]		0x0000	Reserved.
		[15:4]	GNT_XLATE_SLOTS	0x000	GNT_XLATE_SLOTS is the number of translation slots that are currently allocated to connected nodes. You can use this value for debugging purposes.
		[3:0]		0x0	Reserved.
TCU_SCR	0x08E18	[31:4] -		0x0000_000	Reserved.
		[3]	NS_INIT	0x0	Non-secure register access to SMMU_S_INIT. When this bit is set to 0, Non-secure accesses to the SMMU_S_INIT register are RAZ/WI. The sec_override input signal defines the reset

Register	Offset	Bits	Name	Value	Description
					value of this bit.
		[2]		0x0	Reserved.
		[1]	NS_RAS	0x0	Non-secure register access is permitted for RAS registers. When this bit is set to 0, Non-secure accesses to register addresses 0x08E80–0x08EC0 are RAZ/WI. The sec_override input signal defines the reset value of this bit.
		[0]	NS_UARCH	0x0	Non-secure register access is permitted for MMU-600 registers. When this bit is set to 0, Non-secure accesses to register addresses 0x08E00–0x08E7C and 0x09000–0x093FC are RAZ/WI. The sec_override input signal defines the reset value of this bit. If your implementation might use Secure translation, Arm recommends setting this bit to 0.
TCU_NODE_CTRLn	0x09000-0x093FC	[31:5]		0x0000_000	Reserved.
		[4]	DIS_DVM	0x0	Disable DVM. When this bit is set to 1, the corresponding node does not participate in DVM invalidation. Software should set this bit to 1 if all the following are true: • The node is slow to respond to invalidations that are issued over DTI • Software has knowledge that the node does not require to be part of the DVM domain • Software has knowledge that invalidations for the node can be issued using the Command queue Note This bit is ignored for connected DTI-ATS masters, because they never participate in DVM invalidation.
		[3:2]		0x0	Reserved
		[1:0]	PRI_LEVEL	0x0	Priority level. This field indicates the priority level of

Register	Offset	Bits	Name	Value	Description
					the corresponding node. Translation requests from a node with a higher priority level are normally progressed before requests from a node with a lower priority level.
TCU_NODE_STATUSn	0x09400-0x097FC	[31:2]		0x0000_0000	Reserved.
		[1]	ATS	0x0	ATS implemented: 0 The corresponding node is a TBU that is connected to the TCU using the DTI-TBU protocol. 1 The corresponding node is a PCIe Root Complex that supports ATS, and is connected to the TCU using the DTI-ATS protocol.
		[0]	CONNECTED	0x0	DTI link is connected: 0 The DTI link for the corresponding node is not connected. 1 The DTI link for the corresponding node is connected. If a DTI link is not connected, accesses to TBU registers are RAZ/WI. However, the state might change between reading this register and attempting to access the TBU.

TBU architectural registers

Register	Offset	Bits	Name	Value	Description
TBU_CTRL	0x08E00	[31:16]		0x0000	Reserved
		[15:0]	AUX[15:0]	0x0000	Leave each of these bits as 0
TBU_SCR	0x08E18	[31:2] -		0x0000_0000	Reserved.
		[1]	NS_RAS	0x0	Non-secure register access to RAS registers: 0 Non-secure accesses to register addresses 0x08E80–0x08EC0 are RAZ/WI. 1 Non-secure access to RAS registers is permitted
		[0]	NS_UARCH	0x0	Non-secure register access to TBU_CTRL: 0 Non-secure accesses to TBU_CTRL are RAZ/WI. 1 Non-secure

Register	Offset	Bits	Name	Value	Description
					accesses to TBU_CTRL are permitted.

TCU RAS

Register	Offset	Bits	Name	Value	Description
TBU_ERRFR	0x08E80	[31:8]		0x0000_00	Reserved
		[7:6]	FI	0x2	The value 0b10 indicates that the fault handling interrupt is controllable
		[5:2]		0x0	Reserved.
		[1:0]	ED	0x1	The value 0b01 indicates that TBU error detection is always enabled
TBU_ERRCTLR	0x08E88	[31:4]		0x0000_000	Reserved.
		[3]	FI	0x1	Enable fault handling interrupts: 0 No interrupt is generated when a fault occurs 1 An interrupt is generated on ras_irpt when a fault occurs
		[2:0]		0x0	Reserved.
TBU_ERRSTATUS	0x08E90	[31]		0x0	Reserved.
		[30]	V		Register valid. This bit is set to 1 to indicate that at least one RAS error was recorded. Clear this bit by writing a 1 to it. If CE is not 0b00 and is not being cleared, the write is ignored. A write of 0 is ignored.
		[29:28]		0x0	Reserved.
		[27]	OF	0x0	Overflow. This bit is set to 1 to indicate that multiple correctable errors were recorded. That is, at least one correctable error was recorded when CE != 0b00. Clear this bit by writing a 1 to it. A write of 0 is ignored.
		[26]		0x0	Reserved.
		[25:24]	CE	0x0	Correctable Error. This field is set to 0b10 to indicate that a corrected error occurred. Clear this field by writing 0b11 to it. If OF is set to 1 and is not being cleared, the write is ignored. A write of any value other than 0b11 is ignored.
		[23:16]		0x00	Reserved.
		[15:8]	IERR	0x00	IMPLEMENTATION DEFINED error code. When SERR is not set to 0,

Register	Offset	Bits	Name	Value	Description
					this field indicates the source of the error, as follows: 0x00 Stage 1, level 0 walk cache. 0x01 Stage 1, level 1 walk cache. 0x02 Stage 1, level 2 walk cache. 0x03 Stage 1, level 3 walk cache. 0x04 Stage 2, level 0 walk cache. 0x05 Stage 2, level 1 walk cache. 0x06 Stage 2, level 2 walk cache. 0x07 Stage 2, level 3 walk cache. 0x08 Configuration cache. Writes to this field are ignored
		[7:0]	SERR	0x00	Error code. This field provides information about the earliest unacknowledged correctable error, as follows: 0x00 No error. This code occurs when CE = 0b00. 0x07 Main TLB tag is corrupted. This code can occur when CE != 0b00. 0x08 Main TLB data is corrupted. This code can occur when CE != 0b00. Writes to this field are ignored.
TBU_ERRGEN	0x08EC0	[31:2]		0x0000_0000	Reserved.
		[3]	TCC	0x0	Configuration cache tag parity error: 0 No tag parity error is written to the configuration cache. 1 Entries that are written to the configuration cache include a tag parity error. A fault occurs when the entry is used.
		[2]	DCC	0x0	Configuration cache data parity error: 0 No data parity error is written to the configuration cache. 1 Entries that are written to the configuration cache include a data parity error. A fault occurs when the entry is used. Note Tag parity errors mask data parity errors. When testing data parity error functionality, ensure that software does not set this bit and the TCC bit at the same time.
		[1]	TWC	0x0	Walk cache tag parity error: 0 No tag parity error is written to the walk cache. 1 Entries that are written to the walk cache include a tag parity error. A fault occurs when the entry is used.
		[0]	DWC	0x0	Walk cache data parity error: 0 No data parity error is written to the walk cache. 1 Entries that are

Register	Offset	Bits	Name	Value	Description
					written to the walk cache include a data parity error. A fault occurs when the entry is used. Note Tag parity errors mask data parity errors. When testing data parity error functionality, ensure that software does not set this bit and the TWC bit at the same time.

TBU RAS

Register	Offset	Bits	Name	Value	Description
TBU_ERRFR	0x08E80	[31:8]		0x0000_00	Reserved
		[7:6]	FI	0x2	The value 0b10 indicates that the fault handling interrupt is controllable
		[5:2]		0x0	Reserved.
		[1:0]	ED	0x1	The value 0b01 indicates that TBU error detection is always enabled
TBU_ERRCTLR	0x08E88	[31:4]		0x0000_000	Reserved.
		[3]	FI	0x1	Enable fault handling interrupts: 0 No interrupt is generated when a fault occurs 1 An interrupt is generated on ras_irpt when a fault occurs
		[2:0]		0x0	Reserved.
TBU_ERRSTATUS	0x08E90	[31]		0x0	Reserved.
		[30]	V		Register valid. This bit is set to 1 to indicate that at least one RAS error was recorded. Clear this bit by writing a 1 to it. If CE is not 0b00 and is not being cleared, the write is ignored. A write of 0 is ignored.
		[29:28]		0x0	Reserved.
		[27]	OF	0x0	Overflow. This bit is set to 1 to indicate that multiple correctable

Register	Offset	Bits	Name	Value	Description
					errors were recorded. That is, at least one correctable error was recorded when CE != 0b00. Clear this bit by writing a 1 to it. A write of 0 is ignored.
		[26]		0x0	Reserved.
		[25:24]	CE	0x0	Correctable Error. This field is set to 0b10 to indicate that a corrected error occurred. Clear this field by writing 0b11 to it. If OF is set to 1 and is not being cleared, the write is ignored. A write of any value other than 0b11 is ignored.
		[23:8]		0x0000	Reserved.
		[7:0]	SERR	0x00	Error code. This field provides information about the earliest unacknowledged correctable error, as follows: 0x00 No error. This code occurs when CE = 0b00. 0x07 Main TLB tag is corrupted. This code can occur when CE != 0b00. 0x08 Main TLB data is corrupted. This code can occur when CE != 0b00. Writes to this field are ignored.
TBU_ERRGEN	0x08EC0	[31:2]		0x0000_0000	Reserved.
		[1]	TMTLB	0x0	Main TLB tag parity error: 0 No tag parity error is written to the Main TLB. 1 Entries that are written to the Main TLB include a tag parity error. A fault occurs when the entry is used.
		[0]	DMTLB	0x0	Main TLB data parity error: 0 No data parity error is written to the Main TLB. 1 Entries that are written to the Main TLB include a data

Register	Offset	Bits	Name	Value	Description
					parity error. A fault occurs when the entry is used. Note Tag parity errors mask data parity errors. When testing data parity error functionality, ensure that software does not set this bit and the TMTLB bit at the same time.

3.10 Mailbox

3.10.1 Function description

Mailbox is an interrupt module in the multi-core SOC system, because there may be different services between the cores, so the hardware design and allocation of one or two interrupts can no longer meet the needs of the system, and the software is also very difficult to expand, so mailbox can be understood as an interrupt module freely defined by the software.

There are MCPUS, SPU, NPU and DSP in the system, each module can generate interrupts for other modules through Mailbox, and after receiving the data, the module will carry out interrupt processing, and after completing the interrupt processing, the corresponding interrupt will be cleared for the next processing.

Mailbox hais following features:

- Interrupt registers, which are used to enable interrupts.
- Interrupt mask.
- FIFO data width is 64bit，fifo depth is 8.
- FIFO status register.
- Mailbox lock function.

3.10.2 Block diagram

The CPU accesses the mailbox through the APB bus, configures interrupt signals and interrupt enable signals, writes data to the FIFO, then generates an interrupt to the destination CPU after the data is written to the FIFO.The follow diagram is maibox function description.

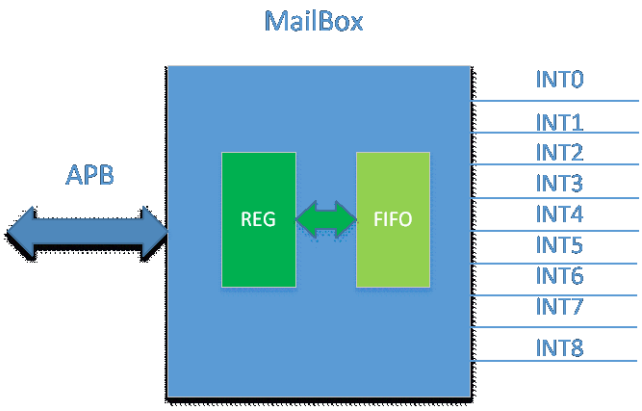


Figure 3-36 Mailbox block diagram

The system contains 16 mailboxes, before operating on the mailbox, you need to write WR_LOCK register, and then read the WR_LOCK register, judge whether the value of read and write is consistent, if the value of read and write is consistent, it means that the mailbox is not occupied by other devices and is in an idle state, otherwise the mailbox is occupied, the operation of the current mailbox should be stopped, and you can choose other mailbox to judge the state. If the MCPU needs to send an interrupt request to the SCPU, the MCPU first selects one of the mailboxes and sets the interrupt register bit1 to 1, which means that the interrupt is output to the first SCPU, and the data is written to the FIFO to generate an interrupt, so as to realize the interaction between the two CPU systems.

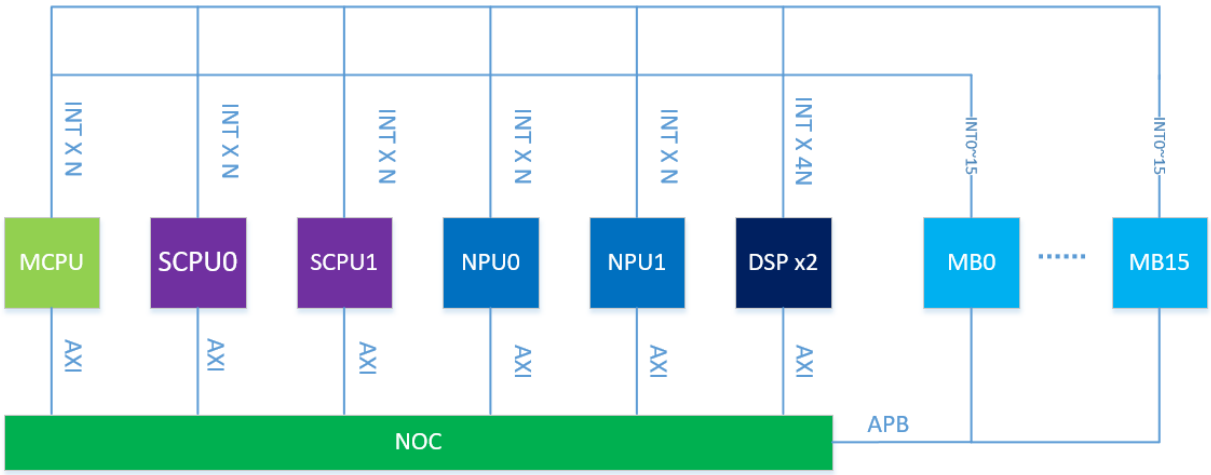


Figure 3-37 Interrupt interconnect

The following table describes the mapping between mailbox's interrupts and CPUs.

Table 3-232 Interrupt Mapping

Interrupt ID	CPU
INT0	MCPU
INT1	SCPU0
INT2	SCPU1
INT3	NPU0
INT4	NPU1

Interrupt ID	CPU
INT5	DSP0
INT6	DSP1

3.10.3 Initialization

Mailbox's Initialization as follow:

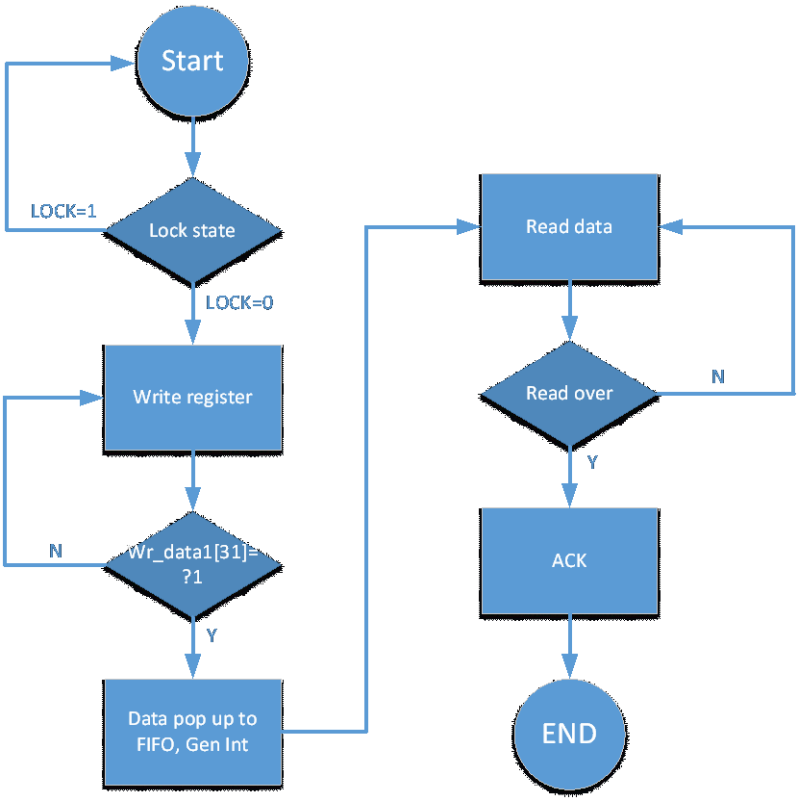


Figure 3-38 Flow of Mailbox Initial

3.10.4 Register Description

Mailbox address mapping：

Table 3-233 Mailbox address mapping

Mailbox Num	Address range	Memory region size
Mailbox0	0x50a00000~0x50a0ffff	64K
Mailbox1	0x50a10000~0x50a1ffff	64K
Mailbox2	0x50a20000~0x50a2ffff	64K
Mailbox3	0x50a30000~0x50a3ffff	64K
Mailbox4	0x50a40000~0x50a4ffff	64K
Mailbox5	0x50a50000~0x50a5ffff	64K
Mailbox6	0x50a60000~0x50a6ffff	64K
Mailbox7	0x50a70000~0x50a7ffff	64K
Mailbox8	0x50a80000~0x50a8ffff	64K

Mailbox Num	Address range	Memory region size
Mailbox9	0x50a90000~0x50a9ffff	64K
Mailbox10	0x50aa0000~0x50aaffff	64K
Mailbox11	0x50ab0000~0x50abffff	64K
Mailbox12	0x50ac0000~0x50acffff	64K
Mailbox13	0x50ad0000~0x50adffff	64K
Mailbox14	0x50ae0000~0x50aeffff	64K
Mailbox15	0x50af0000~0x50afffff	64K

Mailbox's register address：

Table 3-234 Registers of Mailbox

Addr	Name	function	Bit range		R/W	Ini value	description
16'h00	WR_DATA_0	wr_data_0	31	0	RW	32'h0	Register for write fifo
16'h04	WR_DATA_1	wr_data_1	31	0	RW	32'h0	Register for write fifo ,if bit31 is 1'b1, the data will write to fifo.
16'h08	RD_DATA_0	rd_data_0	31	0	RW	32'b0	Register for read fifo
16'h0C	RD_DATA_0	rd_data_1	31	0	RW	32'b0	Register for read fifo, write the rd_data1[31] to zero, and then the next read request can be made, and if it is answered and the FIFO is empty, the current interrupt can be cleared
16'h10	FIFO_STATUS	Reserved	31	2		29'h0	
		empty	1	1	RO	1'b0	FIFO empty
		full	0	0	RO	1'b0	FIFO full
16'h14	MB_ERR	Reserved	31	1		31'h0	
		mb_err	0	0	RC	1'b0	Fifo error instructions
16'h18	MB_INT	Reserved	31	12		24'h0	
		mb_int11	11	11	RW	1'b0	Interrupt 11 enble
		mb_int10	10	10	RW	1'b0	Interrupt 10 enble
		mb_int9	9	9	RW	1'b0	Interrupt 9 enble
		mb_int8	8	8	RW	1'b0	Interrupt 8 enble
		mb_int7	7	7	RW	1'b0	Interrupt 7 enble
		mb_int6	6	6	RW	1'b0	Interrupt 6 enble
		mb_int5	5	5	RW	1'b0	Interrupt 5 enble
		mb_int4	4	4	RW	1'b0	Interrupt 4 enble
		mb_int3	3	3	RW	1'b0	Interrupt 3 enble。

Addr	Name	function	Bit range		R/W	Ini value	description
		mb_int2	2	2	RW	1'b0	Interrupt 2 enble
		mb_int1	1	1	RW	1'b0	Interrupt 1 enble
		mb_int0	0	0	RW	1'b0	Interrupt 0 enble
16'h1C	WR_LOCK	Reserved	31	12		24'h0	
		wr_lock11	11	11	RW	1'b0	Write clock 11
		wr_lock10	10	10	RW	1'b0	Write clock 10
		wr_lock9	9	9	RW	1'b0	Write clock 9
		wr_lock8	8	8	RW	1'b0	Write clock 8
		wr_lock7	7	7	RW	1'b0	Write clock 7
		wr_lock6	6	6	RW	1'b0	Write clock 6
		wr_lock5	5	5	RW	1'b0	Write clock 5
		wr_lock4	4	4	RW	1'b0	Write clock 4
		wr_lock3	3	3	RW	1'b0	Write clock 3
		wr_lock2	2	2	RW	1'b0	Write clock 2
		wr_lock1	1	1	RW	1'b0	Write clock 1
		wr_lock0	0	0	RW	1'b0	Write clock 0

3.11 WDT

3.11.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that caused by conflicting parts or programs in a SOC. When the counter reaches zero, depending on the output response mode selected, either a system reset or an interrupt occurs.

The following functions are supported：

- It supports APB 4.0 and has a bus width of 32 bits
- The counter bit width is 32 bits
- When system reset is released, it is disabled. The module is started according to the user's software configuration
- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
- If a timeout occurs the WDT can perform one of the following operations:
  1. Generate a system reset
  2. First generate an interrupt and even if it is cleared (or not) by the service routine by the time a second timeout occurs then generate a system reset.
- Programmable and hard coded reset pulse length
- Prevention of accidental restart of the WDT
- Prevention of accidental disabling of the WDT
- Outputs a high active interrupt

3.11.2 Block diagram

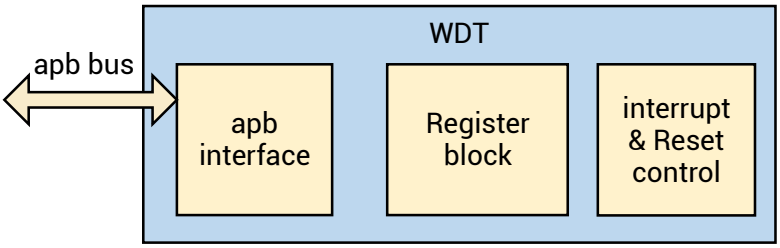


Figure 3-39 WDT block diagram

3.11.3 Function description

3.11.3.1 reset pulse length

The reset pulse width is the number of pclk cycles, which is determined by WDT_CR. RPL

3.11.3.2 Timeout Period Values

WDT has a fixed timeout period range，See Register WDT_TORR. TOP.

3.11.3.3 Counter Restart

As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT_CRR).

3.11.4 Register description

The base address please refer to table 12-1.

3.11.4.1 Register overview

Register	Offset	Description
WDT_CR	0x0	Control Register
WDT_TORR	0x4	Timeout Range Register
WDT_CCVR	0x8	Current Counter Value Register
WDT_CRR	0xc	Counter Restart Register
WDT_STAT	0x10	Interrupt Status Register
WDT_EOI	0x14	Interrupt Clear Register
WDT_PROT_LEVEL	0x1c	WDT Protection level
WDT_COMP_PARAM_5	0xe4	Component Parameters Register 5
WDT_COMP_PARAM_4	0xe8	Component Parameters Register 4
WDT_COMP_PARAM_3	0xec	Component Parameters Register 3
WDT_COMP_PARAM_2	0xf0	Component Parameters Register 2
WDT_COMP_PARAM_1	0xf4	Component Parameters Register 1

Register	Offset	Description
WDT_COMP_VERSION	0xf8	Component Version Register
WDT_COMP_TYPE	0xfc	Component Type Register

3.11.4.2 Register detail

WDT_CR

Name: Control Register

Description: Control Register

Size: 32 bits

Offset: 0x0

Bits	Name	Memory Access	Description
31:6	RSVD_WDT_CR	R	WDT_CR[31:6] Reserved bits and read as zero (0). **Value After Reset:** 0x0
5	NO_NAME	R/W	Redundant R/W bit. Included for ping test purposes, as it is the only R/W register bit that is in every configuration of the wdt. **Value After Reset:** 0x0
4:2	RPL	R/W	Reset pulse length. Writes have no effect when the configuration parameter WDT_HC_RPL is 1, making the register bits read-only. This is used to select the number of pclk cycles for which the system reset stays asserted. The range of values available is 2 to 256 pclk cycles. Note: When WDT_ASYNC_CLK_MODE_ENABLE = 1, the total reset pulse length also includes the reset synchronization delay and the time taken for pclk to be made available. For details, refer to "System Resets" section of wdt Databook. Values: 0x0 (PCLK_CYCLES_2): 2 pclk cycles 0x1 (PCLK_CYCLES_4): 4 pclk cycles 0x2 (PCLK_CYCLES_8): 8 pclk cycles 0x3 (PCLK_CYCLES_16): 16 pclk cycles 0x4 (PCLK_CYCLES_32): 32 pclk cycles 0x5 (PCLK_CYCLES_64): 64 pclk cycles 0x6 (PCLK_CYCLES_128): 128 pclk cycles 0x7 (PCLK_CYCLES_256): 256 pclk cycles **Value After Reset:** 0x0
1	RMOD	R/W	Response mode. Writes have no effect when the parameter WDT_HC_RMOD = 1, thus this register becomes read-only. Selects the output response generated to a timeout. Values:

Bits	Name	Memory Access	Description
			0x0 (RESET): Generate a system reset 0x1 (INTERRUPT): First generate an interrupt and even if it is cleared by the time a second timeout occurs then generate a system reset **Value After Reset:** 0x1
0	WDT_EN	R/W	WDT enable. When the configuration parameter WDT_ALWAYS_EN = 0, this bit can be set; otherwise, it is read-only. This bit is used to enable and disable the wdt. When disabled, the counter does not decrement. Thus, no interrupts or system resets are generated. The wdt is used to prevent system lock-up. To prevent a software bug from disabling the wdt, once this bit has been enabled, it can be cleared only by a system reset. Values: 0x0 (DISABLED): Watchdog timer disabled 0x1 (ENABLED): Watchdog timer enabled **Value After Reset:** 0x0

WDT_TORR

Name: Timeout Range Register

Description: Timeout Range Register

Size: 32 bits

Offset: 0x4

Bits	Name	Memory Access	Description
31:8	RSVD_WDT_TORR	R	WDT_TORR[31:8] Reserved and read as zero (0). **Value After Reset:** 0x0
7:4	RSVD_TOP_INIT	R	WDT_TORR[7:4] Reserved field for TOP_INIT **Value After Reset:** 0x0
3:0	TOP	R/W	Timeout period. Writes have no effect when the configuration parameter WDT_HC_TOP=1, thus making this register read-only. This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick). This register should be written after reset and before the WDT is enabled. The range of values is limited by the WDT_CNT_WIDTH. If TOP is programmed to select a range that is greater than the counter

Bits	Name	Memory Access	Description
			width, the timeout period is truncated to fit to the counter width. This affects only the non-user specified values as users are limited to these boundaries during configuration. The range of values available for a 32-bit watchdog counter are: Where i = TOP and t = timeout period For i = 0 to 15 if WDT_USE_FIX_TOP==1 t = 2^(16 + i) else t = WDT_USER_TOP_(i) Values: 0x0 (USER0_OR_64K): Time out of WDT_USER_TOP_0 or 64K Clocks 0x1 (USER1_OR_128K): Time out of WDT_USER_TOP_1 or 128K Clocks 0x2 (USER2_OR_256K): Time out of WDT_USER_TOP_2 or 256K Clocks 0x3 (USER3_OR_512K): Time out of WDT_USER_TOP_3 or 512K Clocks 0x4 (USER4_OR_1M): Time out of WDT_USER_TOP_4 or 1M Clocks 0x5 (USER5_OR_2M): Time out of WDT_USER_TOP_5 or 2M Clocks 0x6 (USER6_OR_4M): Time out of WDT_USER_TOP_6 or 4M Clocks 0x7 (USER7_OR_8M): Time out of WDT_USER_TOP_7 or 8M Clocks 0x8 (USER8_OR_16M): Time out of WDT_USER_TOP_8 or 16M Clocks 0x9 (USER9_OR_32M): Time out of WDT_USER_TOP_9 or 32M Clocks 0xa (USER10_OR_64M): Time out of WDT_USER_TOP_10 or 64M Clocks 0xb (USER11_OR_128M): Time out of WDT_USER_TOP_11 or 128M Clocks 0xc (USER12_OR_256M): Time out of WDT_USER_TOP_12 or 256M Clocks 0xd (USER13_OR_512M): Time out of WDT_USER_TOP_13 or 512M Clocks 0xe (USER14_OR_1G): Time out of WDT_USER_TOP_14 or 1G Clocks

Bits	Name	Memory Access	Description
			0xf (USER15_OR_2G): Time out of WDT_USER_TOP_15 or 2G Clocks **Value After Reset:** 0x0 Note: **WDT_USE_FIX_TOP** = 1

WDT_CCVR

Name: Current Counter Value Register

Description: Current Counter Value Register.

Size: 32 bits

Offset: 0x8

Bits	Name	Memory Access	Description
31:0	WDT_CCVR	R	WDT Current Counter Value Register. This register, when read, is the current value of the internal counter. This value is read coherently when ever it is read, which is relevant when the APB_DATA_WIDTH is less than the counter width. **Value After Reset:** 0xffff Volatile: true

WDT_CRR

Name: Counter Restart Register

Description: Counter Restart Register

Size: 32 bits

Offset: 0xc

Bits	Name	Memory Access	Description
31:8	RSVD_WDT_CRR	W	WDT_CRR[31:24] Reserved bits - Write Only **Value After Reset:** 0x0
7:0	WDT_CRR	W	Counter Restart Register. This register is used to restart the WDT counter. As a safety feature to prevent accidental restarts, the value 0x76 must be written. A restart also clears the WDT interrupt. Reading this register returns zero.

Bits	Name	Memory Access	Description
			Values: 0x76 (RESTART): Watchdog timer restart command **Value After Reset:** 0x0

WDT_STAT

Name: Interrupt Status Register

Description: Interrupt Status Register

Size: 32 bits

Offset: 0x10

Bits	Name	Memory Access	Description
31:1	RSVD_WDT_STAT	R	WDT_STAT[31:1] Reserved bits - Read Only **Value After Reset:** 0x0 Volatile: true
0	WDT_STAT	R	Interrupt status register This register shows the interrupt status of the WDT. Values: 0x0 (INACTIVE): Interrupt is inactive 0x1 (ACTIVE): Interrupt is active regardless of polarity **Value After Reset:** 0x0 Volatile: true

WDT_EOI

Name: Interrupt Clear Register

Description: Interrupt Clear Register

Size: 32 bits

Offset: 0x14

Bits	Name	Memory Access	Description
31:1	RSVD_WDT_EOI	R	RSVD_WDT_EOI[31:1] Reserved bits and read as zero (0). **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
			Volatile: true
0	WDT_EOI	R	Interrupt Clear Register. Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter. **Value After Reset:** 0x0 Volatile: true

WDT_PROT_LEVEL

Name: WDT Protection level

Description: WDT Protection level register

Size: 32 bits

Offset: 0x1c

Bits	Name	Memory Access	Description
31:3	RSVD_WDT_PROT_LEVEL	R	WDT_PROT_LEVEL[31:3] Reserved field- read-only **Value After Reset:** 0x0
2:0	WDT_PROT_LEVEL	R/W	Protection level register. Enabling protection on any of its three bits would require a match on the input PPROT signal to gain access to protected registers of the WDT. **Value After Reset:** 0x2

WDT_COMP_PARAM_5

Name: Component Parameters Register 5

Description: Component Parameters Register 5

Note: This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).

Size: 32 bits

Offset: 0xe4

Bits	Name	Memory Access	Description
31:0	CP_WDT_USER_TOP_MAX	R	Upper limit of Timeout Period parameters. The value of

Bits	Name	Memory Access	Description
			this register is derived from the WDT_USER_TOP_* coreConsultant parameters. **Value After Reset:** 0x0

WDT_COMP_PARAM_4

Name: Component Parameters Register 4

Description: Component Parameters Register 4

Note: This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).

Size: 32 bits

Offset: 0xe8

Bits	Name	Memory Access	Description
31:0	CP_WDT_USER_TOP_INIT_MAX	R	Upper limit of Initial Timeout Period parameters. The value of this register is derived from the WDT_USER_TOP_INIT_* coreConsultant parameters. **Value After Reset:** 0x0

WDT_COMP_PARAM_3

Name: Component Parameters Register 3

Description: Component Parameters Register 3

Note: This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).

Size: 32 bits

Offset: 0xec

Bits	Name	Memory Access	Description
31:0	CD_WDT_TOP_RST	R	The value of this register is derived from the WDT_TOP_RST coreConsultant parameter. **Value After Reset:** 0x0

WDT_COMP_PARAM_2

Name: Component Parameters Register 2

Description: Component Parameters Register 2

Note: This is a constant read-only register that contains encoded information about the component's parameter settings. The reset value depends on coreConsultant parameter(s).

Size: 32 bits

Offset: 0xf0

Bits	Name	Memory Access	Description
31:0	CP_WDT_CNT_RST	R	The value of this register is derived from the WDT_RST_CNT core Consultant parameter. **Value After Reset:** 0xffff

WDT_COMP_PARAM_1

Name: Component Parameters Register 1

Description: Component Parameters Register 1

Note:This is a constant read-only register that contains encoded information about the component's parameter settings. Some of the reset values depend on coreConsultant parameter(s).

Size: 32 bits

Offset: 0xf4

Bits	Name	Memory Access	Description
31:29	RSVD_31_29	R	WDT_COMP_PARAM_1[31:29]Reserved bits and read as zero (0). **Value After Reset:** 0x0
28:24	WDT_CNT_WIDTH	R	The Watchdog Timer counter width. **Value After Reset:** 0x10
23:20	WDT_DFLT_TOP_INIT	R	Describes the initial timeout period that is available directly after reset. It controls the reset value of the register. If WDT_HC_TOP is 1, then the default initial time period is the only possible period. **Value After Reset:** 0x0
19:16	WDT_DFLT_TOP	R	Selects the timeout period that is available directly after reset. It controls the reset value of the register. If WDT_HC_TOP is set to 1, then the default timeout period is the only possible timeout period. Can choose one of 16 values. **Value After Reset:** 0x0
15:13	RSVD_15_13	R	WDT_COMP_PARAM_1[15:13] Reserved bits and read as zero (0).

Bits	Name	Memory Access	Description
			Value After Reset: 0x0
12:10	WDT_DFLT_RPL	R	The reset pulse length that is available directly after reset. **Value After Reset:** 0x0
9:8	APB_DATA_WIDTH	R	Width of the APB Data Bus to which this component is attached. Values: 0x0 (APB_8BITS): APB data width is 8 bits 0x1 (APB_16BITS): APB data width is 16 bits 0x2 (APB_32BITS): APB data width is 32 bits **Value After Reset:** 0x2
7	WDT_PAUSE	R	Configures the peripheral to have a pause enable signal (pause) on the interface that can be used to freeze the watchdog counter during pause mode. Values: 0x0 (DISABLED): Pause enable signal is non existent 0x1 (ENABLED): Pause enable signal is included **Value After Reset:** 0x0
6	WDT_USE_FIX_TOP	R	When this parameter is set to 1, timeout period range is fixed. The range increments by the power of 2 from 2^16 to 2^(WDT_CNT_WIDTH-1). When this parameter is set to 0, the user must define the timeout period range (2^8 to 2^(WDT_CNT_WIDTH)-1) using the WDT_USER_TOP_(i) parameter. Values: 0x0 (USERDEFINED): User must define timeout values 0x1 (PREDEFINED): Use predefined timeout values **Value After Reset:** 0x1
5	WDT_HC_TOP	R	When set to 1, the selected timeout period(s) is set to be hard coded. Values: 0x0 (PROGRAMMABLE): Timeout period is programmable 0x1 (HARDCODED): Timeout period is hard coded **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
4	WDT_HC_RPL	R	Configures the reset pulse length to be hard coded. Values: 0x0 (PROGRAMMABLE): Reset pulse length is programmable 0x1 (HARDCODED): Reset pulse length is hardcoded **Value After Reset:** 0x0
3	WDT_HC_RMOD	R	Configures the output response mode to be hard coded. Values: 0x0 (PROGRAMMABLE): Output response mode is programmable 0x1 (HARDCODED): Output response mode is hard coded **Value After Reset:** 0x0
2	WDT_DUAL_TOP	R	When set to 1, includes a second timeout period that is used for initialization prior to the first kick. Values: 0x0 (DISABLED): Second timeout period is not present 0x1 (ENABLED): Second timeout period is present **Value After Reset:** 0x0
1	WDT_DFLT_RMOD	R	Describes the output response mode that is available directly after reset. Indicates the output response the WDT gives if a zero count is reached; that is, a system reset if equals 0 and an interrupt followed by a system reset, if equals 1. If WDT_HC_RMOD is 1, then default response mode is the only possible output response mode. Values: 0x0 (DISABLED): System reset only 0x1 (ENABLED): Interrupt and system reset **Value After Reset:** 0x1
0	WDT_ALWAYS_EN	R	Configures the WDT to be enabled from reset. If this setting is 1, the WDT is always enabled and a write to the WDT_EN field (bit 0) of the Watchdog Timer Control Register (WDT_CR) to disable it has no effect. Values: 0x0 (DISABLED): Watchdog timer disabled on reset 0x1 (ENABLED): Watchdog timer enabled on reset

Bits	Name	Memory Access	Description
			Value After Reset: 0x0

WDT_COMP_VERSION

Name: Component Version Register

Description: Component Version Register

Size: 32 bits

Offset: 0xf8

Bits	Name	Memory Access	Description
31:0	WDT_COMP_VERSION	R	ASCII value for each number in the version, followed by *. For example, 32_30_31_2A represents the version 2.01*. **Value After Reset:** 0x3131322a

WDT_COMP_TYPE

Name: Component Type Register

Description: Component Type Register

Size: 32 bits

Offset: 0xfc

Bits	Name	Memory Access	Description
31:0	WDT_COMP_TYPE	R	Designware Component Type number = 0x44_57_01_20. This assigned unique hex value is constant, and is derived from the two ASCII letters "DW" followed by a 16-bit unsigned number. **Value After Reset:** 0x44570120

3.11.5 Programming example

configured with the following configuration parameters:

- Single timeout period
- Response mode not hard coded
- Reset pulse length not hard coded
- WDT not always enabled
- Generates interrupt, and then system reset

3.12 RTC

1.1.1 Overview

The counter increments on successive positive edges of the input counter clock, rtc_clk. When the Counter Load Register (RTC_CLR) is programmed, the counter is loaded with a start value that allows the counter to increment.

3.12.1 block diagram

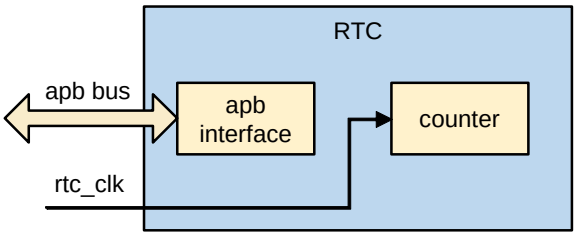


Figure 3-40 RTC block diagram

3.12.2 Function description

3.12.2.1 RTC Pre-scaler Counter

RTC supports preset counters, with the ability to update RTC counters at a rate lower than the RTC clock (rtc_clk). You can also enable/disable the RTC pre-scaler counter through the software using the rtc_psclr_en bit of the RTC_CCR register.

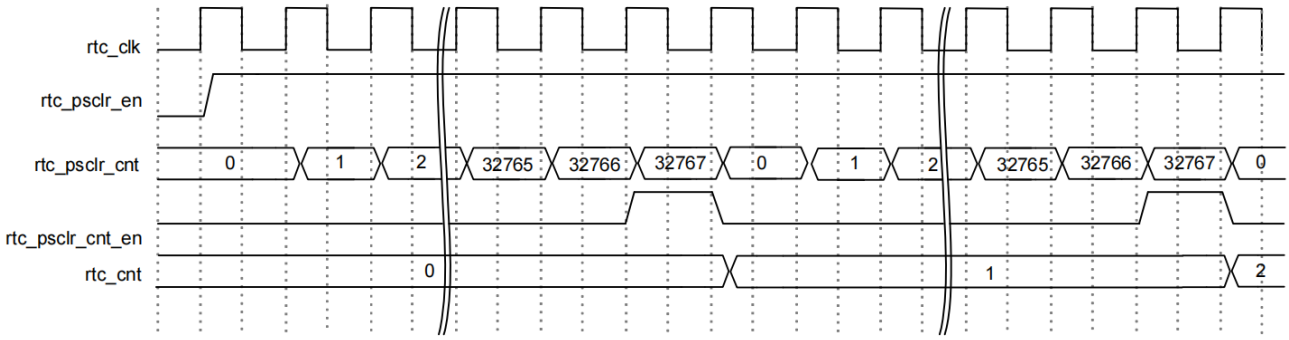


Figure 3-41 Timing Diagram for RTC Pre-scaler Counter with the RTC Counter

3.12.2.2 Match Register and Interrupt Generation

A match register, RTC_CMR, can be programmed and is compared to the internal counter. An interrupt is generated when the match register and the internal counter are equal, but only if interrupt generation is enabled (RTC_CCR, bit 0). This interrupt can be masked if the RTC_CCR register bit 1 is set to 1, which gives the user control of sending interrupts externally. The match register can be read any time.

The interrupt is kept active until it is cleared by an end-of-interrupt clear read access (RTC_EOI). The interrupt status bit is not polarity sensitive. The interrupt is active when the status is read as 1; otherwise, it is inactive. Programming the interrupt mask bit (rtc_mask) within the control register (RTC_CCR) masks the interrupt. The interrupt status can be read at any time. Even when the interrupt is masked, the raw interrupt status can be read.

3.12.3 Register description

The base address please refer to table 12-1.

3.12.3.1 Register overview

Register	Offset	Description
RTC_CCVR	0x0	Current Counter Value Register
RTC_CMR	0x4	Counter Match Register
RTC_CLR	0x8	Counter Load Register
RTC_CCR	0xc	Counter Control Register
RTC_STAT	0x10	Interrupt Status Register
RTC_RSTAT	0x14	Interrupt Raw Status Register
RTC_EOI	0x18	End of Interrupt Register
RTC_COMP_VERSION	0x1c	Component Version Register
RTC_CPSR	0x20	Counter PreScaler Register
RTC_CPCVR	0x24	Current Prescaler Counter Value Register

3.12.3.2 Register detail description

RTC_CCVR

Name: Current Counter Value Register

Description: Current Counter Value Register

Size: 32 bits

Offset: 0x0

Bits	Name	Memory Access	Description
31:y	RSVD_CCVR	R	RTC_CCVR 31 to RTC_CNT_WIDTH（RTC_CNT_WIDTH=32） Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true Range Variable[y]: RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）
x:0	Current_Counter_Value	R	When read, this register is the current value of the internal counter. This value is always read coherently. Bits from RTC_CNT_WIDTH（RTC_CNT_WIDTH=32） to 31 are read as 0 when RTC_CNT_WIDTH（RTC_CNT_WIDTH=32） is less than 31. **Value After Reset:** 0x0 **Volatile:** true **Range Variable[x]**: RTC_CNT_WIDTH（RTC_CNT_WIDTH=32） - 1

RTC_CMR

Name: Counter Match Register

Description: Counter Match Register

Size: 32 bits

Offset: 0x4

Bits	Name	Memory Access	Description
31:y	RSVD_CMR	R	RTC_CMR 31 to RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）Reserved bits - Read Only **Value After Reset:** 0x0 **Range Variable[y]:** RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）
x:0	Counter_Match	R/W	Interrupt Match Register. When the internal counter matches this register, an interrupt is generated, provided interrupt generation is enabled. When appropriate, this value is written coherently. Only when all the bytes are written is the register used by the interrupt detection logic. Bits from RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）and above are read and written as 0 when RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）is less than 31. **Value After Reset:** 0x0 **Range Variable[x]:** RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）- 1

RTC_CLR

Name: Counter Load Register

Description: Counter Load Register

Size: 32 bits

Offset: 0x8

Bits	Name	Memory Access	Description
31:y	RSVD_CLR	R	RTC_CLR 31toRTC_CNT_WIDTH（RTC_CNT_WIDTH=32）Reserved bits - Read Only **Value After Reset:** 0x0 **Range Variable[y]:** RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）

Bits	Name	Memory Access	Description
x:0	Counter_Load	R/W	Loaded into the counter as the loaded value, which is written coherently. Bits from RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）and above are read and written as 0 when RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）is less than 31. **Value After Reset:** 0x0 **Range Variable[x]:** RTC_CNT_WIDTH（RTC_CNT_WIDTH=32）- 1

RTC_CCR

Name: Counter Control Register

Description: Counter Control Register.

Note: If the RTC_RSTAT register indicates that a pending interrupt exists because a masked interrupt has been generated (rtc_mask bit of RTC_CCR is set to 1), then caution should be taken when programming RTC_CCR. That is, if both the rtc_mask and rtc_en bits of the RTC_CCR register are set to 0, then the interrupt asserts for one clock period. To avoid this scenario, you can use the following two-step process:

1. Program the rtc_en bit of the RTC_CCR register to 0, which clears the interrupt.
2. Then program the RTC_CCR register again to set the rtc_mask bit to 0, which unmasks any future interrupts.

Size: 32 bits

Offset: 0xc

Bits	Name	Memory Access	Description
31:8	RSVD_CCR	R	RTC_CCR 31to8 Reserved and read as 0. **Value After Reset:** 0x0
7:5	rtc_prot_level	* Varies	This field holds the protection level value of rtc. **Value After Reset:** PROT_LEVEL_RST **Exists:** (SLVERR_RESP_EN==1) **Memory Access:** {(HC_PROT_LEVEL==0) ? "read-write" : "read-only"}

Bits	Name	Memory Access	Description
4	rtc_psclr_en	* Varies	Optional. Allows user to control the usage of RTC Prescaler feature. **Values:** ■ 0x0 (DISABLED): Disables the Prescaler counter ■ 0x1 (ENABLED): Enables the Prescaler counter **Value After Reset:** 0x0 **Memory Access:** "((RTC_PRESCLR_EN==1) && (RTC_EN_MODE==1)) ? \"read-write\" : \"read-only\""
3	rtc_wen	* Varies	*Optional*. Allows the user to force the counter to wrap when a match occurs instead of waiting until the maximum count is reached. 0 = Wrap disabled, 1 = Wrap enabled, This bit is writable only when RTC_WRAP_MODE = 1 **Values:** ■ 0x0 (DISABLED): Disables the WRAP ■ 0x1 (ENABLED): Enables the WRAP **Value After Reset:** 0x0 **Memory Access:** "(RTC_WRAP_MODE==1) ? \"read-write\" : \"read-only\""
2	rtc_en	* Varies	Optional. Allows the user to control counting in the counter. This bit does not exist if RTC_EN_MODE = 0. Internally, the counter is always enabled. **Values:** ■ 0x0 (DISABLED): Disables the counter ■ 0x1 (ENABLED): Enables the counter **Value After Reset:** 0x0 **Memory Access:** "(RTC_EN_MODE==1) ? \"read-write\" : \"read-only\""
1	rtc_mask	R/W	Allows the user to mask interrupt generation. **Values:** ■ 0x0 (UNMASKED): Interrupt unmasked■ 0x1 (MASKED): Interrupt masked **Value After Reset:** 0x0

Bits	Name	Memory Access	Description
0	rtc_ien	R/W	Allows the user to disable interrupt generation. **Values:** ■ 0x0 (DISABLED): Disables the interrupt generation ■ 0x1 (ENABLED): Enables the interrupt generation **Value After Reset:** 0x0

RTC_STAT

Name: Interrupt Status Register

Description: Interrupt Status Register

Size: 32 bits

Offset: 0x10

Bits	Name	Memory Access	Description
31:1	RSVD_RTC_STAT	R	RTC_STAT 31to1 Reserved and read as 0. **Value After Reset:** 0x0 **Volatile:** true
0	rtc_stat	R	This register is the masked raw status. **Values:** ■ 0x0 (INACTIVE): Interrupt is inactive ■ 0x1 (ACTIVE): Interrupt is active (regardless of polarity) **Value After Reset:** 0x0 **Volatile:** true

RTC_RSTAT

Name: Interrupt Raw Status Register

Description: Interrupt Raw Status Register

Size: 32 bits

Offset: 0x14

Bits	Name	Memory Access	Description
31:1	RSVD_RTC_RSTAT	R	RTC_RSTAT 31to1 Reserved and read as 0. **Value After Reset:** 0x0 **Volatile:** true
0	rtc_rstat	R	Raw Status **Values:** ■ 0x0 (INACTIVE): Interrupt is inactive ■ 0x1 (ACTIVE): Interrupt is active (regardless of polarity) **Value After Reset:** 0x0 **Volatile:** true

RTC_EOI

Name: End of Interrupt Register

Description: End of Interrupt Register

Size: 32 bits

Offset: 0x18

Bits	Name	Memory Access	Description
31:1	RSVD_RTC_EOI	R	RTC_EOI 31to1 Reserved and read as 0. **Value After Reset:** 0x0
0	rtc_eoi	R	By reading this location, the match interrupt is cleared. Performing read-to-clear on interrupt, the interrupt is cleared at the end of the read. **Value After Reset:** 0x0

RTC_COMP_VERSION

Name: Component Version Register

Description: Component Version Register

Size: 32 bits

Offset: 0x1c

Bits	Name	Memory Access	Description
31:0	rtc_comp_version	R	ASCII value for each number in the version, followed by *. For example, 32_30_31_2A represents the version 2.01*. **Value After Reset:** 0x3230_382A

RTC_CPSR

Name: Counter PreScaler Register

Description: Counter PreScaler Register

Size: 32 bits

Offset: 0x20

Exists: RTC_PRESCLR_EN == 1

Bits	Name	Memory Access	Description
31:y	RSVD_CPSR	R	RTC_CPSR 31toRTC_PRESCLR_WIDTH Reserved bits - Read Only **Value After Reset:** 0x0 **Range Variable[y]:** RTC_PRESCLR_WIDTH
x:0	Counter_Prescaler_Value	R/W	Counter Prescaler Register. The RTC counter will be updating at the rate of rtc_clk, rtc_clk_en, or pclk based on the configuration. This register is used to prescale the rate at which the RTC counter updates. When appropriate, this register is written coherently. Only when all the bytes are written, the register used by the prescaler counter logic. **Value After Reset:** RTC_PRESCLR_VAL **Range Variable[x]:** RTC_PRESCLR_WIDTH(RTC_PRESCLR_WIDTH =16) - 1

RTC_CPCVR

Name: Current Prescaler Counter Value Register

Description: Current Prescaler Counter Value Register

Size: 32 bits

Offset: 0x24

Bits	Name	Memory Access	Description
31:y	RSVD_CPCVR	R	RTC_CPCVR 31 to RTC_PRESCLR_WIDTH Reserved bits - Read Only **Value After Reset:** 0x0 **Volatile:** true **Range Variable[y]:** RTC_PRESCLR_WIDTH
x:0	Current_Prescaler_Counter_Value	R	When read, this register provides the current value of the internal prescaler counter. This value always is read coherently. Bits from RTC_PRESCLR_WIDTH to 31 are read as 0 when RTC_PRESCLR_WIDTH is less than 31. **Value After Reset:** 0x0 **Volatile:** true **Range Variable[x]:** (RTC_PRESCLR_WIDTH =16) - 1

3.12.4 Programming example

Make sure that the RTC clock reset is correctly configured, and then configure the RTC:

- Configure the initial count value，Counter_Load
- Configure Counter Match Register，Counter_Match
- Configure Pre-scaler Counter，RTC_CPSR
- Enable RTC，rtc_en

3.13 PVT

3.13.1 Overview

PVT sensors support the following features:

- Operating voltage 0.8V (digital), 1.8V(analog)；
- Operational temperature -40℃~125℃
- Temperature accuracy ±1℃（post trimming），±6℃（untrimmed）
- Voltage accuracy（±3 sigma）±3%
- Sample rate 3000 samples/sec

3.13.2 Register description

The base address please refer to table 12-1.

Reg Name	Offset	Field name	Bit	Access	Description
REG_PVT_TRIM	0x04	cfg_trimg	[4:0]	R/W	pvt control signal, gain

Reg Name	Offset	Field name	Bit	Access	Description
		Reserved	[7:5]	RO	Value: 0
		cfg_trimo	[13:8]	R/W	pvt control signal, offset
		Reserved	[31:14]	RO	Value: 0
REG_PVT_MODE	0x08	cfg_mode	[2:0]	R/W	pvt mode voltage mode: cfg_mode = 3'b100 temperature mode: cfg_mode = 3'b000
		Reserved	[31:3]	RO	Value: 0
REG_PVT_ENA	0x0c	pvt_enable	[0]	R/W	pvt enable
		Reserved	[31:1]	RO	Value: 0
REG_PVT_INT	0x10	pvt_core_intr	[0]	RO	pvt interrupt, When the PVT output data is valid, this register is 1'b1
		pvt_int_clr	[1]	WO	clear interrupt and pvt_enable
REG_PVT_DATA	0x14	data_out_pclk	[9:0]	RO	read data

3.13.3 Programming example

The software operation process is as follows:

- Confirm that the PVT clock is correct
- Configure REG_PVT_MODE to select the measurement mode
- Configure REG_PVT_ENA = 32'h1 to enable PVT
- Wait for the interrupt signal to be valid and read the REG_PVT_DATA
- Configure REG_PVT_INT = 32'h2 to clear interrupts

3.14 Low Power Scheme

3.14.1 Voltage Domain and Power Domain

3.14.1.1 Definition

The voltage domain defined as separate voltage supply from bump(external PMIC or component), the voltage are configured by external PMIC or component.

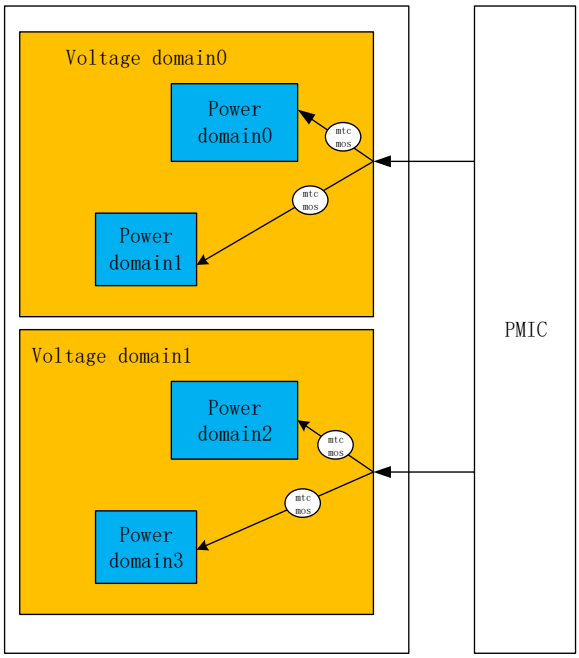


Figure 3-42 Voltage Domain and Power Domain

The power domain defined as independently controlled power on/off domain, normally it is inside one voltage domain. Power domain on/off normally controlled by internal power switch(MTCMOS).

3.14.2 Voltage Domain

Plan of the voltage domains as bellow.

- NPU has a separate 0.8v VDD_NPU. Supports step-up to 1.0v to meet 1Ghz performance.
- CPU has a separate 0.8v VDD_CPU. Supports step-down to 0.75v.
- DDR has a separate 0.8v VDD_LPDDR. Supports step-up to 0.85v.
- Other digital logic share one 0.8v VDD_SOC.
- VDD_CPU, VDD_NPU and VDD_LPDDR's voltage can be changed outside DIE by PMIC(DC-DC).

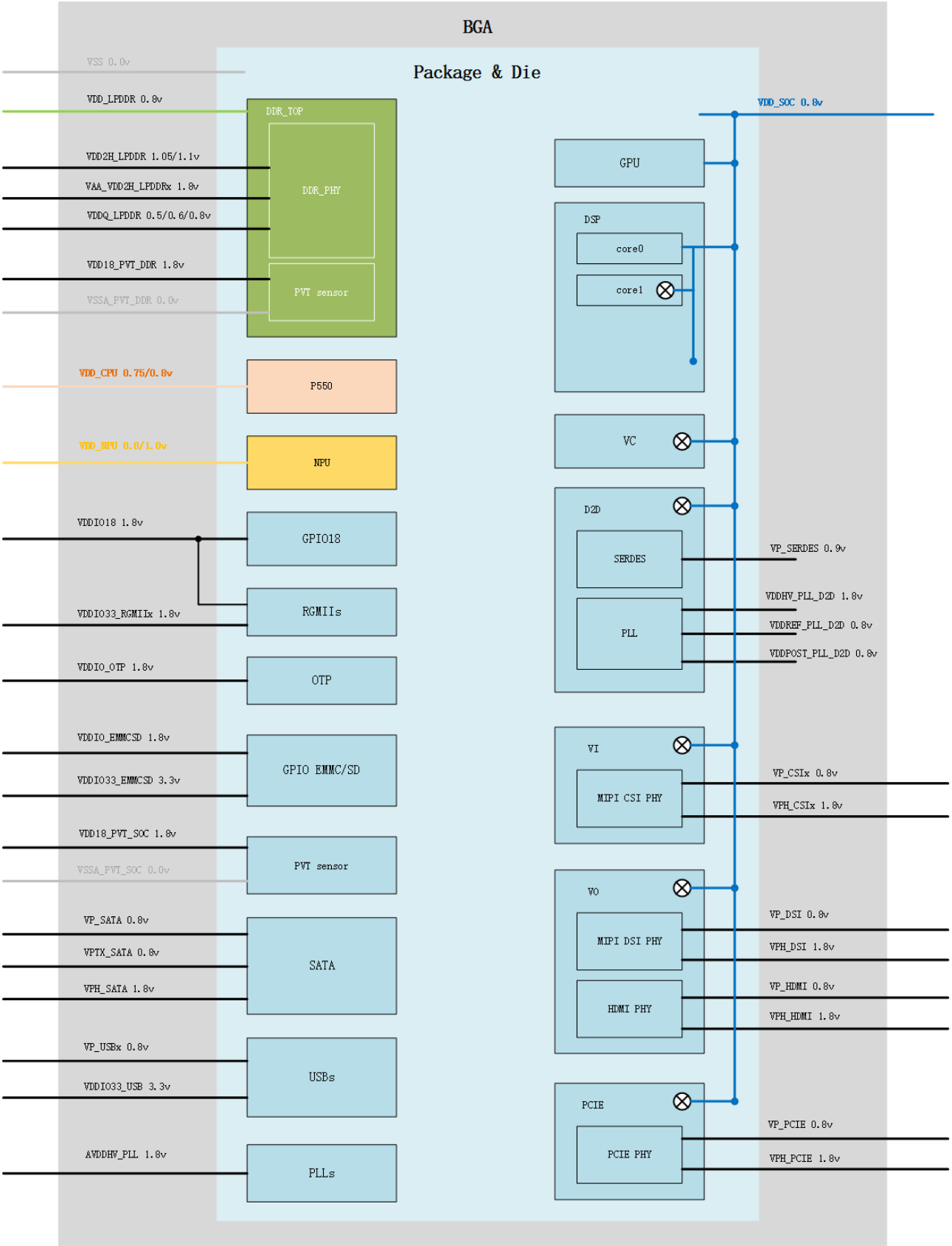


Figure 3-43 voltage domain and main power supplies

Please note that some power supplies are merged within the DIE, and some power supplies are merged in the package or board.

Combination of voltages for all supplies are shown as table below.

Table 3-235 combination of voltages

No.	VDD_SOC(V)	VDD_LPDDR	VDD_CPU(V)	VDD_NPU(V)	sign-off
1	0.8	0.8	0.8	0.8	Y
2	0.8	0.8	0.8	1.0	Y
3	0.8	0.8	0.75	0.8	Y
4	0.8	0.8	0.75	1.0	Y

3.14.3 Power Domain

The Power domain are defined mainly for VDD_NPU, VDD_CPU, VDD_LPDDR and VDD_SOC voltage domain. For other voltage domain, they are not power off, or can be partially power off by it's own control logic(such as PLL.etc)

3.14.3.1 Digital part power domain overview

As shown in the figure below, Isolation Cells, Levelshift Cells, and Enable-levelshift Cells are inserted between different power domains. VDD_SOC, VDD_CPU, and VDD_LPDDR are normally powered on and do not lose power during chip operation. The VDD_NPU power supply can be powered off by controlling the external power supply IC.

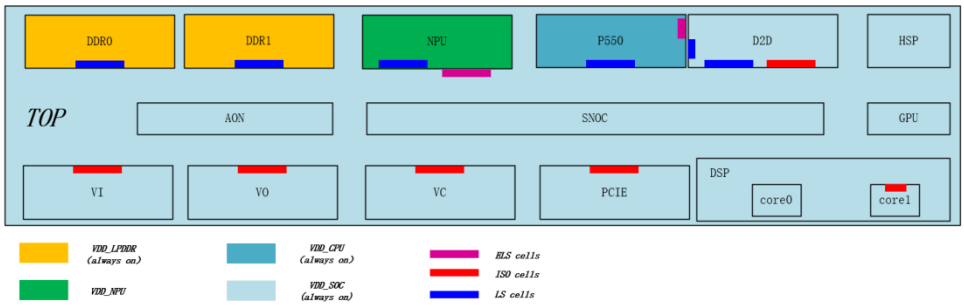


Figure 3-44 power domain and isolation/levelshift cells

3.14.4 Domain description

3.14.4.1 TOP only

• Top_only has 1 main digital power supplys: VDD_SOC, with other dedicated VDDs for PLLs, OPTs, USBPHY and so on.

• TOP_only has 1 power domains: VDD_SOC_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_SOC,which has 1 voltage 0.8V.

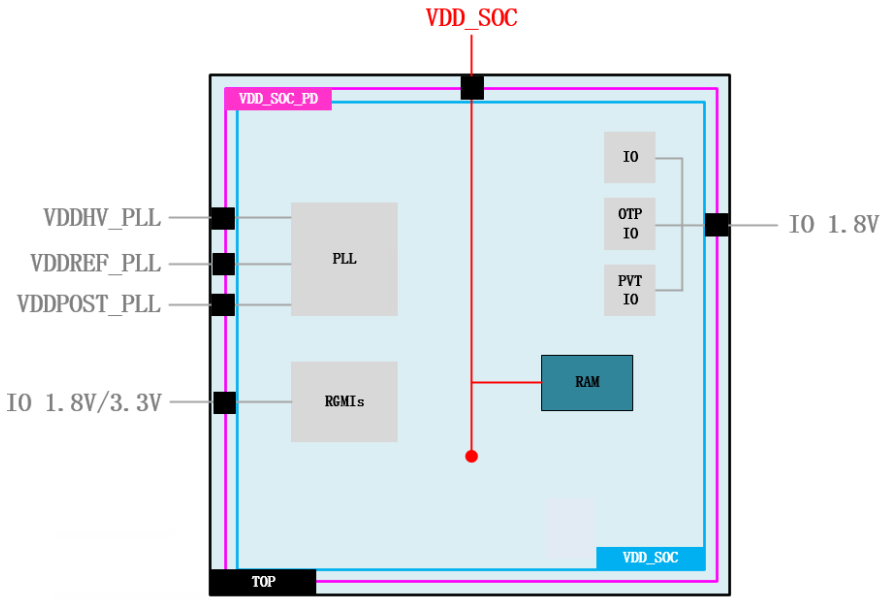


Figure 3-45 TOP only power domain

Note：Black box describes the RTL hierarchy, Purple box describes the power domain scope, Blue box describes the voltage domain.

Table 3-236 TOP only power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain

3.14.4.2 AON

• AON has 1 main digital power supplys: VDD_SOC.

• AON has 1 power domains: VDD_SOC_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_SOC,which has 1 voltage 0.8V.

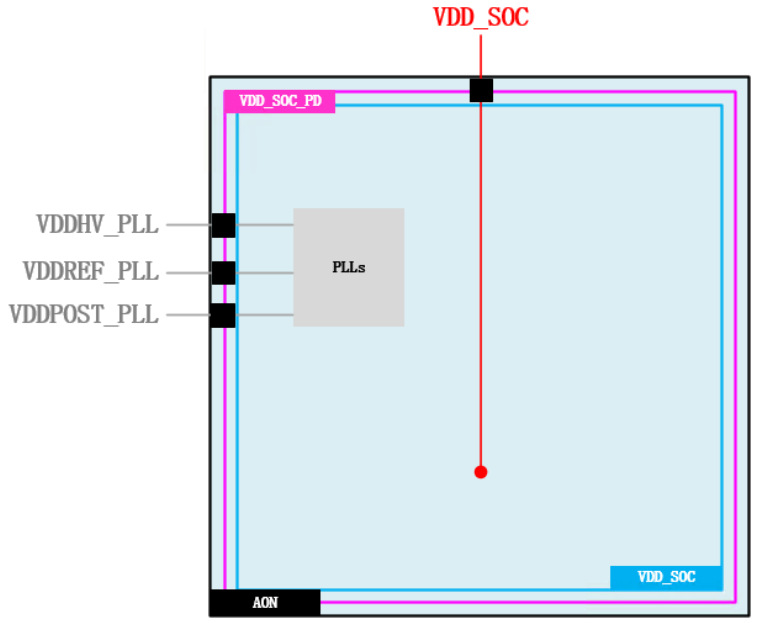


Figure 3-46 AON power domain

Table 3-237 AON only power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain

3.14.4.3 SNOC

• SNOC has 1 main digital power supplys: VDD_SOC.

• SNOC has 1 power domains: VDD_SOC_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_SOC,which has 1 voltage 0.8V.

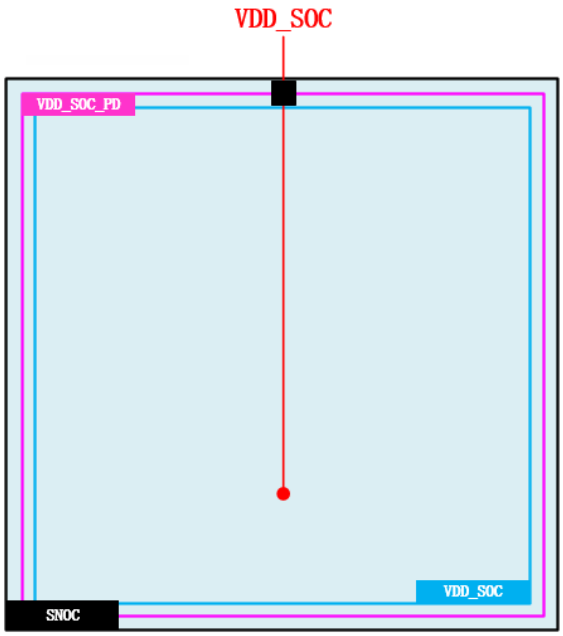


Figure 3-47 SNOC only power domain

Table 3-238 SNOC only power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain

3.14.4.4 GPU

• GPU has 1 main digital power supplys: VDD_SOC.

• GPU has 1 power domains: VDD_SOC_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_SOC,which has 1 voltage 0.8V.

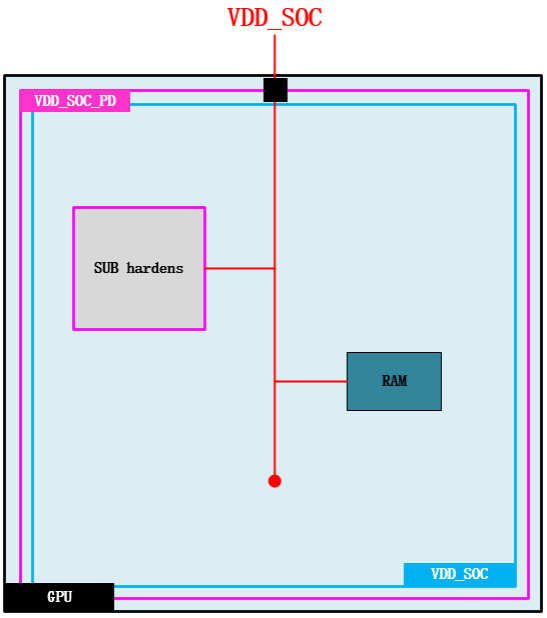


Figure 3-48 GPU power domain

Table 3-239 GPU sys power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain

3.14.4.5 HSP

• HSP has 1 main digital power supplys: VDD_SOC.

• HSP has 1 power domains: VDD_SOC_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_SOC,which has 1 voltage 0.8V.

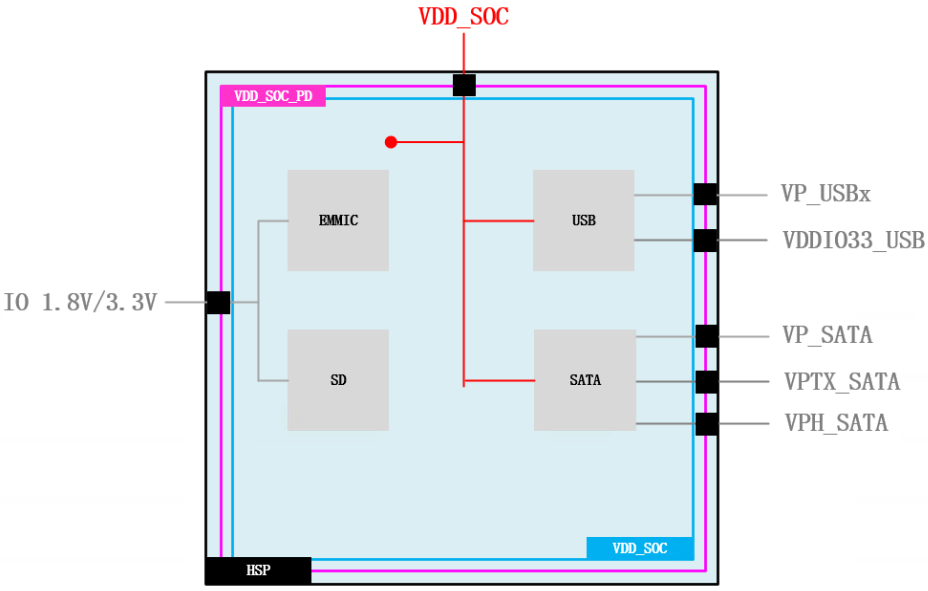


Figure 3-49 HSP power domain

Table 3-240 HSP sys power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain

3.14.4.6 NPU

• NPU has 1 main digital power supplys: VDD_NPU.

• NPU has 1 power domains: VDD_NPU_PD.

• VDD_NPU_PD can be turned off, its primary supply is VDD_NPU, which is switchable by PMIC(DC-DC), VDD_NPU has voltage 0.8V~1.0V.

• As TOP is normally always-on compare to VDD_NPU_PD, Enabled-levelshift cells are needed for NPU to TOP outputs, and all these ELS cells are located parent VDD_NPU_PD. levelshift cells are needed for TOP to NPU inputs, and all these LS cells are located inside VDD_NPU_PD.

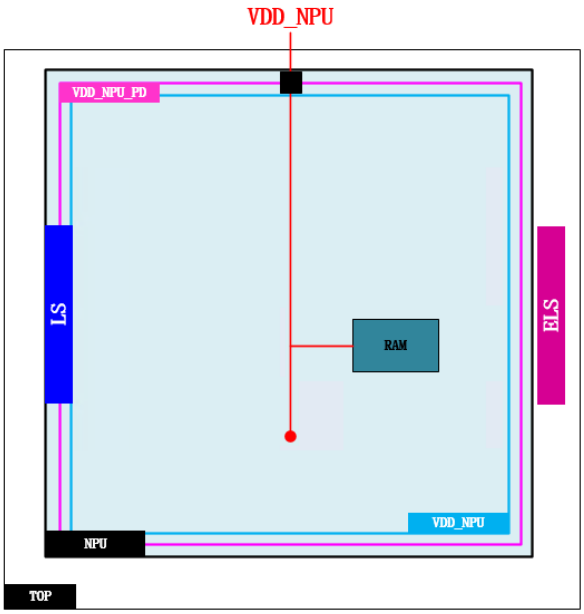


Figure 3-50 NPU power domain

Table 3-241 NPU sys power domain description

Voltage domain	power domain	Notes
VDD_NPU	VDD_NPU_PD	switchable by PMIC(DC-DC)

3.14.4.7 LPDDR0

• DDR0 has 1 main digital power supplys: VDD_LPDDR.

• DDR0 has 1 power domains: VDD_LPDDR_PD.

• VDD_LPDDR_PD is always-on, its primary supply is VDD_LPDDR,which has voltage 0.8V~0.85V.

• As TOP is normally always-on and at different voltages compare to VDD_LPDDR_PD, Levelshift cells are needed for DDR0-TOP inputs&outputs, and all these LS cells are located inside VDD_DDR_PD.

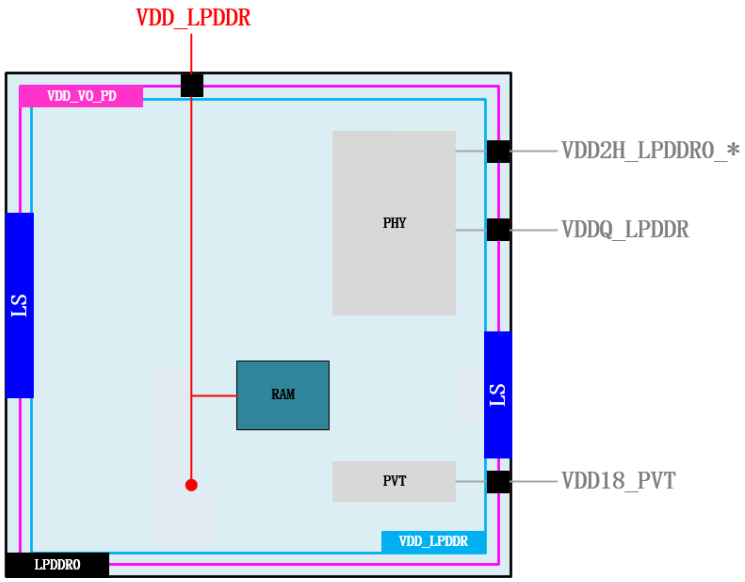


Figure 3-51 LPDDR0 power domain

Table 3-242 LPDDR0 sys power domain description

Voltage domain	power domain	Notes
VDD_LPDDR	VDD_LPDDR_PD	Default Always-on domain

3.14.4.8 LPDDR1

• LPDDR1 has 1 main digital power supplys: VDD_LPDDR.

• LPDDR1 has 1 power domains: VDD_LPDDR_PD.

• VDD_SOC_PD is always-on, its primary supply is VDD_LPDDR,which has voltage 0.8V~0.85V.

• As TOP is normally always-on and at different voltages compare to VDD_LPDDR_PD, Levelshift cells are needed for DDR1-TOP inputs&outputs, and all these LS cells are located inside VDD_DDR_PD.

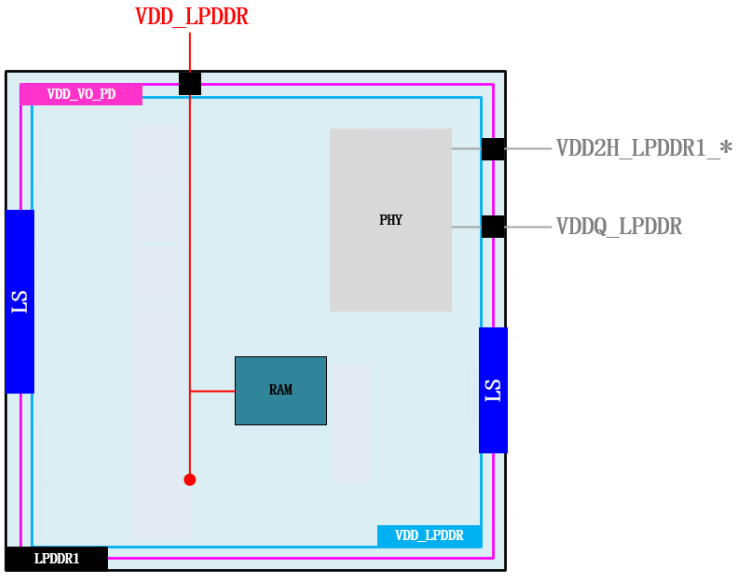


Figure 3-52 LPDDR1 power domain

Table 3-243 LPDDR1 sys power domain description

Voltage domain	power domain	Notes
VDD_LPDDR	VDD_LPDDR_PD	Default Always-on domain

3.14.4.9 MCPU

• MCPU has 1 main digital power supplys: VDD_CPU.

• MCPU has 1 power domains: VDD_CPU_PD.

• VDD_CPU_PD is always-on, its primary supply is VDD_CPU,which has voltage 0.75v~0.8V.

• As TOP is normally always-on and at different voltages compare to VDD_CPU_PD, Levelshift cells are needed for MCPU<->TOP ports, and all these LS cells are located inside VDD_CPU_PD.

• As MCPU is normally always-on and at different voltages compare to VDD_D2D_PD.Enabled-Levelshift cells are needed for D2D to MCPU outputs, and all these ELS cells are located inside VDD_CPU_PD. Levelshift cells are needed for MCPU to D2D inputs, and all these LS cells are located inside VDD_D2D_PD.

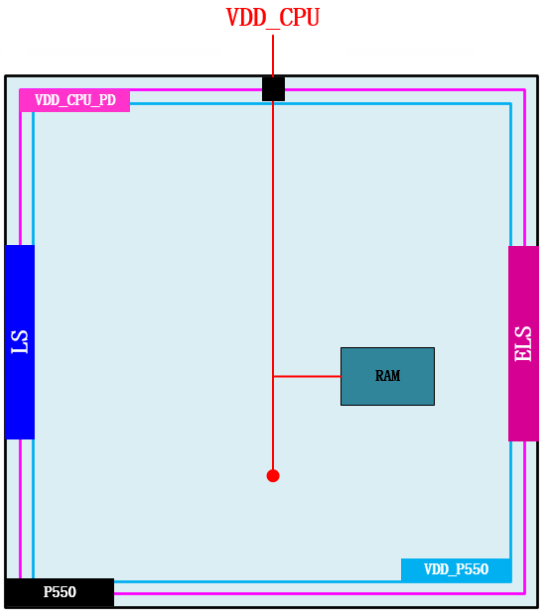


Figure 3-53 MCPU power domain

Table 3-244 MCPU sys power domain description

Voltage domain	power domain	Notes
VDD_CPU	VDD_CPU_PD	Default Always-on domain

3.14.4.10 D2D

• D2D has 1 main digital power supplys: VDD_SOC.

• D2D has 1 power domains: VDD_D2D_PD.

• VDD_D2D_PD can be turned off, its primary supply is VDD_D2D, which is switchable. VDD_D2D's voltage is 0.8V.

• As TOP is normally always-on compare to VDD_D2D_PD, Isolation cells are needed for D2D to TOP outputs, and all these Isolation cells are located inside VDD_D2D_PD.

• As MCPU is normally always-on and at different voltages compare to VDD_D2D_PD.Enabled-Levelshift cells are needed for D2D to MCPU outputs, and all these ELS cells are located inside VDD_CPU_PD. Levelshift cells are needed for MCPU to D2D inputs, and all these LS cells are located inside VDD_D2D_PD.

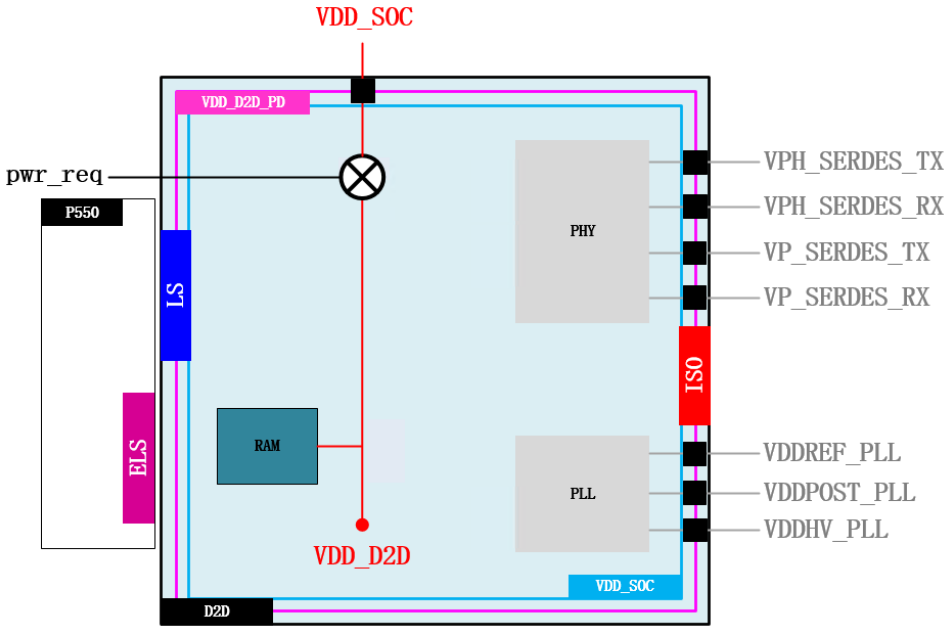


Figure 3-54 D2D power domain

Table 3-245 D2D sys power domain description

Voltage domain	power domain	Notes
VDD_CPU	VDD_D2D_PD	switchable

3.14.4.11 DSP

• DSP has 1 digital power supply: VDD_SOC.

• DSP has 2 power domain:

VDD_SOC_PD ,VDD_DSP_PS1_PD.

• VDD_SOC_PD's primary supply is VDD_SOC, which is always on.

• VDD_DSP_PS1_PD's primary supply is VDD_DSP_PS1, which is switchable.

• Isolation cells are needed for core1 to DSP_TOP outputs, and all these Isolation cells are located inside VDD_DSP_PS1_PD.

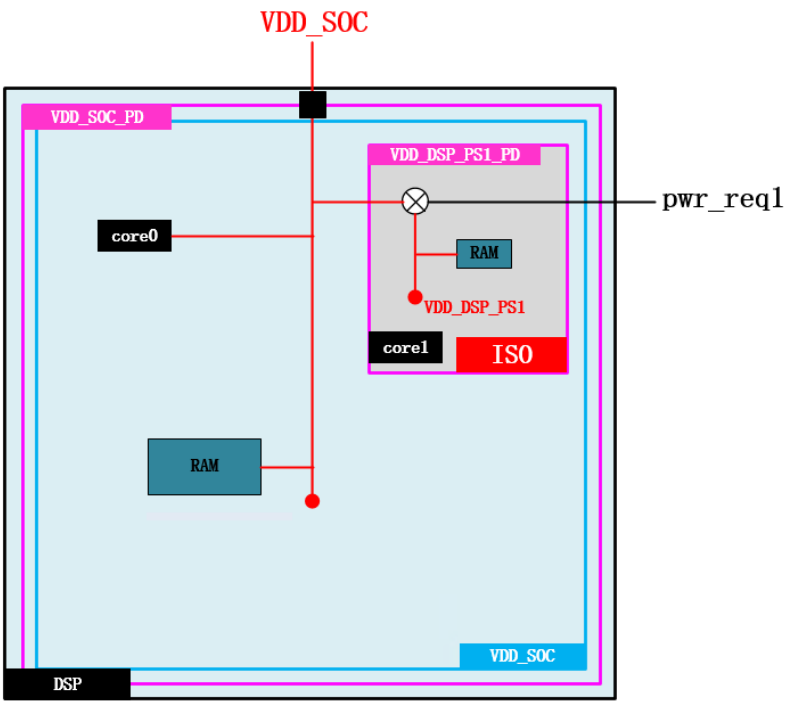


Figure 3-55 DSP power domain

Table 3-246 DSP power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain
	VDD_DSP_PS1_PD	switchable

Table 3-247 DSP power state description

power domain	Power ON/OFF (Y is ON;N is OFF)			
VDD_SOC_PD	Y	Y	Y	Y
VDD_DSP_PS1_PD	N	Y	Y	Y

3.14.4.12 VI

• VI has 1 digital power supply: VDD_SOC.

• VI has 3 power domain: VDD_SOC_PD ,VDD_VI_PD, VDD_MARVIN_PD .

• VDD_SOC_PD's primary supply is VDD_SOC, which is always on.

• VDD_VI_PD's primary supply is VDD_VI, which is switchable;

• VDD_MARVIN_PD's primary supply is VDD_MARVIN, which is switchable.

• Isolation cells are needed for MIPI_CSI inputs, and all these Isolation cells are located parent VDD_SOC_PD.

• As TOP is normally always-on compare to VDD_VI_PD, Isolation cells are needed for VI to TOP outputs, and all these Isolation cells are located inside VDD_VI_PD.

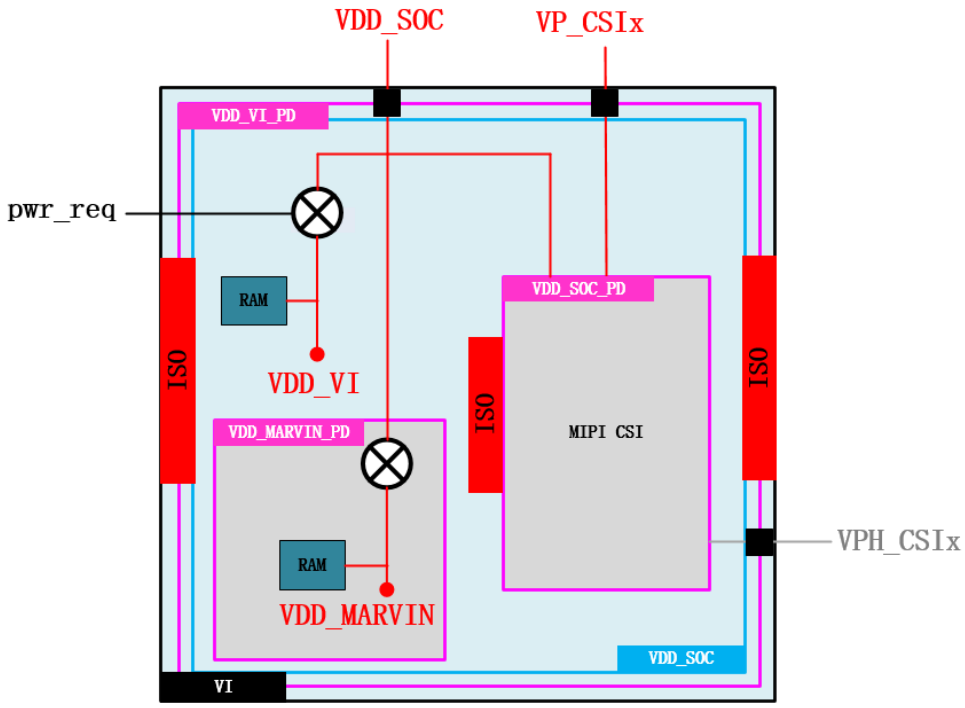


Figure 3-56 VI power domain

Table 3-248 VI power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain
	VDD_VI_PD	switchable
	VDD_MARVIN_PD	switchable

Table 3-249 VI power state description

power domain	Power ON/OFF (Y is ON;N is OFF)	
VDD_SOC_PD	Y	Y
VDD_VI_PD	N	Y
VDD_MARVIN_PD	N	Y

3.14.4.13 VO

- VO has 1 digital power supply: VDD_SOC.
- VO has 2 power domain:
- VDD_SOC_PD ,VDD_VO_PD.
- VDD_SOC_PD's primary supply is VDD_SOC, which is always on.
- VDD_VO_PD's primary supply is VDD_VO, which is switchable.
- Isolation cells are needed for MIPI_DSI/HDMI inputs, and all these Isolation cells are located parent VDD_SOC_PD.
- As TOP is normally always-on compare to VDD_VO_PD, Isolation cells are needed for VO to TOP outputs, and all these Isolation cells are located inside VDD_VO_PD.

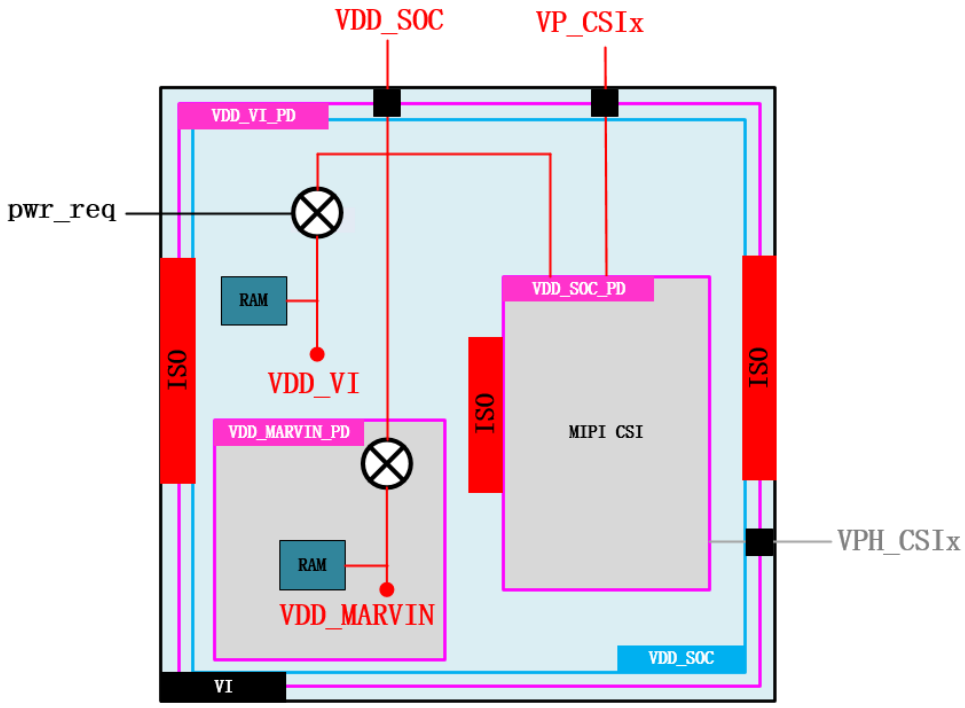


Figure 3-57 VOpower domain

Table 3-250 VO power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_SOC_PD	Default Always-on domain
	VDD_VO_PD	switchable

Table 3-251 VO power state description

power domain	Power ON/OFF (Y is ON;N is OFF)	
VDD_SOC_PD	Y	Y
VDD_VO_PD	N	Y

3.14.4.14 VC

- VC has 1 digital power supply: VDD_SOC.
- VC has 1 power domain: VDD_VC_PD .
- VDD_VC_PD's primary supply is VDD_VC, which is switchable.
- As TOP is normally always-on compare to VDD_VC_PD, Isolation cells are needed for VC to TOP outputs, and all these Isolation cells are located inside VDD_VC_PD.

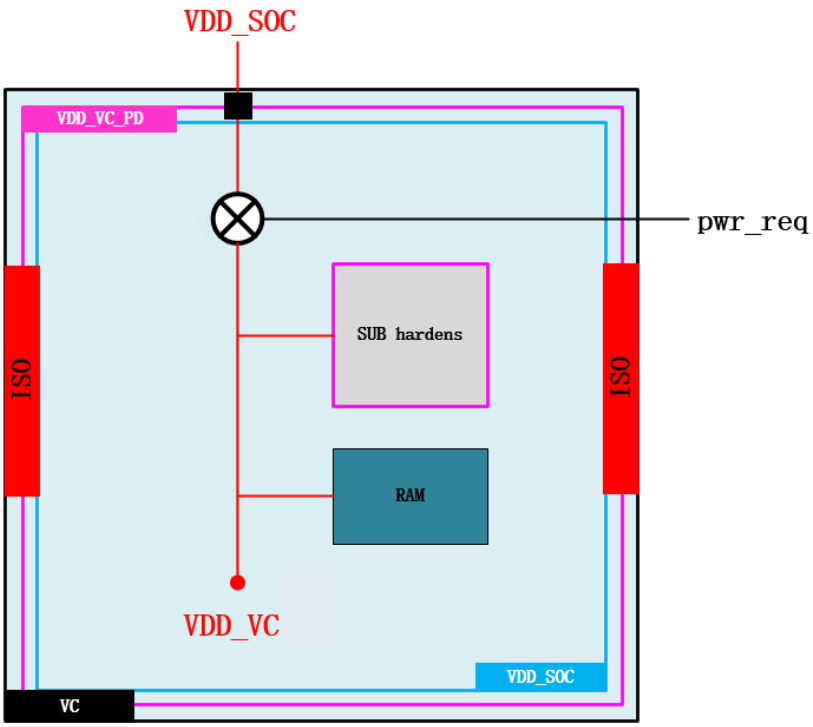


Figure 3-58 VC power domain

Table 3-252 VI power domain description

Voltage domain	power domain	Notes
VDD_SOC	VDD_VI_PD	switchable

3.14.4.15 PCIE

• PCIE has 2 digital power supply: VDD_SOC,VP_PCIE;(VDD_SOC and VP_PCIE is equivalent)

• PCIE has 2 power domain:VDD_PCIE_PD ,VDD_PMA_PD.

• VDD_PCIE_PD's primary supply is VDD_PCIE, which is switchable.

• VDD_PMA_PD's primary supply is VDD_PMA, which is switchable.

• As TOP is normally always-on compare to VDD_PCIE_PD, Isolation cells are needed for PCIE to TOP outputs, and all these Isolation cells are located inside VDD_PCIE_PD.

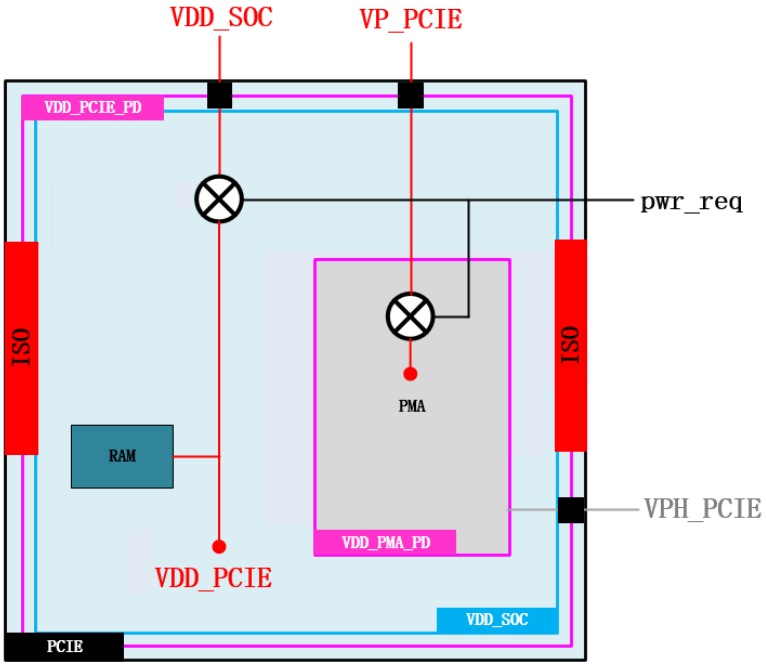


Figure 3-59 PCIE power domain

Table 3-253 PCIE power domain description

Voltage domain	power domain	Notes
VDD_SOC/VP_PCIE	VDD_PCIE_PD	switchable
	VDD_PMA_PD	switchable

Table 3-254 PCIE power state description

power domain	Power ON/OFF (Y is ON;N is OFF)	
VDD_PCIE_PD	Y	N
VDD_PMA_PD	Y	N

3.14.5 Low Power Mode

3.14.5.1 System Low Power Mode scenarioS

Beside each Power domain can be power off respectively, from system point of view, the whole chip has normal/deepsleep power mode.

Deep Sleep state: Defined as if system has already been in idle, then the entry process as:

software configure power gating to VI/VO/VC/DSP/PCIE.

3.14.6 Power Control Unit

3.14.6.1 Overview

Based on background knowledge and application requirements, the following functions need to be realized in the design of Power Control Unit:

- Power on and power off a single power domain.
- The system software can directly control the power-on and power-off sequence of the module.
- The hardware state machine controls the power-on and power-off sequence of the module.
- Supports the default power domain status. The default power domain status can be powered

on or powered off.

- The power domain switch supports the Request-Acknowledge handshake signal operation.
- The hardware state machine can control clock, reset, and clamp signals in any sequence during power-on and power-off.

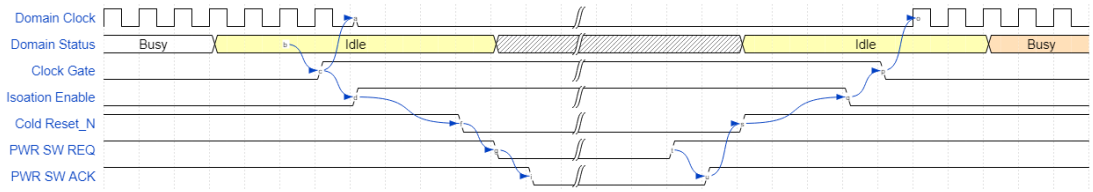


Figure 3-60 Low power Control signals timing diagram

3.14.6.2 Block Diagram

The PCU design architecture is shown in the figure below. In the figure, the black border is the main content of the module design, and the blue dotted line frame is the design supported by the module collocation.

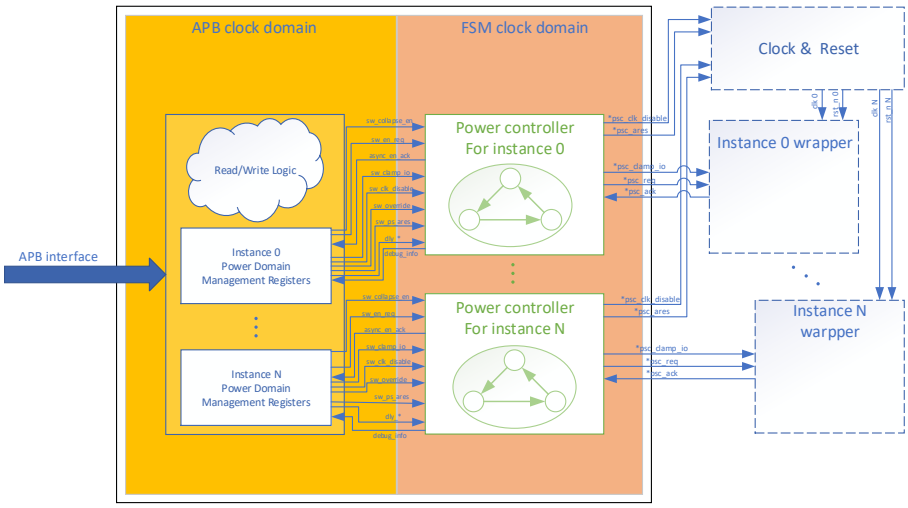


Figure 3-61 PCU Block Diagram

In order to operate the power domain control switch more reliably, the design supports two modes of software control mode and hardware control mode. When the software controls the state order, the software configure the *_GLB_CTRL register and set *sw_override to 1. Otherwise, the power domain is controlled based on the hardware state machine sequence.

The control signal psc_* in the power domain has two sources: one is the control signal of the software. For example, the signal starts with sw_*; The other is a hardware sequence controller, such as a signal starting with fsm_*. When signal sw_override is set to 0, the dual selector selects fsm_* as the control signal of the power domain. When sw_override is set to 1, the dual-selector selects sw_* as the control signal for the power domain. For the *_ack signal in response to the power domain, the decoder decides to feed it back to the software or hardware sequence controller, depending on the state of sw_override.

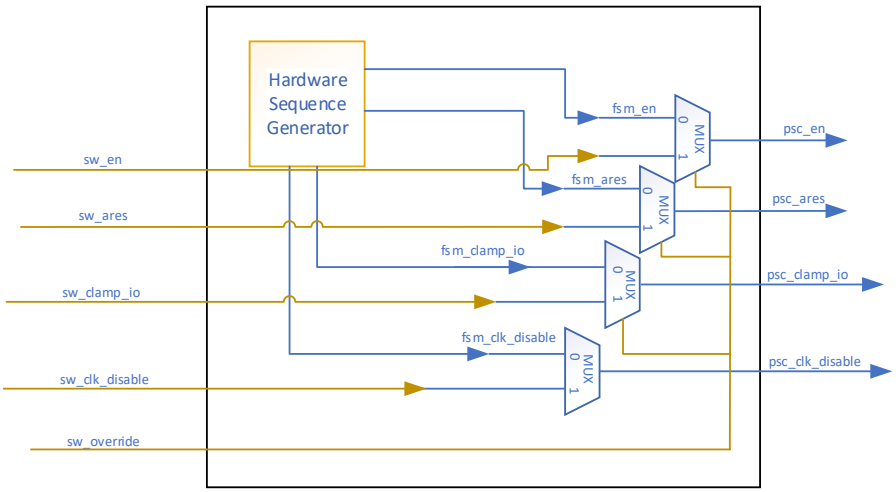


Figure 3-62 Dual power control mode switch

3.14.6.3 Software Control mode

- Software set * sw_override to 1.
- Software config *_sw_ps_en, *_sw_ares,*_sw_clamp_io, *_sw_clk_disable. The timing of these low-power control signals is guaranteed by software.

3.14.6.4 Hardware Control mode

Hardware Control mode is used by default.

When the state machine is triggered by the power-on or power-off trigger signal, it automatically generates handshake signals, counting and waiting operations, etc. and generates the required design time sequence one by one.

The designed state machine can be in the *Power On* state or in the *Power Off* state in the default state, and the different default states of the state machine correspond to the different states of the power domain control signal. The default state of the state machine is indicated by the parameter (*RST_STATE*). When the value is 0, the default state is *Power Off*. If the value is 1, the default status is *Power On*. In a certain state, once the conditions of collapse or failure collapse are triggered, the state machine automatically jumps, generates or releases relevant signals, and produces the required time sequence.

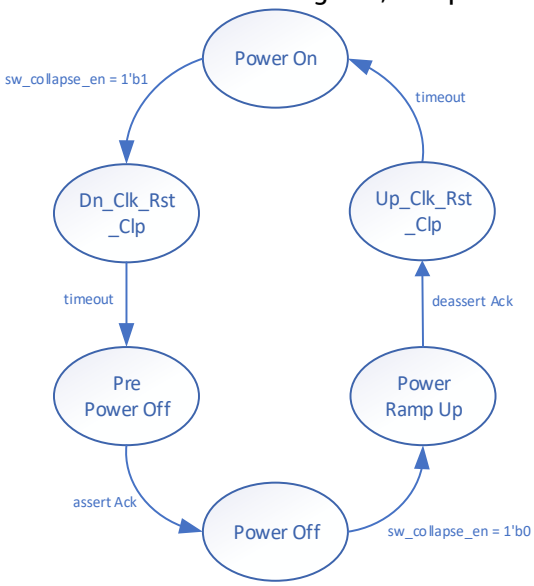


Figure 3-63 HardWare Control Mode State Machine

•In order to introduce the Power-off process, assume that the state machine is currently in the Power On state. The jump process of the state machine is as follows:

1.When the power-off process is triggered, the enable signal *collapses to 1, and the state machine immediately enters the *Dn_Clk_Rst_Clp* state from the *Power On* state. In this state, power domain modules are shut down, clamped across power domain boundaries, and reset. Considering the reset enable and propagation delay, the clamp signal propagation and enable delay, and the delay from declaring clock gating to clock signal stopping, the state machine introduces a count wait, which is passed by the registers *dly_assert_ares, *dly_clamp_io, and *dly_disable_clk, respectively. By configuring the delay register to different values, the shutdown domain clock operation, clamp operation, and reset operation sequence are controlled. After all three groups are counted, a timeout signal is generated.

2. After the counting of Power Off clock signals, reset signals, and clamping signals is complete, the state machine switches to *Pre Power Off*. In this state, the operation to turn off the power switch is initiated, and the state machine waits for the end of the count (*dly_en_down) to pull down the req request signal. After the power switch is turned off, the corresponding ack signal is pulled to a low level.

3. The feedback signal of step 2 is low Power level, and the state machine immediately jumps to the *Power Off* state, and the state machine completes the power off process. At this point, the clock in the power domain is not triggered, the corresponding reset signal is declared, the signal across the power boundary is clamped, and the request and response signals of the power switch are at a low level.

•To describe the power-on process, assume that the state machine is currently in the Power Off state. The jump process of the state machine is as follows:

1. After the power-on process is triggered, the state machine changes from *Power Off* to *Power Ramp Up* immediately. In this state, start the operation of turning on the power switch, that is, pull up the req request signal, and wait for the ack response signal. Taking into account the stable time of the current, after the ack signal is pulled up, the state machine introduces the wait count, because the wait time and the area of the power field, the process, etc., are highly related, often there will be an accurate value on the silicon chip, so the count delay is transmitted by the software register (*dly_en_up).

2. After the first step is complete, the state machine jumps to the *Up_Clk_Rst_Clp* state immediately. In this state, unreset the power domain, unclamp across the power boundary, and turn on the power domain clock are performed. Taking into account the reset enable and propagation delay, the clamp signal propagation and enable delay, and the delay from declaration clock gating to clock on, the state machine introduces a count wait, which is passed by the registers *dly_deassert_ares, *dly_unclamp_io, and *dly_nodisable_clk, respectively. By configuring the delay register to different values, the operation sequence of the unreset operation, the unclamp operation, and the power domain clock are controlled. After all three groups are counted, a timeout signal is generated.

3. After the count in step 2 is complete, the state machine switches to the *Power On* state immediately. The power-on process is complete. The current state is that the clock in the power domain is jumping, the reset operation is released, the signal clamping state across the power domain boundary is transparent, and the request signal and response signal of the power switch are in a high state.

3.14.6.5 PCU Registers

Table 3-255 PCU registers description

Reg Name		Name	Access	Description
*_GLB_CTRL	31:1	reserved	RO	
	0	sw_override	RW	0:Hardware Control mode 1: Software Control mode Default: 0
PD*_SW_COLLAPSE	31:1	reserved	RO	
	0	sw_collapse_en	RW	0: The software triggers the hardware state power-on

Reg Name		Name	Access	Description
				1: The software triggers the hardware state power-off
PD*_SW_PWR	31:2	reserved	RO	
	1	psc_en_ack	RO	0: The power switch is off 1: The power switch is on
	0	sw_ps_en	RW	0: Request to turn off the power switch 1: Request to turn on the power switch
PD*_SW_RESET	31:1	reserved	RO	
	0	sw_ps_ares	RW	1:reset 0:de-reset
PD*_SW_ISO	31:1	Reserved	RO	
	0	sw_clamp_io	RW	0: isolation disable 1: isolation enable
PD*_SW_CLK_DISABLE	31:1	reserved	RO	
	0	sw_clk_disable	RW	0:clock enable 1:clock disable
PD*_PWR_DELAY	31:24	Reserved	RO	
	23:20	en_down_div_exp	RW	Power off delay time Granularity default:4'd3
	19:16	en_up_div_exp	RW	Power on delay time Granularity default:4'd3
	15:8	dly_en_down	RW	Power off delay default:8'd6
	7:0	dly_en_up	RW	Power on delay default:8'd6
PD*_CLK_DELAY	31:24	reserved	RO	
	23:20	disable_div_exp	RW	Gated clock delay time Granularity default:4'd2
	19:16	nodisable_div_exp	RW	Enable clock delay time Granularity default:4'd2
	15:8	dly_disable_clk	RW	Gated clock delay
	7:0	dly_nodisable_clk	RW	Enable clock delay
PD*_RST_DELAY	31:24	reserved	RO	
	23:20	assert_div_exp	RW	Reset delay time Granularity default:4'd2
	19:16	deassert_div_exp	RW	De-Reset delay time Granularity default:4'd2
	15:8	dly_assert_ares	RW	Reset delay
	7:0	dly_deassert_ares	RW	De-Reset delay
PD*_CLAMP_DELAY	31:24	reserved	RO	
	23:20	clamp_div_exp	RW	Enable Isolation delay time Granularity default:4'd2
	19:16	unclamp_div_exp	RW	Disable Isolation delay

Reg Name		Name	Access	Description
				time Granularity default:4'd2
	15:8	dly_clamp_io	RW	Enable Isolation delay
	7:0	dly_unclamp_io	RW	Disable Isolation delay
PD*_DEBUG	31:14	reserved	RO	
	13	smro_clk_disable	RO	0: The hardware state machine control power domain clock is valid 1: The hardware state machine control power domain clock is invalid
	12	smro_ps_ares	RO	1: The hardware state machine controls the reset of the power domain 0: The hardware state machine controls the de-reset of the power domain
	11	smro_clamp_io	RO	0: isolation disable 1: isolation enable
	10	smro_ps_en	RO	0: Request to turn off the power switch 1: Request to turn on the power switch
	9	loc_clk_disable	RO	Clock failure signal for the current state of the hardware state machine
	8	loc_ps_ares	RO	Reset signal for the current state of the hardware state machine
	7	loc_clamp_io	RO	Boundary isolation signal for the current state of the hardware state machine
	6	loc_ps_en	RO	A power request signal for the current state of the hardware state machine
	5	loc_cgc_clk_en	RO	Hardware state machine clock gating enable signal
	4	loc_cntr_load	RO	The counter loading delay time was enabled
	3	loc_cntr_en	RO	Counter enable signal
	2:0	STATE	RO	Hardware state machine statusPower On:3'b011 Power Off:3'b000

Table 3-256 PCU registers address description

Bus Protocol	**apb3**
Base Address	**0x5122_0000**
Addr Width	**10'h000**
Data Width	**32**

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
10'h000	PCIe_GLB_CTRL	reserved	31	1		
		pcie_sw_override	0	0	RW	1'b0
10'h004	PCIe_COLLAPSE_CTRL	reserved	31	1		
		pcie_sw_collapse_en	0	0	RW	1'b0
10'h008	PCIe_PSCHAIN_CTRL	reserved	31	2		
		pcie_psc_en_ack	1	1	RO	1'b1
		pcie_sw_ps_en	0	0	RW	1'b1
10'h0c	PCIe_ASYNC_RST_CTRL	reserved	31	1		
		pcie_sw_ares	0	0	RW	1'b0
10'h010	PCIe_CLAMP_CTRL	reserved	31	1		
		pcie_sw_clamp_io	0	0	RW	1'b0
10'h014	PCIe_CLK_CTRL	reserved	31	1		
		pcie_sw_clk_disable	0	0	RW	1'b0
10'h018	PCIe_PWR_DLY	reserved	31	24		
		pcie_en_down_div_exp	23	20	RW	4'h3
		pcie_en_up_div_exp	19	16	RW	4'h3
		pcie_dly_en_down	15	8	RW	8'h6
		pcie_dly_en_up	7	0	RW	8'h6
10'h01c	PCIe_CLK_DLY	reserved	31	24		
		pcie_disable_div_exp	23	20	RW	4'h2
		pcie_nodisable_div_exp	19	16	RW	4'h2
		pcie_dly_disable_clk	15	8	RW	8'h2
		pcie_dly_nodisable_clk	7	0	RW	8'h4
10'h020	PCIe_RST_DLY	reserved	31	24		
		pcie_assert_div_exp	23	20	RW	4'h2
		pcie_deassert_div_exp	19	16	RW	4'h2
		pcie_dly_assert_ares	15	8	RW	8'h4
		pcie_dly_deassert_ares	7	0	RW	8'h2
10'h024	PCIe_CLAMP_DLY	reserved	31	24		
		pcie_clamp_div_exp	23	20	RW	4'h2
		pcie_unclamp_div_exp	19	16	RW	4'h2
		pcie_dly_clamp_io	15	8	RW	8'h3
		pcie_dly_unclamp_io	7	0	RW	8'h3
10'h028	PCIe_DEBUG	reserved	31	14		
		pcie_smro_clk_disable	13	13	RO	1'b0
		pcie_smro_ps_ares	12	12	RO	1'b0

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
		pcie_smro_clamp_io	11	11	RO	1'b0
		pcie_smro_ps_en	10	10	RO	1'b1
		pcie_loc_clk_disable	9	9	RO	1'b0
		pcie_loc_ps_ares	8	8	RO	1'b0
		pcie_loc_clamp_io	7	7	RO	1'b0
		pcie_loc_ps_en	6	6	RO	1'b1
		pcie_loc_cgc_clk_en	5	5	RO	1'b1
		pcie_loc_cntr_load	4	4	RO	1'b0
		pcie_loc_cntr _en	3	3	RO	1'b0
		pcie_STATE	2	0	RO	3'b011
10'h040	DSP0_GLB_CTRL	reserved	31	1		
		dsp0_sw_override	0	0	RW	1'b0
10'h044	DSP0_COLLAPSE_CTRL	reserved	31	1		
		dsp0_sw_collapse_en	0	0	RW	1'b0
10'h048	DSP0_PSCHAIN_CTRL	reserved	31	2		
		dsp0_psc_en_ack	1	1	RO	1'b1
		dsp0_sw_ps_en	0	0	RW	1'b1
10'h04c	DSP0_ASYNC_RST_CTRL	reserved	31	1		
		dsp0_sw_ares	0	0	RW	1'b0
10'h050	DSP0_CLAMP_CTRL	reserved	31	1		
		dsp0_sw_clamp_io	0	0	RW	1'b0
10'h054	DSP0_CLK_CTRL	reserved	31	1		
		dsp0_sw_clk_disable	0	0	RW	1'b0
10'h058	DSP0_PWR_DLY	reserved	31	24		
		dsp0_en_down_div_exp	23	20	RW	4'h3
		dsp0_en_up_div_exp	19	16	RW	4'h3
		dsp0_dly_en_down	15	8	RW	8'h6
		dsp0_dly_en_up	7	0	RW	8'h6
10'h05c	DSP0_CLK_DLY	reserved	31	24		
		dsp0_disable_div_exp	23	20	RW	4'h2
		dsp0_nodisable_div_exp	19	16	RW	4'h2
		dsp0_dly_disable_clk	15	8	RW	8'h2
		dsp0_dly_nodisable_clk	7	0	RW	8'h4
10'h060	DSP0_RST_DLY	reserved	31	24		
		dsp0_assert_div_exp	23	20	RW	4'h2
		dsp0_deassert_div_exp	19	16	RW	4'h2

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
		dsp0_dly_assert_ares	15	8	RW	8'h4
		dsp0_dly_deassert_ares	7	0	RW	8'h2
10'h064	DSP0_CLAMP_DLY	reserved	31	24		
		dsp0_clamp_div_exp	23	20	RW	4'h2
		dsp0_unclamp_div_exp	19	16	RW	4'h2
		dsp0_dly_clamp_io	15	8	RW	8'h3
		dsp0_dly_unclamp_io	7	0	RW	8'h3
10'h068	DSP0_DEBUG	reserved	31	14		
		dsp0_smro_clk_disable	13	13	RO	1'b0
		dsp0_smro_ps_ares	12	12	RO	1'b0
		dsp0_smro_clamp_io	11	11	RO	1'b0
		dsp0_smro_ps_en	10	10	RO	1'b1
		dsp0_loc_clk_disable	9	9	RO	1'b0
		dsp0_loc_ps_ares	8	8	RO	1'b0
		dsp0_loc_clamp_io	7	7	RO	1'b0
		dsp0_loc_ps_en	6	6	RO	1'b1
		dsp0_loc_cgc_clk_en	5	5	RO	1'b1
		dsp0_loc_cntr_load	4	4	RO	1'b0
		dsp0_loc_cntr _en	3	3	RO	1'b0
		dsp0_STATE	2	0	RO	3'b011
10'h080	VI_GLB_CTRL	reserved	31	1		
		vi_sw_override	0	0	RW	1'b0
10'h084	VI_COLLAPSE_CTRL	reserved	31	1		
		vi_sw_collapse_en	0	0	RW	1'b0
10'h088	VI_PSCHAIN_CTRL	reserved	31	2		
		vi_psc_en_ack	1	1	RO	1'b1
		vi_sw_ps_en	0	0	RW	1'b1
10'h08c	VI_ASYNC_RST_CTRL	reserved	31	1		
		vi_sw_ares	0	0	RW	1'b0
10'h090	VI_CLAMP_CTRL	reserved	31	1		
		vi_sw_clamp_io	0	0	RW	1'b0
10'h094	VI_CLK_CTRL	reserved	31	1		
		vi_sw_clk_disable	0	0	RW	1'b0
10'h098	VI_PWR_DLY	reserved	31	24		
		vi_en_down_div_exp	23	20	RW	4'h3
		vi_en_up_div_exp	19	16	RW	4'h3
		vi_dly_en_down	15	8	RW	8'h6
		vi_dly_en_up	7	0	RW	8'h6
10'h09c	VI_CLK_DLY	reserved	31	24		

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
		vi_disable_div_exp	23	20	RW	4'h2
		vi_nodisable_div_exp	19	16	RW	4'h2
		vi_dly_disable_clk	15	8	RW	8'h2
		vi_dly_nodisable_clk	7	0	RW	8'h4
10'h0a0	VI_RST_DLY	reserved	31	24		
		vi_assert_div_exp	23	20	RW	4'h2
		vi_deassert_div_exp	19	16	RW	4'h2
		vi_dly_assert_ares	15	8	RW	8'h4
		vi_dly_deassert_ares	7	0	RW	8'h2
10'h0a4	VI_CLAMP_DLY	reserved	31	24		
		vi_clamp_div_exp	23	20	RW	4'h2
		vi_unclamp_div_exp	19	16	RW	4'h2
		vi_dly_clamp_io	15	8	RW	8'h3
		vi_dly_unclamp_io	7	0	RW	8'h3
10'h0a8	VI_DEBUG	reserved	31	14		
		smro_clk_disable	13	13	RO	1'b0
		smro_ps_ares	12	12	RO	1'b0
		smro_clamp_io	11	11	RO	1'b0
		smro_ps_en	10	10	RO	1'b1
		loc_clk_disable	9	9	RO	1'b0
		loc_ps_ares	8	8	RO	1'b0
		loc_clamp_io	7	7	RO	1'b0
		loc_ps_en	6	6	RO	1'b1
		loc_cgc_clk_en	5	5	RO	1'b1
		loc_cntr_load	4	4	RO	1'b0
		loc_cntr _en	3	3	RO	1'b0
		STATE	2	0	RO	3'b011
10'h0c0	VO_GLB_CTRL	reserved	31	1		
		vo_sw_override	0	0	RW	1'b0
10'h0c4	VO_COLLAPSE_CTRL	reserved	31	1		
		vo_sw_collapse_en	0	0	RW	1'b0
10'h0c8	VO_PSCHAIN_CTRL	reserved	31	2		
		vo_psc_en_ack	1	1	RO	1'b1
		vo_sw_ps_en	0	0	RW	1'b1
10'h0cc	VO_ASYNC_RST_CTRL	reserved	31	1		
		vo_sw_ares	0	0	RW	1'b0
10'h0d0	VO_CLAMP_CTRL	reserved	31	1		
		vo_sw_clamp_io	0	0	RW	1'b0
10'h0d4	VO_CLK_CTRL	reserved	31	1		
		vo_sw_clk_disable	0	0	RW	1'b0
10'h0d8	VO_PWR_DLY	reserved	31	24		

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
		vo_en_down_div_exp	23	20	RW	4'h3
		vo_en_up_div_exp	19	16	RW	4'h3
		vo_dly_en_down	15	8	RW	8'h6
		vo_dly_en_up	7	0	RW	8'h6
10'h0dc	VO_CLK_DLY	reserved	31	24		
		vo_disable_div_exp	23	20	RW	4'h2
		vo_nodisable_div_exp	19	16	RW	4'h2
		vo_dly_disable_clk	15	8	RW	8'h2
		vo_dly_nodisable_clk	7	0	RW	8'h4
10'h0e0	VO_RST_DLY	reserved	31	24		
		vo_assert_div_exp	23	20	RW	4'h2
		vo_deassert_div_exp	19	16	RW	4'h2
		vo_dly_assert_ares	15	8	RW	8'h4
		vo_dly_deassert_ares	7	0	RW	8'h2
10'h0e4	VO_CLAMP_DLY	reserved	31	24		
		vo_clamp_div_exp	23	20	RW	4'h2
		vo_unclamp_div_exp	19	16	RW	4'h2
		vo_dly_clamp_io	15	8	RW	8'h3
		vo_dly_unclamp_io	7	0	RW	8'h3
10'h0e8	VO_DEBUG	reserved	31	14		
		smro_clk_disable	13	13	RO	1'b0
		smro_ps_ares	12	12	RO	1'b0
		smro_clamp_io	11	11	RO	1'b0
		smro_ps_en	10	10	RO	1'b1
		loc_clk_disable	9	9	RO	1'b0
		loc_ps_ares	8	8	RO	1'b0
		loc_clamp_io	7	7	RO	1'b0
		loc_ps_en	6	6	RO	1'b1
		loc_cgc_clk_en	5	5	RO	1'b1
		loc_cntr_load	4	4	RO	1'b0
		loc_cntr _en	3	3	RO	1'b0
		STATE	2	0	RO	3'b011
10'h140	CODEC_GLB_CTRL	reserved	31	1		
		codec_sw_override	0	0	RW	1'b0
10'h144	CODEC_COLLAPSE_CTRL	reserved	31	1		
		codec_sw_collapse_en	0	0	RW	1'b0
10'h148	CODEC_PSCHAIN_CTRL	reserved	31	2		
		codec_psc_en_ack	1	1	RO	1'b1
		codec_sw_ps_en	0	0	RW	1'b1
10'h14c	CODEC_ASYNC_RST_CTRL	reserved	31	1		
		codec_sw_ares	0	0	RW	1'b0

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
10'h150	CODEC_CLAMP_CTRL	reserved	31	1		
		codec_sw_clamp_io	0	0	RW	1'b0
10'h154	CODEC_CLK_CTRL	reserved	31	1		
		codec_sw_clk_disable	0	0	RW	1'b0
10'h158	CODEC_PWR_DLY	reserved	31	24		
		codec_en_down_div_exp	23	20	RW	4'h3
		codec_en_up_div_exp	19	16	RW	4'h3
		codec_dly_en_down	15	8	RW	8'h6
		codec_dly_en_up	7	0	RW	8'h6
10'h15c	CODEC_CLK_DLY	reserved	31	24		
		codec_disable_div_exp	23	20	RW	4'h2
		codec_nodisable_div_exp	19	16	RW	4'h2
		codec_dly_disable_clk	15	8	RW	8'h2
		codec_dly_nodisable_clk	7	0	RW	8'h4
10'h160	CODEC_RST_DLY	reserved	31	24		
		codec_assert_div_exp	23	20	RW	4'h2
		codec_deassert_div_exp	19	16	RW	4'h2
		codec_dly_assert_ares	15	8	RW	8'h4
		codec_dly_deassert_ares	7	0	RW	8'h2
10'h164	CODEC_CLAMP_DLY	reserved	31	24		
		codec_clamp_div_exp	23	20	RW	4'h2
		codec_unclamp_div_exp	19	16	RW	4'h2
		codec_dly_clamp_io	15	8	RW	8'h3
		codec_dly_unclamp_io	7	0	RW	8'h3
10'h168	CODEC_DEBUG	reserved	31	14		
		smro_clk_disable	13	13	RO	1'b0
		smro_ps_ares	12	12	RO	1'b0
		smro_clamp_io	11	11	RO	1'b0
		smro_ps_en	10	10	RO	1'b1
		loc_clk_disable	9	9	RO	1'b0
		loc_ps_ares	8	8	RO	1'b0
		loc_clamp_io	7	7	RO	1'b0
		loc_ps_en	6	6	RO	1'b1
		loc_cgc_clk_en	5	5	RO	1'b1
		loc_cntr_load	4	4	RO	1'b0

Offset	Reg Name	Field			Access	Default
		Name	MSB	LSB		
		loc_cntr _en	3	3	RO	1'b0
		STATE	2	0	RO	3'b011
10'h1c0	NPU_ASYNC_RST_CTRL	reserved	31	1		
		npu_sw_ares	0	0	RW	1'b0
10'h1c4	NPU_CLAMP_CTRL	reserved	31	1		
		npu_sw_clamp_io	0	0	RW	1'b1
10'h1c8	NPU_CLK_CTRL	reserved	31	1		
		npu_sw_clk_disable	0	0	RW	1'b0