Interim Release

AndesCore™ AX46MPV Data Sheet

Document Number DS280-02
Date Issued 2025-05-19

Copyright Notice

AndesCore™, AndeSight™, AndeShape™, AndESLive™, AndeSoft™, AndeStar™, Andes Custom Extension™, CoDense™, StackSafe™, QuickNap™, AndesClarity™, Andes AutoOpTune™, AndeSim™, AndeSysC™, AndeSentry™, AndesAIRE™, AnDLA™, NNPilot™, Andes-Embedded and Driving Innovations are trademarks owned by Andes Technology Corporation. All other trademarks used herein are the property of their respective owners.

This document contains confidential information pertaining to Andes Technology Corporation. Use of this copyright notice is precautionary and does not imply publication or disclosure. Neither the whole nor part of the information contained herein may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the written permission of Andes Technology Corporation.

The product described herein is subject to continuous development and improvement. Thus, all information herein is provided by Andes in good faith but without warranties. This document is intended only to assist the reader in the use of the product. Andes Technology Corporation shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

Contact Information

Should you have any problems with the information contained herein, you may contact Andes Technology Corporation through:

Email — support@andestech.com
Website — https://es.andestech.com/eservice/

Please include the following information in your inquiries:

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of the problem.

General suggestions for improvements are welcome.

Revision History

Rev.	Rev. Date	Revised Content
0.2	2025-05-19	Draft Release
0.1	2025-03-19	Draft Release

Contents

List of Figures

List of Tables

1 Overview

This document provides information about the AndesCore AX46MPV multi-core processor, and associated platform/peripheral IPs that come with the AX46MPV release.

Note

- The RISC-V specification refers to the hardware thread as "hart". This is equivalent to "core" herein and they are interchangeably used throughout this document.

1.1 AX46MPV Processor Features

The main features of the AX46MPV processor are:

Cluster

- Support of 1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16 Cores
- Support of MESI cache coherence protocol with Coherence Manager
- Core Interface
  - Low-latency/synchronous/asynchronous
- Bus Interface
  - AXI4 Protocol
  - Memory Interface and Memory Mapped I/O (MMIO) Interface
  - Optional I/O Coherence Port (IOCP)
  - Configurable clock
    - Synchronous SMEM-to-bus clock ratio as N:1
    - Asynchronous Bus clock
  - Configurable data width: 128/256/512
  - Configurable address width: 32–64 bits
  - Optional Shared Peripheral Port (SPP)
    - 64-bit data width
  - Optional Private Peripheral Interface (PPI)
    - AXI4/AHB-Lite Protocol
    - 64-bit data width
    - Configurable clock
      - Synchronous core to bus clock ratio as N:1
      - Asynchronous bus clock

CPU Core

- 8-stage in-order dual-issue execution pipeline

- Hardware multiplier
  - radix-2/radix-4/radix-16/radix-256/fast
- Hardware divider
- Branch prediction
  - Dynamic branch prediction
    - 256-entry branch target buffer (BTB)
    - 768-entry branch history table
    - 8-bit global branch history
    - 4-entry return address stack (RAS)
  - Optional ECC error protection
- Machine mode, Supervisor mode, and User mode
- Optional performance monitors
- Misaligned memory accesses
- RISC-V physical memory protection (PMP)
- Optional Programmable physical memory attributes (PPMA)

AndeStar V5 ISA

- RV64I base integer instruction set
- RISC-V Zifencei extension for instruction-fetch fence
- RISC-V Zicsr extension for control and status instructions
- RISC-V Zicntr extension for base counters and timers
- RISC-V "M" standard extension for integer multiplication and division
- RISC-V "A" standard extension for atomic instructions
- RISC-V "C" standard extension for compressed instructions
- RISC-V CMO extension for base cache management operation
  - Zicbom, Zicboz and Zicbop
- Optional RISC-V "F" and "D" standard extensions for single/double-precision floating-point
  - Zfh extension for 16-bit half-precision binary floating-point instruction
  - RISC-V BF16, Zfbfmin, extension
- Optional RISC-V Zc* extension for code size reduction
- Optional RISC-V "B" extension for bit-manipulation
  - Zba, Zbb, Zbc and Zbs extensions
- Optional RISC-V "P" extension for DSP/SIMD instructions (draft)
- RISC-V "V" extension for vector operations
  - Configurable VLEN: 128–2048
  - Zve32x, Zve32f, Zve64x, Zve64f, and Zve64d
  - The single-letter V extension
  - RISC-V BF16 extensions: Zvfbfmin and Zvfbfwma
- AndeStar V5 instruction extensions

  - Andes Performance Extension
  - Andes CoDense extension
  - Andes new CoDense extension
  - Andes Scalar BFLOAT16 conversion extension
  - Andes Vector BFLOAT16 conversion extension
  - Andes Vector INT4 load extension
  - Andes Vector packed FP16 extension
  - Andes Vector dot product FP16 extension
  - Andes Vector small INT handling extension
  - Andes Vector quad-widening integer multiply-add extension
- Andes Custom Extension (ACE)
- Andes BFLOAT16 Computation
  - Use CSR field to select a format of floating-point half precision: FP16 or BFLOAT16
  - Supported floating-point instructions including:
    * RISC-V "F" and "D" standard extension
    * RISC-V "V" standard extension
    * Andes V5 Vector Small INT Handling Extension
    * Andes V5 Vector Packed FP16 Extension

Andes Custom Extension

- Simple and flexible interface for user-defined instructions
- ACE description file and the concise RTL design file in Verilog form
- COPILOT (Custom-OPtimized Instruction deveLOpment Tools) can generate extended components
  - Development tools
  - Instruction set simulator
  - AndesCore RTL

Memory Management Unit

- sv39/sv48/sv57/bare
- 4/8-entry fully associative *i*TLB
- 4/8/16-entry fully associative *d*TLB
- 32/64/128/256/512-entry 4-way set-associative shared TLB
- Optional ECC error protection

Memory Subsystem

- Support for non-blocking memory operations
- I & D-Caches
  - I-Cache is virtually indexed and physically tagged

  - D-Cache is virtually indexed and physically tagged
  - Cache size: 16KiB/32KiB/64KiB
  - Cache line size: 64 bytes
  - Set associativity: 4-way
  - Custom cache control operation through CSR read/write
  - D-Cache prefetch support
  - D-Cache write-around support
- I & D local memories
  - Size: 4KiB to 16MiB
  - Interface: RAM/Wait Cycle Interface
- High-bandwidth vector memory (HVM)
  - Size: 4KiB to 4GiB
- Optional Memory subsystem soft-error protection
  - Protection scheme: parity-checking or error-checking-and-correction (ECC)
  - Automatic hardware error correction
  - Protected memories:
    * I-Cache tag RAM and data RAM
    * D-Cache tag RAM and data RAM
    * I & D local memories
    * Level-3 cache tag RAM, data RAM and snoop-filter RAM

Power Management

- Wait-for-interrupt (WFI) mode
- Wait-for-event (WFE) mode

Debug

- RISC-V external debug support
- Configurable number of breakpoints: 2/4/8
- JTAG debug port for private debug transport module
  - JTAG: IEEE Std 1149.1 style 4-wire JTAG interface
- Optional secure debug

Trace

- Optional instruction trace

AndeStar Extension

- StackSafe hardware stack protection extension
- PowerBrake simple power/performance scaling extension
- Custom performance counter events

Platform-Level Interrupt Controller (PLIC)

- Configurable number of interrupts: 1–1023
- Configurable number of interrupt priorities: 3/7/15/31/63/127/255
- Configurable number of targets: 1–16
- Andes Vectored Interrupt extension

1.2 Block Diagram

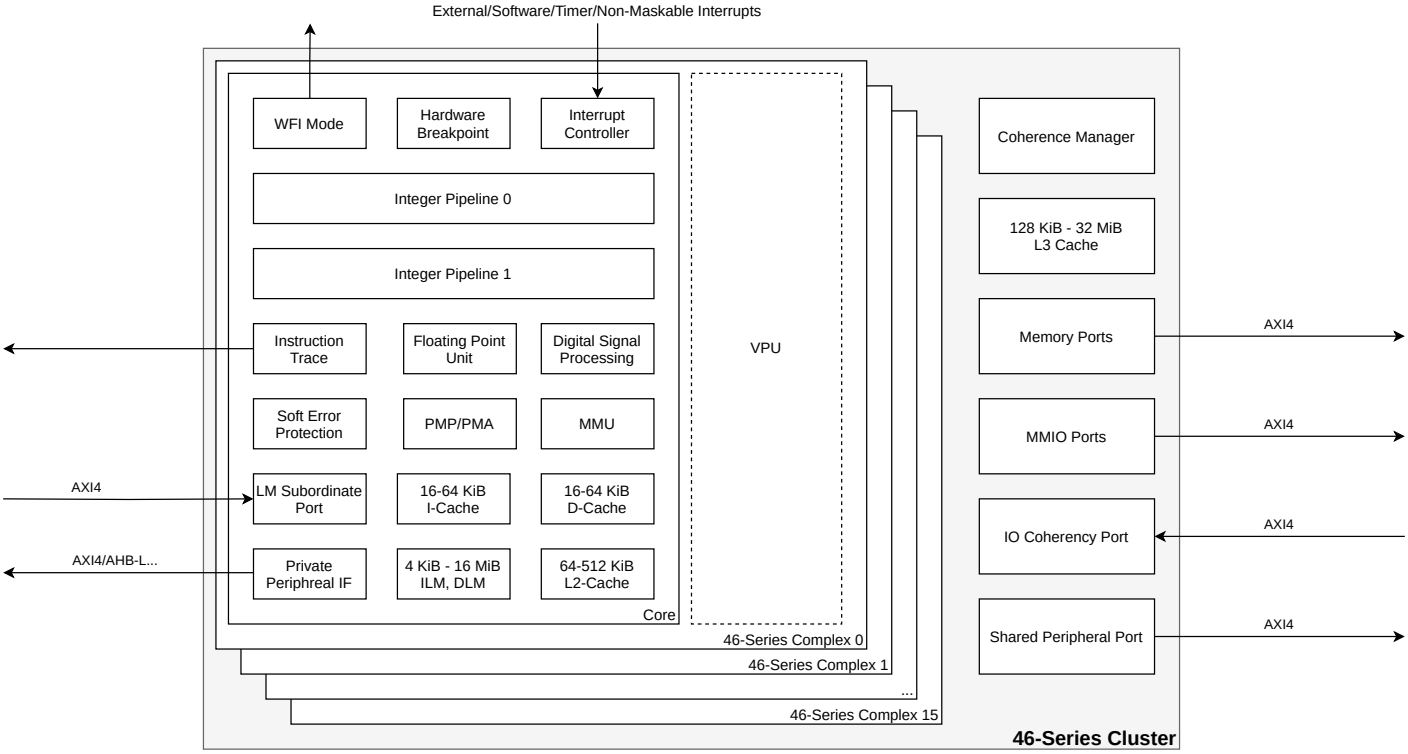


Figure 1: AX46MPV Block Diagram

Note

Depending on license agreements, some products include a non-removable VPU while others do not have a VPU.

1.3 Major Components

The following list describes the major components in the ax46mpv_core design (the inner core of the AX46MPV multi-core processor):

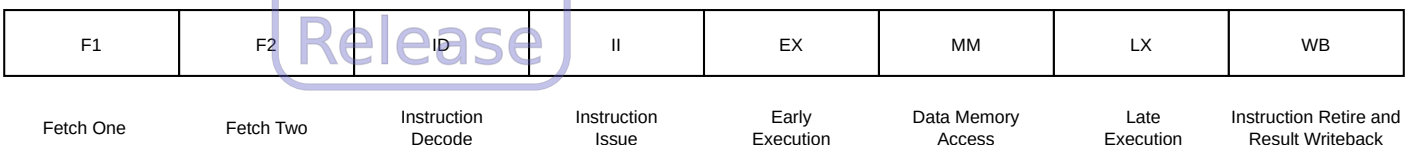
ACE	Andes Custom Extension
ALU	Arithmetic Logic Unit
BIU	Bus Interface Unit
BPU	Branch Prediction Unit
CSR	Control and Status Register
DCU	Data Cache Unit
DLM	Data Local Memory Controller
DSP	Digital Signal Processing
FASTMUL	Fast Multiplier
FPU	Floating Point Unit
ICU	Instruction Cache Unit
IFU	Instruction Fetch Unit
ILM	Instruction Local Memory Controller
IPIPE	Integer Pipeline
LSU	Load Store Unit
MDU	Multiplication and Division Unit
MMU	Memory Manager Unit
***i*TLB**	Instruction Translation Lookaside Buffer
***d*TLB**	Data Translation Lookaside Buffer
PMA	Programmable Physical Memory Attributes Unit
PMP	Physical Memory Protection Unit
RF	Integer Register File
TRIGM	Trigger Module
SMEM	Shared Memory
CM	Coherence Manager
IOCP	I/O Coherence Port
BITMANIP	Bit-Manipulation
PPI	Private Peripheral Interface
SPP	Shared Peripheral Port
VPU	Vector Processing Unit
VRF	Vector Register File
VIQ	Vector Instruction Queue
VLSU	Vector Load Store Unit

VALU	Vector Arithmetic Logic Unit
VMAC	Vector Multiply Accumulate Unit
VMAC2	The Second Vector Multiply Accumulate Unit
VDIV	Vector Divide Unit
VPERMUT	Vector Permutation Unit
VMASK	Vector Mask Unit
VFMIS	Vector Floating-Point Miscellaneous Unit
VFDIV	Vector Floating-Point Divide Unit
VACE	Vector Andes Custom Extension Unit
VSCB	Vector Scoreboard Unit
VCMT	Vector Commit Unit
VDIS	Vector Dispatch Unit
VROB	Vector Reorder Buffer

1.4 Micro-Architecture

1.4.1 Core Pipeline Stages and Activities

AX46MPV implements an 8-stage dual-issue pipeline architecture. The following figure shows the pipeline stages of the processor.



The pipeline activities of the corresponding stages are:

F1—Instruction Fetch Stage 1

- Fetching an instruction block from ILM/Instruction cache/bus
- Dynamic branch prediction

F2—Instruction Fetch Stage 2

- Fetch block data replies from ILM/Instruction cache/bus
- Branch prediction target acquired from BPU
- Redirect fetch address to prediction target

ID—Instruction Decode

- 16/32-bit instruction alignment
- Instruction decoding

II—Instruction Issue

- Register file read
- Resolving data dependency
- Instruction issue scheduling

EX—Early Execution

- Early ALU instruction execution
- Load/Store address generation
- Division instruction execution
- Multiplication instruction execution
- DSP/SIMD instruction execution

MM—Memory Access

- DLM/D-Cache access

- Early ALU branch resolution

LX—Late Execution

- Late ALU instruction execution

WB—Instruction Retire and Result Write-Back

- Interrupt resolution
- Instruction retire
- Register file write back
- Late ALU branch resolution
- ACE instruction issue

1.4.2 VPU Micro-Architecture

Figure 2 shows the VPU micro-architecture.

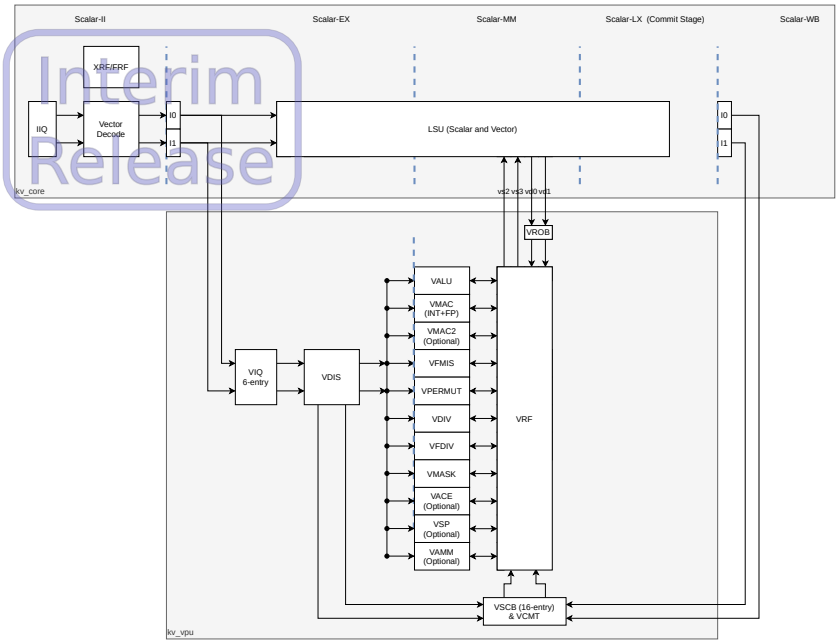


Figure 2: AX46MPV VPU Micro-Architecture

The core can issue two vector instructions per cycle to VIQ in the VPU. The VDIS can dispatch two instructions per cycle from VIQ to two different functional units, including:

- VALU
- VMAC
- VMAC2 (optional)
- VFMIS
- VPERMUT
- VDIV
- VFDIV
- VMASK
- VACE (optional)
- VSP (optional)

Instructions targeting different functional units are executed in parallel and out-of-order. However, instructions targeting the same functional unit are executed in program order. Each functional unit has a predetermined number of read and write ports (see Section 15.17.2). The VSCB tracks vector register dependencies between vector instructions and arbitrates the VRF read and write ports. The VCMT ensures that vector instructions are committed in program order.

2 Instruction Set Overview

2.1 Introduction

AX46MPV implements *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA Document Version 20250504 (TD004)*. The following instruction set extensions are implemented:

- RV64I base integer instruction set
- RISC-V Zifencei extension for instruction-fetch fence
- RISC-V Zicsr extension for control and status instructions
- RISC-V Zicntr extension for base counters and timers
- RISC-V "M" standard extension for integer multiplication and division
- RISC-V "A" standard extension for atomic instructions
- RISC-V "C" standard extension for compressed instructions
- RISC-V CMO extension for base cache management operation
  - Zicbom, Zicboz and Zicbop
- Optional RISC-V "F" and "D" standard extensions for single/double-precision floating-point
  - Zfh extension for 16-bit half-precision binary floating-point instruction
  - RISC-V BF16, Zfbfmin, extension
- Optional RISC-V Zc* extension for code size reduction
- Optional RISC-V "B" extension for bit-manipulation
  - Zba, Zbb, Zbc and Zbs extensions
- Optional RISC-V "P" extension for DSP/SIMD instructions (draft)
- RISC-V "V" extension for vector operations
  - Configurable VLEN: 128–2048
  - Zve32x, Zve32f, Zve64x, Zve64f, and Zve64d
  - The single-letter V extension
  - RISC-V BF16 extensions: Zvfbfmin and Zvfbfwma
- AndeStar V5 instruction extensions
  - Andes Performance Extension
  - Andes CoDense extension
  - Andes new CoDense extension
  - Andes Scalar BFLOAT16 conversion extension
  - Andes Vector BFLOAT16 conversion extension
  - Andes Vector INT4 load extension
  - Andes Vector packed FP16 extension
  - Andes Vector dot product FP16 extension
  - Andes Vector small INT handling extension
  - Andes Vector quad-widening integer multiply-add extension

- Andes Custom Extension (ACE)

For detailed information, please see *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA Document Version 20250504 (TD004)* and *AndeStar V5 Instruction Extension Specification (UM165)*.

2.2 Integer Registers

Table 1 lists all general-purpose integer registers.

Table 1: Integer Registers

Register	Signal Name	Description
x0	zero	Hardwired zero
x1	ra	Return address
x2	sp	Stack pointer
x3	gp	Global pointer
x4	tp	Thread pointer
x5	t0	Temporary/alternate link register
x6–x7	t1–t2	Temporaries
x8	s0/fp	Saved register/frame pointer
x9	s1	Saved register
x10–x11	a0–a1	Function arguments/return values
x12–x17	a2–a7	Function arguments
x18–x27	s2–s11	Saved registers
x28–x31	t3–t6	Temporaries

2.3 Atomic Instructions

The RVA extension includes load-reserved/store-conditional and atomic memory operation (AMO) instructions.

Atomic instructions are handled based on the physical memory attributes. For non-cacheable or device regions, atomic operations are finished by sending exclusive accesses to the AXI bus. For cacheable memory regions, atomic operations depend on the cache state and do not need to send exclusive access transactions. Proper cache coherency must be enabled.

2.3.1 Load-Reserved/Store-Conditional Instructions

The processor tracks at most one physical address location for `LR-SC` instructions at a time. An LR instruction registers a reservation. An SC instruction succeeds only if its address matches the reserved address and the reservation is still valid. The reservation is canceled under any of the following conditions:

- A load or store instruction is executed.
- An AMO instruction is executed.
- mcctlcommand or ucctlcommand is written.
- A trap or an NMI is taken.
- A FENCE.I, FENCE, or SFENCE.VMA instruction is executed.
- The processor enters debug mode.
- The cache line of the reservation is invalidated by the coherence manager.
  - LR/SC to read-no-allocate memory (MTYP=8/10) is not supported. Hardware sets NAMO when writing 8/10 to MTYP. See Section 16.15.1.

2.3.2 Atomic Memory Operation Instructions

An atomic memory operation is expanded to an LR-modify-SC sequence in the processor. First, the LR instruction loads the memory content. The required operation is then performed on the retrieved data, and the SC instruction writes the result back to the memory. If the SC instruction fails, the sequence will be retried until it succeeds.

2.4 Misaligned Memory Access

AX46MPV implements the misaligned memory access to support accessing misaligned addresses without triggering Address Misaligned exceptions. This scheme can be enabled or disabled through the mmisc_ctl CSR.

2.4.1 Exceptions

When the misaligned memory access scheme is enabled, Access Fault exceptions will still be triggered under the following cases:

- Accesses to device regions
- Accesses across ILM or DLM boundary
- Accesses across HVM boundary
- Accesses with inconsistency PMA attributes
- Atomic accesses

If the misaligned memory access scheme is disabled, misaligned accesses will trigger Access Fault exceptions or Address Misaligned exceptions. Access fault exceptions are triggered when the following cases occur:

- Atomic accesses
- Address is located in a device region

Other misaligned accesses trigger Address Misaligned exceptions.

2.5 CMO Extension

AX46MPV supports all subgroups (Zicbom, Zicboz, Zicbop) in the CMO extension. The cache line size is fixed at 64 bytes, and CMO instructions ignore bits 0-5 of the address. CMO instructions do not support hardware watchpoints (type 2 address/data match triggers).

Table 2: Behaviors of Zicbom Instructions

Instruction	Description
CBO.INVAL	Invalidates the cache line in the cluster, including all D-Caches of all cores and the L3-Cache
CBO.CLEAN	Writes back the cache line in the cluster, including all D-Caches of all cores and the L3-Cache
CBO.FLUSH	Writes back and invalidates the cache line in the cluster, including all D-Caches of all cores and the L3-Cache

Note

- Zicbom instructions have no operation to the I-Cache.
- Zicbom instructions have no operation to caches when the address is in ILM or DLM.
- Zicbom instructions take effect on caches when the D-Cache is disabled.
- Zicbom instructions take effect on caches when the physical memory attribute of the address is cacheable, non-cacheable, or device memory.
- Zicbom instructions raise a store access fault when the physical memory attribute of the address is an empty hole.

AX46MPV breaks a CBO.ZERO instruction into multiple store micro-operations, each targeting an aligned 64-byte memory region. An exception may occur during one of the middle micro-operations, while earlier store micro-operations have been visible. In other words, the 64-byte region is partially written, and the later store micro-operations are not visible. AX46MPV raises a store access fault exception when the address is in the device region.

Table 3 shows the behaviors of CBO.ZERO instructions.

Table 3: Behaviors of Zicboz Instructions

Physical Memory Attribute	Description
Local Memory	Writes zero to local memory
Device	Raises store access faults
Cacheable memory	Writes zero to memory

Continued on next page...

Table 3: (continued)

Physical Memory Attribute	Description
Non-Cacheable memory	Writes zero to memory
HVM	Writes zero to memory
Empty hole	Raises store access faults

Table 4 shows the throughput of CBO.ZERO instructions in different memory regions.

Table 4: Throughput of CBO.ZERO Instructions

Physical Memory Attribute	Throughput (Cycle/Instruction)
ILM	TBD
DLM	TBD
Cacheable memory, MTYP = 8, Write-Hit	TBD
Cacheable memory, MTYP = 8, Write-Miss	TBD
Cacheable memory, MTYP = 9, Write-Hit	TBD
Cacheable memory, MTYP = 9, Write-Miss	TBD
Cacheable memory, MTYP = 10, Write-Hit	TBD
Cacheable memory, MTYP = 10, Write-Miss	TBD
Cacheable memory, MTYP = 11, Write-Hit	TBD
Cacheable memory, MTYP = 11, Write-Miss	TBD
Non-Cacheable memory	TBD
HVM	TBD

Zicbop instructions serve as hints to the hardware, indicating that the software plans to execute a specific type of memory access in the future. Their behaviors are as follows:

- PREFETCH.R and PREFETCH.W fill the cache line into the D-Cache:
  - No operation occurs when the D-Cache is disabled.
  - No operation occurs when the address is in ILM or DLM.
  - No operation occurs when the PMA is device memory, non-cacheable memory, or an empty hole.
- PREFETCH.I fills the cache line into the I-Cache:
  - No operation occurs when the I-Cache is disabled.
  - No operation occurs when the address is in ILM or DLM.
  - No operation occurs when the PMA is device memory, non-cacheable memory, or an empty hole.
- Zicbop instructions do not raise access faults or page faults.

2.6 Floating-Point ISA Extension

AX46MPV supports the "F" and "D" Standard Extensions, which enhance the performance of floating-point heavy applications. The supported configuration is indicated in the misa configuration register.

AX46MPV supports the following FPU features:

- Fully pipelined MAC instructions
- Hardware subnormal handling
- All rounding modes

2.6.1 Support for Half-Precision and BFLOAT16 Formats

AX46MPV supports half-precision (FP16) instructions as well as Andes extension instructions for conversion between BFLOAT16 and single-precision formats.

Support for half-precision instructions is implemented using standard RISC-V floating-point instruction formats, with the width field set to "H".

AX46MPV implements BFLOAT16 conversion extensions:

- Andes scalar BFLOAT16 conversion extension
- RISC-V BF16, Zfbfmin extension.
- Andes vector BFLOAT16 conversion extension
- RISC-V BF16 extensions: Zvfbfmin and Zvfbfwma

2.7 Vector Instructions ISA Extension

2.7.1 Extensions

AX46MPV implements RISC-V "V" Vector Extension Version 1.0.

- Configurable VLEN: 128–2048
- Configurable ELEN: 32 or 64 bits
- Vector Floating-point operation: none, FP32, or FP32 + FP64
  - Half-precision floating-point (FP16) *vector* operations are implemented.
  - RISC-V BF16 extensions, Zvfbfmin and Zvfbfwma, are implemented.
    - RISC-V BF16 specification version v0.6.9 is implemented.
- Vector load/store
  - Vector unit-stride instructions
  - Vector strided instructions
  - Vector indexed instructions
  - Unit-stride fault-only-first loads
  - Vector load/store segment instructions
  - Vector load/store whole register instructions
  - Andes V5 INT4 vector load extension
- Andes V5 vector small INT handling extension
- Andes V5 vector BFLOAT16 conversion extension
- Andes V5 vector packed FP16 extension
- Andes V5 vector dot product extension
- Andes V5 vector quad-widening integer multiply-add extension

2.7.2 Vector Start Index CSR (VSTART)

AX46MPV can execute most vector instructions with a non-zero `vstart` value, except for the following vector instructions:

- Vector reduction instructions

- Streaming port instructions
- vcpop, vfirst, vmsbf, vmsif, vmsof, viota
- vcompress

2.7.3 Vector Memory Alignment Constraints

If an element accessed by a vector memory instruction is not naturally aligned to the size of the element, an address misaligned exception is raised.

2.8 DSP ISA Extension

The processor implements the RISC-V "P" extension (draft) for DSP/SIMD ISA. The supported configuration is indicated in the mmsc_cfg register. This extension enables the processor to execute various DSP applications with lower power consumption and higher performance.

The supported DSP features include:

- SIMD Data Processing Instructions
- Partial-SIMD Data Processing Instructions
- 64-bit Profile Instructions
- Non-SIMD Instructions
- RV64 Only Instructions
- Overflow Status Manipulation Instructions

Please see the *AndeStar V5 DSP ISA Extension Specification (UM199)* for instruction details.

2.9 User Mode XLEN (UXL) Extension

2.9.1 Overview

AX46MPV implements 32-bit User mode XLEN (UXL) support while maintaining 64-bit operation in Supervisor and Machine modes.

2.9.2 Features

- Configurable User mode XLEN (UXL)
  - 32-bit or 64-bit operation in User mode

  - Fixed 64-bit operation in Supervisor and Machine modes
- Memory access limitations in 32-bit mode
  - 4-GiB memory access space when UXL=32
  - Address bits [VALEN-1:32] are forced to 0
  - Supported virtual address lengths (VALEN): 39, 48, 57
- Memory Management
  - Maintains RV64 MMU configuration
  - Supports SV39 page table
  - Wraps address at 32-bit boundary for UXL=32

2.9.3 Memory Operations

- Load/Store operations are limited to 4-GiB address space in UXL32 mode
- Wraps address at 32-bit boundary
- Supports full 64-bit address range when needed
  - Addresses exceeding 4-GiB limit use standard 64-bit handling
  - Uses SV39 page table translation

2.9.4 Extension Compatibility

- DSP Extension
  - Not supported in UXL32 mode
- ACE Extension
  - Not supported in UXL32 mode

2.9.5 Limitations and Restrictions

- User mode-only feature
- Memory access is limited to 4 GiB in 32-bit user mode
- DSP and ACE extensions are not available in UXL32 mode

3 Branch Prediction Unit

The processor implements a Branch Prediction Unit (BPU) for branch prediction during instruction fetch. The BPU contains a two-way 128-entry branch target buffer (BTB), a 4-entry return address stack (RAS), and a branch history table (BHT).

The BTB holds target addresses for unconditional jumps and conditional branches, while the RAS keeps return addresses for function calls. The BHT predicts whether conditional branches will be taken or not.

Branch predictions can be disabled by setting mmisc_ctl.BRPE to 0x0. When disabled, the IFU fetches instructions sequentially without using the prediction mechanism.

3.1 CCTL Operation for BTB

When BTB Soft Error Protection is configured as “ecc”, BTB_IX_RDATA and BTB_IX_WDATA CCTL operations are supported. These operations are invoked by writing CCTL commands to the mcctlcommand CSR register.

To define a BTB RAM word for manipulation, the mcctlbeginaddr CSR is provided. Table 5 shows the address format for BTB CCTL operations. In this format, the **INDEX** field defines the word address for the operation, while the **PARTITION** field specifies the RAM number of the BTB SRAM involved. The amount for BTB SRAMs may be two or four. If only two RAMs are present, a **PARTITION** value of 3 is treated as 1, and a value of 2 as 0.

The following diagram is an example specifying the word at address 0x10 in SRAM 0.

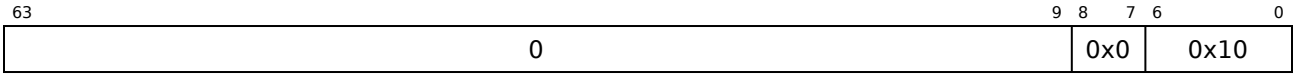


Table 5: Address Format for BTB CCTL Operations

Field	Bit Position	Description
PARTITION	mcctlbeginaddr[8:7]	RAM number of the BTB SRAM
INDEX	mcctlbeginaddr[6:0]	Index of the BTB SRAM

3.1.1 Reading Data from BTB SRAMs (BTB_IX_RDATA)

This operation reads data from a specific word in a BTB SRAM, as defined in mcctlbeginaddr, and transfers it to the mcctldata register. Details about the data format in mcctldata. Additionally, this operation observes the BTB_RWECC settings in mcache_ctl to read the corresponding ECC codes in the RAM to mecc_code.

3.1.2 Writing Data to BTB SRAMs (BTB_IX_WDATA)

This operation writes the contents of the mcctldata register to a specific word in a BTB RAM defined in mcctlbeginaddr. Additionally, this operation observes the BTB_RWECC settings in mcache_ctl to write the mecc_code ECC code to the data RAM.

4 Memory Management Unit

4.1 Introduction

The Memory Management Unit (MMU) handles virtual-to-physical address translation. It interfaces with the Instruction Fetch Unit (IFU) and the Load/Store Unit (LSU). To accelerate instruction address translation, an *i*TLB is used for the IFU, while a *d*TLB is used for the LSU. If the address translation information is not available in these two buffers, a TLB lookup request will be sent to the Shared TLB (STLB), a larger TLB defined in the MMU. If a TLB miss happens in the STLB, the hardware page table walker (PTW) will automatically traverse through page tables for the translation information.

By default, the MMU is disabled. It must be enabled and initialized before use, as no virtual-to-physical address translation occurs otherwise. Initialization is performed through the satp CSR.. Furthermore, an `SFENCE.VMA` instruction must be issued after satp is updated.

4.2 Address Translation

Virtual-to-physical address translation is page-based. Each virtual page is associated with a page table entry (PTE), which describes the physical page mapping information. Page table entries are inserted into the MMU (*i*TLB/*d*TLB/STLB) by the hardware page table walker (PTW) automatically.

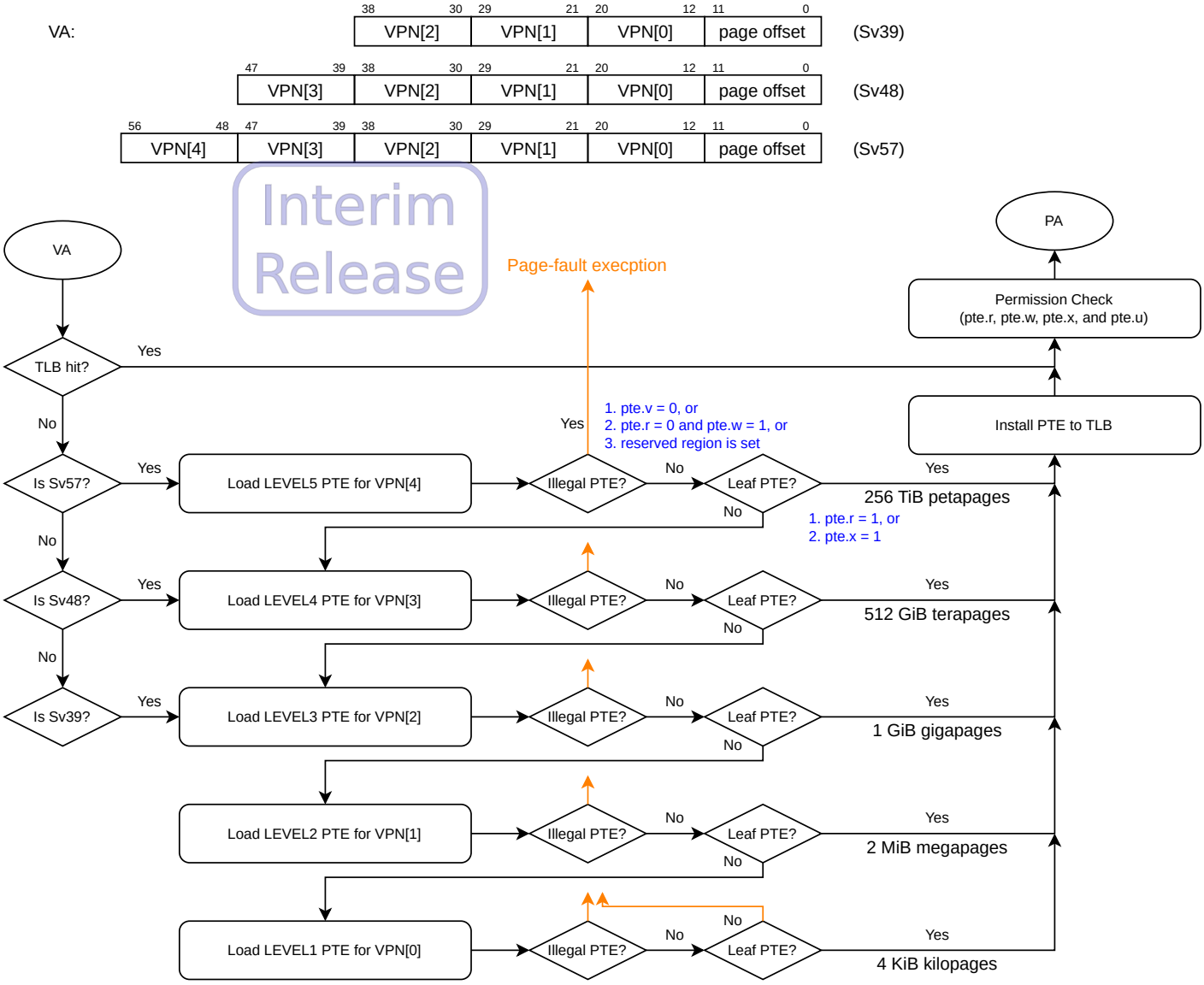


Figure 3: Virtual-to-Physical Address Translation

4.3 Translation Lookaside Buffer

The following sections describe the TLB operations.

4.3.1 Instruction *u*TLB (*i*TLB)

The Instruction *u*TLB (*i*TLB) is a fully associative cache that stores address translations (i.e., Page Table Entries) for instruction fetches. The number of entries is determined by the Number of iTLB Entries option. The *i*TLB retrieves the translation information from the STLB under *i*TLB misses. All non-global entries in the *i*TLB are cleared by `SFENCE.VMA` operations or write to satp.

4.3.2 Data *u*TLB (*d*TLB)

The Data *u*TLB (*d*TLB) is a fully associative cache that stores address translations (i.e., Page Table Entries) for data accesses. The number of entries is determined by the Number of dTLB Entries option. The *d*TLB retrieves the translation information from the STLB under *d*TLB misses. All non-global entries in the *d*TLB are cleared by `SFENCE.VMA` operations or write to satp.

4.3.3 Shared TLB (STLB)

The Shared TLB (STLB) contains a 4-way set-associative structure for 4K pages and a 4-entry fully associative structure for superpages that store address translations (i.e., the Page Table Entries) for both instruction fetches and data accesses. The number of entries is determined by the Number of Shared TLB Entries option.

4.3.4 Replacement Policy

TLBs (*i*TLB/*d*TLB/STLB) all implement pseudo-LRU replacement policy.

4.4 Page Table Walker (PTW)

4.4.1 Introduction

The PTW automatically fills the STLB with PTE entries in the system memory on an STLB lookup miss. Each TLB miss requires two to four memory references. To speed up page table hierarchy traversal, non-leaf PTEs are cached. There are two non-leaf PTE caches: one for instruction fetch misses and the other one for data load/store misses.

4.4.2 Page Table Address Formation

To find the correct PTE from the page table in memory, the PTW needs to perform up to three or four memory read operations to get PTEs. The physical addresses for these read operations are formed by the PTW as follows:

1. Let a be `satp`.$ppn \times$ PAGESIZE, and let i = LEVELS - 1 (PAGESIZE=2^{12}; LEVELS=3 for Sv39; LEVELS=4 for Sv48; LEVELS=4 for Sv57).
2. The physical address for PTE of VPN[i] is $a + va.vpn[i] \times$ PTESIZE (PTESIZE=8 for Sv39, Sv48, and Sv57). Perform a memory read at this address to retrieve the PTE. PMP checks are also applied to memory reads accessing the PTEs. If a violation occurs, an access exception will be raised based on the access type of the instructions triggering this page table walk. Let *pte* be the value of the memory read if no access violation occurs.
3. If $pte.v = 0$, or if *(w, r)* of *pte* is *(1, 0)*, stop and raise a page fault exception based on the access type of the instructions triggering this page table walk.
4. Otherwise, the PTE is valid.
   - If *(x, w, r)* of *pte* is *(0, 0, 0)*, this PTE is a pointer to the next level of the page table. Let $i = i$ - 1, $a = pte.ppn \times$ PAGESIZE and go to step 2 unless $i < 0$, in which case a page fault exception should be raised based on the access type of the instructions triggering this page table walk.
   - If *(x, w, r)* of *pte* is not *(0, 0, 0)*, then the leaf PTE for the page is found.

4.5 Attributes for Address Spaces

4.5.1 Attributes for Virtual Memory Pages

Each memory page is associated with attributes controlling page accesses. These attributes are stored in the page table entries along with their physical address mappings. The following table describes the format for the page table entries.

X	W	R	Meaning
0	0	0	Pointer to the next level of the page table
0	0	1	Read-only page
0	1	0	Reserved for future use
0	1	1	Read-write page
1	0	0	Execute-only page

Continued on next page...

X	W	R	Meaning
1	0	1	Read-execute page
1	1	0	Reserved for future use
1	1	1	Read-write-execute page

Table 6: Translated Address Space Attribute

Field	Bits	Description
PPN	53:10	Physical Page Number of the physical memory page
RSW	9:8	This field is reserved for use by supervisor software.
D	7	This bit indicates whether the virtual page has been written to since it was last cleared. When a virtual page is written and this bit is clear, a page fault exception is raised.
A	6	This bit indicates whether the virtual page has been read, written, or fetched from since it was lat cleared. When a virtual page is accessed and this bit is clear, a page fault exception is raised.
G	5	This bit designates a global mapping, meaning the mapping is shared among all address spaces. For non-leaf PTEs, setting the global mapping implies that all mappings in the subsequent levels of the page table are global. Note that failing to mark a global mapping as global merely reduces performance, whereas marking a non-global mapping as global is an error.
U	4	This bit indicates whether a page is accessible from user mode. U-mode software can access a page only when `U` = 1. If the `SUM` bit in the `sstatus` register is set, supervisor-mode software can also access pages with `U` = 1. However, supervisor code normally operates with the `SUM` bit cleared, in which case, it will fault on accesses to user-mode pages. Regardless of the `SUM` setting, supervisor may not execute code on pages with `U` = 1.
X	3	This bit indicates whether a page is executable. Attempting to fetch an instruction from a page that does not have execution permissions raises a fetch page fault exception.
W	2	This bit indicates whether a page is writable. Attempting to execute a store, store-conditional (regardless of success), or AMO instruction whose effective address lies within a page without write permissions raises a store page fault exception.
R	1	This bit indicates whether a page is readable. Attempting to execute a load or load-reserved instruction whose effective address lies within a page without read permissions raises a load page fault exception.

Continued on next page...

Table 6: (continued)

Field	Bits	Description
V	0	This bit indicates whether a PTE is valid.

5 Local Memory

5.1 Introduction

Local memories are used to store data or instructions that are accessed frequently or require deterministic access latency, such as interrupt service routines, system calls, video data, and real-time systems.

AX46MPV supports both instruction local memory (ILM) and data local memory (DLM). They are dedicated address spaces independent of the memory subsystem. Accesses to ILM and DLM bypass the cache and memory subsystems to achieve minimal latency.

5.2 Local Memory Spaces

The ILM address space is defined by the milmb.IBPA and micm_cfg.ILMSZ, while the DLM address space is defined by mdlmb.DBPA and mdcm_cfg.DLMSZ. Any addresses outside the local memory address spaces belong to the memory subsystem address space.

Instruction fetches go to the ILM or the memory subsystem, while load/store data accesses can access ILM, DLM, or the memory subsystem. To achieve maximum compatibility across Andes processors, the address spaces for ILM and DLM should not overlap. The address space access priorities for AX46MPV are defined in Table 7 for instruction fetches and Table 8 for load/store data accesses.

It is not recommended to configure ILM and DLM with the same base address; otherwise, UNPREDICTABLE behavior might happen.

Table 7: Priorities for Instruction Fetches

Address Hit the ILM Space	Address Hit the DLM Space	Actual Space Accessed
No	No	Memory Subsystem
No	Yes	Memory Subsystem
Yes	No	ILM
Yes	Yes	UNPREDICTABLE (not recommended; the ILM and DLM spaces should not overlap.)

Table 8: Priorities for Data Accesses

Address Hit the ILM Space	Address Hit the DLM Space	Actual Space Accessed
No	No	Memory Subsystem
No	Yes	DLM
Yes	No	ILM
Yes	Yes	UNPREDICTABLE (not recommended; the ILM and DLM spaces should not overlap.)

5.3 Local Memory Address Range

The local memory address ranges are listed in Table 9. LM_BASE represents the base address field of the ILM and DLM local memory base address system registers (milmb.IBPA and mdlmb.DBPA).

Table 9: Local Memory Address Range

LM Size	Start	End
4KiB	(LM_BASE[63:12]<<12)	(LM_BASE[63:12]<<12) + 0x000000FFF
8KiB	(LM_BASE[63:13]<<13)	(LM_BASE[63:13]<<13) + 0x000001FFF
16KiB	(LM_BASE[63:14]<<14)	(LM_BASE[63:14]<<14) + 0x000003FFF
32KiB	(LM_BASE[63:15]<<15)	(LM_BASE[63:15]<<15) + 0x000007FFF
64KiB	(LM_BASE[63:16]<<16)	(LM_BASE[63:16]<<16) + 0x00000FFFF
128KiB	(LM_BASE[63:17]<<17)	(LM_BASE[63:17]<<17) + 0x00001FFFF
256KiB	(LM_BASE[63:18]<<18)	(LM_BASE[63:18]<<18) + 0x00003FFFF
512KiB	(LM_BASE[63:19]<<19)	(LM_BASE[63:19]<<19) + 0x00007FFFF
1MiB	(LM_BASE[63:20]<<20)	(LM_BASE[63:20]<<20) + 0x0000FFFFF
2MiB	(LM_BASE[63:21]<<21)	(LM_BASE[63:21]<<21) + 0x0001FFFFF
4MiB	(LM_BASE[63:22]<<22)	(LM_BASE[63:22]<<22) + 0x0003FFFFF
8MiB	(LM_BASE[63:23]<<23)	(LM_BASE[63:23]<<23) + 0x0007FFFFF
16MiB	(LM_BASE[63:24]<<24)	(LM_BASE[63:24]<<24) + 0x000FFFFFF

5.4 Local Memory Usage Constraints

Local memories are optimized for access latency, but the design imposes the following usage restrictions:

- The addresses of VA and PA should be the same. Otherwise, UNPREDICTABLE behavior might happen.
- Accesses to the local memory are speculative. Therefore, devices with side effects on accesses should not be mapped to this region.
- Read-modify-write operations to local memories are not atomic. These operations include:
  - LR and SC instructions
  - AMO instructions
  - ECC error correction
  - ECC read-modify-write for stores with partial byte accesses

In addition, vector load and store instructions cannot access local memory, and attempting to do so will raise access fault exceptions.

6 Physical Memory Attributes

6.1 Introduction

Memory locations can have various attributes and are basically categorized into two types:

- Memory Regions: They can be cacheable or non-cacheable.
- Device Regions: They are non-cacheable, and accessing them may cause side effects.

AX46MPV uses the MTYP (memory type) field to define the cacheability and idempotency of memory regions. Please see descriptions in Section 16.15.1 for possible MTYP values.

AX46MPV supports two mechanisms for physical memory attributes (PMA):

- Static physical memory attributes
- Programmable physical memory attributes

6.2 Static Physical Memory Attributes

Up to 8 device regions can be statically configured in the processor through the Device Region configuration options. These regions have memory type attributes defined as (device, non-bufferable). If a physical address does not match any of the configured regions, its memory attributes are statically assigned to (cacheable memory, write-back, read and write-allocate) (MTYP = 11).

Device regions and ILM/DLM should not overlap with each other; otherwise, the behavior is UNDEFINED. See AndesCore AX46MPV Integration Guide (IG094) for how to configure static device regions.

6.3 Programmable Physical Memory Attributes

Programmable PMA allows dynamic adjustment of memory attributes at runtime to control the attributes of specific memory locations. If the settings in these entries conflict with the static Device Region settings, PMA entries will have higher priorities.

PMA entries are statically prioritized. The lowest-numbered entry that matches the physical address (PA) of an access determines the attribute type and whether AMO instructions are supported. If no PMA entry matches the address, the attribute type is determined by the statically configured PMA. See Section 16.15.1 for more information about PMA entries.

The amount of PMA entries implemented is configurable (see AndesCore AX46MPV Integration Guide (IG094)).

6.4 Memory Access Ordering

Accesses to device regions are guaranteed to be issued in program order and are non-speculative. Read accesses to device regions are allowed only after all previous accesses to device regions have completed. Similarly, write accesses to non-bufferable device regions will not be initiated until all prior accesses to device regions are completed. However, writes to bufferable device regions can proceed without waiting for a response from earlier writes to the same region.

Accesses to non-cacheable regions are also guaranteed to be issued in program order and are non-speculative. An access to a non-cacheable region may follow a non-cacheable region access without waiting for a response. In non-cacheable memory regions, a load may bypass an earlier store access if there is no data dependency. To ensure proper ordering, explicit `FENCE` instructions are necessary. When there is a data dependency (i.e., addresses overlapping), a load will wait for prior stores finished, and a store will wait for prior loads and stores finished.

On the other hand, accesses to cacheable memory regions can be speculative, and their order is not guaranteed. A load access to a cacheable memory region may bypass an earlier store if there is no data dependency. In such cases, explicit `FENCE` instructions are required to guarantee the order.

Table 10 shows the ordering of two instructions, A and B, where A < B (A comes earlier than B) in program order.

Table 10: Memory Access Ordering

A<B in Program Order	B				
A	Cacheable or Non-cacheable Memory	Device Bufferable Write	Device Bufferable Read	Device Non-bufferable Write	Device Non-bufferable Read
Cacheable or Non-cacheable Memory	-	-	-	-	-
Device Bufferable Write	-	-	-	<	<
Device Bufferable Read	-	<	<	<	<
Device Non-bufferable Write	-	<	<	<	<
Device Non-bufferable Read	-	<	<	<	<

AX46MPV supports sending multiple requests to bufferable device regions. Requests to a device region may not follow the program order. See AndesCore AX46MPV Integration Guide (IG094) for details.

6.5 Non-Cacheable Memory and Device Accesses

Accessing to non-cacheable memory and device will bypass private caches and shared caches regardless of whether the data is cached. These accesses do not affect the cache states.

6.6 Cacheable Memory Accesses to Private Caches

This section describes operations to private caches, including I-Cache and D-Cache.

6.6.1 Write-Back

Table 11: Behaviors upon Accessing Write-Back and No-Allocate Regions (MTYP = 8)

Scenario	Operations
Read-Hit	Read from a private cache
Write-Hit	Write to a private cache. Change the line state to “modified”
Read-Miss	Read from shared memory
Write-Miss	Write to shared memory

Table 12: Behaviors upon Accessing Write-Back and Read-Allocate Regions (MTYP = 9)

Scenario	Operations
Read-Hit	Read from a private cache
Write-Hit	Write to a private cache. Change the line state to “modified”
Read-Miss	Read from shared memory. Allocate the line in the private cache with exclusive/shared state.
Write-Miss	Write to shared memory

Table 13: Behaviors upon Accessing Write-Back and Write-Allocate Regions (MTYP = 10)

Scenario	Operations
Read-Hit	Read from a private cache
Write-Hit	Write to a private cache. Change the line state to “modified”
Read-Miss	Read from shared memory
Write-Miss	Allocate the line in the private cache with modified state. Write to shared memory

Table 14: Behaviors upon Accessing Write-Back and Read-and-Write-Allocate Regions (MTYP = 11)

Scenario	Operations
Read-Hit	Read from a private cache
Write-Hit	Write to a private cache. Change the line state to “modified”
Read-Miss	Read from shared memory. Allocate the line in the private cache with exclusive/shared state
Write-Miss	Allocate the line in the private cache with modified state. Write to shared memory

When a write-miss happens in a write-no-allocate region (MTYP = 8 or 9), the missed line is not allocated into the D-Cache. Instead, the write data is written to the L3-Cache. If the Write-Around feature is configured, AX46MPV implements write buffers to improve write bandwidth to cacheable memory. See Section 7.3.5 for more details.

AX46MPV does not improve the read bandwidth for read-no-allocate regions (MTYP = 8 or 10). When a read miss happens in such regions, the D-Cache sends a single transfer to shared memory. Reads to the same line are blocked until the transfer completes.

6.6.2 I-Cache Disabled Behaviors

When mcache_ctl.IC_EN is set to 0, all fetches to cacheable memory regions are treated as non-cacheable and bufferable. The value of mcache_ctl.IC_EN also affects the behavior of the ARCACHE signal. When I-Cache is disabled, the requests are sent to the MMIO port without passing through the private and shared caches.

6.6.3 D-Cache Disabled Behaviors

When mcache_ctl.DC_EN is 0, all load and store instructions to cacheable memory regions are treated as non-cacheable and bufferable. When the D-Cache is disabled, requests are sent to the MMIO port without passing through the private and shared caches.

6.6.4 Debug Mode

In the debug mode, allocate attributes are ignored. AX46MPV does not allocate lines to the D-Cache and I-Cache in this mode.

6.7 Cacheable Memory Accesses to L3-Cache

This section describes operations to the L3-Cache. L3-Cache read requests include cache filling and non-allocate read. L3-Cache write requests include cache eviction and non-allocate write.

Table 15: Behaviors upon Accessing Write-Back and No-Allocate Regions (MTYP = 8)

Scenario	Operations
Read-Hit	Read from the L3-Cache
Write-Hit	Write to the L3-Cache. Change the line state to “dirty”
Read-Miss	Read from the MEM interface
Write-Miss	Write to the MEM interface

Table 16: Behaviors upon Accessing Write-Back and Read-Allocate Regions (MTYP = 9)

Scenario	Operations
Read-Hit	Read from the L3-Cache
Write-Hit	Write to the L3-Cache. Change the line state to “dirty”
Read-Miss	Read from the MEM interface. Allocate the line in the L3-Cache with clean state
Write-Miss	Write to the MEM interface

Table 17: Behaviors upon Accessing Write-Back and Write-Allocate Regions (MTYP = 10)

Scenario	Operations
Read-Hit	Read from the L3-Cache
Write-Hit	Write to the L3-Cache. Change the line state to “dirty”
Read-Miss	Read from the L3-Cache
Write-Miss	Allocate the line in the L3-Cache with dirty state. Write to the MEM interface

Table 18: Behaviors upon Accessing Write-Back and Read-and-Write-Allocate Regions (MTYP = 11)

Scenario	Operations
Read-Hit	Read from the L3-Cache
Write-Hit	Write to the L3-Cache. Change the line state to “dirty”
Read-Miss	Read from the MEM interface. Allocate the line in the L3-Cache with clean state
Write-Miss	Allocate the line in the L3-Cache with dirty state. Write to the MEM interface

6.8 HVM Read Data Buffer

A HVM read data buffer is used to improve the performance of scalar loads to HVM region. When a scalar load hits in the buffer, the buffer provides the data directly without issuing memory requests. However, if a scalar load does not hit a buffer entry, the buffer issues a memory request and fills an entry.

Note

When memory is updated by another core or an external manager, software must invalidate the read data buffer for ensuring the coherence between memory and the buffer.

The HVM read data buffer is invalidated under the following conditions:

- All entries are filled, and the pseudo-LRU replacement policy determines the entry for replacement.
- A store address overlaps with a read data buffer entry.
- A `FENCE` instruction is executed.
- A `FENCE.I` instruction is executed
- A `WFI` instruction is executed.
- An `SFENCE` instruction is executed.
- An atomic instruction is executed.
- A CCTL instruction is executed.
- A `CBO.CLEAN`, `CBO.FLUSH`, or `CBO.INVAL` instruction is executed.
- The core in the debug mode.

7 Private Caches

7.1 Introduction

Each core contains private caches: an I-Cache, a D-Cache, and a private L2-Cache. The cache line size is fixed to 64 bytes.

Cache organization information can be collected from the micm_cfg register for the I-Cache and the mdcm_cfg register for the D-Cache. The configuration choices are listed below.

Table 19: Configuration Choices for the I-Cache

I-Cache Size	Ways (`micm_cfg.IWAY`)	Cache Lines Per Way (`micm_cfg.ISET`)
16 KiB	4	64
32 KiB	4	128
64 KiB	4	256

Table 20: Configuration Choices for the D-Cache

D-Cache Size	Ways (`mdcm_cfg.DWAY`)	Cache Lines Per Way (`mdcm_cfg.DSET`)
16 KiB	4	64
32 KiB	4	128
64 KiB	4	256

7.2 I-Cache

The I-Cache is virtually indexed and physically tagged (VIPT).

7.2.1 I-Cache Fill Operation

The I-Cache fill operation starts when a cacheable line is not present in the I-Cache. On a cache miss, a burst read request for the missed cache line is issued to the system bus to reduce access time. Up to two outstanding requests can be issued to system bus.

The fill operation may be aborted by system bus errors. If the error is on the critical word, a precise instruction access fault is triggered for the instruction fetch that caused the cache miss. If the error occurs on non-critical words, the fill operation will be canceled and the missed line will not be installed into the I-Cache. Instruction fetches before non-critical error words will not be affected since they have received the required data.

In Debug Mode, instruction fetches do not affect I-Cache contents, and all I-Cache misses do not cause cache replacements.

7.2.2 I-Cache Prefetch

After a cache miss, a series of sequential cache lines will be probed to check whether they are in the I-Cache. If not, streaming prefetch will be performed for these sequential cache lines. The instruction prefetch is turned off by default, and can be turned on by setting mcache_ctl.IPREF_EN to 0x1.

7.3 D-Cache

The D-Cache is virtually indexed and physically tagged (VIPT).

7.3.1 Cache Access Latency

Table 21: D-Cache Access Latency

	D-Cache Hit Latency
Load to ALU	0
Load to Address Generation	2

Note

- The 0-cycle latency assumes that the dependent instruction and the load instruction are dual-issued.
- The latency assumes that the dependent instruction is executed in the late ALU stage.
- The latency assumes that load word (LW) instructions are used.

Table 22: Load-to-Use Access Latency of D-Cache Miss in Cycles (**Private L2-Cache Size** Is 0, **L3-Cache Size** Is 0)

Number of Processor Cores	L3-Cache Tag/Snoop RAM Setup Cycle	L3-Cache Tag/Snoop RAM Output Cycle	D-Cache Miss Latency	Modified in Another Core
1	-	-	7	N/A
2-4	1-cycle	1-cycle	20	34/37/40/43
2-4	1-cycle	2-cycle	21	35/38/41/44
2-4	1-cycle	3-cycle	22	36/39/42/45
2-4	2-cycle	1-cycle	21	35/38/41/44
2-4	2-cycle	2-cycle	22	36/39/42/45
2-4	2-cycle	3-cycle	23	37/40/43/46
5-16	1-cycle	1-cycle	22	38/41/44/47
5-16	1-cycle	2-cycle	23	39/42/45/48
5-16	1-cycle	3-cycle	24	40/43/46/49
5-16	2-cycle	1-cycle	23	39/42/45/48
5-16	2-cycle	2-cycle	24	40/43/46/49
5-16	2-cycle	3-cycle	25	41/44/47/50

Note

- The latency assumes that the dependent instruction and the load instruction are dual-issued.
- The latency assumes that the dependent instruction is executed in the late ALU stage.
- The latency assumes that load word (LW) instructions are used.
- The latency assumes that the miss address is the first transfer of a cache line.
- The latency assumes that bus latency is one cycle after the request address is issued.
- The latency assumes that Core Interface is synchronous.
- The latency assumes that Bus Clock is synchronous.
- The latency assumes that the SMEM_CLK and BUS_CLK clock ratio is 1:1.
- The latency assumes that the bloom filter effectively bypasses the synonym check for the VIPT D-Cache.
- The latency of “Modified in Another Core” depends on the number of times the synonym index is looked up.

Table 23: Load-to-Use Access Latency of D-Cache Miss in Cycles (link:AndesCore_AX46MPV_IG094_V1.0.pdf[Number of Processor Cores] Is 1-4, **Private L2-Cache Size** Is 0)

L3-Cache Tag/Snoop RAM Setup Cycle	L3-Cache Tag/Snoop RAM Output Cycle	L3-Cache Data RAM Setup Cycle	L3-Cache Data RAM Output Cycle	L3-Cache Hit Latency	L3-Cache Miss Latency	Modified in Another Core
1-cycle	1-cycle	1-cycle	2-cycle	14	20	34/37/40/43
1-cycle	1-cycle	1-cycle	3-cycle	15	20	34/37/40/43
1-cycle	1-cycle	1-cycle	4-cycle	16	20	34/37/40/43
1-cycle	1-cycle	2-cycle	2-cycle	16	20	34/37/40/43
1-cycle	1-cycle	2-cycle	3-cycle	17	20	34/37/40/43
1-cycle	1-cycle	2-cycle	4-cycle	18	20	34/37/40/43
1-cycle	2-cycle	1-cycle	2-cycle	15	21	35/38/41/44
1-cycle	2-cycle	1-cycle	3-cycle	16	21	35/38/41/44
1-cycle	2-cycle	1-cycle	4-cycle	17	21	35/38/41/44
1-cycle	2-cycle	2-cycle	2-cycle	17	21	35/38/41/44
1-cycle	2-cycle	2-cycle	3-cycle	18	21	35/38/41/44
1-cycle	2-cycle	2-cycle	4-cycle	19	21	35/38/41/44
1-cycle	3-cycle	1-cycle	2-cycle	16	22	36/39/42/45
1-cycle	3-cycle	1-cycle	3-cycle	17	22	36/39/42/45
1-cycle	3-cycle	1-cycle	4-cycle	18	22	36/39/42/45
1-cycle	3-cycle	2-cycle	2-cycle	18	22	36/39/42/45
1-cycle	3-cycle	2-cycle	3-cycle	19	22	36/39/42/45
1-cycle	3-cycle	2-cycle	4-cycle	20	22	36/39/42/45
2-cycle	1-cycle	1-cycle	2-cycle	15	21	35/38/41/44
2-cycle	1-cycle	1-cycle	3-cycle	16	21	35/38/41/44
2-cycle	1-cycle	1-cycle	4-cycle	17	21	35/38/41/44
2-cycle	1-cycle	2-cycle	2-cycle	17	21	35/38/41/44
2-cycle	1-cycle	2-cycle	3-cycle	18	21	35/38/41/44
2-cycle	1-cycle	2-cycle	4-cycle	19	21	35/38/41/44
2-cycle	2-cycle	1-cycle	2-cycle	16	22	36/39/42/45
2-cycle	2-cycle	1-cycle	3-cycle	17	22	36/39/42/45
2-cycle	2-cycle	1-cycle	4-cycle	18	22	36/39/42/45
2-cycle	2-cycle	2-cycle	2-cycle	18	22	36/39/42/45
2-cycle	2-cycle	2-cycle	3-cycle	19	22	36/39/42/45

Continued on next page...

Table 23: (continued)

L3-Cache Tag/Snoop RAM Setup Cycle	L3-Cache Tag/Snoop RAM Output Cycle	L3-Cache Data RAM Setup Cycle	L3-Cache Data RAM Output Cycle	L3-Cache Hit Latency	L3-Cache Miss Latency	Modified in Another Core
2-cycle	2-cycle	2-cycle	4-cycle	20	22	36/39/42/45
2-cycle	3-cycle	1-cycle	2-cycle	17	23	37/40/43/46
2-cycle	3-cycle	1-cycle	3-cycle	18	23	37/40/43/46
2-cycle	3-cycle	1-cycle	4-cycle	19	23	37/40/43/46
2-cycle	3-cycle	2-cycle	2-cycle	19	23	37/40/43/46
2-cycle	3-cycle	2-cycle	3-cycle	20	23	37/40/43/46
2-cycle	3-cycle	2-cycle	4-cycle	21	23	37/40/43/46

Note

- The latency assumes that the dependent instruction and the load instruction are dual-issued.
- The latency assumes that the dependent instruction is executed in the late ALU stage.
- The latency assumes that load word (LW) instructions are used.
- The latency assumes that the miss address is the first transfer of a cache line.
- The latency assumes that bus latency is one cycle after the request address is issued.
- The latency assumes that Core Interface is synchronous.
- The latency assumes that Bus Clock is synchronous.
- The latency assumes that the SMEM_CLK and BUS_CLK clock ratio is 1:1.
- The latency assumes that the bloom filter effectively bypasses the synonym check for the VIPT D-Cache.
- The latency of “Modified in Another Core” depends on the number of times the synonym index is looked up.

Table 24: Load-to-Use Access Latency of D-Cache Miss in Cycles (link:AndesCore_AX46MPV_IG094_V1.0.pdf[Number of Processor Cores] Is 5-16, **Private L2-Cache Size** Is 0)

L3-Cache Tag/Snoop RAM Setup Cycle	L3-Cache Tag/Snoop RAM Output Cycle	L3-Cache Data RAM Setup Cycle	L3-Cache Data RAM Output Cycle	L3-Cache Hit Latency	L3-Cache Miss Latency	Modified in Another Core
1-cycle	1-cycle	1-cycle	2-cycle	16	22	38/41/44/47
1-cycle	1-cycle	1-cycle	3-cycle	17	23	38/41/44/47
1-cycle	1-cycle	1-cycle	4-cycle	17	23	38/41/44/47
1-cycle	1-cycle	2-cycle	2-cycle	18	24	38/41/44/47
1-cycle	1-cycle	2-cycle	3-cycle	18	24	38/41/44/47
1-cycle	1-cycle	2-cycle	4-cycle	19	25	38/41/44/47
1-cycle	2-cycle	1-cycle	2-cycle	18	22	39/42/45/48
1-cycle	2-cycle	1-cycle	3-cycle	19	23	39/42/45/48
1-cycle	2-cycle	1-cycle	4-cycle	19	23	39/42/45/48
1-cycle	2-cycle	2-cycle	2-cycle	20	24	39/42/45/48
1-cycle	2-cycle	2-cycle	3-cycle	20	24	39/42/45/48
1-cycle	2-cycle	2-cycle	4-cycle	21	25	39/42/45/48
1-cycle	3-cycle	1-cycle	2-cycle	17	22	40/43/46/49
1-cycle	3-cycle	1-cycle	3-cycle	18	23	40/43/46/49
1-cycle	3-cycle	1-cycle	4-cycle	18	23	40/43/46/49
1-cycle	3-cycle	2-cycle	2-cycle	19	24	40/43/46/49
1-cycle	3-cycle	2-cycle	3-cycle	19	24	40/43/46/49
1-cycle	3-cycle	2-cycle	4-cycle	20	25	40/43/46/49
2-cycle	1-cycle	1-cycle	2-cycle	19	22	39/42/45/48
2-cycle	1-cycle	1-cycle	3-cycle	20	23	39/42/45/48
2-cycle	1-cycle	1-cycle	4-cycle	20	23	39/42/45/48
2-cycle	1-cycle	2-cycle	2-cycle	21	24	39/42/45/48
2-cycle	1-cycle	2-cycle	3-cycle	21	24	39/42/45/48
2-cycle	1-cycle	2-cycle	4-cycle	22	25	39/42/45/48
2-cycle	2-cycle	1-cycle	2-cycle	18	22	40/43/46/49
2-cycle	2-cycle	1-cycle	3-cycle	19	23	40/43/46/49
2-cycle	2-cycle	1-cycle	4-cycle	19	23	40/43/46/49
2-cycle	2-cycle	2-cycle	2-cycle	20	24	40/43/46/49
2-cycle	2-cycle	2-cycle	3-cycle	20	24	40/43/46/49

Continued on next page...

Table 24: (continued)

L3-Cache Tag/Snoop RAM Setup Cycle	L3-Cache Tag/Snoop RAM Output Cycle	L3-Cache Data RAM Setup Cycle	L3-Cache Data RAM Output Cycle	L3-Cache Hit Latency	L3-Cache Miss Latency	Modified in Another Core
2-cycle	2-cycle	2-cycle	4-cycle	21	25	40/43/46/49
2-cycle	3-cycle	1-cycle	2-cycle	20	22	41/44/47/50
2-cycle	3-cycle	1-cycle	3-cycle	21	23	41/44/47/50
2-cycle	3-cycle	1-cycle	4-cycle	21	23	41/44/47/50
2-cycle	3-cycle	2-cycle	2-cycle	22	24	41/44/47/50
2-cycle	3-cycle	2-cycle	3-cycle	22	24	41/44/47/50
2-cycle	3-cycle	2-cycle	4-cycle	23	25	41/44/47/50

Note

- The latency assumes that the dependent instruction and the load instruction are dual-issued.
- The latency assumes that the dependent instruction is executed in the late ALU stage.
- The latency assumes that load word (LW) instructions are used.
- The latency assumes that the miss address is the first transfer of a cache line.
- The latency assumes that bus latency is one cycle after the request address is issued.
- The latency assumes that Core Interface is synchronous.
- The latency assumes that Bus Clock is synchronous.
- The latency assumes that the SMEM_CLK and BUS_CLK clock ratio is 1:1.
- The latency assumes that the bloom filter effectively bypasses the synonym check for the VIPT D-Cache.
- The latency of “Modified in Another Core” depends on the number of times the synonym index is looked up.

7.3.2 D-Cache Fill Operations

The D-Cache fill operation starts when a cacheable line is not in the D-Cache. A burst read request for the missed cache line is always sent first to the system bus to minimize the miss latency. The read request will be followed by a burst write request if cache eviction is required.

The fill operation may be aborted by system bus errors. If the bus error is on the critical word, a precise load access fault is triggered for the load instruction causing the cache miss. The bus error on non-critical words does not trigger an exception, but the fill operation will be canceled and the missed line will not be installed into the D-Cache. Store is always non-blocking. Bus errors on store cache miss will trigger bus-write transaction errors.

System bus errors will no longer be reported as precise exceptions if the processor is in the non-blocking mode (mmisc_ctl.NBLD_EN). Instead, *Bus Read/Write Transaction Error Local Interrupts* (mip.BWEI) will be reported. Please see Section 10.1 for details.

In Debug Mode, load/store instructions will minimally affect D-Cache contents. All cache misses will not cause cache replacements, and only dirty bits may be affected by accesses to cache lines that are already in the D-Cache.

7.3.3 D-Cache Eviction Operations

A burst write request will be sent to the next level memory if a dirty line is evicted out of the D-Cache. An imprecise bus-write error exception is triggered if the burst write request encounters system bus errors.

7.3.4 D-Cache Replacement Policy

The D-Cache implements a tree pseudo-LRU replacement policy.

7.3.5 D-Cache Write Buffers

The D-Cache implements three write buffers, each buffer capable of holding a 64-byte line. When the D-Cache evicts a line, the evicted line is temporarily buffered in one of the write buffers before being sent to the next level memory.

When the Write-Around feature is configured, stores to write-no-allocate memory are also buffered in one of the write buffers. Multiple stores can be merged into a single transaction. The buffered line is flushed in any of the following conditions:

- The line is completely filled by subsequent store operations.
- A store operation writes to another line in write-no-allocate memory.
- A load operation accesses the buffered line.
- A `FENCE`, `FENCE.I`, or `SFENCE.VMA` instruction is under execution.
- A `LR`, `SC`, or `AMO` instruction is under execution.
- A CCTL operation is under execution.
- The processor is in debug mode.
- The processor enters WFI mode.
- The D-Cache is disabled (mcache_ctl.DC_EN is cleared).

Write-no-allocate memory consists of:

- A D-Cache miss, which occurs for a store operation when the D-Cache is in the write-no-allocate mode (See Section 10.2).
- PMA of the memory is Memory, Write-back, No-allocate (MTYP=8).
- PMA of the memory is Memory, Write-back, Read-allocate (MTYP=9).

7.4 FENCE and FENCE.I Operations

`FENCE` and `FENCE.I` instructions may affect caches when caches are enabled. If mcache_ctl.IC_EN is 0, FENCE.I instructions will not perform any operation on the I-Cache. Similarly, if mcache_ctl.DC_EN is 0, FENCE.I instructions will not perform any operation on the D-Cache. `FENCE` instructions do not perform any operation on either the I-Cache or D-Cache. The behavior of `FENCE` and `FENCE.I` is summarized in Table 25.

Table 25: Effects of FENCE and FENCE.I Instructions

Cache	FENCE	FENCE.I
I-Cache	None	Invalidates all cache lines
D-Cache	None	Writes back all cache lines

7.5 CCTL Operations

CCTL operations provide direct control to manipulate instruction caches, data caches, or TLB tag/data RAM (cache maintenance operations). They are invoked by writing CCTL commands to the mcctlcommand CSR register. The operations can be grouped into two main types, virtual-address (VA) based or index (IX) based, and they are summarized in Table 26. These two addressing types affect how cache lines or TLB entries are specified for CCTL operations, and how the content of mcctlbeginaddr are interpreted.

Table 26: Addressing Type of CCTL Commands

Type	Usage
IX	Use the content of the mcctlbeginaddr register directly as a (Way, Index), (Way, Index, Double-Word/word), or (Target, Way, Index) pair/tuple to specify a cache line in the cache or TLB entry in the TLB tag/data RAM without going through any translation mechanism. The format is defined in Table 27, Table 28, and Table 29.
VA	Use the content of the mcctlbeginaddr register as a virtual address to access the cache. The virtual address has to go through the same address translation mechanism in the processor pipeline as that used for regular load/store instructions (for the D-Cache) or instruction fetches (for the I-Cache). The specified operation is performed only if the addressed cache line is in the corresponding cache.

Table 27: Index Format for D-Cache Index Type of CCTL Operations

Field	Bit Position	Description
OFFSET	mcctlbeginaddr[A-1:3]	A = log_2(#Double-Words in a Cache Line) + 3
INDEX	mcctlbeginaddr[B-1:A]	B = log_2(Cache Size / #Ways)
WAY	mcctlbeginaddr[C-1:B]	C = Ceiling(log_2(Cache Size))
TARGET	mcctlbeginaddr[C]	This field is only meaningful when performing L1D_IX_RTAG or L1D_IX_WTAG.

Value	Meaning
0	D-Cache primary tag RAM
1	D-Cache duplicate tag RAM

Table 28: Index Format for I-Cache Index Type of CCTL Operations

Field	Bit Position	Description
OFFSET	`mcctlbeginaddr[A-1:2]`	A = log_2(#Words in a Cache Line) + 2

Continued on next page...

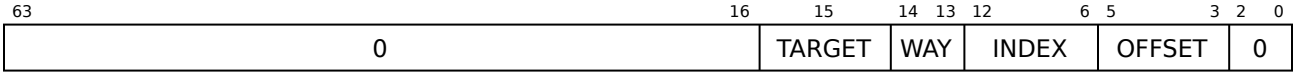
Table 28: (continued)

Field	Bit Position	Description
INDEX	`mcctlbeginaddr[B-1:A]`	B = log_2(Cache Size / #Ways)
WAY	`mcctlbeginaddr[C-1:B]`	C = Ceiling(log_2(Cache Size))

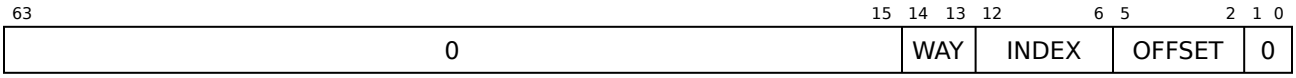
Table 29: Index Format for TLB Index Type of CCTL Operations

Field	Bit Position	Description
INDEX	mcctlbeginaddr[STLB_RAM_AW-1:0]	Index of TLB SRAM
WAY	mcctlbeginaddr[17:16]	Way of TLB SRAM
TARGET	mcctlbeginaddr[27:24]	Target of TLB SRAM

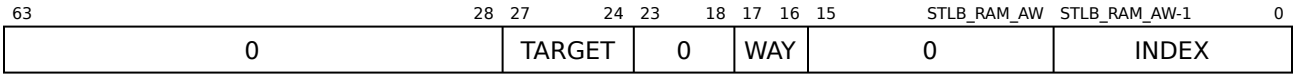
The following diagram shows an example of mcctlbeginaddr for the index type of CCTL cache operations, assuming that cache is 4-way, 32-KiB with 64-byte cache lines for the D-Cache.



The following diagram shows an example of mcctlbeginaddr for the index type of CCTL cache operations, assuming that cache is 4-way, 32-KiB with 64-byte cache lines for the I-Cache.



The following diagram shows an example of mcctlbeginaddr for the index type of CCTL TLB operations.



All available CCTL operations are summarized in Table 128. Their detailed definitions are grouped and described in the following categories.

7.5.1 Invalidating Cache Blocks (L1D_VA_INVAL, L1I_VA_INVAL, L1D_IX_INVAL, L1I_IX_INVAL)

These operations invalidate the specified cache lines. Locked cache lines are unlocked and invalidated.

7.5.2 Writing Back Cache Blocks (L1D_VA_WB, L1D_IX_WB, L1D_WB_ALL)

These operations write the data of the specified cache lines back to the system memory, if the specified cache lines are present in the cache with dirty states. The specified cache lines will still be kept in the cache and locked cache lines remain locked.

7.5.3 Writing Back & Invalidating Cache Blocks (L1D_VA_WBINVAL, L1D_IX_WBINVAL, L1D_WBINVAL_ALL)

These operations write the data of the specified cache lines back to the system memory, if the specified cache lines are present in the cache with dirty states. The specified cache lines will then be invalidated as long as they are valid in the cache, regardless of whether their states are dirty or locked.

7.5.4 Filling and Locking Cache Blocks (L1D_VA_LOCK, L1I_VA_LOCK)

These operations lock the specified cache lines in the cache. The specified cache lines are first brought into the cache if they are not already present in the cache, then the cache lines are locked by setting their lock states. It is not an error to lock an already locked line—the same line is just locked again.

The lock state only affects the cache replacement policy. On cache miss, unlocked lines will be replaced first. When all ways are locked, the missed line is not allocated in the cache.

The status of lock operations are written to the mcctldata register:

- A value of 1 indicates that the lock operation finished successfully;
- A value of 0 indicates that the lock operation was aborted or failed.
  - Locking an address in device or non-cacheable memory.
  - Locking an address in local memory (ILM/DLM).
  - Locking a line when all ways are locked, and the line is not present in the cache.

7.5.5 Unlocking Cache Blocks (L1D_VA_UNLOCK, L1I_VA_UNLOCK)

These operations clear the lock state of the specified cache lines if the specified cache lines are present in the cache.

7.5.6 Reading Tag Data from Caches (L1D_IX_RTAG, L1I_IX_RTAG)

These operations read the contents of the tag part of the target cache line into the mcctldata register. The target cache line is specified in the mcctlbeginaddr register by its way and index information. Additionally, these operations observe DC_RWECC/IC_RWECC settings in mcache_ctl to read the corresponding ECC/Parity codes in the tag RAM to mecc_code.

7.5.7 Reading Data from D-Cache (L1D_IX_RDATA)

This operation reads an 8-byte data from a cache line into the `mcctldata` register. The target cache line is specified in the `mcctlbeginaddr` register by its way, index, and doubleword information. Additionally, this operation observes DC_RWECC settings in mcache_ctl to read the corresponding ECC codes in the data RAM to mecc_code.

7.5.8 Reading Data from I-Caches (L1I_IX_RDATA)

This operation reads a 4-byte data from a cache line into the mcctldata register. The target cache line is specified in the mcctlbeginaddr register by its way, index, and word information. Additionally, this operation observes IC_RWECC settings in mcache_ctl to read the corresponding Parity codes in the data RAM to mecc_code.

7.5.9 Writing Tag Data to Caches (L1D_IX_WTAG, L1I_IX_WTAG)

These operations write the contents of the mcctldata register to the tag part of the target cache line. The target cache line is specified in the mcctlbeginaddr register by its way and index information. Additionally, these operations observe DC_RWECC/IC_RWECC settings in mcache_ctl to write the mecc_code ECC/Parity code to the corresponding tag RAM.

Note

Writing unexpected tag data to a cache might lead to UNPREDICTABLE behavior when the cache is enabled. It is recommended to clear TAG SRAM before enabling the cache.

7.5.10 Writing Data to D-Cache (L1D_IX_WDATA)

This operation writes an 8-byte data in the `mcctldata` register into the target cache line. The target cache line is specified in the `mcctlbeginaddr` register by its way, index, and doubleword information. Additionally, this operation observes DC_RWECC settings in mcache_ctl to write the mecc_code ECC code to the corresponding data RAM.

7.5.11 Writing Data to I-Caches (L1I_IX_WDATA)

This operation writes a 4-byte data in the mcctldata register into the target cache line. The target cache line is specified in the mcctlbeginaddr register by its way, index, and word information. Additionally, this operation observes IC_RWECC settings in mcache_ctl to write the mecc_code Parity code to the corresponding data RAM.

7.5.12 Invalidating All Cache Blocks (L1D_INVAL_ALL)

This operation invalidates all valid lines of the D-Cache. Locked cache lines are unlocked and invalidated.

7.5.13 Writing Back All Cache Blocks (L1D_WB_ALL)

This operation writes the data of all dirty cache lines of the D-Cache back to the system memory. All cache lines will still remain in the D-Cache and locked lines remain locked.

7.5.14 Writing Back & Invalidating All Cache Blocks (L1D_WBINVAL_ALL)

This operation writes the data of all dirty cache lines of the D-Cache back to the system memory and all valid cache lines will be invalidated, including locked and/or clean cache lines.

7.5.15 Reading Tag/Data from TLB SRAMs (TLB_IX_RTAG, TLB_IX_RDATA)

These operations read the contents of the tag/data part of the target TLB entry into the `mcctldata` register. The format of the contents in `mcctldata` is defined in Section 16.10.14. The target TLB entry is specified in the `mcctlbeginaddr` register by its target, way, and index information. Additionally, these operations observe TLB_RWECC settings in `mcache_ctl` to read the corresponding ECC/Parity codes in the tag/data RAM to `mecc_code`.

7.5.16 Writing Tag/Data to TLB SRAMs (TLB_IX_WTAG, TLB_IX_WDATA)

These operations write the contents of the mcctldata register to the tag/data part of the target TLB entry. The target TLB entry is specified in the mcctlbeginaddr register by its target, way, and index information. Additionally, these operations observe TLB_RWECC settings in mcache_ctl to write the mecc_code ECC/Parity code to the corresponding tag/data RAM.

7.6 Supervisor/User CCTL Operations

CCTL operations are available to Supervisor/User-mode software under the control of the mcache_ctl.CCTL_SUEN control bit. These operations are triggered in both modes by accessing ucctlbeginaddr, ucctlcommand, and scctldata registers, while mcache_ctl.CCTL_SUEN controls access permission to these registers. When CCTL_SUEN is 0, accessing them in Supervisor and User modes would cause illegal instruction exceptions. It should be set to 1 to enable Supervisor (and User) CCTL operations.

All CCTL operations can be made available to Supervisor-mode software while only four CCTL operations listed in the table below are available to User-mode software.

Table 30: User CCTL Operations

Value		Command	Type	Exception Entry
0	0b00_000	L1D_VA_INVAL	VA	Store related Fault
1	0b00_001	L1D_VA_WB	VA	Store related Fault
2	0b00_010	L1D_VA_WBINVAL	VA	Store related Fault
8	0b01_000	L1I_VA_INVAL	VA	Store related Fault

7.7 Private L2-Cache

The private L2-cache is not available in this release.

8 Level-3 Cache

8.1 Introduction

A level-3 cache (L3-Cache) improves system performance by providing a larger amount of cache line entries and reasonable access latency. It is an optional feature and can be deconfigured by setting the cache size to 0. Its features include:

- Configurable cache size: 128 KiB – 32 MiB
- 64 bytes cache line size
- 12-way or 16-way set-associative
- Non-inclusive and non-exclusive with private caches of each core
- Support ECC protection on L3-Cache tag and data RAMs and snoop filter RAMs
- Multi-bank structure of L3-Cache for improving physical implementation
- Configurable number of cache setup/output cycles
- Hardware stride prefetcher for data access and instruction fetch
- Cache control (CCTL) operations for cache maintenance
- Way allocation mask for each core and IOCP

8.2 L3-Cache Multi-Bank Structure

L3-Cache is partitioned into two or four banks to improve bandwidth and frequency, depending on the option Number of L3-Cache Banks. These banks can be accessed in parallel. Each bank contains data RAMs, tag RAMs, and associated logic. The interfaces of the L3-Cache are summarized in Table 31.

Table 31: L3-Cache Interfaces

Interface	Number of L3C Banks Is 2	Number of L3C Banks Is 4
bank0_icf	Memory accesses whose address[6] is 0x0	Memory accesses whose address[7:6] is 0x0
bank1_icf	Memory accesses whose address[6] is 0x1	Memory accesses whose address[7:6] is 0x1
bank2_icf	N/A	Memory accesses whose address[7:6] is 0x2
bank3_icf	N/A	Memory accesses whose address[7:6] is 0x3
mmr	L3-Cache register accesses	L3-Cache register accesses
sys	Memory accesses to the L3 memory subsystem	Memory accesses to the L3 memory subsystem

8.3 L3-Cache Prefetch

L3-Cache implements a hardware prefetch engine, and the key features of the L3-Cache prefetch include:

- Non-unit stride detection for data access stream within a 4K page
- Consecutive instruction fetch within a 4K page
- Programmable prefetch depth
  - 0, 1, 2, 3 prefetch depth for instruction fetch
  - 0, 2, 4, 8 prefetch depth for data access
- Programmable threshold of available handling register count for prefetch throttling

8.4 L3-Cache Control Operation

L3-Cache provides a set of cache control (CCTL) operations, which are divided into the following types:

- IX type (index)
- PA type (physical address)
- TGT type (target)
- Flush type
- F-Flush type (Filtered Flush)

Table 32 lists the supported L3-Cache CCTL operations. These CCTL operations are performed/controlled by writing the related CCTL control registers. The CCTL Status Register is responsible for reporting the status of these CCTL operations.

8.4.1 L3-Cache Control Operation

Table 32: Supported L3-Cache CCTL Operations

OPCODE	Mnemonics	Type	Operation
0b00_000	L3_IX_INVAL	IX	Invalidates an L3-Cache entry
0b00_001	L3_IX_WB	IX	Writes back an L3-Cache entry
0b00_010	L3_IX_WBINVAL	IX	Writes back and invalidates an L3-Cache entry
0b00_100	SNP_IX_INVAL	IX	Invalidates a snoop filter entry
0b00_101	SNP_IX_WB	IX	Writes back a snoop filter entry
0b00_110	SNP_IX_WBINVAL	IX	Writes back and invalidates an snoop filter entry
0b01_000	L3_PA_INVAL	PA	Invalidates an L3-Cache entry
0b01_001	L3_PA_WB	PA	Writes back an L3-Cache entry
0b01_010	L3_PA_WBINVAL	PA	Writes back and invalidates an L3-Cache entry
0b10_000	L3_TGT_WRITE	TGT	Writes an L3-Cache entry
0b10_001	L3_TGT_READ	TGT	Reads an L3-Cache entry
0b10_100	SNP_TGT_WRITE	TGT	Writes a snoop filter entry
0b10_101	SNP_TGT_READ	TGT	Reads a snoop filter entry

Continued on next page...

Table 32: (continued)

OPCODE	Mnemonics	Type	Operation
0b10_010	L3_WBINVAL_ALL	Flush	Writes back and invalidates all L3-Cache entries (Cache flush)
0b10_011	L3_FILTERED_FLUSH	F-Flush	Writes back and invalidates selected L3-Cache entries

`L3_TGT_WRITE` and `L3_TGT_READ` directly access the raw data of tag or data RAM entries. If the L3-Cache is configured to support ECC, the ECC code for each RAM entry will also be written to or read through the `TGT_ECC` register. As a result, for raw data, ECC checking is not performed during CCTL operations when ECC is enabled. However, if `L3_TGT_WRITE` writes a corrupted ECC code to a tag or a valid data RAM entry, the error will be detected during normal cache operations when this entry is used and ECC is enabled.

8.4.2 Cache Line State Transition

The five types of L3-Cache CCTL operations have distinct behaviors. The `IX`, `PA`, `Flush`, and `F-Flush` operations always access the L3-Cache tag SRAMs to determine the line state in the D-Cache, regardless of whether the L3-Cache is enabled. When data coherence handling is necessary, the L3-Cache writes back or invalidates the copies in the D-Cache. On the other hand, the `TGT` type of L3-Cache CCTL operations directly access the L3-Cache RAMs and bypass the coherence management logic. Table 33 lists the line state transitions of the D-Cache and L3-Cache during L3-Cache CCTL operations.

Note that none of the L3-Cache CCTL operations affect copies in the I-Cache.

Table 33: Cache Line State Transition of L3-Cache CCTL

Operation	Cache Line State Before Operation		Cache Line State After Operation		Transaction to L3 Memory
	D-Cache	L3-Cache	D-Cache	L3-Cache	
`L3_IX_WB`, `L3_PA_WB`	Invalid	Invalid	Invalid	Invalid	None
	Invalid	Clean	Invalid	Clean	None
	Invalid	Dirty	Invalid	Clean	Write back the dirty data
	Shared	Invalid	Shared	Invalid	None
	Shared	Clean	Shared	Clean	None
	Exclusive	Invalid	Shared	Invalid	None
	Exclusive	Clean	Shared	Clean	None
	Modified	Invalid	Shared	Invalid	Write back the dirty data
`L3_IX_INVAL`, `L3_PA_INVAL`	Invalid, Shared, Exclusive, or Modified	Invalid, Clean, or Dirty	Invalid	Invalid	None
`L3_IX_WBINVAL`, `L3_PA_WBINVAL`, `L3_WBINVAL_ALL`, `L3_FILTERED_FLUSH`	Invalid	Invalid	Invalid	Invalid	None
	Invalid	Clean	Invalid	Invalid	None
	Invalid	Dirty	Invalid	Invalid	Write back the dirty data
	Shared	Invalid	Invalid	Invalid	None
	Shared	Clean	Invalid	Invalid	None
	Exclusive	Invalid	Invalid	Invalid	None
	Exclusive	Clean	Invalid	Invalid	None
	Modified	Invalid	Invalid	Invalid	Write back the dirty data

8.4.3 L3-Cache Filtered Flush CCTL Operation

The L3-Cache Filtered Flush (F-Flush) CCTL operation allows selective flushing of L3-Cache contents. Since the L3-Cache is inclusive, flushing an address from the L3-Cache also flushes the same address from the D-Cache.

Four L3-Cache CCTL F-Flush Range Registers (0, 1, 2, and 3) are used to specify the operation (polarity), the address mode (MODE), and the physical address (PA) range.

- When the MODE field of F-Flush Range Registers are set to `RANGE`, the range is specified through the PA fields of register pairs: Register 0 and Register 1 form one begin/end pair, respectively, and Register 2 and Register 3 form another. The end address of the range is not inclusive to the specified range. If the begin address is larger than the end address, the range is ineffective.
- When the MODE field of Register 0 or Register 2 is `NAPOT` (naturally aligned power of two ranges), the PA fields specify the desired range, as listed in Table 34. Only certain combinations of MODE fields are valid, as specified in Table 35 and Table 36.

When a F-Flush CCTL operation is completed, the MODE fields of F-Flush Range registers will be reset to 0 (Disable).

By default, all cache lines within the specified range will be flushed, while other cache lines in the L3-Cache will remain intact. On the other hand, setting the POLARITY bit of L3-Cache CCTL F-Flush Range Register 0 preserves the specified address range and flushes all other cache lines in the L3-Cache.

All L3-Cache CCTL F-Flush Range Registers are shared among all CPU cores. As such, the F-Flush CCTL operation software sequences must be protected by a mutex lock to prevent race conditions and ensure operation integrity. An example sequence is demonstrated in the following pseudo-code.

```
l3c_filtered_flush_cctl_operation (
    hart_id,
    OPCODE_L3_FILTERED_FLUSH, // 0b10_011
    mode,                     // Mode of address filter
                              //     1: Begin/end pair range
                              //     3: Naturally aligned power of two range
    polarity                  // Polarity of address filter
                              //     0: Flush the specified range
                              //     1: Preserve the specified range
) {
    mutex_lock();
    /* Critical Section Begin */
    set_specified_range_and_polarity(mode, polarity);
    perform_l3c_cctl_command(hart_id, OPCODE_L3_FILTERED_FLUSH);
    wait_until_l3c_cctl_command_is_completed(hart_id);
    /* Critical Section End */
    mutex_unlock()
}
```


Table 34: NAPOT Range Encoding for L3-Cache F-Flush CCTL Operations

Bit 0 Position in PA	PA	Description
1	0	64-byte range starting at (PA << 5)
2	01	128-byte range starting at ((PA & ~0b01) << 5)
3	011	256-byte range starting at ((PA & ~0b011) << 5)
4	0111	512-byte range starting at ((PA & ~0b0111) << 5)
..	...	...
60	1111...1111	all cache lines in L3-Cache

Table 35: Valid MODE Settings for L3-Cache CCTL F-Flush Range Register 0 and 1

F-Flush Range Register 0	F-Flush Range Register 1	Specified Ranges
Disabled	Don't Care	Disabled
RANGE	RANGE	1 Begin/End range
NAPOT	Disabled	1 NAPOT range
NAPOT	NAPOT	2 NAPOT ranges

Table 36: Valid MODE Settings for L3-Cache CCTL F-Flush Range Register 2 and 3

F-Flush Range Register 2	F-Flush Range Register 3	Specified Ranges
Disabled	Don't Care	Disabled
RANGE	RANGE	1 Begin/End range
NAPOT	Disabled	1 NAPOT range
NAPOT	NAPOT	2 NAPOT ranges

8.4.4 Processing Cycles of L3-Cache Flush CCTL Operation

Table 37: Processing Cycles of L3-Cache Flush Operation in 2-Bank L3-Cache

Cache Line State		Bus Latency			
D-Cache	**L3-Cache**	**1**	**10**	**60**	**200**
Invalid	**Clean**	TBD	TBD	TBD	TBD
Exclusive	**Clean**	TBD	TBD	TBD	TBD
Invalid	**Dirty**	TBD	TBD	TBD	TBD
Modified	**Invalid**	TBD	TBD	TBD	TBD

Table 38: Processing Cycles of L3-Cache Flush Operation in 4-Bank L3-Cache

Cache Line State		Bus Latency			
D-Cache	**L3-Cache**	**1**	**10**	**60**	**200**
Invalid	**Clean**	TBD	TBD	TBD	TBD
Exclusive	**Clean**	TBD	TBD	TBD	TBD
Invalid	**Dirty**	TBD	TBD	TBD	TBD
Modified	**Invalid**	TBD	TBD	TBD	TBD

Note

The number assumes the following conditions:

- **D-Cache Size** is 32 KiB.
- **L3C Cache Size** is 128 KiB.
- **Core Interface** is asynchronous.
- **Bus Data Width** is 512
- CORE_CLK and BUS_CLK clock ratio is 1:1.
- L3C Tag RAM Output Cycle is 1 cycle.
- L3C Tag RAM Setup Cycle is 1 cycle.
- L3C Data RAM Output Cycle is 2 cycles.
- L3C Data RAM Setup Cycle is 1 cycle.
- All outstanding transactions from the MEM interface are accepted.

8.5 L3-Cache Way Allocation

When a cache miss occurs, the missed line is allocated to a specific way. The L3-Cache uses a pseudo-random replacement policy and Way Allocation Mask registers (WAYMASK) to determine the way for allocation. Each core and IOCP operate in an individual domain, and each domain has a programmable Way Allocation Mask register. These registers specify which ways can be allocated or replaced by accesses from the bus manager in the corresponding domain. Domain numbers are listed in Table 39.

Table 39: Allocation Domain Assignment

Domain Number	1 Core + IOCP	2 Cores + IOCP	3 Cores + IOCP	4 Cores + IOCP	5 Cores + IOCP	6 Cores + IOCP	7 Cores + IOCP	8 Cores + IOCP
0	Core 0	Core 0	Core 0	Core 0	Core 0	Core 0	Core 0	Core 0
1	IOCP	Core 1	Core 1	Core 1	Core 1	Core 1	Core 1	Core 1
2		IOCP	Core 2	Core 2	Core 2	Core 2	Core 2	Core 2
3			IOCP	Core 3	Core 3	Core 3	Core 3	Core 3
4				IOCP	Core 4	Core 4	Core 4	Core 4
5					IOCP	Core 5	Core 5	Core 5
6						IOCP	Core 6	Core 6
7							IOCP	Core 7
8								IOCP

8.6 L3-Cache Error Handling

There are two sources of L3-Cache requests:

- Upstream bus managers: The requests include core fetches, core load/store operations, and IOCP accesses.
- L3-Cache controller: The requests include prefetches, CCTL operations, and cache line write-back to L3.

When handling a request, the L3-Cache may encounter any of the following errors:

- Correctable RAM errors. These are silently corrected without any logging.
- Uncorrectable RAM errors
- Bus errors
- Cache coherency protocol errors
- WAYMASK set to an invalid value for a request

Errors can be synchronous or asynchronous. Synchronous errors are reported in the upstream response of an upstream request, while asynchronous errors are reported through `l3c_err_int`.

A core ignores errors of speculative accesses (e.g., instruction prefetches and data prefetches). For demanded accesses, the following errors are reported by traps:

- Load access faults
- Store/AMO access faults
- Instruction access faults
- Local interrupts (mie.BWEI)

The IOCP reports SLVERR or DECERR errors through `iocp0_bresp` and `iocp0_rresp`.

Asynchronous errors include:

- A CCTL operation encounters uncorrectable RAM errors (RAM error).
- A CCTL operation encounters bus errors (Bus error).
- A CCTL operation probes the D-Cache when D-Cache coherency is disabled (Probe error).
- The D-Cache writes back a line not in the L3-Cache (Release error).
- The L3-Cache sends an L3 request to write back a line, and the response has bus errors (Bus error).

When an asynchronous error occurs, the Asynchronous Error Register and Error Register are updated. The error status can be cleared by writing any value to the corresponding field in the Asynchronous Error Register.

Non-Repairable Soft Faults

A non-repairable soft fault is resolved differently depending on which cache RAM detects the error.

When an asynchronous non-repairable soft fault (a 2-bit error) is detected in the tag RAM or data RAM, the error is reported in the Error Register with the ERRTYPE field set to 0x0, and the `cm_async_error` interrupt pin is asserted.

If the error is detected in the snoop filter, the snoop filter enters broadcast mode by deasserting the SFEN bit in the Control Register 1. The error is also reported in the Error Register with the ERRTYPE field set to 0x0, and the `cm_async_error` interrupt pin is asserted. To handle this error, software should read the Error Register, resolve, and acknowledge the error by writing to the Asynchronous Error Register. Then, re-enable the snoop filter as described in the Enabling and Disabling Snoop Filter section.

Hard Faults

If a cache error is found consecutively at the *same* cache location, it is classified a hard fault. Hard fault handling is the same as for a non-repairable soft fault, except it is reported in the Error Register with the ERRTYPE field set to 0x3.

8.7 L3-Cache Disabled Behaviors

When the L3-Cache is disabled (SCEN = 0 or CEN = 0), regular memory transactions will not affect the shared cache. In contrast, L3-Cache CCTL commands can continue to operate on the L3-Cache.

When the Snoop Filter is disabled (SFEN = 0 or CEN = 0), regular memory transactions, CMO transactions and IX/PA CCTL commands will continue to operate but at the cost of lower performance. Furthermore, L3-CCTL WBINVAL_ALL, INVAL_ALL, and FFLUSH commands will not flush the upstream caches; software should ensure the upstream caches are flushed through the use of local CCTL commands.

8.8 L3-Cache Programming Guide

8.8.1 L3-Cache Initialization

When the L3-Cache exits the reset state, its tag RAMs are initialized. During this initialization process, requests to the L3-Cache are blocked. The `INITSTATUS` field in the Control Register 0 indicates whether the initialization is complete. To disable the initialization, assert the `l3c_disable_init` signal.

8.8.2 L3-Cache Enablement

The enablement of the L3-Cache is controlled by the `CEN` field of the Control Register 0. It is enabled by default but can be disabled by clearing this field. Because the L3-Cache implements an inclusive policy, it should be enabled before the D-Cache is enabled.

8.8.3 L3-Cache Registers

L3-Cache controller registers are mapped to a 64KiB or 1MiB memory space. For more details, see AndesCore AX46MPV Integration Guide (IG094).

Bus transactions to this region should follow the rules below:

- Use a single transfer.
- Access size must be 8 bytes or smaller
- The cacheability must be device.

Using other burst types of transactions to access this region may result in UNPREDICTABLE behavior.

8.8.4 Register Type

Term	Description
IM	Implementation dependent/determined
RO	Read-Only register/field. Any software write to RO registers/fields will be silently ignored by hardware.
RW	Read/Write register/field
W1C	Write-One-Clear register/field. When written 1, the corresponding bit of the register/field is cleared to 0.

Continued on next page...

Term	Description
WC	Write-Clear register/field. When any value is written, the register/field is cleared to 0.

8.8.5 Summary of Registers

The memory map of the registers is specified by **Number of Processor Cores**.

Table 40: L3-Cache Registers

Address Offset	Description	Section
0x0000 – 0x0007	Configuration Register	Section 8.8.6
0x0008 – 0x000f	Control Register 0	Section 8.8.7
0x0010 – 0x0017	HPM Control Register 0	Section 8.8.8
0x0030 – 0x0037	Asynchronous Error Register	Section 8.8.9
0x0038 – 0x003f	Error Register	Section 8.8.10
0x0040 – 0x0047	CCTL Command Register 0	Section 8.8.11
0x0048 – 0x004f	CCTL Access Line Register 0	Section 8.8.12
0x0080 – 0x0087	CCTL Status Register 0	Section 8.8.13
0x0090 – 0x00c8	CCTL TGT Data Register 0 to 7	Section 8.8.14
0x00d0 – 0x00d7	CCTL TGT ECC Code Register	Section 8.8.15
0x00f0 – 0x00f8	Control Register 1	Section 8.8.16
0x0200 – 0x0207	HPM Counter Register 0	Section 8.8.17
0x0208 – 0x020f	HPM Counter Register 1	Section 8.8.17
0x0300 – 0x0307	Way Allocation Mask Register 0	Section 8.8.18
0x0308 – 0x030f	Way Allocation Mask Register 1	Section 8.8.18
0x0310 – 0x0317	Way Allocation Mask Register 2	Section 8.8.18
0x0318 – 0x031f	Way Allocation Mask Register 3	Section 8.8.18
0x0320 – 0x0327	Way Allocation Mask Register 4	Section 8.8.18
0x0328 – 0x032f	Way Allocation Mask Register 5	Section 8.8.18
0x0330 – 0x0337	Way Allocation Mask Register 6	Section 8.8.18
0x0338 – 0x033f	Way Allocation Mask Register 7	Section 8.8.18
0x0340 – 0x0347	Way Allocation Mask Register 8	Section 8.8.18
0x0500 – 0x0507	CCTL F-Flush Status Register	Section 8.8.19
0x0508 – 0x050f	CCTL F-Flush Range Register 0	Section 8.8.20
0x0510 – 0x0517	CCTL F-Flush Range Register 1	Section 8.8.20
0x0518 – 0x051f	CCTL F-Flush Range Register 2	Section 8.8.20
0x0520 – 0x0527	CCTL F-Flush Range Register 3	Section 8.8.20

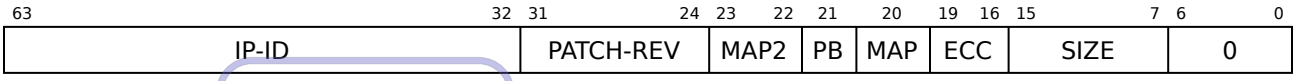
Continued on next page...

Table 40: (continued)

Address Offset	Description	Section
0x0580 – 0x0587	SW Prefetch Status Register	Section 8.8.21
0x0588 – 0x058f	SW Prefetch Control Register	Section 8.8.22
0x0590 – 0x0597	SW Prefetch Size Register	Section 8.8.23
0x1040 – 0x1047	CCTL Command Register 1	Section 8.8.11
0x1048 – 0x104f	CCTL Access Line Register 1	Section 8.8.12
0x1080 – 0x1087	CCTL Status Register 1	Section 8.8.13
0x2040 – 0x2047	CCTL Command Register 2	Section 8.8.11
0x2048 – 0x204f	CCTL Access Line Register 2	Section 8.8.12
0x2080 – 0x2087	CCTL Status Register 2	Section 8.8.13
0x3040 – 0x3047	CCTL Command Register 3	Section 8.8.11
0x3048 – 0x304f	CCTL Access Line Register 3	Section 8.8.12
0x3080 – 0x3087	CCTL Status Register 3	Section 8.8.13
0x4040 – 0x4047	CCTL Command Register 4	Section 8.8.11
0x4048 – 0x404f	CCTL Access Line Register 4	Section 8.8.12
0x4080 – 0x4087	CCTL Status Register 4	Section 8.8.13
0x5040 – 0x5047	CCTL Command Register 5	Section 8.8.11
0x5048 – 0x504f	CCTL Access Line Register 5	Section 8.8.12
0x5080 – 0x5087	CCTL Status Register 5	Section 8.8.13
0x6040 – 0x6047	CCTL Command Register 6	Section 8.8.11
0x6048 – 0x604f	CCTL Access Line Register 6	Section 8.8.12
0x6080 – 0x6087	CCTL Status Register 6	Section 8.8.13
0x7040 – 0x7047	CCTL Command Register 7	Section 8.8.11
0x7048 – 0x704f	CCTL Access Line Register 7	Section 8.8.12
0x7080 – 0x7087	CCTL Status Register 7	Section 8.8.13

8.8.6 Configuration Register

Offset: 0x0000



This register indicates the cache size, ECC type, and version of the L3-Cache implementation.

Field Name	Bits	Description	Type	Reset
SIZE	[15:7]	L3-Cache size in KiB	RO	IM

Value	Size
0x000	0 KiB
0x001	128 KiB
0x002	256 KiB
0x004	512 KiB
0x008	1 MiB
0x00c	1.5 MiB
0x010	2 MiB
0x018	3 MiB
0x020	4 MiB
0x030	6 MiB
0x040	8 MiB
0x060	12 MiB
0x080	16 MiB
0x0c0	24 MiB
0x100	32 MiB
Others	Reserved

Field Name	Bits	Description	Type	Reset
ECC	[19:16]	L3-Cache ECC type	RO	IM

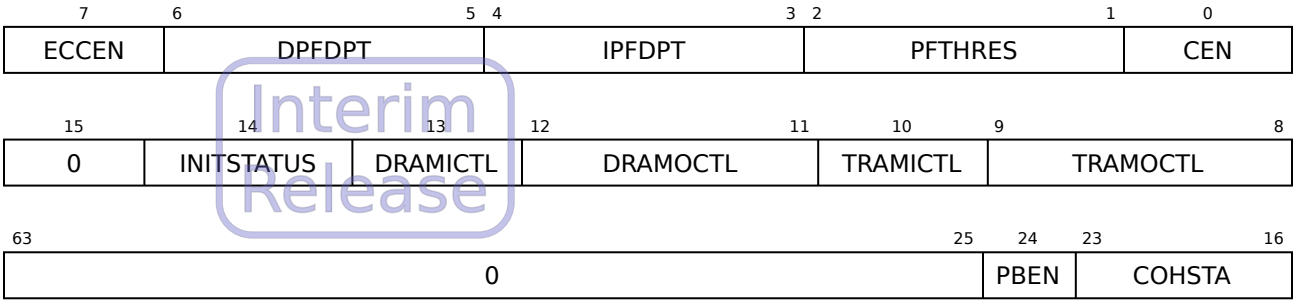
Value	ECC Type
0x0	No protection
0x1	One-bit error correction and two-bit error detection
Others	Reserved

Continued on next page...

Field Name	Bits	Description	Type	Reset
MAP	[20]	Memory map **Value** / **Meaning** 0x00: V0 map for **Number of Processor Cores** 1 – 4 (Not supported) 0x01: V1 map for **Number of Processor Cores** 5 – 8	RO	1
PB	[21]	Prefetch buffer functionality **Value** / **Meaning** 0x0: Prefetch buffer absent 0x1: Prefetch buffer present	RO	IM
MAP2	[23:22]	Memory map 2 **Value** / **Definition** 0x0: Reserved 0x1: V1 map is extended to support up to 16 cores, with additional registers for NCECM500. The MAP field must be 0x1. 0x2: Reserved 0x3: Reserved	RO	1
PATCH-REV	[31:24]	Design patch revision	RO	0x0
IP-ID	[63:32]	Indicates the IP ID and the IP revision. Internal use only.	RO	0x0C105010

8.8.7 Control Register 0

Offset: 0x0008



This register controls various functions of the L3-Cache.

Field Name	Bits	Description	Type	Reset
CEN	[0]	L3-Cache and Snoop Filter control **Value** / **Meaning** 0 — Disable the L3-Cache and Snoop Filter 1 — Enable the L3-Cache and Snoop Filter	RW	1
PFTHRES	[2:1]	Prefetch threshold value. To avoid excessive use of L3-Cache handling registers, prefetching can be throttled based on their occupancy status. **Value** / **Meaning** 0 — Balanced prefetching: Roughly share resources such that prefetches and demand-requests have similar throughput. 1 — Balanced prefetching: Same behavior as *0*. 2 — Opportunistic prefetching: Limits prefetching to minimize its impact on critical code performance. 3 — Aggressive prefetching: Prefetching does not stop and may degrade performance of demand requests or critical code.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
IPFDPT	[4:3]	Instruction prefetch depth. This field indicates the depth of L3 instruction prefetch. It is the number of prefetch requests ahead of the required request stream. **Value** / **Meaning** 0 — 0 request (disable L3 instruction prefetch) 1 — 1 request 2 — 2 requests 3 — 3 requests	RW	0
DPFDPT	[6:5]	Data prefetch depth. This field indicates the depth of L3 data prefetch. It is the number of prefetch requests ahead of the required request stream. **Value** / **Meaning** 0 — 0 request (disable L3 data prefetch) 1 — 2 requests 2 — 4 requests 3 — 8 requests	RW	0
ECCEN	[7]	ECC control **Value** / **Meaning** 0 — Disable L3 ECC 1 — Enable L3 ECC **Note** This field is only meaningful when the L3-Cache Soft Error Protection configuration option is set to “ecc”.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
TRAMOCTL	[9:8]	Tag RAM output cycle	RO	IM

Value	Meaning
0	1 cycle
1	2 cycles
2	3 cycles
Others	Reserved

TRAMICTL	[10]	Tag RAM setup cycle	RO	IM

Value	Meaning
0	1 cycle
1	2 cycles

DRAMOCTL	[12:11]	Data RAM output cycle	RO	IM

Value	Meaning
1	2 cycles
2	3 cycles
3	4 cycles

DRAMICTL	[13]	Data RAM setup cycle	RO	IM

Value	Meaning
0	1 cycle
1	2 cycles

INITSTATUS	[14]	Self-initialization status	RO	1

Value	Meaning
0	Initialization done
1	Under initialization

COHSTA	[23:16]	Per core coherent status. When asserted, the corresponding core (from 0 to 7) is in coherent mode.	RO	0

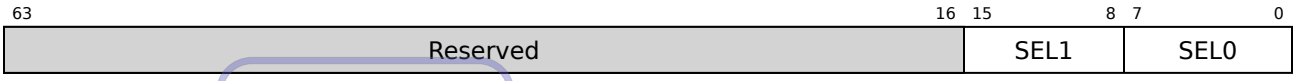
Continued on next page...

Field Name	Bits	Description	Type	Reset
PBEN	[24]	Prefetch buffer control. When this field is enabled and a cache line is fetched from a 128-byte aligned address, its subsequent cache line will be prefetched from the sequential address.	RW	IM

Value	Meaning
0	Disable
1	Enable

8.8.8 HPM Control Register 0

Offset: 0x0010



This register selects events to be monitored by performance counters. It is only present when the number of configured L3-Cache Hardware Performance Monitor is greater than 0.

Field Name	Bits	Description	Type	Reset
SEL0	[7:0]	Monitored event selection of performance counter 0	RW	0xff
SEL1	[15:8]	Monitored event selection of performance counter 1	RW	0xff

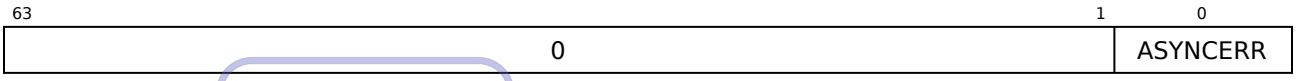
Table 41 lists the supported events of performance counters.

Table 41: Monitored Event Definitions of the L3C Performance Counter

Select	Event Description
0x00	Total access count
0x01	L3-Cache access count
0x02	L3-Cache miss count
0x03–0xff	Reserved

8.8.9 Asynchronous Error Register

Offset: 0x0030

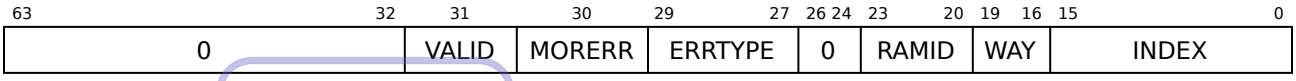


This register is write-clear and indicates the occurrence of asynchronous errors in the L3-Cache.

Field Name	Bits	Description	Type	Reset
ASYNCERR	[0]	Indicates if any asynchronous error has happened	WC	0x0

8.8.10 Error Register

Offset: 0x0038

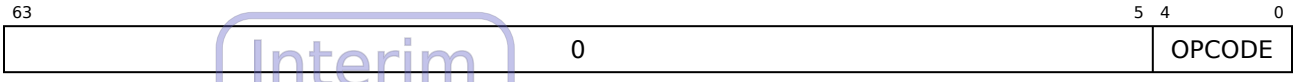


This register holds information about the first asynchronous error. If more than one errors occur, the MORERR bit is set. The INDEX and WAY fields are only meaningful if the first error is a RAM error. Otherwise, these fields are "Don't Care". Writing any value to this register clears the whole register to zero.

Field Name	Bits	Description	Type	Reset
INDEX	[15:0]	Index address of the first error	WC	0
WAY	[19:16]	Way of the RAM, where the first error occurred	WC	0
RAMID	[23:20]	RAM ID, where the first error occurred **Value** / **Description** 0b0000 / L3 tag RAM 0b0001 / L3 data RAM 0b0010 / snoop filter RAM Others / Reserved	WC	0
ERRTYPE	[29:27]	The first error type **Value** / **Description** 0b000 / Soft fault RAM error 0b001 / Release error 0b010 / Probe error 0b011 / Hard fault RAM error detected 0b100 / Bus error 0b101 / Corrupt data received from an external source 0b111 / Received illegal or unsupported transactions Others / Reserved	WC	0
MORERR	[30]	More error indicator, set when more errors are reported.	WC	0
VALID	[31]	Valid bit, set to 1 when an error is reported.	WC	0

8.8.11 CCTL Command Registers

Offset: 0x0040, 0x0050, 0x0060, 0x0070, 0x1040, 0x2040, 0x3040, 0x4040, 0x5040, 0x6040, 0x7040



Writing to CCTL Command Register n (where $n = 0 - 7$) triggers a CCTL operation, with the operation specified by the written value and the line specified by CCTL Access Line Register n. Each register corresponds to one CPU core, so each core should only use its respective registers. If CPU core n does not exist, CCTL Command Register n is reserved. The L3-Cache uses a round-robin strategy to determine the execution order when two or more CCTL Command Registers are written at the same time.

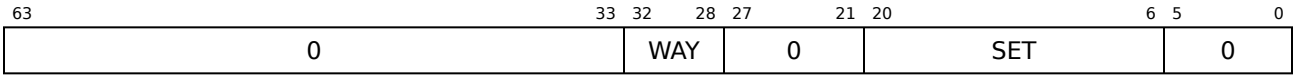
Field Name	Bits	Description	Type	Reset
OPCODE	[4:0]	CCTL operation	RW	0

8.8.12 CCTL Access Line Registers

Offset: 0x0048, 0x1048, 0x2048, 0x3048, 0x4048, 0x5048, 0x6048, 0x7048

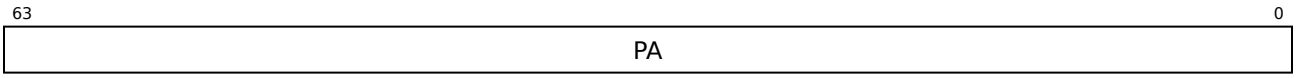
These registers are used to specify the physical address, set, way, and/or RAM ID for CCTL operations. Each register corresponds to one CPU core, so each core should only use its respective register. If CPU core *n* does not exist, CCTL Access Line Register *n* is reserved.

The interpretation of these registers is different based on the CCTL operation type. The register format for IX-type CCTL operations is:



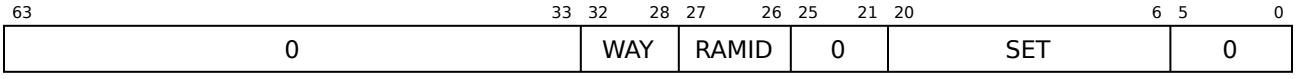
Field Name	Bits	Description	Type	Reset
SET	[19:6]	Cache set of the CCTL operation	RW	0
WAY	[32:28]	Cache way of the CCTL operation. This field is interpreted dynamically based on the configured number of ways for the Tag/Data RAM and Snoop RAM. When the number of ways is 16 or less, this field is only [31:28]. Otherwise, this field is [32:28].	RW	0

The register format for PA-type CCTL operations is:



Field Name	Bits	Description	Type	Reset
PA	[63:0]	Physical address of the CCTL operation	RW	0

The register format for TGT-type CCTL operations is:



Field Name	Bits	Description	Type	Reset
SET	[19:6]	Cache set of the CCTL operation	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
RAMID	[27:26]	RAM ID of the CCTL operation Value: 0b00 — Meaning: L3 tag RAM Value: 0b01 — Meaning: L3 data RAM	RW	0
WAY	[32:28]	Cache way of the CCTL operation. This field is interpreted dynamically based on the configured number of ways for the Tag/Data RAM and Snoop RAM. When the number of ways configured is 16 or less, the way field is only [31:28]. Otherwise, this field is [32:28]. On TGT-Type and IDX-Type CCTL Operations, if the programmable way field indicates a way larger than the configured number of ways for the particular RAM of CCTL operation (Tag RAM, Data RAM, Snoop RAM), then the CCTL operation will be dropped and the corresponding status register will indicate an illegal command.	RW	0

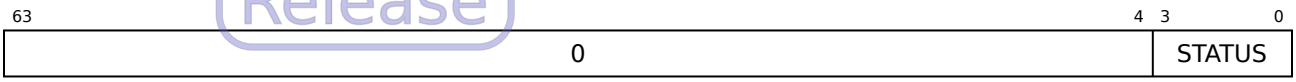
As for the flush type, these registers are not used.

8.8.13 CCTL Status Register

8.8.13.1 Pre-Core CCTL Status *n* Register

Offset: 0x0080, 0x1080, 0x2080, 0x3080, 0x4080, 0x5080, 0x6080, 0x7080

This register provides the status of CCTL operations. The status field of this register of the initiating CPU core also reflects the F-Flush CCTL operation running status.

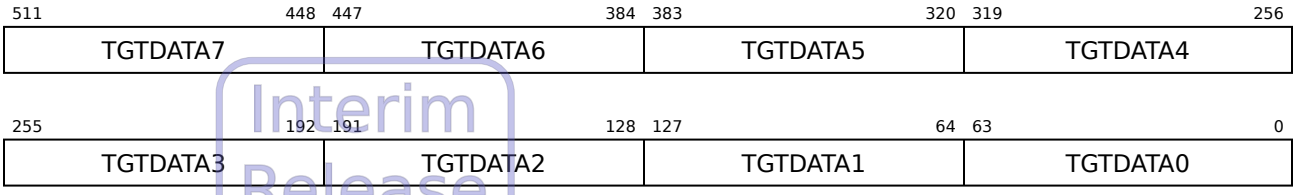


Field Name	Bits	Description	Type	Reset
STATUS	[3:0]	CCTL status of the corresponding CCTL command	RO	0

Value	Definition
0b0000	Idle
0b0001	A CCTL operation is running.
0b0010	An illegal CCTL operation was performed.
0b0100	An incomplete CCTL operation was performed; A CCTL operation may be done but incomplete due to a fatal ECC error.
0b1xxx	An operation was dropped because the CCTL module was busy executing another operation.
Others	Reserved

8.8.14 CCTL TGT Data Registers 0 – 7

Offset: 0x0090 to 0x00c8

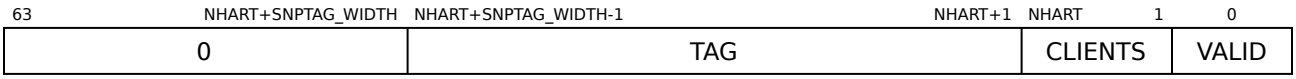


These registers provide 512-bit write or read data for the L3_TGT_WRITE or L3_TGT_READ CCTL operation. They consist of eight 64-bit registers which are readable/writable as below:

Offset	Description
0x90	CCTL TGT Data Register 0 (bits [63:0])
0x98	CCTL TGT Data Register 1 (bits [127:64])
0xa0	CCTL TGT Data Register 2 (bits [191:128])
0xa8	CCTL TGT Data Register 3 (bits [255:192])
0xb0	CCTL TGT Data Register 4 (bits [319:256])
0xb8	CCTL TGT Data Register 5 (bits [383:320])
0xc0	CCTL TGT Data Register 6 (bits [447:384])
0xc8	CCTL TGT Data Register 7 (bits [511:448])

The interpretation of these registers is different based on the TGT operation.

The register format for SNP_TGT_READ/SNP_TGT_WRITE accessing tag RAMs is:

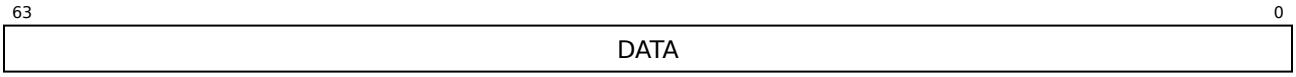


Field Name	Bits	Description	Type	Reset
VALID	[0]	Valid bit of the entry	RW	0
CLIENTS	[NHART:1]	NHART is defined by the parameter CM_CACHING_AGENTS.	RW	0
SNPTAG	[NHART+1 +: SNPTAG_ WIDTH]	Tag address of the Snoop.	RW	0

The register format for L3_TGT_READ/L3_TGT_WRITE accessing tag RAMs is:

Field Name	Bits	Description	Type	Reset
Coherence State	[2:0]	Coherence state of the cache entry **Value** / **Definition** 0b000 – Invalid 0b001 – Shared 0b010 – Shared-Corrupted 0b011 – Reserved (setting this value will cause unexpected behavior) 0b100 – Exclusive-Corrupted 0b101 – Exclusive 0b110 – Exclusive-Corrupted 0b111 – Exclusive-Dirty	RW	0
TAG	[(TAG_WIDTH-1): 3]]	Tag address of the Tag RAM.	RW	0

The register format for `L3_TGT_READ/L3_TGT_WRITE` accessing data RAMs is:



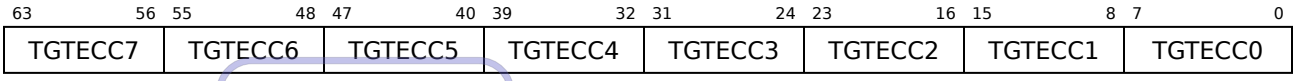
Field Name	Bits	Description	Type	Reset
DATA	[63:0]	Read data from the data RAM / Write data to the data RAM	RW	0

If an `L3_TGT_WRITE` operation is applied to the data RAM, the corresponding line of the data RAM is filled with the content of CCTL TGT Data Registers 0 – 7. Otherwise, the corresponding line of the tag RAM is filled.

If an `L3_TGT_READ` operation is applied to the data RAM, CCTL TGT Data Registers 0 – 7 are filled with the corresponding line of the data RAM. Otherwise, they are filled with the corresponding line of the tag RAM.

8.8.15 CCTL TGT ECC Code Register

Offset: 0x00d0

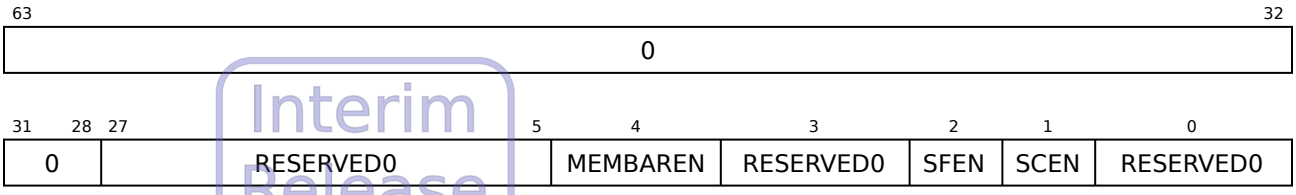


This register provides ECC codes accompanying the write or read data when an `L3_TGT_WRITE` or `L3_TGT_READ` CCTL operation is performed.

Field Name	Bits	Description	Type	Reset
TGTECC0	[7:0]	ECC code of CCTL TGT data 0	RW	0
TGTECC1	[15:8]	ECC code of CCTL TGT data 1	RW	0
TGTECC2	[23:16]	ECC code of CCTL TGT data 2	RW	0
TGTECC3	[31:24]	ECC code of CCTL TGT data 3	RW	0
TGTECC4	[39:32]	ECC code of CCTL TGT data 4	RW	0
TGTECC5	[47:40]	ECC code of CCTL TGT data 5	RW	0
TGTECC6	[55:48]	ECC code of CCTL TGT data 6	RW	0
TGTECC7	[63:56]	ECC code of CCTL TGT data 7	RW	0

8.8.16 Control Register 1

Offset: 0x00f0



This register provides more fine-grain control functions of the L3-Cache controller.

Field Name	Bits	Description	Type	Reset
RESERVED0	[0]	Reserved; must be written 0	RO	0
SCEN	[1]	L3-Cache fine-grain control. This bit is only effective when the CEN bit of the Control Register is asserted. While the L3-cache is disabled, all look-ups are treated as cache misses. However, CCTL operations can continue to operate on the shared cache. 0 = Disabled; 1 = Enabled.	RW	1
SFEN	[2]	Snoop Filter fine-grain control. This bit is only effective when the CEN bit of the Control Register is asserted. While the Snoop Filter is disabled, the coherence manager defaults to a broadcast mode and normal transactions will trigger probes to all CPUs for coherence data. Snoop Filter ECC errors are ignored in this mode. If the Snoop Filter encounters uncorrectable ECC errors, it will be disabled and the Error Register will log the details of the asynchronous error. Only CCTL IX and PA commands can continue to operate on the Snoop Filter, but CCTL WBINVAL_ALL, INVAL_ALL and FFLUSH will not. While the snoop filter is disabled, software bears the responsibility to ensure the upstream caches have already been invalidated before it shuts down the cache or powers down. 0 = Disabled; 1 = Enabled.	RW	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
RESERVED0	[3]	Reserved; must be written 0	RO	0
MEMBAREN	[4]	Memory barrier control. By default, the memory barrier is turned off. When it is turned on, writes to MMR control registers will force an ordering barrier against other transactions. Specifically, operations that were still in-flight or outstanding when the MMR command was executed will complete before the MMR write graduates. MMR registers that possess an inherent ordering hazard are: • all Way Allocation Mask Registers • Control Register 0 • Control Register 1	RW	0
RESERVED0	[27:5]	Reserved value. Must be written 0.	RW	0

8.8.17 HPM Counter Registers 0–1

Offset: 0x0200 and 0x0208

These readable and writable registers provide the counter values of monitored events. The **L3-Cache Hardware Performance Monitor** configuration option determines how many of them are present.

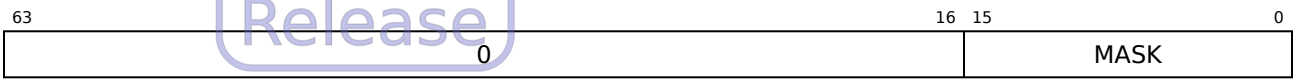
Offset	Description
0x0200	Performance Counter 0
0x0208	Performance Counter 1

8.8.18 Way Allocation Mask

Offset: 0x0300, 0x0308, 0x0310, 0x0318, 0x0320, 0x0328, 0x0330, 0x0338, 0x0340

The Way Allocation Mask (WAYMASK) register controls when a request from a domain can be allocated in ways. A domain is used to classify requests.

Details for the domain classification is introduced in Section 8.5.



If bit-n in WAYMASK_D_ is set, a request from domain_D_ can be allocated to way-n. For example, if bit-2 in WAYMASK1 is set, the request from domain 1 can be allocated to way-2.

Field Name	Bits	Description	Type	Reset
MASK	[15:0]	Way allocation mask	RW	0xFFFF

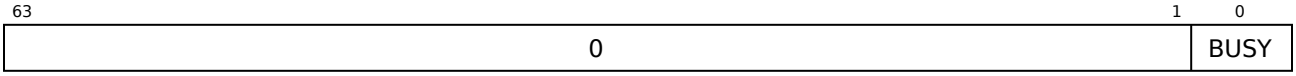
8.8.19 CCTL F-Flush Status Register

Offset: 0x0500

This register indicates the status of an L3-Cache F-Flush CCTL operation.

When an L3-Cache F-Flush CCTL operation is running, the BUSY bit of this register is set to 1. In addition, the STATUS field of the Pre-Core CCTL Status Register in the initiating CPU core will also be set to 1, indicating the operation is running.

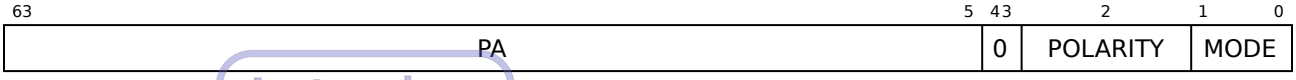
For consistency with other CCTL operations, it is recommended to use the Pre-Core CCTL Status Register to check for the completion of the L3-Cache F-Flush CCTL operation. The information provided by this register is useful to observe the status of L3-Cache F-Flush CCTL operations initiated by other CPU cores for diagnostic purposes only. However, it cannot guarantee mutual exclusivity among F-Flush operations initiated from different cores. Instead, a proper mutex lock should be used.



Field Name	Bits	Description	Type	Reset
BUSY	[0]	An F-Flush CCTL operation is running. All writes to CCTL F-Flush Range registers are ignored when this bit is asserted.	RO	0

8.8.20 CCTL F-Flush Range Registers

Offset: 0x0508, 0x0510, 0x0518, 0x0520



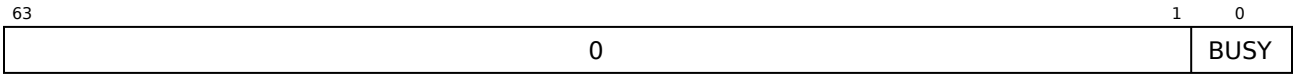
Field Name	Bits	Description	Type	Reset
MODE	[1:0]	Mode of address filter. The field is reset to Disable (0) when a Filtered Flush CCTL operation is completed. **Value** / **Meaning**: 0 Disable; 1 RANGE; 2 Reserved; 3 NAPOT	RW	0
POLARITY	[2]	Polarity of address filter. All cache lines within the specified range will either be flushed or preserved depending on the value of this bit. The polarity of all address filters must be the same. The polarity set in Register 0 applies to all CCTL F-Flush Range registers. Accordingly, the polarity values in CCTL F-Flush Range Registers 1, 2, and 3 are ignored. **Value** / **Meaning**: 0 Flush cache lines in the specified ranges.; 1 Preserve cache lines in the specified ranges.	RW	0
PA	[63:5]	Physical Address. See Section 8.4.3 for details on how a range is formed. The particular format of PA when MODE is NAPOT is described in Table 34.	RW	0

8.8.21 SW Prefetch Status Register

Offset: 0x0580

When a SW prefetch operation is running, the BUSY bit of this register will be set to 1.

The information provided by this register is useful to observe the status of SW prefetch operations initiated by other CPU cores for diagnostic purposes only. However, it cannot guarantee mutual exclusivity amongst other operations initiated from different cores. Instead, a proper mutex lock should be used.

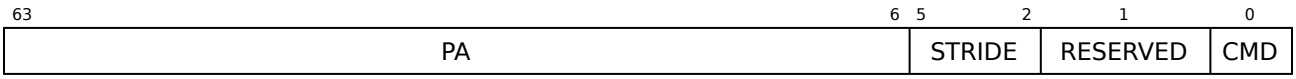


Field Name	Bits	Description	Type	Reset
BUSY	[0]	A SW prefetch operation is running. All writes to the SW Prefetch Status Register are ignored when this bit is asserted.	RO	0

8.8.22 SW Prefetch Control Register

Offset: 0x0588

This register provides controls that allows software to initiate a series of data prefetches into the shared cache. The SW prefetch allows software to control the stride of the prefetches in a 4KiB aligned region. Since the SW Prefetch feature is shared amongst all cores, proper arbitration amongst the CPUs must be implemented to ensure correct behavior.



Field Name	Bits	Description	Type	Reset
CMD	[0]	Command Value / Definition: 0 — Prefetch for read 1 — Prefetch for write	RW	0

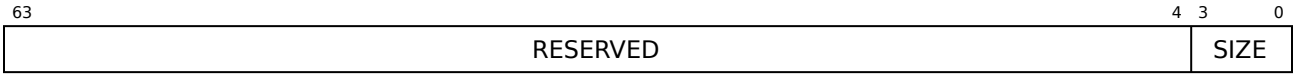
Continued on next page...

Field Name	Bits	Description	Type	Reset
STRIDE	[5:2]	Stride of the prefetch. **Value** / **Definition** 0 — Unit stride: Every cache line is prefetched. 1 — Stride is 2: Every 2nd cache line address is prefetched. 2 — Every 3rd cache line address is prefetched. . . . 15 — Every 16th cache line address is prefetched.	RW	0
PA	[63:6]	Starting physical address for which the prefetch command will begin	RW	0

8.8.23 SW Prefetch Size Register

Offset: 0x0590

This register defines the size of a prefetch in a cache line.



Field Name	Bits	Description	Type	Reset
SIZE	[3:0]	Specifies the size of the prefetch in cache lines. If any errors are encountered, or if the stride crosses a 4KiB boundary, the SW prefetcher will stop.	RW	0

Value	Definition
0	A total number of 4 cache lines is requested per command.
1	A total number of 8 cache lines is requested per command.
2	A total number of 12 cache lines is requested per command.
...	
15	A total number of 64 cache lines is requested per command.

9 Memory Error Protection

AX46MPV provides ECC (error correction code) and parity protection schemes to protect RAMs against soft errors. Soft errors are caused by external particles that temporarily corrupt values stored in the RAMs, and can be recovered or detected with different protection schemes on the RAMs. Supported protection schemes include:

- Parity for I-Cache: Single Error Detection (SED)
- ECC for BTB: Double Error Detection (DED)
- ECC for I-Cache: Double Error Detection (DED)
- ECC for ILM, DLM, D-Cache, and STLB: Single Error Correct and Double Error Detection (SEC-DED)

Table 42: Memory Protection Types

RAM	Protection Scheme	Protection Granularity	Behavior
ILM	ECC	64 bits	See Section 9.2
DLM	ECC	64 bits	See Section 9.2
I-Cache tag	Parity	8 bits	See Section 9.3
I-Cache data	Parity	8 bits	See Section 9.3
I-Cache tag	ECC	32 bits/64 bits	See Section 9.3
I-Cache data	ECC	32 bits	See Section 9.3
D-Cache tag	ECC	64 bits	See Section 9.4
D-Cache data	ECC	64 bits	See Section 9.4
BTB	ECC	64 bits	See Section 9.5
STLB tag	ECC	32 bits/64 bits	See Section 9.6
STLB data	ECC	32 bits/64 bits	See Section 9.6

Note

There are two possible ECC protection granularity for I-Cache tag:

- 32 bits when the SRAM data width is equal to or less than 39 bits, or
- 64 bits when the SRAM data width is larger than 39 bits.

There are two possible ECC protection granularity for STLB tag and STLB data:

- 32 bits when the SRAM data width is equal to or less than 39 bits, or
- 64 bits when the SRAM data width is larger than 39 bits.

9.1 Parity/ECC Control Mode

The milmb.ECCEN, mdlmb.ECCEN mcache_ctl.TLB_ECCEN, mcache_ctl.BTB_ECCEN, mcache_ctl.IC_ECCEN, and mcache_ctl.DC_ECCEN CSR fields control memory protection modes for ILM, DLM, STLB, BTB, I-Cache, and D-Cache respectively. The memory protection modes include:

- Disable parity/ECC
- Generate exceptions on uncorrectable parity/ECC errors
- Generate exceptions on all parity/ECC errors

See the following subsections for detailed information on each mode.

9.1.1 Disable Parity/ECC

In this mode, the parity/ECC logic behaves as follows:

- The logic disables parity/ECC checking, but still updates the newly-generated parity/ECC code into the RAM for each write access.
- The received data is always uncorrected.
- The processor uses read-modify-write operation to store partial double words.
- If any parity/ECC error happens:
  - No exception will be generated.
  - The error will not be corrected.

9.1.2 Generate Exceptions on Uncorrectable Parity/ECC Errors

In this mode, the parity/ECC logic behaves as follows:

- The logic enables parity/ECC checking and updates the newly-generated parity/ECC code into the RAM for each write access.
- Corrected data is always given.
- The processor uses read-modify-write operations to store partial double words.
- No exceptions will be generated for correctable errors, and those correctable errors will be recovered.
- Exceptions will be generated for uncorrectable errors.

- For speculative accesses, such as instruction prefetch and data prefetch:
  - Uncorrectable errors do not generate exceptions.
  - Correctable errors are corrected.

9.1.3 Generate Exceptions on Parity/ECC Errors

In this mode, the parity/ECC logic behaves as follows:

- The logic enables parity/ECC checking and updates the newly-generated parity/ECC code into the RAM for each write access.
- Corrected data is always given.
- The processor uses read-modify-write operations to store partial double words.
- For correctable Parity/ECC errors, exceptions are generated, and the corrupted data in the RAM will be corrected before generating exceptions.
- For uncorrectable Parity/ECC errors, exceptions are generated.
- For speculative accesses, such as instruction prefetch and data prefetch:
  - Uncorrectable and correctable errors do not generate exceptions.
  - Correctable errors are corrected.

9.2 Local Memory Protection

SEC-DED is applied to ILM or DLM when ILM or DLM Soft Error Protection is specified to ECC. One-bit ECC errors detected in ILM and DLM access are correctable, while two-bit ECC errors found on load or store instructions are uncorrectable. For ILM and DLM ECC error exceptions, `mecc_code` will be updated with the ECC error information.

9.3 I-Cache Protection

When I-Cache Soft Error Protection is configured to Parity and ECC, SED and DED are applied to the I-Cache, respectively. Data in the I-Cache are always clean since there are no store accesses to it. If a parity or ECC error is detected, the I-Cache can get the correct copy from the next-level memory system. Therefore, one-bit parity errors and one-bit/two-bit ECC errors in I-Cache accesses are correctable.

To detect soft errors in the lock bit, an additional lock mirror bit is implemented. If a parity or ECC error occurs on a locked cache line, or if the lock bit and the lock mirror bit mismatch, that parity or ECC error will be uncorrectable.

Correctable errors will always be corrected after detection, even though they may trigger an exception. Erroneous cache data of uncorrectable errors will be kept in the I-Cache without correction. When a parity or ECC error occurs in the I-Cache, `mecc_code` will be updated with the parity/ECC error information. If lock bit information is needed , users can get the lock bit and the lock mirror bit by the L1I_IX_RTAG CCTL command.

Table 43: I-Cache RAM Error Types

RAM	RAM Error	Error Type
TAG RAM	Lock Line ECC Error	Uncorrectable Error
TAG RAM	Lock Bit Parity Error	Uncorrectable Error
TAG RAM	Parity Error	Correctable Error
TAG RAM	1-bit ECC Error	Correctable Error
TAG RAM	2-bit ECC Error	Correctable Error
DATA RAM	Parity Error	Correctable Error
DATA RAM	1-bit ECC Error	Correctable Error
DATA RAM	2-bit ECC Error	Correctable Error

Table 44: I-Cache Memory Protection

Case	`mcache_ctl.IC_ECCEN`	Error Type	Behavior
Instruction Fetch	2	Correctable error	Invalidate the line
	2	Uncorrectable error	Raise instruction access fault
	3	Correctable error	Invalidate the line and raise instruction access fault
	3	Uncorrectable error	Raise instruction access fault

Continued on next page...

Table 44: (continued)

Case	`mcache_ctl.IC_ECCEN`	Error Type	Behavior
L1I_VA_LOCK	2	Correctable error	Invalidate the line
	2	Uncorrectable error	Raise Store/AMO access fault
	3	Correctable error	Invalidate the line and raise Store/AMO access fault
	3	Uncorrectable error	Raise Store/AMO access fault
L1I_VA_UNLOCK	2	Correctable error	Invalidate the line
	2	Uncorrectable error	Raise Store/AMO access fault
	3	Correctable error	Invalidate the line and raise Store/AMO access fault
	3	Uncorrectable error	Raise Store/AMO access fault
L1I_VA_INVAL	2	Correctable error	Invalidate the line
	2	Uncorrectable error	Raise Store/AMO access fault
	3	Correctable error	Invalidate the line and raise Store/AMO access fault
	3	Uncorrectable error	Raise Store/AMO access fault

9.4 D-Cache Protection

SEC-DED is applied to D-Cache tag and data SRAMs when D-Cache Soft Error Protection is specified to ECC. Correctable errors include:

- A cache line with a one-bit error on tag or data SRAMs is correctable, and the hardware will automatically repair the error.
- A clean line with a two-bit error on data SRAMs is also correctable, and the hardware will invalidate the line.
  - If `mcache_ctl.DC_ECCEN` is 2, the line is refilled from the next-level memory system.
  - If `mcache_ctl.DC_ECCEN` is 3, an exception is raised without refilling the line.

Uncorrectable errors include:

- An access to a cache line with a two-bit error on tag SRAMs
- An access to a modified line with a two-bit error on data SRAMs

For all D-Cache ECC error exceptions, `mecc_code` will be updated with the ECC error information.

When a D-Cache write-back encounters ECC errors in data SRAMs, the following operations are used:

- The errors are trapped by imprecise exceptions or local interrupts.
  - The mecc_code.CODE field is always written to 0.
- Correctable ECC errors are repaired and written to the next-level memory.
- Uncorrectable ECC errors are written to the next-level memory.

D-Cache TAG RAMs are duplicated for VLSU to check whether a cacheable line is in the D-Cache or not. If the line is in the D-Cache, the VLSU uses the data in the D-Cache. If the line is not in the D-Cache, the VLSU uses the data in the next-level memory.

Table 45: D-Cache Memory Protection for Load, Store and Eviction

Operation	DC_ECCEN	TAG RAM ECC Error	Cache Line State	Data RAM ECC Error	Trap
Load	2	None	Shared or Exclusive	Don't Care	N/A
	2	None	Modified	1-bit	N/A
	2	None	Modified	2-bit	Load access fault
	2	1-bit	Shared or Exclusive	Don't Care	N/A
	2	1-bit	Modified	1-bit	N/A
	2	1-bit	Modified	2-bit	Load access fault
	2	2-bit	Don't Care	Don't Care	Load access fault
	3	None	Don't Care	1-bit	Load access fault
	3	None	Don't Care	2-bit	Load access fault
	3	1-bit	Don't Care	Don't Care	Load access fault
	3	2-bit	Don't Care	Don't Care	Load access fault
Partial store	2	None	Shared or Exclusive	Don't Care	N/A
	2	None	Modified	1-bit	N/A
	2	None	Modified	2-bit	Store/AMO access fault
	2	1-bit	Shared or Exclusive	Don't Care	N/A
	2	1-bit	Modified	1-bit	N/A
	2	1-bit	Modified	2-bit	Store/AMO access fault
	2	2-bit	Don't Care	Don't Care	Store/AMO access fault
	3	None	Don't Care	1-bit	Store/AMO access fault
	3	None	Don't Care	2-bit	Store/AMO access fault
	3	1-bit	Don't Care	Don't Care	Store/AMO access fault
	3	2-bit	Don't Care	Don't Care	Store/AMO access fault
Full store	2	None	Don't Care	Don't Care	N/A
	2	1-bit	Don't Care	Don't Care	N/A
	2	2-bit	Don't Care	Don't Care	Store/AMO access fault
	3	None	Don't Care	Don't Care	N/A
	3	1-bit	Don't Care	Don't Care	Store/AMO access fault
	3	2-bit	Don't Care	Don't Care	Store/AMO access fault

Continued on next page...

Table 45: (continued)

Operation	DC_ECCEN	TAG RAM ECC Error	Cache Line State	Data RAM ECC Error	Trap
Eviction	2	Don't Care	Shared or Exclusive	Don't Care	N/A
	2	Don't Care	Modified	1-bit	N/A
	2	Don't Care	Modified	2-bit	Imprecise ECC error local interrupt
	3	Don't Care	Shared or Exclusive	Don't Care	N/A
	3	Don't Care	Modified	1-bit/2-bit	Imprecise ECC error local interrupt

Note

- Partial store: Byte, halfword and word accesses
- Full store: Doubleword access

Table 46: D-Cache Memory Protection for CCTL Operations

Operation	DC_ECCEN	TAG RAM ECC Error	Cache Line State	Data RAM ECC Error	Trap
L1D_VA_INVAL, L1D_IX_INVAL	2	None	Don't Care	Don't Care	N/A
	2	1-bit	Don't Care	Don't Care	N/A
	2	2-bit	Don't Care	Don't Care	Store/AMO access fault
	3	None	Don't Care	Don't Care	N/A
	3	1-bit	Don't Care	Don't Care	Store/AMO access fault
	3	2-bit	Don't Care	Don't Care	Store/AMO access fault
L1D_VA_WB, L1D_VA_WBINVAL, L1D_IX_WB, L1D_IX_WBINVAL,	2	None	Shared or Exclusive	Don't Care	N/A
	2	None	Modified	1-bit	N/A
	2	None	Modified	2-bit	Store/AMO access fault
	2	1-bit	Shared or Exclusive	Don't Care	N/A
	2	1-bit	Modified	1-bit	N/A
	2	1-bit	Modified	2-bit	Store/AMO access fault
	2	2-bit	Don't Care	Don't Care	Store/AMO access fault
	3	None	Shared or Exclusive	Don't Care	N/A
	3	None	Modified	1-bit	Store/AMO access fault
	3	None	Modified	2-bit	Store/AMO access fault
	3	1-bit	Don't Care	Don't Care	Store/AMO access fault
	3	2-bit	Don't Care	Don't Care	Store/AMO access fault

Note

- L1D_*_INVAL does not read data RAMs, and the ECC codes are not checked.
- L1D_**WB and L1D***WBINVAL do not read data RAMs of a clean line, and the ECC codes are not checked.

9.5 BTB Protection

BTB RAMs save predicted branch target addresses to improve the instruction fetch throughput. When a branch instruction is resolved, the branch resolution logic compares the correct branch target with the predicted target address. If there is a mismatch, the logic updates the BTB entry and fetches instructions from the correct target address.

SED and DED are also applied to the BTB when **BTB Soft Error Protection** is specified to ECC. If an ECC error is detected, the corresponding BTB entry with the ECC error will be invalidated. The BTB entry will be recovered when the same branch information is trained into the BTB.

9.6 STLB Protection

SEC and DED are applied to STLB tag and data SRAMs when STLB Soft Error Protection is specified to ECC.

Table 47 summaries STLB memory protection. The hardware operations are:

- For instruction fetch, load, store, or speculative accesses with mcache_ctl.TLB_ECCEN as 2:
  - One-bit errors: The entry is repaired.
  - Two-bit errors: The entry is invalidated and refilled from the next-level memory system.
- For instruction fetch, load, or store with mcache_ctl.TLB_ECCEN as 3:
  - One-bit errors: The entry is repaired the, and an exception is raised.
  - Two-bit errors: The entry is invalidated, and an exception is raised without refilling the entry.
- For speculative accesses with mcache_ctl.TLB_ECCEN as 3:
  - One-bit errors: The entry is repaired the entry.
  - Two-bit errors: The entry is invalidated the entry.
- For `SFENCE.VMA` with mcache_ctl.TLB_ECCEN as 2 or 3:
  - One-bit or two bit errors: The entry is invalidated.

Table 47: STLB Memory Protection

Case	mcache_ctl.TLB_ECCEN	1-Bit Error	2-Bit Error	Exception
Instruction	2	Repair	Invalidate and refill	No
Fetch/Load/Store	3	Repair	Invalidate	Yes

Continued on next page...

Table 47: (continued)

Case	mcache_ctl.TLB_ECCEN	1-Bit Error	2-Bit Error	Exception
Speculative accesses	2	Repair	Invalidate and refill	No
	3	Repair	Invalidate	No
SFENCE.VMA	2 or 3	Invalidate	Invalidate	No

9.7 Soft Error Injection

9.7.1 ILM/DLM ECC Error Injection

To inject ECC errors in the ILM/DLM for implementing the ECC exception handler, use the following steps:

- Specify the ILM or DLM to "Disable ECC" mode.
- Enable milmb.RWECC or mdlmb.RWECC for the ILM or DLM.
- Perform a load instruction on an address in the ILM or DLM region. mecc_code.CODE will contain the ECC code of the read data.
- Flip one or two bits in mecc_code.CODE to inject a one-bit or two-bit ECC error.
- Perform a store instruction to the address in the ILM or DLM region. The ECC code with error will be written to the ILM/DLM.
- Disable `milmb.RWECC` or `mdlmb.RWECC` for the ILM or DLM.
- Specify the ILM or DLM to "Generate Exceptions on Uncorrectable Parity/ECC Errors" mode or "Generate Exceptions on parity/ECC Errors" mode.
- Perform a load instruction to the address to trigger the ECC error.

9.7.2 I-Cache Parity/ECC Error Injection

To inject ECC errors in the I-Cache for implementing the parity/ECC exception handler, use the following steps:

- Specify the I-Cache to "Disable ECC" mode.
- Enable mcache_ctl.IC_RWECC.

- Perform the L1I_IX_RTAG or L1I_IX_RDATA CCTL command to a specific address to read parity/ECC data from tag RAM or data RAM.
- Flip one bit in mecc_code.CODE or mcctldata to inject a one-bit parity/ECC error or flip the lock duplicate in mcctldata to generate a lock mismatch error.
- Perform the L1I_IX_WTAG or L1I_IX_WDATA CCTL command to the specific address to inject a parity/ECC error.
- Disable mcache_ctl.IC_RWECC.
- Specify the I-Cache to "Generate Exceptions on Uncorrectable Parity/ECC Errors" mode or "Generate Exceptions on parity/ECC Errors" mode.
- Perform a fetch to that specific address to trigger the ECC error.

9.7.3 D-Cache ECC Error Injection

To inject ECC errors on the D-Cache for implementing the ECC exception handler, use the following steps:

- Specify the D-Cache to "Disable ECC" mode.
- Enable mcache_ctl.DC_RWECC.
- Perform the L1D_IX_RTAG or L1D_IX_RDATA CCTL command to a specific address to read ECC data from tag RAM or data RAM.
- Flip one bit or two bits in mecc_code.CODE and mcctldata to inject a one-bit or two-bit ECC error.
- Perform the L1D_IX_WTAG or L1D_IX_WDATA CCTL command to the specific address to inject an ECC error.
- Disable mcache_ctl.DC_RWECC.
- Specify the D-Cache to "Generate Exceptions on Uncorrectable Parity/ECC Errors" mode or "Generate Exceptions on parity/ECC Errors" mode.
- Perform a load/store fetch to that specific address to trigger the ECC error.

9.7.4 STLB ECC Error Injection

To inject ECC errors on Shard TLB SRAMs for implementing the ECC exception handler, use the following steps:

- Specify mcache_ctl.TLB_ECCEN to "Disable ECC" mode.

- Enable mcache_ctl.TLB_RWECC.
- Perform the TLB_IX_RTAG or TLB_IX_RDATA CCTL command to a specific address to read ECC data from tag RAM or data RAM.
- Flip one bit or two bits in mecc_code.CODE and mcctldata to inject a one-bit or two-bit ECC error.
- Perform the TLB_IX_WTAG or TLB_IX_WDATA CCTL command to the specific address to inject an ECC error.
- Disable mcache_ctl.TLB_RWECC.
- Specify mcache_ctl.TLB_ECCEN to "Generate Exceptions on Uncorrectable Parity/ECC Errors" mode or "Generate Exceptions on parity/ECC Errors" mode.
- Perform a fetch or load/store to that specific address to trigger the ECC error.

9.7.5 BTB ECC Error Injection

To inject ECC errors on BTB SRAMs for implementing the ECC exception handler, use the following steps:

- Disable mmisc_ctl.brpe to avoid BTB content update through BTB prediction training.
- Specify mcache_ctl.BTB_ECCEN to "Disable ECC" mode.
- Enable mcache_ctl.BTB_RWECC.
- Perform the BTB_IX_RDATA CCTL command to read ECC data from BTB RAM.
  - Set mcctlbeginaddr[8:7] as 0x0.
  - Set mcctlbeginaddr[6:0] as 0x20.
- Flip one or two bits in mecc_code.CODE and mcctldata to inject a one-bit or two-bit ECC error.
- Perform the BTB_IX_WDATA command to the same address to inject an ECC error.
- Disable mcache_ctl.BTB_RWECC.
- Specify mcache_ctl.BTB_ECCEN to "Generate Exceptions on Uncorrectable Parity/ECC Errors" mode or "Generate Exceptions on parity/ECC Errors" mode.
- Perform a direct jump to PC 0x80 to trigger an ECC error on the BTB.

9.8 L3-Cache Memory Error Protection

See Section 8.6.

10 MemBoost

10.1 Non-Blocking Memory Access

The non-blocking memory access mechanism is used to hide memory latency by allowing instructions and memory accesses to execute concurrently. In AX46MPV, store instructions are always non-blocking. However, load instructions are blocking by default, meaning that they will block all subsequent instructions until required bus accesses are finished. This occurs, for example, on cache misses or when the D-Cache is disabled or non-present.

The control bit mmisc_ctl.NBLD_EN controls the blocking behavior of load instructions. Once set, load instructions will operate in a non-blocking manner. They will wait in the background for bus accesses to finish, allowing subsequent instructions to continue execution until data dependency hazards or structural hazards occur.

Data dependency hazards occur when subsequent instructions access the destination registers of load instructions. Structural hazards refer to resource limitations that allow load/store instructions to wait in the background. The number of cacheable accesses is limited by the maximum outstanding D-Cache misses. On the other hand, the number of non-cacheable memory accesses is restricted by the maximum number of transactions to Non-Cacheable Memory.

When the processor operates in the non-blocking mode, each bus read transaction for the outstanding load/store instructions is assigned a unique AxID, enabling out-of-order return of data.

When load instructions operate in the blocking mode, bus aborts or errors are reported synchronously as *Load Access Faults* (mcause = 5). On the other hand, such errors are no longer reported synchronously in the non-blocking mode. Instead, they are treated as imprecise exceptions and reported as *Bus Read/Write Transaction Error Local Interrupts* (mip.BWEI).

Regardless of mmisc_ctl.NBLD_EN, the following memory access instructions are always blocking:

- Load instructions to device non-bufferable region through the MMIO interface
- Load instructions to local memory
- Load-Reserved/Store-Conditional instructions
- Atomic memory access (AMO) instructions

10.2 D-Cache Write-Around Support

When executing store operations to write-allocate memory, a D-Cache entry is allocated on a cache miss, anticipating that the entry will be reused. However, in certain streaming cases, such as memory copy or memory set operations involving a large amount of data that is is used only once, allocating D-Cache entries can hinder performance. Accordingly, configuring the Write-Around feature can address this issue by merging store operations to the same write-no-allocate line into a burst write transfer. Write-no-allocate lines are memory regions with one of the following physical memory attributes:

- Cacheable memory, write-back, no allocate
- Cacheable memory, write-back, read-allocate
- D-Cache is in write-no-allocate mode (see below)
  - Cacheable memory, write-back, write-allocate
  - Cacheable memory, write-back, read and write-allocate

When mcache_ctl.DC_WAROUND is set, the processor automatically detects streaming memory write patterns and switches the D-Cache to the write-no-allocate mode once the following conditions are met:

- A D-Cache miss happens for a store operation.
- This missed cache line is then completely filled by subsequent store operations, without any other stores causing a miss to a different cache line. Note that the same bytes in the missing cache line may be overwritten multiple times.
- The above mentioned conditions happen consecutively for a configured amount of cache lines. However, the corresponding addresses of those cache lines do not need to be consecutive.

In the write-no-allocate mode, the write-allocate attribute is overridden to write-no-allocate. Both D-Cache allocation policy and AWCACHE are affected.

The processor stays in the write-no-allocate mode until any of the following conditions happens:

- A D-Cache miss arises for a store operation, but the previous store operations have not completely filled their cache line.
- A load operation accesses the same cache line being filled by store operations honored by Write-Around, and that cache line has not completely been filled in.
- The setting of mcache_ctl.DC_WAROUND is changed.

The Write-Around feature should be enabled (see Section 16.10.6) to take effect.

Note

- Stores qualified for Write-Around behavior may produce AXI write transactions with partial or even zero write strobes in certain corner case scenarios. As a result, Write-Around support should not be enabled for accessing regions containing designs that do not support AXI partial and zero write strobes, such as the Andes ATCAXI2AHB100 and ATCAXI2AHB200 bridges.

10.3 D-Cache Prefetch

AX46MPV implements a hardware prefetcher for hiding memory access latency. It has a 4-entry reference prediction table, with each entry capable of detecting an access sequence with a fixed stride. When mcache_ctl.DPREF_EN is set, the following access pattern will enable the hardware prefetcher:

- A load instruction accesses a cacheable and read-allocate memory region.
- This access causes cache misses.
- The same load instruction accesses another two cache lines with a fixed stride (positive or negative).

After the pattern is detected, the hardware prefetcher automatically fills the subsequent cache lines (with the same stride) into the D-Cache. However, it does not send requests to the system bus for the lines already present in the D-Cache.

The prefetcher is stopped in one of the following conditions:

- The stride of the load access is changed.
- Entering debug mode
- The load accesses
  - Cross the LM boundary
  - Have inconsistent PMA
  - Encounter access faults
    - Bus error
    - ECC error
    - PMP violations
    - PMA empty hole
    - Page fault
- SFENCE.VMA is executed.
- satp is updated.
- Hit a load/store debug trigger
- A trap is taken.
- A trap return instruction is executed.

The prefetcher is also stopped when prefetch accesses encounter one of the following conditions. These errors will be ignored without triggering any exception.

- The accesses cross the LM boundary.
- The PMA is inconsistent with the load access.
- Bus error
- ECC error
- PMP violations
- PMA empty hole
- Page fault

10.4 Store Data Buffer

The store buffer is for buffering scalar store data to non-cacheable memory, device or HVM regions. Its depth is specified by Depth of Scalar Store Data Buffer and the each entry has a width of 64 bits.

Buffered data is sent to the bus in program order. The corresponding buffer entry is released and made available for subsequent scalar store instructions after the downstream accepts the store requests. When the store buffer reaches its maximum capacity, any further scalar store instructions are stalled until the buffer becomes available.

11 High-Bandwidth Vector Memory

11.1 Introduction

High-Bandwidth Vector Memory (HVM) is a physical address region that provides high memory bandwidth for vector operations.

The HVM region is for data access only and does not allocate data into any cache. If an instruction fetch accesses the HVM region, a fetch fault exception will be raised. Additionally, the memory attributes of the HVM region will not be overridden by the settings of static or programmable physical memory attributes.

The HVM region is shared across all cores, but atomic accesses are not supported. To maintain data coherence between cores, a software lock mechanism is required. Attempts to access the HVM region with atomic load/store instructions will raise an access fault exception.

11.2 Reference HVM Design

See AndesCore AX46MPV Integration Guide (IG094) for the reference HVM design.

12 Coherence Manager (CM)

AX46MPV supports cache coherency for 1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16 cores with D-Cache and L3-Cache. To enforce cache coherency, D-Cache and L3-Cache settings should be properly enabled.

- D-Cache is enabled by setting mcache_ctl.DC_EN.
- L3-Cache is enabled by setting CEN in L3-Cache control register.
- D-Cache coherency is enabled by setting mcache_ctl.DC_COHEN.

Table 48 summarizes behaviors of different settings. See Section 12.2 and Section 12.1 for detailed information on enabling or disabling cache coherency.

Table 48: Behaviors of Cache Enable/Disable Settings

D-Cache	L3-Cache	D-Cache Coherency	Description
Disabled	Don't Care	Don't Care	Cache coherence is not supported. All load/store requests are sent to the MMIO port.
Enabled	Disabled	Don't Care	Cache coherence is not supported. All cacheable load/store requests bypass the L3-Cache, and are sent to the MEM port.
Enabled	Enabled	Disabled	Illegal usage. Accesses to shared lines may trigger access fault exceptions.
Enabled	Enabled	Enabled	Cache coherence is enforced by the coherence manager.

AX46MPV implements the MESI (Modified, Exclusive, Shared, and Invalid) protocol in the L3-Cache to maintain coherency for cacheable accesses. If the L3-Cache is enabled, it tracks line states of both L3-Cache and D-Cache to maintain the cache coherency. For each read or write requests to the L3-Cache, its tag SRAMs are accessed to determine the cache line state. When data coherence handling is necessary, the L3-Cache sends probe messages to write back or invalidate copies in the D-Cache.

The L3-Cache handles requests without intervening the D-Cache under the following conditions:

- D-Cache coherency is disabled.
- The cache line state indicates coherence handling is not required.

12.1 Disable D-Cache Coherency

To disable D-Cache coherency of a core, follow these steps:

- Disable the D-Cache by clearing mcache_ctl.DC_EN. This changes accesses to non-cacheable regions, preventing the D-Cache from allocating more lines.
- Write back and invalidate the D-Cache by executing the CCTL command L1D_WBINVAL_ALL. All modified lines are flushed, and the D-Cache will not cache stale lines after power-up.
- Disable D-Cache coherency by clearing mcache_ctl.DC_COHEN.
- Wait for mcache_ctl.DC_COHSTA to be cleared to ensure the previous step is completed. At this point, the CM does not issue probe requests.

12.2 Enable D-Cache Coherency

To enable coherency of a core, follow these steps:

- Enable D-Cache coherency by setting mcache_ctl.DC_COHEN.
- Wait for mcache_ctl.DC_COHSTA to be set.
- Enable the D-Cache by setting mcache_ctl.DC_EN.

To maintain a coherent view between cores, D-Cache coherency should be enabled before the D-Cache is enabled.

12.3 Fine-Grain L3-Cache and Snoop Filter Control

The `CEN` bit in L3-Cache `Control Register 0` serves as a global enable for both the L3-Cache and the snoop filter. However, more fine-grained control is available through specific fields in `Control Register 1`:

- The shared cache is enabled or disabled by toggling the `SCEN` field.
  - When the shared cache is disabled, all look-ups result in cache misses.
- The snoop filter is enabled or disabled by toggling the `SFEN` field.
  - When the snoop filter is disabled, the coherence manager enters broadcast mode.

While the `CEN` bit in `Control Register 0` is cleared to 0, all fine-grained controls are ineffective, and both the L3-Cache and snoop filter are disabled. Fine-grained controls are only effective when the `CEN` bit is set to 1.

12.4 Enabling and Disabling Snoop Filter

The snoop filter is enabled under these conditions:

- The `CEN` bit in L3-Cache `Control Register 0` is set to 1.
- The `SFEN` bit in L3-Cache `Control Register 1` is set to 1.

If any of the above conditions are not true, the snoop filter enters broadcast mode, and all upstream caching agents will be probed upon receiving coherent transaction requests.

To synchronize the snoop filter with private caches, it is recommended to flush all coherent private caches prior to re-enabling the snoop filter.

An example programming sequence is as follows:

- Current state: broadcast mode
- Software runs the `WBINVAL` CCTL command on all coherent private caches.
- Software waits for the operation to complete. During this time, coherent caching agents cannot allocate any new lines.
- Software sets the `CEN` and `SFEN` bits to 1.

13 Trap

13.1 Introduction

According to the RISC-V Privileged Architecture, a trap is a control flow change of normal instruction execution caused by an interrupt or an exception. Interrupts are initiated by external sources, while exceptions are generated as a by-product of instruction execution. When a trap happens, the processor halts the current instruction flow, disables interrupts, saves enough states for later resumption, and starts executing a trap handler.

Interrupts can be categorized as local or external. External interrupts are global and arbitrated externally by a platform-level interrupt controller (PLIC). Once selected, an external interrupt is combined with local interrupts for arbitration to trigger a trap.

Exceptions can be precise or imprecise. When a precise exception is triggered, the instruction causing it and all its subsequent instructions in the program order will not have affected the architectural state. Furthermore, the precise exception must be directly attributed to the instruction that caused it. For precise exceptions, the value of mcause register will be greater than zero. In contrast, exceptions that do not meet these criteria are considered imprecise and are delivered as local interrupts (mcause < 0). That is, in the standard RISC-V privileged architecture, only precise exceptions trigger exception handling, while imprecise exceptions trigger local interrupts.

For precise exceptions, mepc is the PC of the faulting instruction. For imprecise exceptions, mepc points to the interrupted instruction. Regardless of preciseness of exceptions, mtval records the effective faulting address for exceptions related to memory operations.

13.2 Interrupt

AX46MPV provides three interrupt inputs: timer, software, and external interrupts. In a RISC-V platform, timer and software interrupts are local interrupts, meaning each processor in the platform receives its own timer and software interrupts. Contrarily, external interrupts are global interrupts and shared across all processors. They are arbitrated and distributed by a platform-level interrupt controller (PLIC) to individual processors. Each external interrupt source can be assigned a priority, and each interrupt target (i.e., RISC-V processors) can select which external interrupt sources it will handle. The PLIC routes the highest priority interrupt source to the target processor. See AndesCore AX46MPV Integration Guide (IG094) for more descriptions on PLIC.

13.2.1 Additional Local Interrupts

In addition to external interrupts, AX46MPV may generate internal interrupts for the following events (imprecise exceptions):

- Local memory subordinate port parity/ECC errors (See mie.IMECCI, and mip.IMECCI)
- Cache write back parity/ECC errors (See mie.IMECCI and mip.IMECCI)
- Bus read/write transaction errors (See mie.BWEI, mip.BWEI, and mdcause)
- Performance monitor overflows (See mie.LCOFI and mip.LCOFI)

13.2.2 Interrupt Status and Masking

The `mip` CSR holds the pending bits of each interrupt, while the `mie` CSR contains enable bits of those interrupts. The processor can selectively enable interrupts by manipulating the `mie` CSR, or globally disable interrupts by clearing the `mstatus.MIE` bit.

13.3 Exception

AX46MPV implements the following (precise) exceptions (mcause > 0). See the tables in Section 16.3.13 (and Section 16.3.9) for how these exceptions can be identified by trap handlers.

- Instruction address misaligned exceptions
  - Jump to misaligned addresses
- Instruction access faults
  - Bus errors caused by instruction fetches
  - Uncorrectable ECC errors when fetching
  - PMP faults caused by instruction fetches
  - PMA faults caused by instruction fetches
- Illegal instructions
  - Unsupported instructions
  - Privileged instructions
  - Accessing non-existent CSRs
  - Accessing privileged CSRs
  - Writing to read-only CSRs
  - Executing Andes-specific instructions in the RISC-V compatibility mode (`mmisc_ctl.RVCOMPM` == 1).
  - Initial value of stack point (SP) is not aligned to 16 bytes when performing PUSH, POP, or POPRET.
  - Executing floating point instructions without enabling FPU context (`mstatus.FS` == 0).

  - Executing ACE instructions without enabling ACE context (`mmisc_ctl.ACES == 0`).
- Breakpoint exceptions
- Load address misaligned exceptions
- Load access faults
  - Bus errors caused by load instructions
  - ECC errors caused by load instructions
  - PMP fault caused by load instructions
  - PMA fault caused by load instructions
- Store/AMO address misaligned exceptions
- Store/AMO access faults
  - Bus errors caused by store instructions
  - ECC errors caused by store instructions
  - PMP fault caused by store instructions
  - PMA fault caused by store instructions
- Environment calls
- Stack overflow/underflow exceptions with StackSafe supported
- ACE exceptions

Some events (for example, parity/ECC and bus errors) listed above may cause imprecise exceptions under certain circumstances. Imprecise exceptions are delivered through local interrupts (mcause < 0) instead of the standard RISC-V exceptions (mcause > 0). It depends on the ability of the pipeline to attribute the errors to the faulting instruction while maintaining the architectural state unpolluted by the faulting instruction and its subsequent instructions. For example, bus read errors on a non-critical word cannot be attributed to any executed instructions and will be imprecise. Similarly, when the processor is in the non-blocking mode (Section 10.1), load instructions may be retired before data is returned, and bus errors will always be imprecise.

Most errors related to address checks are precise, unless the instruction is split into micro-operations and the error is found not on the first micro-operation. For example, the PMP checks errors for the second micro-operation of misaligned memory accesses.

13.3.1 Vector Load and Store Instructions

13.3.1.1 Precise Exceptions

AX46MPV does not raise precise exceptions for inactive elements.

Vector load/store instructions may trigger the following precise exceptions:

- Load address misaligned

  - Address is in cacheable or non-cacheable regions and not naturally aligned to the size of the element.
- Load access fault
  - Address is in the device region and not naturally aligned to the size of the element.
  - PMA faults
  - PMP faults
  - Address in the device region
  - Address in the instruction local memory
  - Address in the data local memory
- Store address misaligned
  - Address is in cacheable or non-cacheable regions and not naturally aligned to the size of the element.
- Store access fault
  - Address is in the device region and not naturally aligned to the size of the element.
  - PMA faults
  - PMP faults
  - Address in the device region
  - Address in the instruction local memory
  - Address in the data local memory
- Load page fault
- Store page fault

Note

- Load and store address match trigger does not support vector load/store instructions.

See Section 17.4 for detailed behavior of precise exceptions.

13.3.1.2 Imprecise Exceptions

AX46MPV may raise imprecise exceptions for inactive elements if a transaction includes both active and inactive elements.

Vector load/store instructions may trigger the following imprecise exceptions:

- Imprecise ECC error interrupts
  - D-Cache ECC errors
- Bus read/write transaction error interrupts
  - Bus errors
  - L2-Cache ECC errors

13.4 Trap Handling

13.4.1 Entering the Trap Handler

When a trap occurs, the following operations are applied:

- mepc is set to the current program counter.
- mstatus is updated.
  - The MPP field is set to the current privilege mode.
  - The MPIE field is set to the MIE field.
  - The MIE field is set to 0.
- mcause is updated.
- mtval is updated on any address-misaligned, access-fault, or page-fault exceptions.
- The privilege mode is changed to M-mode.
- When mmisc_ctl.VEC_PLIC is 0, the program counter is set to the address specified by mtvec.
- When mmisc_ctl.VEC_PLIC is 1, the mtvec register will be the base address register of a vector table with 4-byte entries storing addresses pointing to interrupt service routines.
  - mtvec[0] is for exceptions and non-external local interrupts. For these traps, the mcause register records the trap type based on RISC-V definitions.
  - mtvec[i] is for external PLIC interrupt source *i* triggered through the mip.MEIP pending condition.
  - mtvec[1024+i] is for external PLIC interrupt source *i* triggered through
    - the mip.SEIP pending condition when mideleg.SEI = 0 for M/S/U systems.
    - the mip.UEIP pending condition when mideleg.UEI = 0 for M/U systems.
  - mtvec[2048+i] is for external PLIC interrupt source *i* triggered through the mip.UEIP pending condition when mideleg.UEI = 0 for M/S/U systems.
  - For external PLIC interrupts, the mcause register records the interrupt source ID. The RISC-V architecture defines a two-level stack of interrupt enable bits and privilege modes. To support nested traps, the trap handler should back up trap handling CSRs and enable the interrupt enable bit.

13.4.2 Returning from the Trap Handler

After handling a trap, the MRET instruction can be executed for returning to the instruction and the privilege context before the trap happened. Alternatively, the trap handler can assign new PC, privilege level, and/or interrupt enable status to mepc, mstatus.MPP, and mstatus.MPIE before MRET. Specifically, the following operations take place when an MRET instruction is executed:

- The program counter is set to mepc.
- The privilege mode is set to mstatus.MPP.

- mstatus is updated.
  - The MPP field is set to U-mode (or M-mode if U-mode is not supported).
  - The MIE field is set to the MPIE field.
  - The MPIE field is set to 1.

14 Reset and Non-Maskable Interrupts

14.1 Reset

After the processor core exits reset, the following operations are performed:

- All CSRs are set to their reset values.
- All integer registers (listed in Table 1) are set to zero.
- The BTB is initialized.
- Program execution starts at the reset vector specified in the mnvec CSR.

14.2 Non-Maskable Interrupts

Non-maskable interrupts (NMIs) are designed to handle hardware error conditions and are typically non-resumable. They are triggered through the `coreN_nmi` input signal. On the rising edge of this signal, the hart immediately jumps to an address stored in the mnvec register and transitions to Machine Mode, regardless of the state of the interrupt enable bit.

When an NMI is taken, the following operations are performed:

- The mepc register is written with the address of the next instruction.
- The mcause register is set to 1, indicating that the NMI was caused by the reset vector specified in the mnvec SCR.
- The mstatus.MPP field records the privilege mode before the NMI was taken.
- The mstatus.MPIE field is set to the value of mstatus.xIE before the NMI was taken, where “x” is the previous active privilege mode.
- The mstatus.MIE field is set to 0.

15 Instruction Throughput and Latency

This chapter lists the performance metrics of instructions.

Table 49: Performance Metrics of Instructions

Metric	Definition	Units
Latency	Number of cycles it takes for an instruction to produce a result	Cycles
Reciprocal Throughput	Average number of cycles per independent instructions. A value of 0.5 indicates that two instructions can be executed per clock cycle.	Cycles/Instruction
Penalty	Extra delay after executing an instruction	Cycles
Issue Latency	Minimum number of wait cycles before an instruction can be executed	Cycles

```
I1: lw  t0, 0(a0)                  # T1
I2: add t1, t0, t0                 # T2
I3: nop                            # T3
I4: add t3, t2, t2                 # T4
I5: add t3, t2, t2                 # T5
I6: add t3, t2, t2                 # T6
I7: add t3, t2, t2                 # T7
I8: add t3, t2, t2                 # T8
I9: add t3, t2, t2                 # T9
I10: add t3, t2, t2                # T10
I11: add t3, t2, t2                # T11
I12: add t3, t2, t2                # T12
I13: add t3, t2, t1                # T13
I14: nop                           # T14
I15: j I16                         # T15
I16: nop                           # T16
I17: csrr t0, minstret, zero       # T17
I18: nop                           # T18
```

Figure 4: Code Example for Instruction Performance

Figure 4 shows an example to illustrate instruction performance metrics. Instructions I1 to I18 represent the 1st to 18th instructions, while T1 to T18 represent their respective commit times.

`lw t0, 0(a0)` is a load instruction that produces a result to t0. The next instruction `add t1, t0, t0` is dependent on this result. The instruction latency of the load instruction is `T1-T2`.

I4 to I13 are ten add instructions, and the throughput of add instruction is `(T14-T3-1)/10`. The throughput assumes that instructions have no dependency to other instructions.

`j I16` is a jump instruction targeting I16. The misprediction penalty of this jump is `T16-T15-1`. The instruction penalty excludes cache instruction fetch penalty, such as miss penalty and unaligned fetches.

`csrr t0, minstret, zero` is a CSR read instruction, and I16 is the earlier instruction. The minimum issue latency of the CSR read instruction is `T16-T15`.

15.1 ALU Instructions

The latency of ALU instructions is 1 cycle, and the throughput is 1/2 cycle per instruction. ALU instructions include:

- Add/Sub: ADD, SUB, ADDI, ADDW, SUBW, ADDIW
- Shift: SLL, SRL, SRA, SLLI, SRLI, SRAI, SLLW, SRLW, SRAW, SLLIW, SRLIW, SRAIW
- Logical: AND, OR, XOR, ANDI, ORI, XORI
- Compare: SLT, SLTU, SLTI, SLTIU
- LUI and AUIPC
- Load effective address: LEA.H, LEA.W, LEA.D, LEA.B.ZE, LEA.H.ZE, LEA.W.ZE, LEA.D.ZE
- GP-implied add immediate: ADDIGP
- String processing: FFB, FFZMISM, FFMISM, FLMISM
- Bit field operation: BFOS, BFOZ
- Bit-manipulation ISA extensions
  - Zba and Zbs instructions
  - Logical with negate: ANDN, ORN, XNOR
  - Integer minimum/maximum: MAX, MAXU, MIN, MINU
  - Sign-extension and zero-extension: SEXT.B, SEXT.H, ZEXT.H
  - Bitwise rotation: ROL, ROLW, ROR, RORI, RORIW, RORW
  - OR combine: ORC.B
  - Byte-reverse: REV8
- Zca, Zcb instructions
  - 16 bits encoding Half-word, byte sign-extend and zero-extend instructions: c.zext.b, c.sext.b, c.zext.h, c.sext.h, c.zext.w

- 16 bits encoding NOT instructions: c.not

- Zcmp instructions
  - Move two registers: cm.mva01s, cm.mvsa01

15.2 BITMANIP Instructions

The latency of BITMANIP instructions is 1 cycle, and the throughput is 1/2 cycle per BITMANIP instruction. BITMANIP instructions include:

- Count leading/trailing zero bits: CLZ, CLZW, CTZ, CTZW
- Count population: CPOP, CPOPW
- Carry-less multiplication: CLMUL, CLMULH, CLMULR

15.3 Dual-Issue Capability

15.3.1 Dual-Issue Capability of Integer Instructions

AX46MPV can simultaneously execute two independent instructions. Table 50 shows the dual-issue capability of integer instruction pairs.

Table 50: Dual-Issue Capability

The First Instruction	The Second Instruction	Dual-Issue
ALU/BITMANIP	ALU/BITMANIP	Yes
ALU/BITMANIP	Branch/Jump	Yes
ALU/BITMANIP	Load/Store	Yes
ALU/BITMANIP	MUL/DIV	Yes
ALU/BITMANIP	DSP (2R/1W)	Yes
ALU/BITMANIP	DSP (3R/4R/2W)	No
ALU/BITMANIP	ACE (2R/1W)	Yes
ALU/BITMANIP	ACE (3R/4R/2W)	No
ALU/BITMANIP	ZC (Zcmp/Zcmt)	No
Branch/Jump	ALU/BITMANIP	Yes
Branch/Jump	Branch/Jump	Yes
Branch/Jump	Load/Store	Yes
Branch/Jump	MUL/DIV	Yes

Continued on next page...

Table 50: (continued)

The First Instruction	The Second Instruction	Dual-Issue
Branch/Jump	DSP (2R/1W)	Yes
Branch/Jump	DSP (3R/4R/2W)	No
Branch/Jump	ACE (2R/1W)	Yes
Branch/Jump	ACE (3R/4R/2W)	No
Branch/Jump	ZC (Zcmp/Zcmt)	No
Load/Store	ALU/BITMANIP	Yes
Load/Store	Branch/Jump	Yes
Load/Store	Load/Store	Yes
Load/Store	MUL/DIV	Yes
Load/Store	DSP (2R/1W)	Yes
Load/Store	DSP (3R/4R/2W)	No
Load/Store	ACE (2R/1W)	Yes
Load/Store	ACE (3R/4R/2W)	No
Load/Store	ZC (Zcmp/Zcmt)	No
MUL/DIV	ALU/BITMANIP	Yes
MUL/DIV	Branch/Jump	Yes
MUL/DIV	Load/Store	Yes
MUL/DIV	MUL/DIV	No
MUL/DIV	DSP (2R/1W)	Yes
MUL/DIV	DSP (3R/4R/2W)	No
MUL/DIV	ACE (2R/1W)	Yes
MUL/DIV	ACE (3R/4R/2W)	No
MUL/DIV	ZC (Zcmp/Zcmt)	No
DSP (2R/1W)	ALU/BITMANIP	Yes
DSP (2R/1W)	Branch/Jump	Yes
DSP (2R/1W)	Load/Store	Yes
DSP (2R/1W)	MUL/DIV	Yes
DSP (2R/1W)	DSP (2R/1W)	No
DSP (2R/1W)	DSP (3R/4R/2W)	No
DSP (2R/1W)	ACE (2R/1W)	Yes
DSP (2R/1W)	ACE (3R/4R/2W)	No
DSP (2R/1W)	ZC (Zcmp/Zcmt)	No
DSP (3R/4R/2W)	ALU/BITMANIP	No

Continued on next page...

Table 50: (continued)

The First Instruction	The Second Instruction	Dual-Issue
DSP (3R/4R/2W)	Branch/Jump	No
DSP (3R/4R/2W)	Load/Store	No
DSP (3R/4R/2W)	MUL/DIV	No
DSP (3R/4R/2W)	DSP (2R/1W)	No
DSP (3R/4R/2W)	DSP (3R/4R/2W)	No
DSP (3R/4R/2W)	ACE (2R/1W)	No
DSP (3R/4R/2W)	ACE (3R/4R/2W)	No
DSP (3R/4R/2W)	ZC (Zcmp/Zcmt)	No
ACE (2R/1W)	ALU/BITMANIP	Yes
ACE (2R/1W)	Branch/Jump	Yes
ACE (2R/1W)	Load/Store	Yes
ACE (2R/1W)	MUL/DIV	Yes
ACE (2R/1W)	DSP (2R/1W)	Yes
ACE (2R/1W)	DSP (3R/4R/2W)	No
ACE (2R/1W)	ZC (Zcmp/Zcmt)	No
ACE (2R/1W)	ACE (2R/1W)	No
ACE (2R/1W)	ACE (3R/4R/2W)	No
ACE (3R/4R/2W)	ALU/BITMANIP	No
ACE (3R/4R/2W)	Branch/Jump	No
ACE (3R/4R/2W)	Load/Store	No
ACE (3R/4R/2W)	MUL/DIV	No
ACE (3R/4R/2W)	DSP (2R/1W)	No
ACE (3R/4R/2W)	DSP (3R/4R/2W)	No
ACE (3R/4R/2W)	ZC (Zcmp/Zcmt)	No
ACE (3R/4R/2W)	ACE (2R/1W)	No
ACE (3R/4R/2W)	ACE (3R/4R/2W)	No

Note

- An ACE (2R/1W) instruction reads 2 integer registers, and writes 1 register.
- An ACE (3R/4R/2W) instruction reads 3/4 integer registers, or writes 2 registers.
- A DSP (2R/1W) instruction reads 2 integer registers, and writes 1 register.
- A DSP (3R/4R/2W) instruction reads 3/4 integer registers, or writes 2 registers.
- The dual-issue capability is No if the first or second instruction is a Zcmp (push/pop/popret/popretz/mva01s/mvsa01) or Zcmt (jt/jalt) instruction.

15.3.2 Dual-Issue Capability of Floating-Pointing Instructions

Floating-point instructions are executed in 5 functional units:

- FMAC: FADD.x, FSUB.x, FMUL.x, FMADD.x, FMSUB.x, FNMADD.x, and FNMSUB.x
- FDIV: FDIV.x and FSQRT.x
- FMV: FSGNJ.x, FSGNJN.x, FSGNJX.x, FMV.D.X, FMV.W.X, FMV.H.X, FMV.X.D, FMV.X.W, and FMV.X.H
- FMISC: FMIN.x, FMAX.x, FCLASS.x, FEQ.x, FLT.x, FLE.x, FCVT.D.x, FCVT.S.x, FCVT.H.x, FCVT.BF16.x, FCVT.L[U].x, and FCVT.W[U].x
- LSU: FLH, FSH, FLW, FSW, FLD, FSD, C.FLWSP, C.FSWSP, C.FLDSP, C.FSDSP, C.FLW, C.FSW, C.FLD, and C.FSD

AX46MPV can issue two floating-point load/store instructions simultaneously as an instruction pair. Additionally, AX46MPV can issue two load/store instructions, including floating-point load/store, unless it encounters an issue restriction. The floating-point instruction can be either the first or the second instruction in the pair. Issue restrictions include:

- Any instruction of the instruction pair reads 3 or 4 integer registers.
- Any instruction of the instruction pair writes 2 integer registers.
- Both instructions use the same functional unit.
- The first instruction reads 3 floating-point registers, and the second instruction reads 1, 2, or 3 floating-point registers.
- The first instruction reads 2 or 3 floating-point registers, and the second one reads 3 floating-point registers.

15.4 Throughput and Latency for Aligned Load Instructions

The throughput and latency of load instructions are summarized in the following tables.

Table 51: Load Instruction Throughput and Latency from Private Memory Subsystem

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
load doubleword/word from DLM/D-Cache	1/2	0
load halfword/byte from DLM/D-Cache	1/2	1
load doubleword/word from ILM	1	2
load halfword/byte from ILM	1	3
load doubleword/word from Read Data Buffer	1/2	0
load halfword/byte from Read Data Buffer	1/2	1
load doubleword/word from HVM (`mmisc_ctl.NBLD_EN` = 0)	15/2	7
load halfword/byte from HVM (`mmisc_ctl.NBLD_EN` = 0)	15/2	8
load halfword/byte/doubleword/word from HVM (`mmisc_ctl.NBLD_EN` = 1)	1/2	11
load doubleword/word from PPI (device non-bufferable)	11/2	5
load halfword/byte from PPI (device non-bufferable)	11/2	6
load doubleword/word from PPI (device bufferable)	11/2	5
load halfword/byte from PPI (device bufferable)	11/2	6

The latencies are determined by the configurations used when load instructions access the shared memory subsystem. The following table, Table 52, presents the relevant configurations along with their corresponding latencies.

Table 52: Configurations and Corresponding Aligned Load Performance Table

Number of Processor Cores	L3-Cache Size	Aligned Load Throughput and Latency Table
1	=0	Table 53, Configuration 1
1	>0	Table 54, Configuration 2
2 – 4	any value	Table 54, Configuration 2
4 – 8	any value	Table 55, Configuration 3

Table 53: Load Instruction Throughput and Latency from Shared Memory Subsystem(Configuration 1)

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
load doubleword/word from Device	17/2	8
load halfword/byte from Device	17/2	9
load doubleword/word from Non-cacheable memory (`mmisc_ctl.NBLD_EN` = 0)	17/2	8
load halfword/byte from Non-cacheable memory (`mmisc_ctl.NBLD_EN` = 0)	17/2	9
load halfword/byte/doubleword/word from Non-cacheable memory (`mmisc_ctl.NBLD_EN` = 1)	1/2	11

Table 54: Load Instruction Throughput and Latency from Shared Memory Subsystem(Configuration 2)

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
load doubleword/word from Device	21/2	10
load halfword/byte from Device	21/2	11
load doubleword/word from Non-cacheable memory (`mmisc_ctl.NBLD_EN` = 0)	21/2	10
load halfword/byte from Non-cacheable memory (`mmisc_ctl.NBLD_EN` = 0)	21/2	11

Continued on next page...

Table 54: (continued)

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
load halfword/byte/doubleword/word from Non-cacheable memory (mmisc_ctl.NBLD_EN = 1)	1/2	13
load doubleword/word from SPP (device non-bufferable)	21/2	10
load halfword/byte from SPP (device non-bufferable)	21/2	11
load doubleword/word from SPP (device bufferable)	21/2	10
load halfword/byte from SPP (device bufferable)	21/2	11

Table 55: Load Instruction Throughput and Latency from Shared Memory Subsystem(Configuration 3)

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
load doubleword/word from Device	25/2	12
load halfword/byte from Device	25/2	13
load doubleword/word from Non-cacheable memory (mmisc_ctl.NBLD_EN = 0)	25/2	12
load halfword/byte from Non-cacheable memory (mmisc_ctl.NBLD_EN = 0)	25/2	13
load halfword/byte/doubleword/word from Non-cacheable memory (mmisc_ctl.NBLD_EN = 1)	1/2	15
load doubleword/word from SPP (device non-bufferable)	25/2	12
load halfword/byte from SPP (device non-bufferable)	25/2	13
load doubleword/word from SPP (device bufferable)	25/2	12
load halfword/byte from SPP (device bufferable)	25/2	13

Note

- The latency assumes that the dependent instruction is executed in late ALU.
- The 0-cycle latency assumes that the dependent instruction and the load instruction are dual-issued.
- The latency assumes that the load hits cache.
- The latency assumes that bus latency is one cycle after the requested address is issued.
- The latency assumes that the latency of memory access is 1 cycle.
- The latency assumes that Core Interface is synchronous.
- The latency assumes that the SMEM_CLK and BUS_CLK clock ratio is 1:1.
- The latency assumes that Private Peripheral Clock is synchronous.
- The latency assumes that the CORE_CLK and PPI_CLK clock ratio is 1:1.
- The throughput assumes that Number of Scalar Load/Store Requests is 32.
- The throughput assumes that Number of HVM Banks is 2.
- The throughput assumes that Depth of Scalar Read Data Buffer is 16.

15.5 Throughput and Latency for Misaligned Load Instructions

Misaligned data accesses will incur extra overhead in latency and throughput. See the following table for details.

Table 56: Misaligned Load Throughput and Latency from Private Memory Subsystem

Instruction	Misaligned Address	Throughput (Cycles/Instruction)	Latency (Cycles)
load halfword from DLM/D-Cache	addr[2:0] = 1/3/5	1/2	1
	addr[2:0] = 7	3/2	2
load word from DLM/D-Cache	addr[2:0] = 1/2/3	1/2	1
	addr[2:0] = 5/6/7	3/2	2
load doubleword from DLM/D-Cache	addr[2:0] $\neq$ 0	3/2	2
load halfword from ILM	addr[2:0] = 1/3/5	1	3
	addr[2:0] = 7	3	4
load word from ILM	addr[2:0] = 1/2/3	1	3
	addr[2:0] = 5/6/7	3	4
load doubleword from ILM	addr[2:0] $\neq$ 0	3	4
load halfword from Read Data Buffer	addr[2:0] = 1/3/5	1/2	1
	addr[2:0] = 7	3/2	2
load word from Read Data Buffer	addr[2:0] = 1/2/3	1/2	1
	addr[2:0] = 5/6/7	3/2	2
load doubleword from Read Data Buffer	addr[2:0] $\neq$ 0	3/2	2
load halfword from HVM	addr[2:0] = 1/3/5	15/2	8
	addr[2:0] = 7	31/2	16
load word from HVM	addr[2:0] = 1/2/3	15/2	8
	addr[2:0] = 5/6/7	31/2	16
load doubleword from HVM	addr[2:0] $\neq$ 0	31/2	16

The latencies are determined by the configurations used when load instructions access the shared memory subsystem. The following table, Table 57, presents the relevant configurations along with their corresponding latencies.

Table 57: Configurations and Corresponding Misaligned Load Performance Table

Number of Processor Cores	L3-Cache Size	Misaligned Load Throughput and Latency Table
1	=0	Table 58, Configuration 1
1	>0	Table 59, Configuration 2
2 – 4	any value	Table 59, Configuration 2
4 – 8	any value	Table 60, Configuration 3

Table 58: Misaligned Load Throughput and Latency from Shared Memory Subsystem(Configuration 1)

Instruction	Misaligned Address	Throughput (Cycles/Instruction)	Latency (Cycles)
load halfword from Non-cacheable memory	addr[2:0] = 1/3/5	17/2	9
	addr[2:0] = 7	35/2	18
load word from Non-cacheable memory	addr[2:0] = 1/2/3	17/2	9
	addr[2:0] = 5/6/7	35/2	18
load doubleword from Non-cacheable memory	addr[2:0] $\neq$ 0	35/2	18

Table 59: Misaligned Load Throughput and Latency from Shared Memory Subsystem(Configuration 2)

Instruction	Misaligned Address	Throughput (Cycles/Instruction)	Latency (Cycles)
load halfword from Non-cacheable memory	addr[2:0] = 1/3/5	21/2	11
	addr[2:0] = 7	43/2	22
load word from Non-cacheable memory	addr[2:0] = 1/2/3	21/2	11
	addr[2:0] = 5/6/7	43/2	22
load doubleword from Non-cacheable memory	addr[2:0] $\neq$ 0	43/2	22

Table 60: Misaligned Load Throughput and Latency from Shared Memory Subsystem(Configuration 3)

Instruction	Misaligned Address	Throughput (Cycles/Instruction)	Latency (Cycles)
load halfword from Non-cacheable memory	addr[2:0] = 1/3/5	25/2	13
	addr[2:0] = 7	51/2	26
load word from Non-cacheable memory	addr[2:0] = 1/2/3	25/2	13
	addr[2:0] = 5/6/7	51/2	26
load doubleword from Non-cacheable memory	addr[2:0] $\neq$ 0	51/2	26

15.6 Multiply Instructions

The latency and throughput of multiply instructions depend on the multiplier implementation.

Table 61: Multiply Instruction Throughput and Latency: Radix Multiplier

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
MULHU, MULW	4 + 64 / LOG2(MUL_RADIX)	4 + 64 / LOG2(MUL_RADIX)
MUL, MULH, MULHSU, C.MUL without negative operands	4 + 64 / LOG2(MUL_RADIX)	4 + 64 / LOG2(MUL_RADIX)
MUL, MULH, MULHSU, C.MUL with negative operands	5 + 64 / LOG2(MUL_RADIX)	5 + 64 / LOG2(MUL_RADIX)

Note

- When **Multiplier Implementation** is configured to “radix2”, MUL_RADIX is 2.
- When **Multiplier Implementation** is configured to “radix4”, MUL_RADIX is 4.
- When **Multiplier Implementation** is configured to “radix16”, MUL_RADIX is 16.
- When **Multiplier Implementation** is configured to “radix256”, MUL_RADIX is 256.

Table 62: Multiply Instruction Throughput and Latency: Fast Multiplier

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
MUL, MULH, MULHU, MULHSU, MULW, C.MUL	1	1

15.7 Divide and Remainder Instructions

Divide and remainder instructions are implemented using the non-restoring division algorithm with early termination detection.

Table 63: Divide and Remainder Instruction Throughput and Latency

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
DIVU, REMU	6 – 69	6 – 69
DIV, REM	6 – 69	6 – 69
DIVUW, REMUW	6 – 37	6 – 37
DIVW, REMW	6 – 37	6 – 37

Note

- The latency of 0x8000000000000001 / 0x1 is 69 cycles.
- The latency of 0x0000000080000001 / 0x1 is 37 cycles.
- The latency of 0x0000000000000000 / 0x1 is 6 cycles.

15.8 Branch and Jump Instructions

Branch mis-prediction penalty is 5 cycles when resolved in the EX stage and 7 cycles when resolved in the LX stage. AX46MPV can issue two branch and jump instructions in a cycle. However, the execution throughput of branch and jump instructions is bounded by the branch prediction logics.

Table 64: Branch and Jump Instructions Penalty and Throughput

Extension	Instruction	Resolve Stage	Penalty(Cycles)	Throughput (Cycles/Instruction)
RV32I Control Transfer Instructions	JAL	EX	5	1
	JALR	EX	5	1
	JALR	LX	7	1
	BEQ	EX	5	1/2
	BEQ	LX	7	1/2
	BNE	EX	5	1/2
	BNE	LX	7	1/2
	BLT	EX	5	1/2
	BLT	LX	7	1/2
	BLTU	EX	5	1/2
	BLTU	LX	7	1/2
	BGE	EX	5	1/2
	BGE	LX	7	1/2
	BGEU	EX	5	1/2
	BGEU	LX	7	1/2
C Extension Control Transfer Instructions	C.J	EX	5	1
	C.JAL	EX	5	1
	C.JR	EX	5	1
	C.JR	LX	7	1
	C.JALR	EX	5	1
	C.JALR	LX	7	1
	C.BEQZ	EX	5	1/2
	C.BEQZ	LX	7	1/2
	C.BNEZ	EX	5	1/2
	C.BNEZ	LX	7	1/2
Andes Performance Extension	BBC	EX	5	1/2
	BBC	LX	7	1/2
	BBS	EX	5	1/2
	BBS	LX	7	1/2
	BEQC	EX	5	1/2
	BEQC	LX	7	1/2
	BNEC	EX	5	1/2

Continued on next page...

Table 64: (continued)

Extension	Instruction	Resolve Stage	Penalty(Cycles)	Throughput (Cycles/Instruction)
	BNEC	LX	7	1/2

Note

- The ideal throughput of branch instructions is 1/2 cycle per independent branch instruction, assuming dual-issue execution and that all branches are resolved as not taken.

15.9 EXEC.IT Instructions

An EXEC.IT instruction is decoded at the instruction slot 0 at the ID stage. When an EXEC.IT instruction is decoded, the pipeline requires extra 4 cycles to fetch the instruction from the CoDense instruction table. This 4-cycle penalty may be hidden if the execution pipeline is stalled by earlier instructions.

15.10 ZCE (Zcmt) Instructions

An Zcmt instruction is decoded at the instruction slot 0 at the ID stage. When an ZCE (Zcmt) instruction is decoded, the pipeline requires extra 4 cycles to fetch the instruction from the jump vector table (jvt). This 4-cycle penalty may be hidden if the execution pipeline is stalled by earlier instructions.

15.11 Trap Return Instructions

A trap return instruction flushes the entire pipeline, and the penalty is 8 cycles.

15.12 FENCE Instructions

The `FENCE` instruction maintains memory ordering while executing in the background, allowing the pipeline to proceed without stalling even if prior memory operations are still inflight. However, to ensure proper memory ordering, subsequent load and store instructions may still incur additional penalties. In addition, CSR accesses are classified as device I/O and must be ordered with respect to `FENCE` instruction, which may introduce additional penalties.

The table below summarizes the behavior and performance impact of various instruction sequences involving `FENCE`:

Table 65: Performance Impact of Various Instruction Sequences Involving FENCE

Prior Instruction	Subsequent Instruction	Impact
Load and Store	Fence	No additional issue latency and penalty; FENCE executes in the background.
CSR	Fence	Subsequent instruction stalls at the II stage and waits for outstanding memory access; the issue latency is 4 cycles if there is no outstanding memory access.
Fence	Load and Store	Subsequent instruction stalls at the IX stage and waits for outstanding memory access; the penalty is 3 cycles if there is no outstanding memory access.
Fence	CSR	Subsequent instruction stalls at the II stage and waits for outstanding memory access; the penalty is 4 cycles if there is no outstanding memory access.

15.13 Throughput and Latency for Zicntr Instructions

The throughput and latency for Zicntr instructions are summarized in the following table.

Table 66: Zicntr Instruction Throughput and Latency

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
rdinstret	2	2
rdcycle	2	2
rdtime	13	9

Note

- The throughput and latency of the rdtime instruction are based on the assumption that CORE_CLK and BUS_CLK are synchronous and their frequency ratio is 1:1.

15.14 Scalar Floating-Point Instructions

The instruction latencies of floating-point instructions are shown in the following tables.

Note

- The latencies are not affected by the setting of `umisc_ctl.FP_MODE`.

Table 67: Throughput and Latency for Instructions Executed in FMAC When Double-Precision Is Supported

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles) to FMAC, FDIV, FMV, FMISC	Latency (Cycles) to LSU
FADD.x, FSUB.x	1	4	2
FMUL.x	1	4	2
FMADD.x, FMSUB.x, FNMADD.x, FNMSUB.x	1	4	2

Table 68: Throughput and Latency for Instructions Executed in FMAC When Double-Precision Is Not Supported

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles) to FMAC, FDIV, FMV, FMISC	Latency (Cycles) to LSU
FADD.x, FSUB.x	1	3	1
FMUL.x	1	3	1
FMADD.x, FMSUB.x, FNMADD.x, FNMSUB.x	1	3	1

Table 69: Throughput and Latency for Instructions Executed in FDIV/FMV/FMISC

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles) to FMAC, FDIV, FMV, FMISC	Latency (Cycles) to LSU/ALU
FDIV.D	32	32	33
FSQRT.D	31	31	32
FDIV.S	18	18	19
FSQRT.S	17	17	18

Continued on next page...

Table 69: (continued)

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles) to FMAC, FDIV, FMV, FMISC	Latency (Cycles) to LSU/ALU
FDIV.H	11	11	12
FSQRT.H	10	10	11
FSGNJ.x, FSGNJN.x, FSGNJX.x	1	1	1
FMV.D.X, FMV.W.X, FMV.H.X	1	1	1
FMV.X.D, FMV.X.W, FMV.X.H	1	1	1
FMIN.x, FMAX.x	1	2	1
FCLASS.x, FEQ.x, FLT.x, FLE.x	1	2	1
FCVT.D.x, FCVT.S.x, FCVT.H.x, FCVT.BF16.x	1	2	1
FCVT.L[U].x, FCVT.W[U].x	1	2	1

15.15 DSP Instructions

The instruction latencies of DSP instruction groups are shown in the following tables.

DSP instructions include:

- Non-MUL Arithmetic Operation (8/16/32/64-bit): ADD8, SUB16, CLZ32, RSUB64, ...
- MUL8*8: KHM8, ...
- MUL8*8 with 32-bit ADD/SUB: SMAQA, ...
- MUL16*16: KHM16, ...
- MUL16*16 with 32-bit ADD/SUB: KMDA, ...
- MUL16*16 with 64-bit ADD/SUB: SMAL, ...
- MUL32*32: SMMUL, ...
- MUL32*32 with 32-bit ADD/SUB: KMMAC, ...
- MUL32*32 with 64-bit ADD/SUB: KMAR64, ...
- MUL32*16: SMMWB, ...
- MUL32*16 with 32-bit ADD/SUB: KMMAWB, ...

Table 70: DSP Instruction Throughput and Latency

Instruction	Throughput (Cycles/Instruction)	Latency (Cycles)
Non-MUL Arithmetic Operation	1	1
MUL8*8	1	1
MUL8*8 with 32-bit ADD/SUB	1	2
MUL16*16	1	2
MUL16*16 with 32-bit ADD/SUB	1	2
MUL16*16 with 64-bit ADD/SUB	1	3
MUL32*32	1	2
MUL32*32 with 32-bit ADD/SUB	1	2
MUL32*32 with 64-bit ADD/SUB	1	3
MUL32*16	1	2
MUL32*16 with 32-bit ADD/SUB	1	2

15.16 Throughput and Latency for PUSH/POP/POPRET/POPRETZ Instructions

The throughput and latency for push/pop/popret/popretz instructions are summarized in the following table.

Table 71: PUSH/POP/POPRET/POPRETZ Instruction Throughput and Latency

Instruction	XREG_LIST	Throughput (Cycles/Instruction)	Latency (Cycles)
C.PUSH	ra	3	3
	ra,s0	4	4
	ra,s0-s1	5	5
	ra,s0-s2	6	6
	ra,s0-s3	7	7
	ra,s0-s4	8	8
	ra,s0-s5	9	9
	ra,s0-s6	10	10
	ra,s0-s7	11	11
	ra,s0-s8	12	12
	ra,s0-s9	13	13
	ra,s0-s11	15	15
C.POP	ra	3	3
	ra,s0	4	4
	ra,s0-s1	5	5
	ra,s0-s2	6	6
	ra,s0-s3	7	7
	ra,s0-s4	8	8
	ra,s0-s5	9	9
	ra,s0-s6	10	10
	ra,s0-s7	11	11
	ra,s0-s8	12	12
	ra,s0-s9	13	13
	ra,s0-s11	15	15
C.POPRET	ra	4	4
	ra,s0	5	5
	ra,s0-s1	6	6
	ra,s0-s2	7	7

Continued on next page...

Table 71: (continued)

Instruction	XREG_LIST	Throughput (Cycles/Instruction)	Latency (Cycles)
	ra,s0-s3	8	8
	ra,s0-s4	9	9
	ra,s0-s5	10	10
	ra,s0-s6	11	11
	ra,s0-s7	12	12
	ra,s0-s8	13	13
	ra,s0-s9	14	14
	ra,s0-s11	16	16
C.POPRETZ	ra	5	5
	ra,s0	6	6
	ra,s0-s1	7	7
	ra,s0-s2	8	8
	ra,s0-s3	9	9
	ra,s0-s4	10	10
	ra,s0-s5	11	11
	ra,s0-s6	12	12
	ra,s0-s7	13	13
	ra,s0-s8	14	14
	ra,s0-s9	15	15
	ra,s0-s11	17	17

Note

- The latency assumes that the dependent instruction and the push/pop/popret/popretz instruction are not dual-issued.
- The latency assumes that the non-blocking load is disabled.
- The latency assumes that the load hits cache.
- The latency assumes that the popret/popretz instruction is executed without misprediction.

15.17 Vector Instructions

The instruction throughput and latency of a vector instruction depend on the settings of the LMUL and SEW fields in the `vtype` CSR. In this section, LMUL and SEW specifically refer to the fields in `vtype`, not to EMUL and EEW of the source and destination operands. Throughput of a vector instruction refers to the number of cycles needed to complete a vector instruction. Result latency of a vector instruction signifies the number of cycles before the first result becomes available for its subsequent instruction.

15.17.1 Dual-Issue Capability of Vector Instructions

Vector instructions are executed in function units:

- VALU: See Section 15.17.4
- VMAC: See Section 15.17.5
- VFMIS: See Section 15.17.6
- VLSU: See Section 15.17.7
- VMASK: See Section 15.17.8
- VPERMUT: See Section 15.17.9
- VDIV: See Section 15.17.10
- VFDIV: See Section 15.17.11
- VMAC2: See Section 15.17.5
- VACE: Andes Custom Extension - Vector instructions
- VSP: Andes Custom Extension - Streaming port instructions

AX46MPV can issue a vector instruction with another instruction as an instruction pair, unless restricted by certain conditions. The vector instruction can be either the first or the second instruction in the pair. The instruction pair may consist of two vector instructions. Instruction pairing is not allowed under the following conditions:

- The first instruction is a Load/Store instruction, and the second is a vector instruction.
- The first instruction is a Branch/Jump instruction, and the second is a vector instruction.
- The first instruction is a VLSU instruction, and the second is a vector instruction.
- The first instruction is a VSP instruction, and the second is a VLSU instruction.
- The first instruction reads more than 2 integer registers.
- The first instruction writes 2 integer registers.
- The first instruction reads 3 floating-point registers, and the second reads 1 floating-point register.
- The second instruction reads more than 2 integer registers.

15.17.2 VRF Ports

Vector functional units share VRF read and write ports. Table 72 and Table 73 show the assignments of the VRF ports. VRF port conflicts may introduce extra bubbles.

Table 72: VRF Read Ports

Read Port	0	1	2	3	4	5	6	7	8	9	10
VALU	vs1	vs2	vs3								
VFMIS	vs3	vs1	vs2								
VMASK	vs1	vs2	vs3								
VACE	vs1	vs2	vs3								
VMAC				vs1	vs2	vs3					
VMAC2									vs1	vs2	vs3
VLSU							vs2	vs3			
VSP								vs3			
VPERMUT							vs1	vs2			
VDIV							vs1	vs2			
VFDIV							vs1	vs2			

Table 73: VRF Write Ports

Write Port	0	1	2	3	4	5	6
VALU	vd						
VMASK	vd						
VACE	vd						
VMAC		vd					
VMAC2						vd	
VFMIS			vd				
VLSU/VROB				vd			vd
VSP					vd		
VPERMUT					vd		
VDIV					vd		
VFDIV					vd		

15.17.3 Vector Chaining

AX46MPV supports vector chaining between two instructions in different functional units. Individual results can be forwarded before the instruction is completed. An example code is shown below, and Table 74 shows the pipeline diagram.

```
vadd.vx         v0, v0, a4
vfcvt.f.x.v     v0, v0
vadd.vx         v0, v0, a5
```


Table 74: A Vector Chaining Example (LMUL = 4)

Cycle	VALU-VS	VALU Stage 0	VALU-VD	VFMIS-VS	VFMIS Stage 0	VFMIS Stage 1	VFMIS-VD
0	v0						
1	v1	v0+a4					
2	v2	v1+a4	v0	v0			
3	v3	v2+a4	v1	v1	vfcvt.f.x.v v0		
4		v3+a4	v2	v2	vfcvt.f.x.v v1	vfcvt.f.x.v v0	
5	v0		v3	v3	vfcvt.f.x.v v2	vfcvt.f.x.v v1	v0
6	v1	v0+a5			vfcvt.f.x.v v3	vfcvt.f.x.v v2	v1
7	v2	v1+a5	v0			vfcvt.f.x.v v3	v2
8	v3	v2+a5	v1				v3
9		v3+a5	v2				
10			v3				

15.17.4 VALU Instructions

Table 75: VALU Instruction Throughput and Latency

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vadd	(VLEN/DLEN)*LMUL	2
vsub	(VLEN/DLEN)*LMUL	2
vwadd.w* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwadd.v* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwsub.w* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwsub.v* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwadd.w* (LMUL<1)	(VLEN != DLEN) ? 4 : 1	2
vwadd.v* (LMUL<1)	(VLEN != DLEN) ? 4 : 2	2
vwsub.w* (LMUL<1)	(VLEN != DLEN) ? 4 : 1	2
vwsub.v* (LMUL<1)	(VLEN != DLEN) ? 4 : 2	2
vwaddu.w* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwaddu.v* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwsubu.w* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwsubu.v* (LMUL≥1)	(VLEN/DLEN)*LMUL*2	2
vwaddu.w* (LMUL<1)	(VLEN != DLEN) ? 4 : 1	2
vwaddu.v* (LMUL<1)	(VLEN != DLEN) ? 4 : 2	2
vwsubu.w* (LMUL<1)	(VLEN != DLEN) ? 4 : 1	2
vwsubu.v* (LMUL<1)	(VLEN != DLEN) ? 4 : 2	2
vrsub	(VLEN/DLEN)*LMUL	2
vadc	(VLEN/DLEN)*LMUL	2
vsbc	(VLEN/DLEN)*LMUL	2
vminu	(VLEN/DLEN)*LMUL	2
vmin	(VLEN/DLEN)*LMUL	2
vmaxu	(VLEN/DLEN)*LMUL	2
vmax	(VLEN/DLEN)*LMUL	2
vand	(VLEN/DLEN)*LMUL	2
vor	(VLEN/DLEN)*LMUL	2
vxor	(VLEN/DLEN)*LMUL	2
vmv	(VLEN/DLEN)*LMUL	2
vmerge	(VLEN/DLEN)*LMUL	2
vsadd	(VLEN/DLEN)*LMUL	2

Continued on next page...

Table 75: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vsaddu	(VLEN/DLEN)*LMUL	2
vssubu	(VLEN/DLEN)*LMUL	2
vssub	(VLEN/DLEN)*LMUL	2
vsll	(VLEN/DLEN)*LMUL	2
vsrl	(VLEN/DLEN)*LMUL	2
vsra	(VLEN/DLEN)*LMUL	2
vssrl	(VLEN/DLEN)*LMUL	2
vssra	(VLEN/DLEN)*LMUL	2
vaaddu	(VLEN/DLEN)*LMUL	2
vaadd	(VLEN/DLEN)*LMUL	2
vasubu	(VLEN/DLEN)*LMUL	2
vasub	(VLEN/DLEN)*LMUL	2
vnclipu (LMUL≥1)	(VLEN/DLEN)*LMUL*2	3
vnclip (LMUL≥1)	(VLEN/DLEN)*LMUL*2	3
vnsrl (LMUL≥1)	(VLEN/DLEN)*LMUL*2	3
vnsra (LMUL≥1)	(VLEN/DLEN)*LMUL*2	3
vnclipu (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 3 : 2
vnclip (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 3 : 2
vnsrl (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 3 : 2
vnsra (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 3 : 2
vmseq	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsne	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsltu	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmslt	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsleu	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsle	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsgtu	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsgt	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmadc	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmsbc	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vredsum	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredand	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredor	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1

Continued on next page...

Table 75: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vredxor	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredminu	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredmin	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredmaxu	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vredmax	vredsum-micro-ops + (VLEN != DLEN)	vredsum-micro-ops + 1
vwredsum	vwredsum-micro-ops + (VLEN != DLEN)	vwredsum-micro-ops + 1
vwredsumu	vwredsum-micro-ops + (VLEN != DLEN)	vwredsum-micro-ops + 1

Note

- vredsum-micro-ops = (VLEN/DLEN)*LMUL + LOG2(DLEN/64) * 2 + LOG2(64/SEW)
- vwredsum-micro-ops = (VLEN/DLEN)*LMUL*2 + LOG2(DLEN/64) * 2 + LOG2(64/2/SEW)
- When fractional LMUL is used, the **LMUL** used in throughput and latency calculation is 1.

15.17.5 VMAC Instructions

15.17.5.1 VMAC Integer Instructions

Table 76: VMAC Integer Instruction Throughput and Latency

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
RISC-V Vector Extension		
vmul	(VLEN/DLEN)*LMUL	4
vmulh	(VLEN/DLEN)*LMUL	4
vmulhu	(VLEN/DLEN)*LMUL	4
vmulhsu	(VLEN/DLEN)*LMUL	4
vmacc	(VLEN/DLEN)*LMUL	4
vnmsac	(VLEN/DLEN)*LMUL	4
vmadd	(VLEN/DLEN)*LMUL	4
vnmsub	(VLEN/DLEN)*LMUL	4
vsmul	(VLEN/DLEN)*LMUL	4
vwmul (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmulu (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmulsu (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmaccu (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmaccsu (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmaccus (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vwmul (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmulu (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmulsu (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmaccu (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmacc (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmaccsu (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
vwmaccus (LMUL<1)	(VLEN != DLEN) ? 4 : 1	4
Andes V5 Vector Quad-Widening Integer Multiply-Add Extension		
vqmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*4	4
vqmaccu (LMUL≥1)	(VLEN/DLEN)*LMUL*4	4
vqmaccsu (LMUL≥1)	(VLEN/DLEN)*LMUL*4	4
vqmaccsu (LMUL≥1)	(VLEN/DLEN)*LMUL*4	4
vqmacc (LMUL==1/2)	(VLEN != DLEN) ? 8 : 2	4

Continued on next page...

Table 76: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vqmaccu (LMUL==1/2)	(VLEN != DLEN) ? 8 : 2	4
vqmaccsu (LMUL==1/2)	(VLEN != DLEN) ? 8 : 2	4
vqmaccsu (LMUL==1/2)	(VLEN != DLEN) ? 8 : 2	4
vqmacc (LMUL<1/2)	(VLEN != DLEN) ? 8 : 1	4
vqmaccu (LMUL<1/2)	(VLEN != DLEN) ? 8 : 1	4
vqmaccsu (LMUL<1/2)	(VLEN != DLEN) ? 8 : 1	4
vqmaccus (LMUL<1/2)	(VLEN != DLEN) ? 8 : 1	4
Andes V5 Vector Dot Product Extension		
vd4dots_vv	(VLEN/DLEN)*LMUL	4
vd4dotu_vv	(VLEN/DLEN)*LMUL	4
vd4dotsu_vv	(VLEN/DLEN)*LMUL	4

15.17.5.2 VMAC Floating-Point Instructions

Please see Table 77 and Table 78 when **Vector Floating-Point Support** is configured as "FP32" and "FP32+FP64", respectively.

Table 77: VMAC Floating-Point Instruction Throughput and Latency When Vector Floating-Point Support Is FP32

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
RISC-V Vector Extension		
vfadd	(VLEN/DLEN)*LMUL	5
vfsub	(VLEN/DLEN)*LMUL	5
vfrsub	(VLEN/DLEN)*LMUL	5
vfmul	(VLEN/DLEN)*LMUL	5
vfmadd	(VLEN/DLEN)*LMUL	5
vfnmadd	(VLEN/DLEN)*LMUL	5
vfmsub	(VLEN/DLEN)*LMUL	5
vfnmsub	(VLEN/DLEN)*LMUL	5
vfmacc	(VLEN/DLEN)*LMUL	5
vfnmacc	(VLEN/DLEN)*LMUL	5
vfmsac	(VLEN/DLEN)*LMUL	5
vfnmsac	(VLEN/DLEN)*LMUL	5
vfwadd (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5

Continued on next page...

Table 77: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vfwsub (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwadd (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwsub (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwmul (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwnmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwmsac (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwnmsac (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwmul (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwmacc (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwnmacc (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwmsac (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfwnmsac (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
vfredusum	1 or 3*(vfredusum-micro-ops - 1) + 1	3*vfredusum-micro-ops + 2
vfwredusum	1 or 3*(vfwredusum-micro-ops - 1) + 1	3*vfwredusum-micro-ops + 2
vfredosum	1 or 3*(vfredosum-micro-ops - 1) + 1	3*vfredosum-micro-ops + 2
vfwredosum	1 or 3*(vfwredosum-micro-ops - 1) + 1	3*vfwredosum-micro-ops + 2
RISC-V BF16 extensions		
vfwmaccbf16 (LMUL≥1)	(VLEN/DLEN)*LMUL*2	5
vfwmaccbf16 (LMUL<1)	(VLEN != DLEN) ? 4 : 1	5
Andes V5 Vector Packed FP16 Extension		
vfpmadt	(VLEN/DLEN)*LMUL	5
vfpmadb	(VLEN/DLEN)*LMUL	5

Table 78: VMAC Floating-Point Instruction Throughput and Latency When Vector Floating-Point Support Is FP32+FP64

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
RISC-V Vector Extension		
vfadd	(VLEN/DLEN)*LMUL	6
vfsub	(VLEN/DLEN)*LMUL	6
vfrsub	(VLEN/DLEN)*LMUL	6
vfmul	(VLEN/DLEN)*LMUL	6
vfmadd	(VLEN/DLEN)*LMUL	6

Continued on next page...

Table 78: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vfnmadd	(VLEN/DLEN)*LMUL	6
vfmsub	(VLEN/DLEN)*LMUL	6
vfnmsub	(VLEN/DLEN)*LMUL	6
vfmacc	(VLEN/DLEN)*LMUL	6
vfnmacc	(VLEN/DLEN)*LMUL	6
vfmsac	(VLEN/DLEN)*LMUL	6
vfnmsac	(VLEN/DLEN)*LMUL	6
vfwadd (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwsub (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwadd (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwsub (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwmul (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwnmacc (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwmsac (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwnmsac (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwmul (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwmacc (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwnmacc (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwmsac (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfwnmsac (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
vfredusum	1 or 4*(vfredusum-micro-ops - 1) + 1	4*vfredusum-micro-ops + 2
vfwredusum	1 or 4*(vfwredusum-micro-ops - 1) + 1	4*vfwredusum-micro-ops + 2
vfredosum	1 or 4*(vfredosum-micro-ops - 1) + 1	4*vfredosum-micro-ops + 2
vfwredosum	1 or 4*(vfwredosum-micro-ops - 1) + 1	4*vfwredosum-micro-ops + 2
RISC-V BF16 extensions		
vfwmaccbf16 (LMUL≥1)	(VLEN/DLEN)*LMUL*2	6
vfwmaccbf16 (LMUL<1)	(VLEN != DLEN) ? 4 : 1	6
Andes V5 Vector Packed FP16 Extension		
vfpmadt	(VLEN/DLEN)*LMUL	6
vfpmadb	(VLEN/DLEN)*LMUL	6

Note

- The latencies and throughputs are not affected by the setting of `umisc_ctl.FP_MODE`.
- vfredusum-micro-ops = (VLEN/DLEN)*LMUL + LOG2(DLEN/SEW)
- vfwredusum-micro-ops = (VLEN/DLEN)*LMUL*2 + LOG2(DLEN/SEW) - 1
- vfredosum-micro-ops = VLEN * LMUL / SEW
- vfwredosum-micro-ops = VLEN * LMUL / SEW
- VMAC can execute at most three reductions simultaneously in the pipeline.
  - Only back-to-back reductions can be executed simultaneously in the pipeline.
  - The throughput is 1 when two reductions are executed simultaneously in the pipeline.
- When fractional LMUL is used, the **LMUL** used in throughput and latency calculation is 1.

15.17.6 VFMIS Instructions

Table 79: VFMIS Instruction Throughput and Latency

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
RISC-V Vector Extension		
vfmin	(VLEN/DLEN)*LMUL	2
vfmax	(VLEN/DLEN)*LMUL	2
vfsgnj	(VLEN/DLEN)*LMUL	2
vfsgnjn	(VLEN/DLEN)*LMUL	2
vfsgnjx	(VLEN/DLEN)*LMUL	2
vfclass	(VLEN/DLEN)*LMUL	2
vfmerge	(VLEN/DLEN)*LMUL	2
vfmv	(VLEN/DLEN)*LMUL	2
vfcvt	(VLEN/DLEN)*LMUL	3
vfwcvt	(VLEN/DLEN)*LMUL*2	3
vfncvt (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vfncvt (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 4 : 3
vmfeq	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmfne	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmflt	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmfle	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmfgt	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vmfge	(VLEN/DLEN)*LMUL + (VLEN != DLEN)	(VLEN/DLEN)*LMUL+2
vfredmin	vfredmin-micro-ops + (VLEN != DLEN)	vfredmin-micro-ops + 1
vfredmax	vfredmin-micro-ops + (VLEN != DLEN)	vfredmin-micro-ops + 1
RISC-V BF16 extensions		
vfwcvt	(VLEN/DLEN)*LMUL*2	2
vfncvt (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vfncvt (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 4 : 3
AndeStar V5 Vector Small INT Handling Extension		
vfwcvt.f.n.v (LMUL>SEW/4)	(VLEN/DLEN)*LMUL	3
vfwcvt.f.n.v (LMUL≤SEW/4)	(VLEN/DLEN)*(SEW/4)	3
vfwcvt.f.nu.v (LMUL>SEW/4)	(VLEN/DLEN)*LMUL	3
vfwcvt.f.nu.v (LMUL≤SEW/4)	(VLEN/DLEN)*(SEW/4)	3
vfwcvt.f.b.v (LMUL>SEW/8)	(VLEN/DLEN)*LMUL	3

Continued on next page...

Table 79: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vfwcvt.f.b.v (LMUL≤SEW/8)	(VLEN/DLEN)*(SEW/8)	3
vfwcvt.f.bu.v (LMUL>SEW/8)	(VLEN/DLEN)*LMUL	3
vfwcvt.f.bu.v (LMUL≤SEW/8)	(VLEN/DLEN)*(SEW/8)	3
AndeStar V5 Vector BFLOAT16 Conversion Extension		
vfwcvt	(VLEN/DLEN)*LMUL*2	2
vfncvt (LMUL≥1)	(VLEN/DLEN)*LMUL*2	4
vfncvt (LMUL<1)	(VLEN != DLEN) ? 4 : 1	(VLEN != DLEN) ? 4 : 3

Note

- The latencies and throughputs are not affected by the setting of `umisc_ctl.FP_MODE`.
- vfredmin-micro-ops = (VLEN/DLEN)*LMUL + LOG2(DLEN/64) * 2 + LOG2(64/SEW)
- When fractional LMUL is used, the **LMUL** used in throughput and latency calculation is 1.

15.17.7 VLSU Instructions

The cycles of processing data and latency of VLSU instructions depend on the instruction type, memory type, VLSU_MEM_DW, and resource restrictions. The data width (VLSU_MEM_DW) for memory accesses varies based on the access region:

- For memory accesses within the HVM region, VLSU_MEM_DW equals `DLEN`.
- For memory accesses within a cacheable or non-cacheable region, VLSU_MEM_DW equals `BIU_DATA_WIDTH`.

In AX46MPV, the Load-Store Unit (LSU) can process up to two load transactions or one store transaction per cycle with vector load store instructions. The number of transactions that can be processed in parallel is influenced by the configurations, access region, and the address pattern. For detailed information, see AndesCore AX46MPV Integration Guide (IG094).

- Number of MMIO Port
- MMIO Port Interleaving Mask
- Number of MEM Port
- Number of L3-Cache Banks
- Number of HVM Banks
- Address Bit for HVM Bank Selection

Cacheable vector load or store requests are sent to L3-Cache without allocating the data into the D-Cache. If the requested data is already allocated in the D-Cache, AX46MPV will handle the data coherency as follows:

- If a vector load hits the D-Cache, the data will be read from the D-Cache.
- If a vector store hits the D-Cache, the data will be sent to the L3-Cache, and the cache line is evicted from the D-Cache.

15.17.7.1 VLSU Memory Latency

Table 80: VLSU Read Access Latency (VLSU_MEM_LATENCY) When L3-Cache Is Configured

Memory Attribute	D-Cache Line State	L3-Cache Line State	Latency (Cycle)
Non-Cacheable Memory	N/A	N/A	7
Cacheable Memory	TBD	TBD	TBD
HVM	N/A	N/A	5

Note

1. The latency assumes that:
   - **Number of Processor Cores** is 4.
   - **Core Interface** is synchronous.
   - **Bus Data Width** is 512 bits.
   - Bus latency is one cycle after the requested address is issued.
   - HVM latency is one cycle after the requested address is issued.
   - **HVM Clock** is synchronous.

15.17.7.2 Resource Restrictions For VLSU

The following lists the configurable resources used to handle memory requests and vector load and store data. See AndesCore AX46MPV Integration Guide (IG094) for more information.

- Number of VRF Reorder Buffers: Specifies the number of vector reorder buffers, which are temporary storage units used to handle out-of-order memory responses during vector processing.
- Number of VLSU Requests: Determines the maximum number of memory requests.

If any of the above resources is not available, the VLSU will stop sending out the memory requests.

15.17.7.3 Vector Unit-Stride Load and Store Instructions

For unmasked vector unit-stride load and store instructions, if a memory access falls within a cacheable or non-cacheable region, the same 64-byte memory requests are merged into a transaction. Additionally, if a memory access falls within the HVM region, the same (DLEN/8)-byte requests are merged into a transaction.

For vector unit-stride load and store instructions with masked or `vstart` != 0, each active element issues an individual memory transaction. Regardless of whether the instruction is unmasked, masked, or has `vstart` != 0, inactive, prestart, and tail elements consume the load store unit pipeline cycles as the active elements.

Table 81, Table 82, Table 83, and Table 84 present the estimated number of LSU cycles required to process the same amount of data under different EMUL, EEW, and port configurations.

Table 81: Cycles of Unmasked Vector Unit-Stride Load Within HVM Region

Processing Data	EMUL	Number of Transactions Per Cycle	Cycles
16384 bytes (16KiB)	1	2	281
	2	2	138
	4	2	138
	8	2	138
16384 bytes (16KiB)	1	1	281
	2	1	267
	4	1	267
	8	1	267

Table 82: Cycles of Masked Or Vstart !=0 Vector Unit-Stride Load Within HVM Region

Processing Data	EMUL	EEW	Number of Transactions Per Cycle	Cycles
16384 bytes (16KiB)	1	8	2	TBD
	1	16	2	TBD
	1	32	2	TBD
	1	64	2	TBD
	2	8	2	TBD
	2	16	2	TBD
	2	32	2	TBD
	2	64	2	TBD
	4	8	2	TBD
	4	16	2	TBD
	4	32	2	TBD
	4	64	2	TBD
	8	8	2	TBD
	8	16	2	TBD
	8	32	2	TBD
	8	64	2	TBD
16384 bytes (16KiB)	1	8	1	TBD
	1	16	1	TBD
	1	32	1	TBD
	1	64	1	TBD
	2	8	1	TBD
	2	16	1	TBD
	2	32	1	TBD
	2	64	1	TBD
	4	8	1	TBD
	4	16	1	TBD
	4	32	1	TBD
	4	64	1	TBD
	8	8	1	TBD
	8	16	1	TBD
	8	32	1	TBD

Continued on next page...

Table 82: (continued)

Processing Data	EMUL	EEW	Number of Transactions Per Cycle	Cycles
	8	64	1	TBD

Table 83: Cycles of Unmasked Vector Unit-Stride Store Within HVM Region

Processing Data	EMUL	Number of Transactions Per Cycle	Cycles
16384 bytes (16KiB)	1	2	281
	2	2	266
	4	2	266
	8	2	266
16384 bytes (16KiB)	1	1	281
	2	1	267
	4	1	267
	8	1	267

Table 84: Cycles of Masked Or Vstart !=0 Vector Unit-Stride Store Within HVM Region

Processing Data	EMUL	EEW	Number of Transactions Per Cycle	Cycles
16384 bytes (16KiB)	1	8	1 or 2	TBD
	1	16	1 or 2	TBD
	1	32	1 or 2	TBD
	1	64	1 or 2	TBD
	2	8	1 or 2	TBD
	2	16	1 or 2	TBD
	2	32	1 or 2	TBD
	2	64	1 or 2	TBD
	4	8	1 or 2	TBD
	4	16	1 or 2	TBD
	4	32	1 or 2	TBD

Continued on next page...

Table 84: (continued)

Processing Data	EMUL	EEW	Number of Transactions Per Cycle	Cycles
	4	64	1 or 2	TBD
	8	8	1 or 2	TBD
	8	16	1 or 2	TBD
	8	32	1 or 2	TBD
	8	64	1 or 2	TBD

Table 85: Vector Unit-Stride Load Latency

Condition	Result Latency (Cycle)
Unmasked	4 + VLSU_MEM_LATENCY
Masked or vstart != 0	TBD

Note

- The above tables assume that:
  - Vector Register Length (VLEN) is 512.
  - Vector Datapath Width (DLEN) is 512.
  - VLSU_MEM_DW is 512.
  - Number of VRF Reorder Buffers is 32.
  - Number of VLSU Requests is 32.
  - The transactions respond in order.
- Number of Transactions Per Cycle = 2 assumes that:
  - Number of HVM Banks is 2
  - Address Bit for HVM Bank Selection is 6
- Number of Transactions Per Cycle = 1 assumes that:
  - Number of HVM Banks is 1
- For unmasked unit-stride load and store instructions, the latency assumes the access address is aligned to (VLSU_MEM_DW/8) bytes.
- For masked or `vstart` != 0 unit-stride store instructions, the cycles assumes that no stall is caused by memory dependency, as described in Section 6.4.

15.17.7.4 Vector Strided Load and Store Instructions

TBD

15.17.7.5 Vector Indexed Load and Store Instructions

TBD

15.17.7.6 Vector Unit-Stride Segment Load and Store Instructions

TBD

15.17.7.7 Vector Strided Segment Load and Store Instructions

TBD

15.17.7.8 Vector Indexed Segment Load and Store Instructions

TBD

15.17.7.9 AndeStar V5 INT4 Vector Load Extension

TBD

15.17.7.10 AndeStar V5 Vector Small INT Handling Extension

TBD

15.17.7.11 VLSU Cycle Measurement Example Code

An example showing how to measure the cycles of processing data is provided below.

```
        # VLEN: 512
        # Processing Data: 16K bytes
        la      a0, data
        li      a1, 0x200 # data bytes per instruction
        li      a2, 0x8   # loop count
        vsetvli t0, zero, e32, m8, ta, ma
        csrr    t0, mcycle
TEST_LOOP_START:
        vle32.v v0,  (a0)    # 7 + 8
        add     a0,  a0, a1
        vle32.v v8,  (a0)    #
        add     a0,  a0, a1
        vle32.v v16, (a0)
        add     a0,  a0, a1
        vle32.v v24, (a0)
        add     a0,  a0, a1
        addi    a2, a2, -1
        bnez    a2, TEST_LOOP_START
TEST_LOOP_END:
        fence
        csrr    t1, mcycle
        sub     t1, t1, t0 # t1: cycles
```

15.17.8 VMASK Instructions

Table 86: VMASK Instruction Throughput and Latency

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN = 1:1		
vmand	1	3
vmnand	1	3
vmandn	1	3
vmxor	1	3
vmor	1	3
vmnor	1	3
vmnorn	1	3
vmxnor	1	3
vcpop	1	4
vfirst	1	4
vmsbf	1	3
vmsif	1	3
vmsof	1	3
viota	LMUL	4
vid	LMUL	4
VLEN:DLEN = 2:1		
vmand	2	3
vmnand	2	3
vmandn	2	3
vmxor	2	3
vmor	2	3
vmnor	2	3
vmnorn	2	3
vmxnor	2	3
vcpop	2	5
vfirst	2	5
vmsbf	2	3
vmsif	2	3
vmsof	2	3

Continued on next page...

Table 86: (continued)

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
viota (LMUL≤4)	(2*LMUL)+1	4
vid (LMUL≤4)	(2*LMUL)+1	4
viota (LMUL = 8, SEW = 8)	(2*LMUL)	4
vid (LMUL = 8, SEW = 8))	(2*LMUL)	4
viota (LMUL = 8, SEW = 16,32, or 64)	(2*LMUL)+1	4
vid (LMUL = 8, SEW = 16,32, or 64))	(2*LMUL)+1	4

Note

- When fractional LMUL is used, the **LMUL** used in throughput and latency calculation is 1.

15.17.9 VPERMUT Instructions

Table 87: Whole Register Move Instructions

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vmv.x.s	VLEN/DLEN	2
vmv.s.x	VLEN/DLEN	3
vfmv.f.s	VLEN/DLEN	2
vfmv.s.f	VLEN/DLEN	3
vmv1r	(VLEN/DLEN) * 1	2
vmv2r	(VLEN/DLEN) * 2	2
vmv4r	(VLEN/DLEN) * 4	2
vmv8r	(VLEN/DLEN) * 8	2
vzext.vf2	(VLEN/DLEN) * LMUL	2
vsext.vf2	(VLEN/DLEN) * LMUL	2
vzext.vf4	(VLEN/DLEN) * LMUL	2
vsext.vf4	(VLEN/DLEN) * LMUL	2
vzext.vf8	(VLEN/DLEN) * LMUL	2
vsext.vf8	(VLEN/DLEN) * LMUL	2

Note

- When fractional LMUL is used, the LMUL used in throughput and latency calculation is 1.

Table 88: Vector Slide1 Up/Down Instructions

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
v(f)slide1up	LMUL*VLEN/VPERMUT_DLEN	2 + LOG2(DLEN/VPERMUT_DLEN)
v(f)slide1down	(LMUL*VLEN/VPERMUT_DLEN) + `vslide1down_ratio`	2 + LOG2(DLEN/VPERMUT_DLEN) + `vslide1down_ratio`

Note

- When fractional LMUL is used, the LMUL used in LMUL*VLEN/VPERMUT_DLEN expression is 1.
- `vslide1down_ratio` = (LMUL*VLEN>VPERMUT_DLEN)? 1 : 0. This LMUL is the real number. For example, when LMUL is f2 and VLEN/VPERMUT_DLEN equals 4, `vslide1down_ratio` is 1 because VLEN/2 > VPERMUT_DLEN.

Table 89: `vslidedown.vx` Instruction Throughput and Latency

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1		
≤1	1	2
2	2 – 4	2 – 4
4	4 – 8	2 – 6
8	8 – 16	2 – 10
VLEN:DLEN:VPERMUT_DLEN = 2:2:1		
≤f2	2	3
1	2 – 4	3 – 5
2	4 – 8	3 – 7
4	8 – 16	3 – 11
8	16 – 32	3 – 19
VLEN:DLEN:VPERMUT_DLEN = 2:1:1		
≤f2	2	2
1	2 – 4	2 – 4
2	4 – 8	2 – 6
4	8 – 16	2 – 10
8	16 – 32	2 – 18
VLEN:DLEN:VPERMUT_DLEN = 4:2:1		
≤f4	4	3
f2	4 – 6	3 – 5
1	4 – 8	3 – 7
2	8 – 16	3 – 11
4	16 – 32	3 – 19
8	32 – 64	3 – 35

Note

- The minimum value is reached when `rs1` is 0.
- The maximum value is reached when `rs1` is (LMUL*VLEN/SEW-1).

Table 90: `vslidedown.vi` Instruction Throughput and Latency

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1			
128	≤1	1	2

Continued on next page...

Table 90: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
128	2	2 – 4	2 – 4
128	4	4 – 8	2 – 6
128	8	8 – 16	2 – 10
256	≤1	1	2
256	2	2 – 4	2 – 4
256	4	4 – 8	2 – 6
256	8	8 – 16	2 – 10
512	≤1	1	2
512	2	2 – 4	2 – 4
512	4	4 – 8	2 – 6
512	8	8 – 12	2 – 6
1024	≤1	1	2
1024	2	2 – 4	2 – 4
1024	4	4 – 6	2 – 4
1024	8	8 – 10	2 – 4
VLEN:DLEN:VPERMUT_DLEN = 2:2:1			
256	≤f2	2	3
256	1	2 – 4	3 – 5
256	2	4 – 8	3 – 7
256	4	8 – 16	3 – 11
256	8	16 – 32	3 – 19
512	≤f2	2	3
512	1	2 – 4	3 – 5
512	2	4 – 8	3 – 7
512	4	8 – 16	3 – 11
512	8	16 – 24	3 – 11
1024	≤f2	2	3
1024	1	2 – 4	3 – 5
1024	2	4 – 8	3 – 7
1024	4	8 – 12	3 – 7
1024	8	16 – 20	3 – 7
VLEN:DLEN:VPERMUT_DLEN = 2:1:1			

Continued on next page...

Table 90: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
256	≤f2	2	2
256	1	2 – 4	2 – 4
256	2	4 – 8	2 – 6
256	4	8 – 16	2 – 10
256	8	16 – 32	2 – 18
512	≤f2	2	2
512	1	2 – 4	2 – 4
512	2	4 – 8	2 – 6
512	4	8 – 16	2 – 10
512	8	16 – 24	2 – 10
1024	≤f2	2	2
1024	1	2 – 4	2 – 4
1024	2	4 – 8	2 – 6
1024	4	8 – 12	2 – 6
1024	8	16 – 20	2 – 6
2048	≤f2	2	2
2048	1	2 – 4	2 – 4
2048	2	4 – 6	2 – 4
2048	4	8 – 10	2 – 4
2048	8	16 – 18	2 – 4
VLEN:DLEN:VPERMUT_DLEN = 4:2:1			
512	≤f4	4	3
512	f2	4 – 6	3 – 5
512	1	4 – 8	3 – 7
512	2	8 – 16	3 – 11
512	4	16 – 32	3 – 19
512	8	32 – 48	3 – 19
1024	≤f4	4	3
1024	f2	4 – 6	3 – 5
1024	1	4 – 8	3 – 7
1024	2	8 – 16	3 – 11
1024	4	16 – 24	3 – 11

Continued on next page...

Table 90: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
1024	8	32 – 40	3 – 11
2048	≤f4	4	3
2048	f2	4 – 6	3 – 5
2048	1	4 – 8	3 – 7
2048	2	8 – 12	3 – 7
2048	4	16 – 20	3 – 7
2048	8	32 – 36	3 – 7

Note

- The minimum value is reached when `uimm` is 0.
- The maximum value is reached when SEW is 64 and one of the following conditions is true:
  - (LMUL*VLEN) is greater than (31*64), and `uimm` is 31.
  - (LMUL*VLEN) is less than (31*64), and `uimm` is (LMUL*VLEN/64-1).

Table 91: `vslideup.vx` Instruction Throughput and Latency

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1		
≤1	1	2
2	2 – 3	2
4	4 – 7	2
8	8 – 15	2
VLEN:DLEN:VPERMUT_DLEN = 2:2:1		
≤f2	2	3
1	2 – 3	3
2	4 – 7	3
4	8 – 15	3
8	16 – 31	3
VLEN:DLEN:VPERMUT_DLEN = 2:1:1		
≤f2	2	2
1	2 – 3	2
2	4 – 7	2
4	8 – 15	2
8	16 – 31	2

Continued on next page...

Table 91: (continued)

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 4:2:1		
≤f4	4	3
f2	4 – 5	3
1	4 – 7	3
2	8 – 15	3
4	16 – 31	3
8	32 – 63	3

Note

- The minimum value is reached when `rs1` is 0.
- The maximum value is reached when `rs1` is (LMUL*VLEN/SEW-1).

Table 92: `vslideup.vi` Instruction Throughput and Latency

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1			
128	≤1	1	2
128	2	2 – 3	2
128	4	4 – 7	2
128	8	8 – 15	2
256	≤1	1	2
256	2	2 – 3	2
256	4	4 – 7	2
256	8	8 – 15	2
512	≤1	1	2
512	2	2 – 3	2
512	4	4 – 7	2
512	8	8 – 11	2
1024	≤1	1	2
1024	2	2 – 3	2
1024	4	4 – 5	2
1024	8	8 – 9	2
VLEN:DLEN:VPERMUT_DLEN = 2:2:1			
256	≤f2	2	3

Continued on next page...

Table 92: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
256	1	2 – 3	3
256	2	4 – 7	3
256	4	8 – 15	3
256	8	16 – 31	3
512	≤f2	2	3
512	1	2 – 3	3
512	2	4 – 7	3
512	4	8 – 15	3
512	8	16 – 23	3
1024	≤f2	2	3
1024	1	2 – 3	3
1024	2	4 – 7	3
1024	4	8 – 11	3
1024	8	16 – 19	3
VLEN:DLEN:VPERMUT_DLEN = 2:1:1			
256	≤f2	2	2
256	1	2 – 3	2
256	2	4 – 7	2
256	4	8 – 15	2
256	8	16 – 31	2
512	≤f2	2	2
512	1	2 – 3	2
512	2	4 – 7	2
512	4	8 – 15	2
512	8	16 – 23	2
1024	≤f2	2	2
1024	1	2 – 3	2
1024	2	4 – 7	2
1024	4	8 – 11	2
1024	8	16 – 19	2
2048	≤f2	2	2
2048	1	2 – 3	2

Continued on next page...

Table 92: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
2048	2	4 – 5	2
2048	4	8 – 9	2
2048	8	16 – 17	2
VLEN:DLEN:VPERMUT_DLEN = 4:2:1			
512	≤f4	4	3
512	f2	4 – 5	3
512	1	4 – 7	3
512	2	8 – 15	3
512	4	16 – 31	3
512	8	32 – 47	3
1024	≤f4	4	3
1024	f2	4 – 5	3
1024	1	4 – 7	3
1024	2	8 – 15	3
1024	4	16 – 23	3
1024	8	32 – 39	3
2048	≤f4	4	3
2048	f2	4 – 5	3
2048	1	4 – 7	3
2048	2	8 – 11	3
2048	4	16 – 19	3
2048	8	32 – 35	3

Note

- The minimum value is reached when `uimm` is 0.
- The maximum value is reached when SEW is 64 and one of the following conditions is true:
  - (LMUL*VLEN) is greater than (31*64), and `uimm` is 31.
  - (LMUL*VLEN) is less than (31*64), and `uimm` is (LMUL*VLEN/64-1).

Table 93: `vrgather.vx` Instruction Throughput and Latency

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1		
≤1	1	2

Continued on next page...

Table 93: (continued)

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
2	2 – 3	2 – 3
4	4 – 7	2 – 5
8	8 – 15	2 – 9
VLEN:DLEN:VPERMUT_DLEN = 2:2:1		
≤f2	2	3
1	2 – 3	3 – 4
2	4 – 7	3 – 6
4	8 – 15	3 – 10
8	16 – 31	3 – 18
VLEN:DLEN:VPERMUT_DLEN = 2:1:1		
≤f2	2	2
1	2 – 3	2 – 3
2	4 – 7	2 – 5
4	8 – 15	2 – 9
8	16 – 31	2 – 17
VLEN:DLEN:VPERMUT_DLEN = 4:2:1		
≤f4	4	3
f2	4 – 5	3 – 4
1	4 – 7	3 – 6
2	8 – 15	3 – 10
4	16 – 31	3 – 18
8	32 – 63	3 – 34

Note

- The minimum value is reached when `rs1` is 0.
- The maximum value is reached when `rs1` is (LMUL*VLEN/SEW-1).

Table 94: `vrgather.vi` Instruction Throughput and Latency

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1			
128	≤1	1	2
128	2	2 – 3	2 – 3
128	4	4 – 7	2 – 5

Continued on next page...

Table 94: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
128	8	8 – 15	2 – 9
256	≤1	1	2
256	2	2 – 3	2 – 3
256	4	4 – 7	2 – 5
256	8	8 – 15	2 – 9
512	≤1	1	2
512	2	2 – 3	2 – 3
512	4	4 – 7	2 – 5
512	8	8 – 11	2 – 5
1024	≤1	1	2
1024	2	2 – 3	2 – 3
1024	4	4 – 5	2 – 3
1024	8	8 – 9	2 – 3
VLEN:DLEN:VPERMUT_DLEN = 2:2:1			
256	≤f2	2	3
256	1	2 – 3	3 – 4
256	2	4 – 7	3 – 6
256	4	8 – 15	3 – 10
256	8	16 – 31	3 – 18
512	≤f2	2	3
512	1	2 – 3	3 – 4
512	2	4 – 7	3 – 6
512	4	8 – 15	3 – 10
512	8	16 – 23	3 – 10
1024	≤f2	2	3
1024	1	2 – 3	3 – 4
1024	2	4 – 7	3 – 6
1024	4	8 – 11	3 – 6
1024	8	16 – 19	3 – 6
VLEN:DLEN:VPERMUT_DLEN = 2:1:1			
256	≤f2	2	2
256	1	2 – 3	2 – 3

Continued on next page...

Table 94: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
256	2	4 – 7	2 – 5
256	4	8 – 15	2 – 9
256	8	16 – 31	2 – 17
512	≤f2	2	2
512	1	2 – 3	2 – 3
512	2	4 – 7	2 – 5
512	4	8 – 15	2 – 9
512	8	16 – 23	2 – 9
1024	≤f2	2	2
1024	1	2 – 3	2 – 3
1024	2	4 – 7	2 – 5
1024	4	8 – 11	2 – 5
1024	8	16 – 19	2 – 5
2048	≤f2	2	2
2048	1	2 – 3	2 – 3
2048	2	4 – 5	2 – 3
2048	4	8 – 9	2 – 3
2048	8	16 – 17	2 – 3
VLEN:DLEN:VPERMUT_DLEN = 4:2:1			
512	≤f4	4	3
512	f2	4 – 5	3 – 4
512	1	4 – 7	3 – 6
512	2	8 – 15	3 – 10
512	4	16 – 31	3 – 18
512	8	32 – 47	3 – 18
1024	≤f4	4	3
1024	f2	4 – 5	3 – 4
1024	1	4 – 7	3 – 6
1024	2	8 – 15	3 – 10
1024	4	16 – 23	3 – 10
1024	8	32 – 39	3 – 10
2048	≤f4	4	3

Continued on next page...

Table 94: (continued)

VLEN	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
2048	f2	4 – 5	3 – 4
2048	1	4 – 7	3 – 6
2048	2	8 – 11	3 – 6
2048	4	16 – 19	3 – 6
2048	8	32 – 35	3 – 6

Note

- The minimum value is reached when `uimm` is 0.
- The maximum value is reached when SEW is 64 and one of the following conditions is true:
  - (LMUL*VLEN) is greater than (31*64), and `uimm` is 31.
  - (LMUL*VLEN) is less than (31*64), and `uimm` is (LMUL*VLEN/64-1).

Table 95: Vector Compress Instruction Throughput and Latency

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1		
≤1	3	4
2	4 – 5	4 – 5
4	6 – 9	4 – 7
8	10 – 17	4 – 11
VLEN:DLEN:VPERMUT_DLEN = 2:2:1		
≤f2	5	6
1	5 – 6	6 – 7
2	7 – 10	6 – 9
4	11 – 18	6 – 13
8	19 – 34	6 – 21
VLEN:DLEN:VPERMUT_DLEN = 2:1:1		
≤f2	5	5
1	5 – 6	5 – 6
2	7 – 10	5 – 8
4	11 – 18	5 – 12
8	19 – 34	5 – 20
VLEN:DLEN:VPERMUT_DLEN = 4:2:1		
≤f4	9	8

Continued on next page...

Table 95: (continued)

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
f2	9 – 10	8 – 9
1	9 – 12	8 – 11
2	13 – 20	8 – 15
4	21 – 36	8 – 23
8	37 – 68	8 – 39

Note

- When all mask bits are zero, the minimum value is reached.
- When the mask bit at VLMAX-1 is 1 and the others are zero, the maximum value is reached.

Table 96: `vrgather.vv` Instruction Throughput and Latency

LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1		
≤1	2	3
2	6	5
4	20	9
8	72	17
VLEN:DLEN:VPERMUT_DLEN = 2:2:1		
≤1	3	4
2	10	7
4	36	13
8	136	25
VLEN:DLEN:VPERMUT_DLEN = 2:1:1		
≤1	4	4
2	12	7
4	40	13
8	144	25
VLEN:DLEN:VPERMUT_DLEN = 4:2:1		
≤1	6	5
2	20	9
4	72	17
8	272	33

Table 97: `vrgatherei16.vv` Instruction Throughput and Latency

SEW	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
VLEN:DLEN:VPERMUT_DLEN = 1:1:1			
8	≤f2	2	3
8	1	3	4
8	2	7	5
8	4	23	9
≥16	≤1	2	3
≥16	2	6	5
≥16	4	20	9
≥16	8	72	17
VLEN:DLEN:VPERMUT_DLEN = 2:2:1			
8	≤f4	3	4
8	f2	4	5
8	1	5	6
8	2	13	8
8	4	43	14
16	≤1	3	4
16	2	10	7
16	4	36	13
16	8	136	25
32	≤1	3	4
32	2	10	7
32	4	36	13
32	8	136	25
64	≤1	4	4
64	2	10	7
64	4	36	13
64	8	136	25
VLEN:DLEN:VPERMUT_DLEN = 2:1:1			
8	≤f2	4	4
8	1	5	4
8	2	15	7
8	4	47	13

Continued on next page...

Table 97: (continued)

SEW	LMUL	Throughput (Cycle/Instruction)	Result Latency (Cycle)
16	≤1	4	4
16	2	12	7
16	4	40	13
16	8	144	25
32	≤1	4	4
32	2	12	7
32	4	40	13
32	8	144	25
64	≤1	5	4
64	2	12	7
64	4	40	13
64	8	144	25
VLEN:DLEN:VPERMUT_DLEN = 4:2:1			
8	f8	6	5
8	f4	7	6
8	f2	7	6
8	1	9	6
8	2	27	10
8	4	87	18
16	≤1	6	5
16	2	20	9
16	4	72	17
16	8	272	33
32	≤1	7	5
32	2	20	9
32	4	72	17
32	8	272	33
64	≤1	8	5
64	2	21	9
64	4	72	17
64	8	272	33

15.17.10 VDIV Instructions

Table 98: VDIV Instruction Throughput and Latency

Instruction	VLEN/DLEN	LMUL	SEW	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vdiv(u)/vrem(u)	1	<=1	8	11	12
vdiv(u)/vrem(u)	1	<=1	16	19	20
vdiv(u)/vrem(u)	1	<=1	32	35	36
vdiv(u)/vrem(u)	1	<=1	64	67	68
vdiv(u)/vrem(u)	1	2	8	21	12
vdiv(u)/vrem(u)	1	2	16	37	20
vdiv(u)/vrem(u)	1	2	32	69	36
vdiv(u)/vrem(u)	1	2	64	133	68
vdiv(u)/vrem(u)	1	4	8	41	12
vdiv(u)/vrem(u)	1	4	16	73	20
vdiv(u)/vrem(u)	1	4	32	137	36
vdiv(u)/vrem(u)	1	4	64	265	68
vdiv(u)/vrem(u)	1	8	8	81	12
vdiv(u)/vrem(u)	1	8	16	145	20
vdiv(u)/vrem(u)	1	8	32	273	36
vdiv(u)/vrem(u)	1	8	64	529	68
vdiv(u)/vrem(u)	2	<=1	8	21	12
vdiv(u)/vrem(u)	2	<=1	16	37	20
vdiv(u)/vrem(u)	2	<=1	32	69	36
vdiv(u)/vrem(u)	2	<=1	64	133	68
vdiv(u)/vrem(u)	2	2	8	41	12
vdiv(u)/vrem(u)	2	2	16	73	20
vdiv(u)/vrem(u)	2	2	32	137	36
vdiv(u)/vrem(u)	2	2	64	265	68
vdiv(u)/vrem(u)	2	4	8	81	12
vdiv(u)/vrem(u)	2	4	16	145	20
vdiv(u)/vrem(u)	2	4	32	273	36
vdiv(u)/vrem(u)	2	4	64	529	68
vdiv(u)/vrem(u)	2	8	8	161	12

Continued on next page...

Table 98: (continued)

Instruction	VLEN/DLEN	LMUL	SEW	Throughput (Cycle/Instruction)	Result Latency (Cycle)
vdiv(u)/vrem(u)	2	8	16	289	20
vdiv(u)/vrem(u)	2	8	32	545	36
vdiv(u)/vrem(u)	2	8	64	1057	68

15.17.11 VFDIV Instructions

Table 99: VFDIV Instruction Throughput and Latency

Instruction	Throughput (Cycle/Instruction)	Result Latency (Cycle)
DLEN:VFDIV_DLEN = 1:1		
vfrsqrt7	(VLEN/DLEN) * LMUL * 3	4
vfrec7	(VLEN/DLEN) * LMUL * 3	4
vf(r)div (SEW = 16)	(VLEN/DLEN) * LMUL * 11	13
vf(r)div (SEW = 32)	(VLEN/DLEN) * LMUL * 18	20
vf(r)div (SEW = 64)	(VLEN/DLEN) * LMUL * 32	34
vfsqrt (SEW = 16)	(VLEN/DLEN) * LMUL * 10	12
vfsqrt (SEW = 32)	(VLEN/DLEN) * LMUL * 17	19
vfsqrt (SEW = 64)	(VLEN/DLEN) * LMUL * 31	33
DLEN:VFDIV_DLEN = 2:1		
vfrsqrt7	(VLEN/DLEN) * LMUL * 2 * 3	7
vfrec7	(VLEN/DLEN) * LMUL * 2 * 3	7
vf(r)div (SEW = 16)	(VLEN/DLEN) * LMUL * 2 * 11	24
vf(r)div (SEW = 32)	(VLEN/DLEN) * LMUL * 2 * 18	38
vf(r)div (SEW = 64)	(VLEN/DLEN) * LMUL * 2 * 32	66
vfsqrt (SEW = 16)	(VLEN/DLEN) * LMUL * 2 * 10	22
vfsqrt (SEW = 32)	(VLEN/DLEN) * LMUL * 2 * 17	36
vfsqrt (SEW = 64)	(VLEN/DLEN) * LMUL * 2 * 31	64

Note

- The latencies and throughputs are not affected by the setting of `umisc_ctl.FP_MODE`.
- The data width of VFDIV logic is DLEN/2, and operations are not pipelined.
- When fractional LMUL is used, the **LMUL** used in throughput and latency calculation is 1.

16 Control and Status Registers

16.1 Introduction

The sections below describe the registers in detail.

16.1.1 Definitions of Terms

Term	Description
IM	Implementation dependent/determined
RO	Read-Only register/field. Any software write to RO register/field will be silently ignored by hardware.
RW	Read/Write register/field
W1	Write-only. Only writing 1 has an effect.
W1S	Write 1 to Set
W1C	Write 1 to Clear
WLRL	Write/Read Only Legal Values
WARL	Write-Any-Read-Legal
DC	“Don’t Care”

16.1.2 CSR Listing

Table 100: Machine Information Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
mvendorid	0xf11	Section 16.2.1	N/A	1
marchid	0xf12	Section 16.2.2	N/A	1
mimpid	0xf13	Section 16.2.3	N/A	1
mhartid	0xf14	Section 16.2.4	N/A	1
mconfigptr	0xf15	Section 16.2.5	N/A	2

Table 101: Machine Trap Related Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
mstatus	0x300	Section 16.3.1	9	1 or 5
misa	0x301	Section 16.3.2	9	1
medeleg	0x302	Section 16.3.3	9	1
mideleg	0x303	Section 16.3.4	9	1
mie	0x304	Section 16.3.5	9	1
mtvec	0x305	Section 16.3.6	9	1
mseccfg	0x747	Section 16.4.7	9	2
menvcfg	0x30a	Section 16.4.8	9	2
mscratch	0x340	Section 16.3.7	0	1
mepc	0x341	Section 16.3.8	0	1
mcause	0x342	Section 16.3.9	0	1
mtval	0x343	Section 16.3.10	0	1
mip	0x344	Section 16.3.11	9	1
mxstatus	0x7c4	Section 16.3.12	9	1
mdcause	0x7c9	Section 16.3.13	0	1
mslideleg	0x7D5	Section 16.3.14	9	1

Note

- The read issue latency of mstatus is 1 without the ACE extension and 5 with the ACE extension.

Table 102: Counter Related Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
mcycle	0xb00	Section 16.4.1	0	2
minstret	0xb02	Section 16.4.2	0	2
mhpmcounter3 – mhpmcounter31	0xb03 – 0xb1f	Section 16.4.3	0	2
mcounteren	0x306	Section 16.4.6	9	2
mhpmevent3 – mhpmevent31	0x323 – 0x33f	Section 16.4.5	9	2
mcountinhibit	0x320	Section 16.4.4	9	2

Table 103: Configuration Control & Status Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
micm_cfg	0xfc0	Section 16.5.1	N/A	2
mdcm_cfg	0xfc1	Section 16.5.2	N/A	2
mmsc_cfg	0xfc2	Section 16.5.3	N/A	2
mmsc_cfg3	0xfc4	Section 16.5.4	N/A	2
mvec_cfg	0xfc7	Section 16.5.5	N/A	2
mrvarch_cfg	0xfca	Section 16.5.6	N/A	2
mrvarch_cfg3	0xfcc	Section 16.5.10	N/A	2
ml1dbf_cfg	0xfcd	Section 16.5.11	N/A	2
mccache_ctl_base	0xfcf	Section 16.5.7	N/A	2
mhvm_cfg	0xfd0	Section 16.5.8	N/A	2
mhvmb	0xfd1	Section 16.5.9	N/A	2

Table 104: Trigger Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
tselect	0x7a0	Section 16.6.1	0	2
tdata1	0x7a1	Section 16.6.2	9	2
tdata2	0x7a2	Section 16.6.3	9	2
tdata3	0x7a3	Section 16.6.4	9	2
tinfo	0x7a4	Section 16.6.5	9	2
tcontrol	0x7a5	Section 16.6.6	9	2
mcontext	0x7a8	Section 16.6.7	9	2
scontext	0x7aa	Section 16.6.8	9	2
mcontrol	0x7a1	Section 16.6.9	9	2
icount	0x7a1	Section 16.6.10	9	2
itrigger	0x7a1	Section 16.6.11	9	2
etrigger	0x7a1	Section 16.6.12	9	2
textra	0x7a3	Section 16.6.13	9	2

Table 105: Debug Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
dcsr	0x7b0	Section 16.7.1	9	2
dpc	0x7b1	Section 16.7.2	0	2
dscratch0	0x7b2	Section 16.7.3	0	2
dscratch1	0x7b3	Section 16.7.4	0	2
dexc2dbg	0x7e0	Section 16.7.5	9	2
ddcause	0x7e1	Section 16.7.6	0	2

Table 106: Supervisor Trap Related Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
sstatus	0x100	Section 16.8.1	9	1 or 5
sie	0x104	Section 16.8.2	9	1
stvec	0x105	Section 16.8.3	9	1
scounteren	0x106	Section 16.8.4	9	2
senvcfg	0x10a	Section 16.9.2	9	2
scountovf	0xda0	Section 16.8.5	N/A	2
sscratch	0x140	Section 16.8.6	0	1
sepc	0x141	Section 16.8.7	0	1
scause	0x142	Section 16.8.8	0	1
stval	0x143	Section 16.8.9	0	1
sip	0x144	Section 16.8.10	9	1
slie	0x9c4	Section 16.8.11	9	1
slip	0x9c5	Section 16.8.12	9	1
sdcause	0x9c9	Section 16.8.13	0	1

Note

- The read issue latency of sstatus is 1 without the ACE extension and 5 with the ACE extension.

Table 107: Supervisor Page Translation Related Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
satp	0x180	Section 16.9.1	9	2

Table 108: Memory and Miscellaneous Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
milmb	0x7c0	Section 16.10.1	9	2
mdlmb	0x7c1	Section 16.10.2	4	2
mecc_code	0x7c2	Section 16.10.3	9	2
mnvec	0x7c3	Section 16.10.4	9	2
mpft_ctl	0x7c5	Section 16.10.5	9	2
mcache_ctl	0x7ca	Section 16.10.6	9	2
mcctlbeginaddr	0x7cb	Section 16.10.12	0	2
mcctlcommand	0x7cc	Section 16.10.13	9	2
mcctldata	0x7cd	Section 16.10.14	0	2
scctldata	0x9cd	Section 16.10.15	9	2
ucctlbeginaddr	0x80b	Section 16.10.16	9	2
ucctlcommand	0x80c	Section 16.10.17	9	2
mmisc_ctl	0x7d0	Section 16.10.7	9	2 or 5
smisc_ctl	0x9d0	Section 16.10.8	9	2
umisc_ctl	0x813	Section 16.10.9	9	2
mclk_ctl	0x7df	Section 16.10.10	0	2
mppib	0x7f0	Section 16.10.11	9	2

Note

- The read issue latency of mmisc_ctl is 2 without the ACE extension and 5 with the ACE extension.

Table 109: Hardware Stack Protection and Recording Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
mhsp_ctl	0x7c6	Section 16.11.1	9	2
msp_bound	0x7c7	Section 16.11.2	9	2
msp_base	0x7c8	Section 16.11.3	9	2

Table 110: CoDense Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
uitb	0x800	Section 16.12.1	9	2

Table 111: DSP Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
ucode	0x801	Section 16.13.1	9	2

Table 112: PMP Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
pmpcfg0	0x3a0	Section 16.14.1	9	2
pmpcfg2	0x3a2	Section 16.14.1	9	2
pmpcfg4	0x3a4	Section 16.14.1	9	2
pmpcfg6	0x3a6	Section 16.14.1	9	2
pmpcfg8	0x3a8	Section 16.14.1	9	2
pmpcfg10	0x3aa	Section 16.14.1	9	2
pmpcfg12	0x3ac	Section 16.14.1	9	2
pmpcfg14	0x3ae	Section 16.14.1	9	2
pmpaddr0	0x3b0	Section 16.14.2	9	2
pmpaddr1	0x3b1	Section 16.14.2	9	2
pmpaddr2	0x3b2	Section 16.14.2	9	2
pmpaddr3	0x3b3	Section 16.14.2	9	2
pmpaddr4	0x3b4	Section 16.14.2	9	2
pmpaddr5	0x3b5	Section 16.14.2	9	2
pmpaddr6	0x3b6	Section 16.14.2	9	2
pmpaddr7	0x3b7	Section 16.14.2	9	2
pmpaddr8	0x3b8	Section 16.14.2	9	2
pmpaddr9	0x3b9	Section 16.14.2	9	2
pmpaddr10	0x3ba	Section 16.14.2	9	2
pmpaddr11	0x3bb	Section 16.14.2	9	2
pmpaddr12	0x3bc	Section 16.14.2	9	2
pmpaddr13	0x3bd	Section 16.14.2	9	2
pmpaddr14	0x3be	Section 16.14.2	9	2
pmpaddr15	0x3bf	Section 16.14.2	9	2
pmpaddr16	0x3c0	Section 16.14.2	9	2
pmpaddr17	0x3c1	Section 16.14.2	9	2
pmpaddr18	0x3c2	Section 16.14.2	9	2

Continued on next page...

Table 112: (continued)

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
pmpaddr19	0x3c3	Section 16.14.2	9	2
pmpaddr20	0x3c4	Section 16.14.2	9	2
pmpaddr21	0x3c5	Section 16.14.2	9	2
pmpaddr22	0x3c6	Section 16.14.2	9	2
pmpaddr23	0x3c7	Section 16.14.2	9	2
pmpaddr24	0x3c8	Section 16.14.2	9	2
pmpaddr25	0x3c9	Section 16.14.2	9	2
pmpaddr26	0x3ca	Section 16.14.2	9	2
pmpaddr27	0x3cb	Section 16.14.2	9	2
pmpaddr28	0x3cc	Section 16.14.2	9	2
pmpaddr29	0x3cd	Section 16.14.2	9	2
pmpaddr30	0x3ce	Section 16.14.2	9	2
pmpaddr31	0x3cf	Section 16.14.2	9	2
pmpaddr32	0x3d0	Section 16.14.2	9	2
pmpaddr33	0x3d1	Section 16.14.2	9	2
pmpaddr34	0x3d2	Section 16.14.2	9	2
pmpaddr35	0x3d3	Section 16.14.2	9	2
pmpaddr36	0x3d4	Section 16.14.2	9	2
pmpaddr37	0x3d5	Section 16.14.2	9	2
pmpaddr38	0x3d6	Section 16.14.2	9	2
pmpaddr39	0x3d7	Section 16.14.2	9	2
pmpaddr40	0x3d8	Section 16.14.2	9	2
pmpaddr41	0x3d9	Section 16.14.2	9	2
pmpaddr42	0x3da	Section 16.14.2	9	2
pmpaddr43	0x3db	Section 16.14.2	9	2
pmpaddr44	0x3dc	Section 16.14.2	9	2
pmpaddr45	0x3dd	Section 16.14.2	9	2
pmpaddr46	0x3de	Section 16.14.2	9	2
pmpaddr47	0x3df	Section 16.14.2	9	2
pmpaddr48	0x3e0	Section 16.14.2	9	2
pmpaddr49	0x3e1	Section 16.14.2	9	2
pmpaddr50	0x3e2	Section 16.14.2	9	2

Continued on next page...

Table 112: (continued)

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
pmpaddr51	0x3e3	Section 16.14.2	9	2
pmpaddr52	0x3e4	Section 16.14.2	9	2
pmpaddr53	0x3e5	Section 16.14.2	9	2
pmpaddr54	0x3e6	Section 16.14.2	9	2
pmpaddr55	0x3e7	Section 16.14.2	9	2
pmpaddr56	0x3e8	Section 16.14.2	9	2
pmpaddr57	0x3e9	Section 16.14.2	9	2
pmpaddr58	0x3ea	Section 16.14.2	9	2
pmpaddr59	0x3eb	Section 16.14.2	9	2
pmpaddr60	0x3ec	Section 16.14.2	9	2
pmpaddr61	0x3ed	Section 16.14.2	9	2
pmpaddr62	0x3ee	Section 16.14.2	9	2
pmpaddr63	0x3ef	Section 16.14.2	9	2

Table 113: PMA Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
pmacfg0	0xbc0	Section 16.15.1	9	2
pmacfg2	0xbc2	Section 16.15.1	9	2
pmaaddr0	0xbd0	Section 16.15.2	9	2
pmaaddr1	0xbd1	Section 16.15.2	9	2
pmaaddr2	0xbd2	Section 16.15.2	9	2
pmaaddr3	0xbd3	Section 16.15.2	9	2
pmaaddr4	0xbd4	Section 16.15.2	9	2
pmaaddr5	0xbd5	Section 16.15.2	9	2
pmaaddr6	0xbd6	Section 16.15.2	9	2
pmaaddr7	0xbd7	Section 16.15.2	9	2
pmaaddr8	0xbd8	Section 16.15.2	9	2
pmaaddr9	0xbd9	Section 16.15.2	9	2
pmaaddr10	0xbda	Section 16.15.2	9	2
pmaaddr11	0xbdb	Section 16.15.2	9	2
pmaaddr12	0xbdc	Section 16.15.2	9	2

Continued on next page...

Table 113: (continued)

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
pmaaddr13	0xbdd	Section 16.15.2	9	2
pmaaddr14	0xbde	Section 16.15.2	9	2
pmaaddr15	0xbdf	Section 16.15.2	9	2

Table 114: Floating-Point CSRs

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
fflags	0x001	Section 16.16.1	0	2
frm	0x002	Section 16.16.2	9	2
fcsr	0x003	Section 16.16.3	9	2

Table 115: User Mode Counter Related Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
cycle	0xc00	Section 16.17.1	N/A	2
time	0xc01	Section 16.17.2	N/A	2
instret	0xc02	Section 16.17.3	N/A	2
hpmcounter3 – hpmcounter31	0xc03 – 0xc1f	Section 16.17.4	N/A	2

Table 116: Vector CSRs

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
vstart	0x008	Section 16.18.1	9	2
vxsat	0x009	Section 16.18.2	9	2
vxrm	0x00a	Section 16.18.3	9	2
vcsr	0x00f	Section 16.18.4	9	2
vl	0xc20	Section 16.18.5	N/A	2
vtype	0xc21	Section 16.18.6	N/A	2
vlenb	0xc22	Section 16.18.7	N/A	2

Table 117: Machine Indirect CSRs

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
miselect	0x350	Section 16.22.1	9	1
mireg2	0x352	Section 16.22.2	9	1
mireg3	0x353	Section 16.22.3	9	1

Table 118: Supervisor Timer CSRs

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
stimecmp	0x14D	Section 16.19.1	4	2

Table 119: Wait for Event CSRs

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
wfe	0x810	Section 16.20.1	9	2
sleepvalue	0x811	Section 16.20.2	9	2
txevt	0x812	Section 16.20.3	9	2

Table 120: RISC-V Zc* Extension Registers

Mnemonic Name	CSR Address	Definition	Write Penalty (Cycles)	Read Issue Latency (Cycles)
jvt	0x017	Section 16.21.1	9	2

16.2 Machine Information Registers

16.2.1 Machine Vendor ID Register

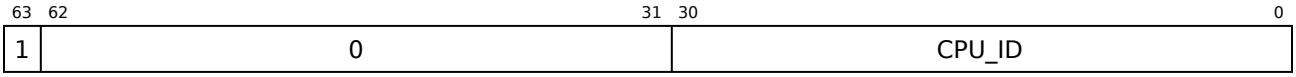
Mnemonic Name: mvendorid
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xf11 (standard read only)

This read-only register provides the Andes JEDEC manufacturer ID, which is 0x0000031e.

Field Name	Bits	Description	Type	Reset
MVENDORID	[63:0]	The manufacturer ID of Andes	RO	0x0000031e

16.2.2 Machine Architecture ID Register

Mnemonic Name: marchid
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xf12 (standard read only)

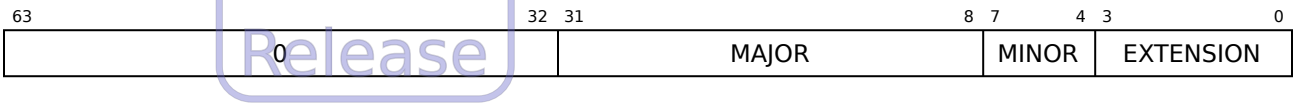


This register provides the micro-architecture ID of AndesCore processor implementations. For AX46MPV, `marchid.CPU_ID` is 0x8a46. Note that the MSB of this register is 1 for commercial implementations of RISC-V processors.

Field Name	Bits	Description	Type	Reset
CPU_ID	[30:0]	Andes CPU ID	RO	0x8a46

16.2.3 Machine Implementation ID Register

Mnemonic Name: mimpid
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xf13 (standard read only)



This register provides the revision number of AX46MPV. Please see *AndesCore AX46MPV Release Note (RN477)* for the exact values. It is documented in the Release Note as MAJOR.MINOR.EXTENSION.

Field Name	Bits	Description	Type	Reset
EXTENSION	[3:0]	Revision extension	RO	IM
MINOR	[7:4]	Revision minor	RO	IM
MAJOR	[31:8]	Revision major	RO	IM

16.2.4 Hart ID Register

Mnemonic Name: mhartid
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xf14 (standard read only)

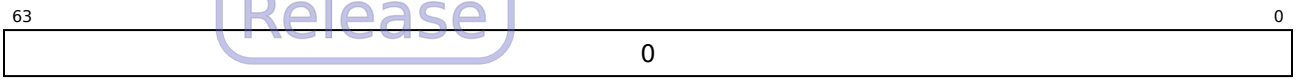
This register provides the ID of the hardware thread. On a RISC-V platform, one of the hart IDs must be zero.

The value of this register is determined by the `hart_id` input port and must be unique across the system.

Field Name	Bits	Description	Type	Reset
MHARTID	[63:0]	Hart ID	RO	IM

16.2.5 Pointer to Configuration Data Structure

Mnemonic Name: mconfigptr
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xf15 (standard, read-only)

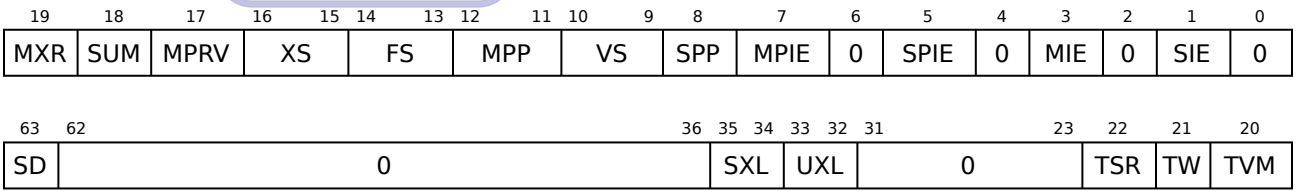


This register holds the physical address of a configuration data structure. AX46MPV does not support this discovering method.

16.3 Machine Trap Related CSRs

16.3.1 Machine Status

Mnemonic Name: mstatus
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x300 (standard read/write)



Field Name	Bits	Description	Type	Reset
SIE	[1]	S-mode interrupt enable bit. **Value** / **Meaning**: 0 Disabled; 1 Enabled	RW	0
MIE	[3]	M-mode interrupt enable bit. **Value** / **Meaning**: 0 Disabled; 1 Enabled	RW	0
SPIE	[5]	`SPIE` holds the value of the `SIE` bit prior to a trap.	RW	0
MPIE	[7]	`MPIE` holds the value of the `MIE` bit prior to a trap.	RW	0
SPP	[8]	`SPP` holds the privilege mode prior to a trap. Encoding is 1 for S-mode and 0 for U-mode.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
VS	[10:9]	VS holds the status of the architectural states of the vector unit, including the vxsat, vxrm, vcsr, vl, vtype, vlenb, vstart CSRs, and v0 – v31 vector registers. The value of this field is zero and read-only if the processor does not implement the vector extension. This field is primarily managed by software. The processor hardware assists the state managements in two regards: • When VS is Off, attempting to access the vxsat, vxrm, vcsr, vl, vtype, vlenb, or vstart CSR upon executing any vector instruction will raise an illegal instruction exception. • When the VS field is Initial or Clean, executing any instruction that changes vector states will change VS to Dirty. Vector instructions that read or write fcsr or any f register are subject to the control of FS bits. They will raise illegal instruction exceptions when FS is Off. However, these CSR registers can be accessed using CSR-access instructions when FS is not Off, regardless of the value of VS. Changing the setting of this field has no effect on the contents of the vector register states. In particular, setting VS to Off does not destroy the states, nor does setting VS to Initial clear the contents.	RW	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Continued on next page...

Field Name	Bits	Description	Type	Reset
MPP	[12:11]	MPP holds the privilege mode prior to a trap. **Value** / **Meaning** 0 – User 1 – Supervisor 2 – Reserved 3 – Machine When U-mode is not available, this field is hardwired to 3.	WARL	3
FS	[14:13]	FS holds the status of the architectural states of the floating-point unit, including the fcsr CSR and f0 – f31 floating-point data registers. The value of this field is zero and read-only if the processor does not have FPU. This field is primarily managed by software. The processor hardware assists the state managements in two regards: • When attempting to access fcsr or any f register, an illegal instruction exception will be raised if FS is set to Off. • FS is updated to Dirty with the execution of any instruction that updates fcsr or any f register when FS is Initial or Clean. Changing the setting of this field has no effect on the contents of the floating-point register states. In particular, setting FS to Off does not destroy the states, nor does setting FS to Initial clear the contents. **Value** / **Meaning** 0 – Off 1 – Initial 2 – Clean 3 – Dirty	WLRL	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
XS	[16:15]	XS holds the status of the architectural states (ACE registers) of ACE instructions. The value of this field is zero if ACE extension is not configured. This field is primarily managed by software. The processor hardware assists the state managements in two regards: • Illegal instruction exceptions are triggered when XS is Off. • XS is updated to Dirty with the execution of ACE instructions when XS is not Off. Changing the setting of this field has no effect on the contents of ACE states. In particular, setting XS to Off does not destroy the states, nor does setting XS to Initial clear the contents.	RO	0
MPRV	[17]	When this bit is set, the memory access privilege for load and store is specified by the MPP field. When U-mode is not available, this field is hardwired to 0.	RW	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Continued on next page...

Field Name	Bits	Description	Type	Reset
SUM	[18]	SUM controls whether an S-mode load/store instruction can access a user-accessible page when page translation is enabled. It is in effect in two scenarios: (a) when in M-mode with MPRV = 1 and MPP = S, and (b) when in S-mode. This bit has no effect when page-based virtual memory is not in effect. A page is considered user-accessible when the U bit of the corresponding PTE entry is 1. SUM is hardwired to 0 when S-mode is not supported.	RW	0

Value	Meaning
0	Not allowed
1	Allowed

Field Name	Bits	Description	Type	Reset
MXR	[19]	MXR controls whether execute-only pages are readable. It has no effect when page-based virtual memory is not in effect. This bit is hardwired to 0 when S-mode is not supported.	RW	0

Value	Meaning
0	Execute-only pages are not readable.
1	Execute-only pages are readable.

Field Name	Bits	Description	Type	Reset
TVM	[20]	TVM controls whether performing certain virtual memory operations in S-mode will raise illegal instruction exceptions. The operations include accessing the satp register and executing the SFENCE.VMA instruction. This bit is hardwired to 0 when S-mode is not supported.	RW	0

Value	Meaning
0	Normal execution
1	Raising exceptions

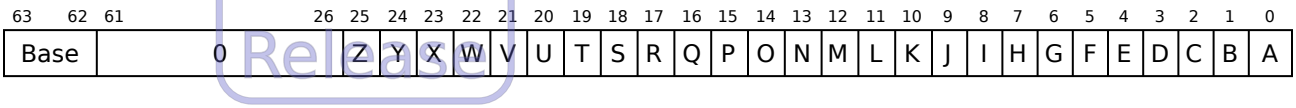
Continued on next page...

Field Name	Bits	Description	Type	Reset
TW	[21]	TW controls whether executing WFI instructions in S-mode will raise illegal instruction exceptions. It is hardwired to 0 when S-mode is not supported. **Value** / **Meaning** 0 — Normal execution 1 — Raising exceptions	RW	0
TSR	[22]	TSR controls whether executing SRET instructions in S-mode will raise illegal instruction exceptions. It is hardwired to 0 when S-mode is not supported. **Value** / **Meaning** 0 — Normal execution 1 — Raising exceptions	RW	0
UXL	[33:32]	UXL controls the value of XLEN for U-mode. When U-mode is not available, this field is hardwired to 0. **Value** / **Meaning** 1 — 32 2 — 64	WARL	2/0
SXL	[35:34]	SXL controls the value of XLEN for S-mode. When S-mode is not available, this field is hardwired to 0. **Value** / **Meaning** 1 — 32 2 — 64	RO	2/0
SD	[63]	SD summarizes whether the FS, XS, or VS field is dirty.	RO	0

When supervisor mode or N extension is not supported, the corresponding bits in mstatus are hard-wired to zero.

16.3.2 Machine ISA Register

Mnemonic Name: misa
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x301 (standard read/write)



Field Name	Bits	Description	Type	Reset
A	[0]	Atomic extension	RO	1
B	[1]	B extension 0: no 1: yes	RO	IM
C	[2]	Compressed extension RISC-V Zce Extension / RISC-V Floating-Point Extension / C No / Don't Care / 1 Yes / None / 1 Yes / Single precision / 1 Yes / Double + Single precision / 0	RO	IM
D	[3]	Double-precision floating-point extension 0: Single precision/none 1: Double + single precision	RO	IM
E	[4]	RV32E base ISA	RO	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
F	[5]	Single-precision floating-point extension **Value** / **RISC-V Floating-Point Instruction Extension** 0 – None 1 – Double + single precision/single precision	RO	IM
G	[6]	Additional standard extensions present	RO	0
H	[7]	Reserved	RO	0
I	[8]	RV32I/64I/128I base ISA	RO	1
J	[9]	*Tentatively reserved for Dynamically Translated Languages extension*	RO	0
K	[10]	Reserved	RO	0
L	[11]	*Tentatively reserved for Decimal Floating-Point extension*	RO	0
M	[12]	Integer Multiply/Divide extension	RO	1
N	[13]	User-level interrupts supported **Value** / **RISC-V User-Level Interrupt Extension** 0 – No 1 – Yes	RO	IM
O	[14]	Reserved	RO	0
P	[15]	*Tentatively reserved for Packed-SIMD extension*	RO	0
Q	[16]	Quad-precision floating-point extension	RO	0
R	[17]	Reserved	RO	0
S	[18]	Supervisor mode implemented **Value** / **Privilege Modes** 0 – Machine/Machine + User 1 – Machine + Supervisor + User	RO	IM
T	[19]	*Tentatively reserved for Transactional Memory extension*	RO	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
U	[20]	User mode implemented Value: 0 — Machine; 1 — Machine + User/Machine + Supervisor + User	RO	IM
V	[21]	*Tentatively reserved for Vector extension*	RO	0
W	[22]	Reserved	RO	0
X	[23]	Non-standard extensions present	RO	1
Y	[24]	Reserved	RO	0
Z	[25]	Reserved	RO	0
Base	[63:62]	The general-purpose register width of the native base integer ISA. Value: 0 — Reserved; 1 — 32; 2 — 64; 3 — 128	RO	2

16.3.3 Machine Exception Delegation

Mnemonic Name: medeleg
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x302 (standard read/write)

63 … 16	15	14	13	12	11 10	9	8	7	6	5	4	3	2	1	0
0	SPF	0	LPF	IPF	0	SEC	UEC	SAF	SAM	LAF	LAM	B	II	IAF	IAM

Field Name	Bits	Description	Type	Reset
IAM	[0]	IAM indicates whether an Instruction Address Misaligned Exception will be delegated to S-mode. **Value** / **Meaning**: 0 — Not delegated; 1 — Delegated	RW	0
IAF	[1]	IAF indicates whether an Instruction Access Fault Exception will be delegated to S-mode. **Value** / **Meaning**: 0 — Not delegated; 1 — Delegated	RW	0
II	[2]	II indicates whether an Illegal Instruction Exception will be delegated to S-mode. **Value** / **Meaning**: 0 — Not delegated; 1 — Delegated	RW	0
B	[3]	B indicates whether an exception triggered by a breakpoint will be delegated to S-mode. **Value** / **Meaning**: 0 — Not delegated; 1 — Delegated	RW	0
LAM	[4]	LAM indicates whether a Load Address Misaligned Exception will be delegated to S-mode. **Value** / **Meaning**: 0 — Not delegated; 1 — Delegated	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
LAF	[5]	LAF indicates whether a Load Access Fault Exception will be delegated to S-mode. Value: 0 = Not delegated; 1 = Delegated	RW	0
SAM	[6]	SAM indicates whether a Store/AMO Address Misaligned Exception will be delegated to S-mode. Value: 0 = Not delegated; 1 = Delegated	RW	0
SAF	[7]	SAF indicates whether a Store/AMO Access Fault Exception will be delegated to S-mode. Value: 0 = Not delegated; 1 = Delegated	RW	0
UEC	[8]	UEC indicates whether an exception triggered by environment call from U-mode will be delegated to S-mode. Value: 0 = Not delegated; 1 = Delegated	RW	0
SEC	[9]	SEC indicates whether an exception triggered by environment call from S-mode will be delegated to S-mode. Value: 0 = Not delegated; 1 = Delegated	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
IPF	[12]	`IPF` indicates whether an Instruction Page Fault Exception will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0
LPF	[13]	`LPF` indicates whether a Load Page Fault exception will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0
SPF	[15]	`SPF` indicates whether a Store/AMO Page Fault Exception will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0

When supervisor mode or N extension is not supported, the corresponding bits in `medeleg` are hard-wired to zero.

16.3.4 Machine Interrupt Delegation

Mnemonic Name: mideleg
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x303 (standard read/write)

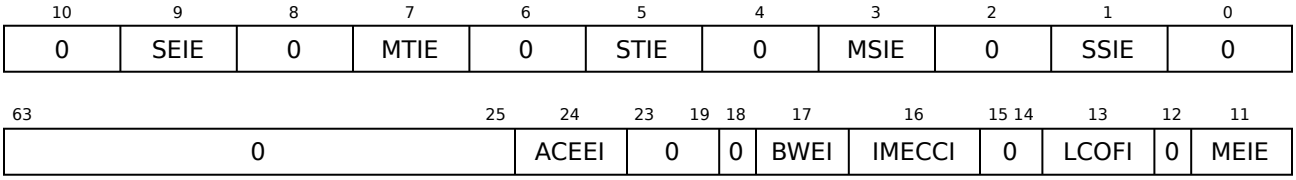
63 … 14	13	12 10	9	8	7 6	5	4	3 2	1	0
0	LCOFI	0	SEI	0	0	STI	0	0	SSI	0

Field Name	Bits	Description	Type	Reset
SSI	[1]	`SSI` indicates whether an S-mode software interrupt will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0
STI	[5]	`STI` indicates whether an S-mode timer interrupt will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0
SEI	[9]	`SEI` indicates whether an S-mode external interrupt will be delegated to S-mode. **Value** / **Meaning**: 0 Not delegated; 1 Delegated	RW	0
LCOFI	[13]	Delegate S-mode performance monitor overflow local interrupts to S-mode. The privileged mode of `LCOFI` determined by the current privileged mode. **Value** / **Meaning**: 0 Do not delegate to S-mode.; 1 Delegate to S-mode.	RW	0

When supervisor mode or N extension is not supported, the corresponding bits in `mideleg` are hard-wired to zero.

16.3.5 Machine Interrupt Enable

Mnemonic Name: mie
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x304 (standard read/write)



Field Name	Bits	Description	Type	Reset
SSIE	[1]	S-mode software interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
MSIE	[3]	M-mode software interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
STIE	[5]	S-mode timer interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
MTIE	[7]	M-mode timer interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
SEIE	[9]	S-mode external interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
MEIE	[11]	M-mode external interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Continued on next page...

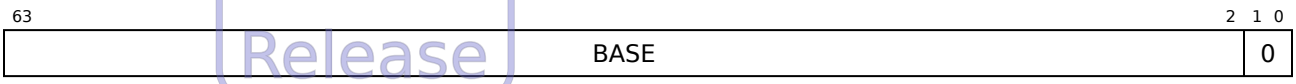
Field Name	Bits	Description	Type	Reset
LCOFI	[13]	Performance monitor overflow local interrupt enable bit. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0
IMECCI	[16]	Imprecise ECC error local interrupt enable bit. The processor may receive imprecise ECC errors on LM subordinate port accesses or cache writebacks. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0
BWEI	[17]	Bus read/write transaction error local interrupt enable bit. The processor may receive bus errors on load/store instructions or cache writebacks. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0
ACEEI	[24]	ACE error local interrupt enable bit. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0

When supervisor mode or N extension is not supported, the corresponding bits in `mie` are hardwired to zero.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields above show the default bit location.

16.3.6 Machine Trap Vector Base Address

Mnemonic Name: mtvec
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x305 (standard read/write)



This register determines the base address of the trap vector. The least significant 2 bits are hardwired to zero. When the configured address width is less than 64, the upper bits are hardwired to zero. When `mmisc_ctl.VEC_PLIC` is 0 (PLIC is not in the vector mode), this register indicates the entry points for the trap handler, and it may point to any 4-byte aligned location in the memory space.

On the other hand, when `mmisc_ctl.VEC_PLIC` is 1 (PLIC is operating in the vector mode), this register will be the base address of a vector table. Each entry is 4 bytes and stores addresses pointing to interrupt service routines. When XLEN = 64, the upper 32-bit address of the interrupt service routines is equal to `mtvec` [63:32].

- This register should be aligned to a $2^{\log_2 N+2}$-byte boundary, where N is the number of PLIC interrupt sources. For example, if N is 1023, the minimum alignment requirement is 4096 bytes (4 KiB).
- `mtvec[0]` is for exceptions and non-external local interrupts.
- `mtvec[`*i*`]` is for external PLIC interrupt source *i* triggered through the `mip.MEIP` pending condition.
- `mtvec[1024+`*i*`]` is for external PLIC interrupt source *i* triggered through
  - the `mip.SEIP` pending condition when `mideleg.SEI` = 0 for M/S/U systems.
  - the `mip.UEIP` pending condition when `mideleg.UEI` = 0 for M/U systems.
- `mtvec[2048+`*i*`]` is for external PLIC interrupt source *i* triggered through the `mip.UEIP` pending condition when `mideleg.UEI` = 0 for M/S/U systems.

Field Name	Bits	Description	Type	Reset
BASE[63:2]	[63:2]	Base address for interrupt and exception handlers. See the above descriptions for alignment requirements when PLIC is operating in the vector mode.	RW	0

16.3.7 Machine Scratch Register

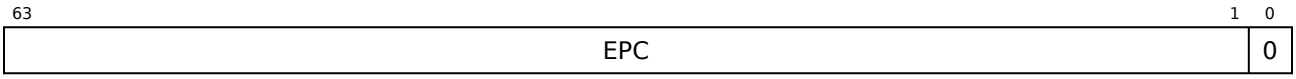
Mnemonic Name: mscratch
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x340 (standard read/write) **Write Penalty**: 0 cycle

This is a scratch register for temporary data storage, which is typically used by the M-mode trap handler.

Field Name	Bits	Description	Type	Reset
MSCRATCH	[63:0]	Scratch register storage	RW	0

16.3.8 Machine Exception Program Counter

Mnemonic Name: mepc
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x341 (standard read/write)



This register is written with the virtual address of the instruction that encountered a trap and/or NMI. The interpretation of this address depends on the configured **Page-Based Virtual Memory** mode:

- "sv39": bits 63–40 are hardwired to bit 39.
- "sv48": bits 63–49 are hardwired to bit 48.
- "sv57": bits 63–57 are hardwired to bit 56.
- "bare": bits 63–BIU_ADDR_WIDTH are hardwired to 0.

In call cases, AX46MPV ignores the corresponding hardwired bits of any written values.

Field Name	Bits	Description	Type	Reset
EPC	[63:1]	Exception program counter	RW	0

16.3.9 Machine Cause Register

Mnemonic Name: mcause
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x342 (standard read/write)

This register records the cause of a trap, reset, NMI, or the interrupt source ID of a vector interrupt. It is updated whenever one of these events occurs. Please see Section 13 on how AX46MPV handles interrupts and exceptions. When multiple events may cause a trap to be taken with the same `mcause` value, the value of `mdcause` records the exact event that causes the trap.

In the RISC-V architecture, exceptions can be precise or imprecise. Precise exceptions are triggered as the standard RISC-V exceptions, with the `mcause.INTERRUPT` bit cleared. On the other hand, imprecise exceptions are considered as local interrupts, with the `mcause.INTERRUPT` bit set.

Field Name	Bits	Description	Type	Reset
EXCEPTION_CODE	[9:0]	Exception code	RW	0
INTERRUPT	[63]	Interrupt	RW	0

The following tables show the possible values of `mcause`:

Table 121: Possible Values of `mcause` After a Trap

Interrupt	Exception Code	Description
1	1	Supervisor software interrupt
1	3	Machine software interrupt
1	5	Supervisor timer interrupt
1	7	Machine timer interrupt
1	9	Supervisor external interrupt
1	11	Machine external interrupt
1	16	Imprecise ECC error interrupt (LM subordinate port accesses, D-Cache evictions, and nonblocking load/stores) (M-mode)
1	17	Bus transaction error interrupt (M-mode)
1	18	Performance monitor overflow interrupt (M-mode)
1	24	ACE error interrupt (M-mode)

Continued on next page...

Table 121: (continued)

Interrupt	Exception Code	Description
1	256+16	Imprecise ECC error interrupt (LM subordinate port accesses, D-Cache evictions, and nonblocking load/stores) (S-mode)
1	256+17	Bus transaction error interrupt (S-mode)
1	256+18	Performance monitor overflow interrupt (S-mode)
1	256+24	ACE error interrupt (S-mode)
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
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	15	Store/AMO page fault
0	32	Stack overflow exception
0	33	Stack underflow exception
0	40–47	Andes Custom Extension exception (see *Andes Custom Extension Specification* for more details)

Table 122: Possible Values of `mcause` After Reset

Interrupt	Exception Code	Description
0	0	Initial value when the processor exits reset (by `coreN_reset_n`)

Table 123: Possible Values of mcause After an NMI

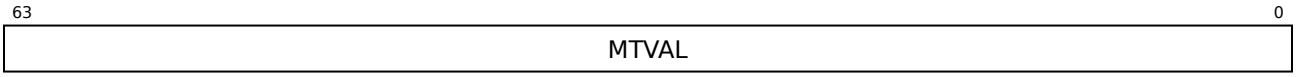
Interrupt	Exception Code	Description
0	0x001	NMI triggered

Table 124: Possible Values of mcause After a Vector Interrupt

mcause	Description
Interrupt source ID	Interrupt source ID when a vector interrupt occurs

16.3.10 Machine Trap Value

Mnemonic Name: mtval
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x343 (standard read/write)



This register is updated when a trap is taken to M-mode. The updated value is dependent on the cause of traps:

- The updated value is the effective faulting addresses for Hardware Breakpoint exceptions, Address Misaligned exceptions, Page Fault exceptions, or Access Fault exceptions.
- The updated value is the faulting instruction for illegal instruction exceptions. If the length of the instruction is less than XLEN bits, the upper bits of mtval are cleared. For exceptions caused by EXEC.IT instructions, the faulting instruction is the translated instruction. Note that if an EXEC.IT instruction is translated to a 16-bit instruction, the translated instruction is considered illegal even if it is normally a valid one.
- For other exceptions, mtval is set to zero.
- For instruction-fetch access faults, this register will be updated with the address indicating the portion of the instruction that caused the fault, while the mepc register will be updated with the address pointing to the beginning of the instruction.

When the configured address width is less than 64, the upper bits of mtval are hardwired to zeros.

Field Name	Bits	Description	Type	Reset
MTVAL	[63:0]	Exception-specific information for software trap handling	RW	0

16.3.11 Machine Interrupt Pending

Mnemonic Name: mip
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x344 (standard read/write)

Field Name	Bits	Description	Type	Reset
SSIP	[1]	S-mode software interrupt pending bit. Value: 0 = Not pending; 1 = Pending	RW	0
MSIP	[3]	M-mode software interrupt pending bit. Value: 0 = Not pending; 1 = Pending	RO	0
STIP	[5]	S-mode timer interrupt pending bit. Value: 0 = Not pending; 1 = Pending	RW	0
MTIP	[7]	M-mode timer interrupt pending bit. Value: 0 = Not pending; 1 = Pending	RO	0

Continued on next page...

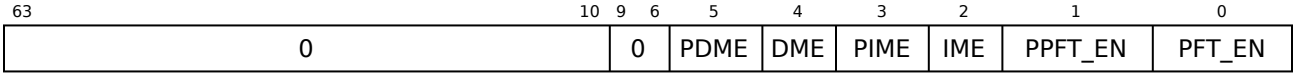
Field Name	Bits	Description	Type	Reset
SEIP	[9]	S-mode external interrupt pending bit. **Value** / **Meaning** 0 — Not pending 1 — Pending	RW	0
MEIP	[11]	M-mode external interrupt pending bit. **Value** / **Meaning** 0 — Not pending 1 — Pending	RO	0
LCOFI	[13]	Performance monitor overflow local interrupt pending bit. **Value** / **Meaning** 0 — Not pending 1 — Pending	RW	0
IMECCI	[16]	Imprecise ECC error local interrupt pending bit. The processor may receive imprecise ECC errors on LM subordinate port accesses or cache writebacks. **Value** / **Meaning** 0 — Not pending 1 — Pending	RW	0
BWEI	[17]	Bus read/write transaction error local interrupt pending bit. The processor may receive bus errors on load/store instructions or cache writebacks. **Value** / **Meaning** 0 — Not pending 1 — Pending	RW	0
ACEEI	[24]	ACE error local interrupt pending bit. **Value** / **Meaning** 0 — Not pending 1 — Pending	RO	0

When supervisor mode or N extension is not supported, the corresponding bits in `mip` are hardwired to zero.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields above show the default bit location.

16.3.12 Machine Extended Status

Mnemonic Name: mxstatus
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x7c4 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
PFT_EN	[0]	Enable performance throttling. When throttling is enabled, the processor executes instructions at the performance level specified in `mpft_ctl.T_LEVEL`. On entering a trap: • `PPFT_EN` ⇐ `PFT_EN`; • `PFT_EN` ⇐ `mpft_ctl.FAST_INT` ? 0 : `PFT_EN`; On executing an `MRET` instruction: • `PFT_EN` ⇐ `PPFT_EN`; This field is hardwired to 0 if the PowerBrake feature is not supported.	RW	0
PPFT_EN	[1]	For saving the previous `PFT_EN` state on entering a trap. This field is hardwired to 0 if the PowerBrake feature is not supported.	RW	0

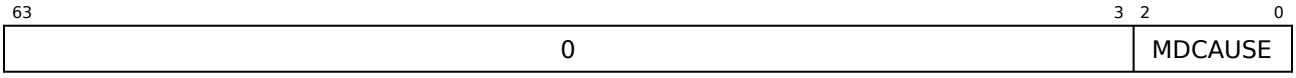
Continued on next page...

Field Name	Bits	Description	Type	Reset
IME	[2]	Instruction Machine Error flag. It indicates an ECC exception occurred in the instruction cache or instruction local memory (ILM). Instruction accesses will bypass the I-Cache when this bit is set. The exception handler should clear this bit after the machine error has been dealt with. Note that when a load/store to ILM access causes an ECC error exception, DME is set instead of IME. IME is overwritten by PIME on MRET.	RW	0
PIME	[3]	For saving the previous `IME` state on entering a trap. This field is hardwired to 0 if instruction cache and instruction local memory are not supported.	RW	0
DME	[4]	Data Machine Error flag. It indicates an ECC exception occurred in the data cache or data local memory (DLM). Load/store accesses will bypass the D-Cache when this bit is set. The exception handler should clear this bit after the machine error has been dealt with. Note that when a load/store to ILM access causes an ECC error exception, DME is set instead of IME. DME is overwritten by PDME on MRET.	RW	0
PDME	[5]	For saving the previous `DME` state on entering a trap. This field is hardwired to 0 if data cache and data local memory are not supported.	RW	0

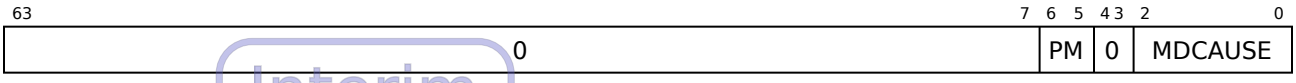
16.3.13 Machine Detailed Trap Cause

Mnemonic Name: mdcause
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x7c9 (non-standard read/write)

For precise exceptions:



For imprecise exceptions (local interrupts):



When multiple events cause traps to be taken with the same `mcause` value, this register helps to further disambiguate them. Certain events can trigger either precise exceptions or imprecise exceptions (local interrupts), depending on when they are detected. Consequently, they can appear in multiple tables below.

Imprecise exceptions are triggered as local interrupts. Thus, the tables below for `mcause` == Local Interrupt *n* summarize imprecise exceptions delivered as local interrupt *n*. The `mcause` == Local Interrupt *n* notation standing for (`INTERRUPT`, `EXCEPTION_CODE`) fields of `mcause` is (1, *n*).

Field Name	Bits	Description	Type	Reset
MDCAUSE	[2:0]	This register further disambiguates causes of traps recorded in the `mcause` register. See the list below for details.	RW	0
PM	[6:5]	When `mcause` is imprecise exception (in the form of an interrupt), the PM field records the current privileged mode. The PM field encoding is defined as follows: **Value** / **Meaning** 0 — User mode 1 — Supervisor mode 2 — Reserved 3 — Machine mode	RW	0

The value of `MDCAUSE` for precise exception:

- When `mcause` == 1 (Instruction access fault):

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP instruction access violation
3	Bus error

Continued on next page...

Value	Meaning
4	PMA empty hole access

- When `mcause` == 2 (Illegal instruction):

Value	Meaning
0	The actual faulting instruction is stored in the `mtval` CSR.
1	FP disabled exception
2	ACE disabled exception
3	RVV disabled exception

- When `mcause` == 5 (Load access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP load access violation
3	Bus error
4	Misaligned address
5	PMA empty hole access
6	PMA attribute inconsistency
7	PMA NAMO exception

- When `mcause` == 7 (Store access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP store access violation
3	Bus error
4	Misaligned address
5	PMA empty hole access
6	PMA attribute inconsistency
7	PMA NAMO exception

The value of MDCAUSE for imprecise exception:

- When mcause == *Local Interrupt 16* or *Local Interrupt 272 (16 + 256)* (ECC error local interrupt)

Value	Meaning
0	Reserved
1	LM subordinate port ECC/Parity error
2	Imprecise store ECC/Parity error
3	Imprecise load ECC/Parity error

- When mcause == *Local Interrupt 17* or *Local Interrupt 273 (17 + 256)* (Bus read/write transaction error local interrupt)

Value	Meaning
0	Reserved
1	Reserved
2	Bus error
3	PMP error caused by load instructions
4	PMP error caused by store instructions
5	PMA error caused by load instructions
6	PMA error caused by store instructions

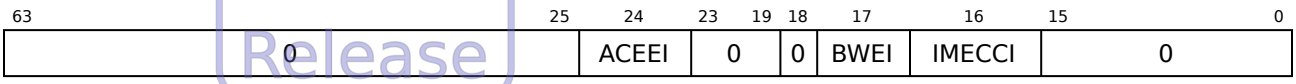
- For other exceptions and interrupts, this register are not updated.

16.3.13.1 Detailed Exception Priority

For instruction, load, and store access fault exceptions, a PMP exception has higher priority than a PMA exception when both occur on the same instruction.

16.3.14 Machine Supervisor Local Interrupt Delegation

Mnemonic Name: mslideleg
IM Requirement: misa[18]==1
Access Mode: Machine
CSR Address: 0x7D5 (non-standard read/write)



This register controls the delegation of supervisor local interrupts. If a supervisor local interrupt is not delegated, the supervisor local interrupt will be taken in M-mode. If a supervisor local interrupt is delegated, the supervisor local interrupt will be taken in S-mode.

The privileged mode of `ACEEI`, `IMECCI`, and `BWEI` are determined by the current privileged mode. For M/S/U configuration, if the current privileged mode is User or Supervisor, the local interrupt generated is a supervisor local interrupt. For M/U configuration, if the current privileged mode is User, the local interrupt generated is a machine local interrupt.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields below show the default bit location.

Field Name	Bits	Description	Type	Reset
IMECCI	[16]	Delegate S-mode imprecise ECC error local interrupts to S-mode. **Value** / **Meaning** 0 — Do not delegate to S-mode. 1 — Delegate to S-mode.	RW	0
BWEI	[17]	Delegate S-mode bus read/write transaction error local interrupts to S-mode. **Value** / **Meaning** 0 — Do not delegate to S-mode. 1 — Delegate to S-mode.	RW	0
ACEEI	[24]	Delegate S-mode ACE error local interrupts to S-mode. **Value** / **Meaning** 0 — Do not delegate to S-mode. 1 — Delegate to S-mode.	RW	0

16.4 Counter Related CSRs

16.4.1 Machine Cycle Counter

Mnemonic Name: mcycle
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xb00 (standard read/write)

The `mcycle` CSR counts the number of cycles executed by the hart since some arbitrary time in the past. This register provides a precision of 64 bits.

16.4.2 Machine Instruction-Retired Counter

Mnemonic Name: minstret
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xb02 (standard read/write)

The `minstret` CSR counts the number of instructions that the hart has retired since some arbitrary time in the past. This register provides a precision of 64 bits.

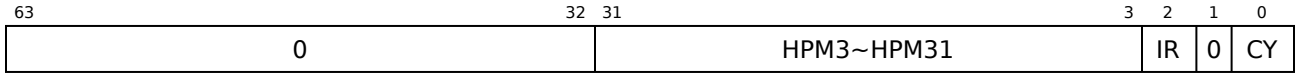
16.4.3 Machine Performance Monitoring Counter

Mnemonic Name: mhpmcounter3–mhpmcounter31
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xb03 to 0xb1f (standard read/write)

The `mhpmcounter3 - mhpmcounter31` CSRs count the number of events selected by `mhpmevent3 - mhpmevent31`.

16.4.4 Machine Counter-Inhibit

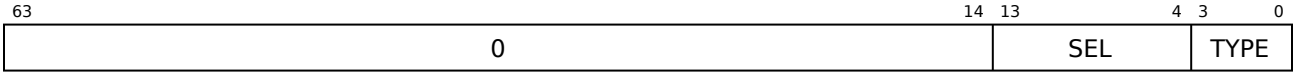
Mnemonic Name: mcountinhibit
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x320 (non-standard read/write)



The counter-inhibit register controls which counters should not be incremented. When the CY, IR, or HPMn bit is set, the corresponding counter will not be incremented on the event.

16.4.5 Machine Performance Monitoring Event Selector

Mnemonic Name: mhpmevent3–mhpmevent31
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x323 to 0x33f (standard read/write)



The event selectors are defined in Table 125. Micro-architectural events are mostly speculative in nature. The counted events include those caused by speculative actions, unless they are defined as non-speculative in the **Comment** column in the table below. In particular, all *retired* instruction counts are considered non-speculative.

Table 125: Event Selectors

TYPE	SEL	Event Name	Comment
0	1	Cycle count	Number of elapsed processor clock cycles
0	2	Retired instruction count	Number of retired instructions
0	3	Integer load instruction count	Number of retired load instructions (including LR)
0	4	Integer store instruction count	Number o retired store instructions (including SC)
0	5	Atomic instruction count	Number of retired atomic instructions (excluding LR and SC)
0	6	System instruction count	Number of retired SYSTEM instructions (instructions with major opcode equal to 0b1110011)
0	7	Integer computational instruction count	Number of retired integer computational instructions
0	8	Conditional branch instruction count	Number of retired conditional branch instructions
0	9	Taken conditional branch instruction count	Number of retired conditional branch instructions that are taken
0	10	JAL instruction count	Number of retired JAL instructions

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
0	11	JALR instruction count	Number of retired JALR instructions. This event selector also counts the events monitored by the *return instruction count* defined in the next row.
0	12	Return instruction count	Number of retired return instructions. • JALR instructions with zero immediate offset and the following operands: – (rd != x1/x5) and (rs1 == x1/x5) – rd == x1 and rs1 == x5 – rd == x5 and rs1 == x1 • POPRET instructions
0	13	Control transfer instruction count	Number of retired unconditional jumps (JAL and JALR) and conditional branch instructions
0	14	EXEC.IT instruction count	Number of retired EXEC.IT instructions
0	15	Integer multiplication instruction count	Number of retired integer multiplication instructions
0	16	Integer division instruction count	Number of retired integer division/remainder instructions
0	17	Floating-point load instruction count	Number of retired floating-point load instructions
0	18	Floating-point store instruction count	Number of retired floating-point store instructions
0	19	Floating-point addition instruction count	Number of retired floating-point addition/subtraction instructions
0	20	Floating-point multiplication instruction count	Number of retired floating-point multiplication instructions
0	21	Floating-point fused multiply-add instruction count	Number of retired floating-point fused multiply-add/subtraction instructions (FMADD, FMSUB, FNMSUB, and FNMADD)
0	22	Floating-point division or square-root instruction count	Number of retired floating-point division/square-root instructions
0	23	Other floating-point instruction count	Number of retired floating-point instructions not counted by the previous floating-point instruction event selectors

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
0	24	Integer multiplication and add/sub instruction count	Number of retired integer multiplication and add/sub instructions
0	25	Retired operation count	Number of retired operations • Vector multiply-add instructions are counted as 2*VL operations. • The following vector instructions are counted as 1 operation: – vsetvl(i), – vector integer/FP scalar move, – vector load/store whole register, – whole vector register move. • Other vector instructions (including reduction) are counted as VL operations. • Floating-point multiply-add instructions are counted as 2 operations. • `CM.PUSH` instructions are counted as N+1 operations. (N: Number of registers in a `CM.PUSH` instruction) • `CM.POP` instructions are counted as N+1 operations. (N: Number of registers in a `CM.POP` instruction) • `CM.POPRET` instructions are counted as N+2 operations. (N: Number of registers in a `CM.POPRET` instruction) • `CM.POPRETZ` instructions are counted as N+3 operations. (N: Number of registers in a `CM.POPRETZ` instruction) • other instructions are counted as 1 operation.
0	26	Integer push instruction count	Number of retired integer push instructions
0	27	Integer pop instruction count	Number of retired integer pop instructions
0	64	Vector instructions (all vector instructions are counted as 1 operation. Zvb* is included and Zvk* is excluded)	Number of retired vector instructions

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
0	65	Vector integer arithmetic instructions	Number of retired vector integer arithmetic instructions
0	66	Vector integer multiply instructions	Number of retired vector integer multiply instructions
0	67	Vector integer multiply-add instructions	Number of retired vector integer multiply-add instructions
0	68	Vector integer divide instructions	Number of retired vector integer divide instructions
0	69	Vector integer reduction instructions	Number of retired vector integer reduction instructions
0	70	Vector integer fixed-point arithmetic instructions	Number of retired vector integer fixed-point arithmetic instructions
0	96	Vector floating-point arithmetic instructions	Number of retired vector floating-point arithmetic instructions
0	97	Vector floating-point multiply instructions	Number of retired vector floating-point multiply instructions
0	98	Vector floating-point fused MAC instructions	Number of retired vector floating-point fused MAC instructions
0	99	Vector floating-point divide instructions	Number of retired vector floating-point divide instructions
0	100	Vector floating-point square-root instructions	Number of retired vector floating-point square-root instructions
0	101	Vector floating-point/integer type-convert instructions	Number of retired vector floating-point/integer type-convert instructions
0	102	Vector floating-point reduction instructions	Number of retired vector floating-point reduction instructions
0	128	Vector mask instructions	Number of retired vector mask instructions
0	129	Vector permutation instructions	Number of retired vector permutation instructions
0	130	Vector Dot Product instructions	Number of retired vector Dot Product instructions
0	131	Vector Small INT Handling instructions	Number of retired vector small INT handling instructions
0	132	Vector Quad-Widening Integer Multiply-Add instructions	Number of retired vector quad-widening integer multiply-add instructions

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
0	133	Vector packed FP16 instructions	Number of retired vector packed FP16 instructions
0	160	Vector Unit-Stride Loads	Number of retired vector unit-stride loads instructions
0	161	Vector Unit-Stride Stores	Number of retired vector unit-stride stores instructions
0	162	Vector Strided Loads	Number of retired vector strided loads instructions
0	163	Vector Strided Stores	Number of retired vector strided stores instructions
0	164	Vector Indexed Loads	Number of retired vector indexed loads instructions
0	165	Vector Indexed Stores	Number of retired vector indexed stores instructions
0	166	Vector Unit-Stride Segment Loads	Number of retired vector unit-stride segment loads instructions
0	167	Vector Unit-Stride Segment Stores	Number of retired vector unit-stride segment stores instructions
0	168	Vector Strided Segment Loads	Number of retired vector strided segment loads instructions
0	169	Vector Strided Segment Stores	Number of retired vector strided segment stores instructions
0	170	Vector Indexed Segment Loads	Number of retired vector indexed segment loads instructions
0	171	Vector Indexed Segment Stores	Number of retired vector indexed segment stores instructions
0	172	Vector Whole-Register Loads	Number of retired vector whole-register loads instructions
0	173	Vector Whole-Register Stores	Number of retired vector whole-register stores instructions
0	174	Vector configuration Setting instructions	Number of retired vector configuration setting instructions
0	192	ACE instructions	Number of retired ACE instructions
0	193	ACE-Scalar instructions	Number of retired ACE-Scalar instructions
0	194	ACE Streaming-port instructions	Number of retired ACE Streaming-port instructions
0	195	ACE-RVV instructions	Number of retired ACE-RVV instructions

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
0	256	CMO prefetch hints	Number of retired CMO prefetch hints
0	256	Reserved	
1	0	ILM access	Number of ILM transfers, including speculative instruction fetch, load/store accesses, ECC repair, and LM subordinate port accesses
1	1	DLM access	Number of DLM transfers, including speculative load/store accesses, ECC repair, and LM subordinate port accesses
1	2	I-Cache access	Number of completed I-Cache fetch access
1	3	I-Cache miss	Number of I-Cache fetch miss
1	4	D-Cache access*	Number of completed D-Cache load-and-store access. Misaligned load/store accesses may increase this counter by either one or two, depending on access sizes and alignments. Only misaligned accesses crossing two cache lines are guaranteed to result in an increment of two.
1	5	D-Cache miss*	The event counts the number of D-Cache load-and-store miss. Misaligned load/store accesses may increase this counter by either zero, one or two, depending on access sizes, alignments and whether the accessed lines are in D-Cache.
1	6	D-Cache load access*	Number of completed D-Cache load access. See the D-Cache access count event selector for the handling of misaligned load accesses.
1	7	D-Cache load miss*	Number of D-Cache load miss. See the D-Cache miss count event selector for the handling of misaligned load accesses.
1	8	D-Cache store access*	Number of completed D-Cache store access. See the D-Cache access count event selector for the handling of misaligned load accesses.
1	9	D-Cache store miss*	Number of D-Cache store miss. See the D-Cache miss count event selector for the handling of misaligned load accesses.
1	10	D-Cache writeback*	Number of D-Cache writeback

Continued on next page...

Table 125: (continued)

TYPE	SEL	Event Name	Comment
1	11	Cycles waiting for I-Cache fill data*	Number of cycles waiting for the return of the critical word of I-Cache misses from the system bus. This event selector does not monitor accesses to I/O regions or cacheable regions when I-Cache is turned off.
1	12	Cycles waiting for D-Cache fill data*	Number of cycles waiting for the return of the critical word of D-Cache misses from the system bus. This event selector does not monitor accesses to I/O regions or cacheable regions when D-Cache is turned off. Additionally, non-blocking loads do not increment this counter since they do not cause pipeline stalls under D-Cache misses.
1	13	Uncached fetch data access from bus*	Number of accesses of uncached instruction data returning from the system bus. This event selector monitors accesses to I/O regions or cacheable regions when I-Cache is not configured or off.
1	14	Uncached load data access from bus*	Number of accesses of uncached load data returning from the system bus. This event selector monitors accesses to I/O regions or cacheable regions when D-Cache is not configured or off.
1	15	Cycles waiting for uncached fetch data from bus*	Number of cycles waiting for the instruction data to return from the system bus. This event selector monitors accesses to I/O regions or cacheable regions when I-Cache is not configured or off.
1	16	Cycles waiting for uncached load data from bus*	Number of cycles waiting for the load data to return from the system bus. This event selector monitors accesses to I/O regions or cacheable regions when D-Cache is not configured or off.
1	17	Main ITLB access	Number of address translation requests performed by the shared TLB for instruction fetches
1	18	Main ITLB miss	Number of address translation requests from the instruction fetch that misses the Shared TLB and invokes the hardware page table walker

Continued on next page...

Table 125: (continued)

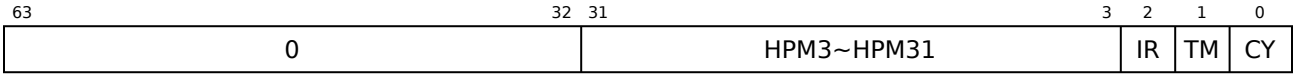
TYPE	SEL	Event Name	Comment
1	19	Main DTLB access	Number of address translation requests performed by the Shared TLB for load/store accesses
1	20	Main DTLB miss	Number of address translation requests from load/store accesses that miss the Shared TLB and invoke the hardware page table walker
1	21	Cycles waiting for Main ITLB fill data	Number of instruction fetch stall cycles attributable to TLB misses
1	22	Pipeline stall cycles caused by Main DTLB miss	Number of pipeline stall cycles attributable to address translation for load/store accesses
1	23	Hardware prefetch bus access	This event counts the bus accesses generated by the hardware data prefetcher
1	24	Cycles waiting for source operand ready in the integer register file scoreboard	Number of cycles waiting for source operand ready in the integer register file scoreboard. This event monitors the waiting cycles caused by nonblocking execution instructions.
1	25–28	Reserved	
2	0	Misprediction of conditional branches (direction)	Number of misprediction of committed conditional branches
2	1	Misprediction of taken conditional branches (direction)	Number of misprediction of committed taken conditional branches
2	2	Misprediction of targets of Return instructions	Number of misprediction of committed Return instructions
2	4–19	Reserved	
3	0-31	MSHR entry accumulated miss counts	Number of miss counts in the selected MSHR entry. If the SEL number is a nonexistent MSHR entry, the counter value will not change.
4	0-31	MSHR entry accumulated miss latencies	Number of cycles waiting for filling data in the selected MSHR entry. If the SEL number is a nonexistent MSHR entry, the counter value will not change.

Note

- Interrupts are expected to be disabled when monitoring D-Cache related events and cycles waiting related events.

16.4.6 Machine Counter Enable

Mnemonic Name: mcounteren
IM Requirement: Required if User mode is implemented
Access Mode: Machine
CSR Address: 0x306 (standard read/write)



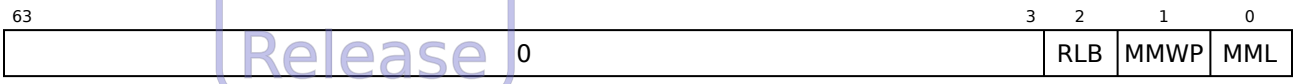
The machine counter-enable register controls the availability of the hardware performance monitoring counters to the next-lowest privileged mode. The default value of this register is 0.

When `CY`, `TM`, `IR`, or `HPM3` to `HPM31` in this register is 0, attempts to read the `cycle`, `time`, `instret`, or `hpmcounter3` to `hpmcounter31` register while executing in U-mode (M/U configuration) or S-mode (M/S/U configuration) will cause an illegal instruction exception. When one of these bits is set, accessing to the corresponding register is permitted in the next implemented privilege mode.

AX46MPV implements the RDTIME instruction which reads the value from the `time` CSR. The `time` CSR is a read-only shadow of the `mtime` register in NCEPLMT210. The RDTIME instruction has the XCSR latency type, which may introduce additional latency due to bus transactions and hardware synchronization protocols. See AndesCore AX46MPV Integration Guide (IG094) for details.

16.4.7 Machine Security Configuration Register

Mnemonic Name: mseccfg
IM Requirement: mrvarch_cfg.Smepmp==1
Access Mode: Machine
CSR Address: 0x747 (standard read/write)



Field Name	Bits	Description	Type	Reset
MML	[0]	Machine Mode Lockdown. If this bit is set, it can only be cleared to 0 by a system reset.	RW	0
MMWP	[1]	Machine Mode Whitelist Policy.	RW	0

Value	Meaning
0	M-mode access without a matching PMP rule is permitted.
1	M-mode access without a matching PMP rule is not permitted. This bit can only be cleared to 0 by a system reset.

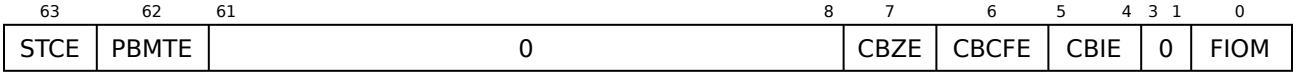
Continued on next page...

Field Name	Bits	Description	Type	Reset
RLB	[2]	Rule locking bypass control.	RW	0

Value	Meaning
0	If pmpicfg.L is 1, writes to pmpicfg and pmpiaddr are ignored. Additionally, if pmpicfg.A is set to TOR, writes to pmpaddr(i-1) is ignored as well. If any pmpicfg.L is 1, any modifications to RLB are ignored until a system reset.
1	If pmpicfg.L is 1, writes to pmpicfg and pmpaddr(i) are allowed. Additionally, if pmpicfg.A is set to TOR, writes to pmpaddr(i-1) is allowed as well.

16.4.8 Machine Environment Configuration Register

Mnemonic Name: menvcfg
IM Requirement: misa.U==1
Access Mode: Machine
CSR Address: 0x30a (standard read/write)



This register controls certain characteristics of the execution environment for modes less privileged than M.

Field Name	Bits	Description	Type	Reset
FIOM	[0]	FENCE of I/O implies Memory. 0: FENCE instructions used to order accesses to device I/O are not modified. 1: FENCE instructions executed in less privileged modes (e.g., U-mode) are modified such that ordering accesses to device I/O also implies ordering accesses to memory (PI implies PR, PO implies PW, SI implies SR, and SO implies SW). In addition, if an AMO instruction that accesses an I/O region has its aq and/or rl bit set, it is ordered as though it accesses both device I/O and memory.	RW	0
CBIE	[5:4]	Cache Block Invalidate instruction Enable. 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed and performs a flush operation. 2: Reserved 3: The instruction is executed and performs an invalidate operation.	RW	0
CBCFE	[6]	Cache Block Clean and Flush instruction Enable. 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
CBZE	[7]	Cache Block Zero instruction Enable. 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed.	RW	0
PBMTE	[62]	PBMTE controls whether the Svpbmt extension is available for use in S-mode address translation. If Svpbmt is not implemented, PBMTE is read-only zero. 0: Svpbmt is not available for S-mode address translation 1: Svpbmt is available for S-mode address translation.	RW	0
STCE	[63]	For Sstc extension	RW	0

16.5 Configuration Control & Status Registers

16.5.1 Instruction Cache/Memory Configuration Register

Mnemonic Name: micm_cfg
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xfc0 (non-standard read only)

63 27	26 25	24	23	22 21	20	19 15	14 12	11 10	9	8 6	5 3	2 0
0	IC_REPL	SETH	0	ILM_ECC	0	ILMSZ	ILMB	IC_ECC	ILCK	ISZ	IWAY	ISET

This register provides information about configurations of the instruction cache and instruction memory.

Field Name	Bits	Description	Type	Reset
ISET	[2:0]	I-Cache settings (# of cache lines per way): When `micm_cfg.SETH` = 0:	RO	IM

Value	Meaning
0	64
1	128
2	256
3	512
4	1024
5	2048
6	4096
7	Reserved

When `micm_cfg.SETH` = 1:

Value	Meaning
0	32
1	16
2	8
3~7	Reserved

- When the I-Cache is not configured, this field should be ignored.

Continued on next page...

Field Name	Bits	Description	Type	Reset
IWAY	[5:3]	Associativity of I-Cache	RO	IM

Value	Meaning
0	Direct-mapped
1	2-way
2	3-way
3	4-way
4	5-way
5	6-way
6	7-way
7	8-way

- When the I-Cache is not configured, this field should be ignored.

Field Name	Bits	Description	Type	Reset
ISZ	[8:6]	I-Cache block (line) size	RO	IM

Value	Meaning
0	No I-Cache
1	8 bytes
2	16 bytes
3	32 bytes
4	64 bytes
5	128 bytes
6,7	Reserved

- When the I-Cache is not configured, this field should be ignored.

Field Name	Bits	Description	Type	Reset
ILCK	[9]	I-Cache locking support	RO	IM

Value	Meaning
0	No locking support
1	With locking support

- When the I-Cache is not configured, this field should be ignored.

Continued on next page...

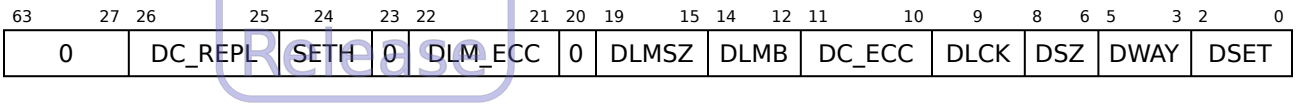
Field Name	Bits	Description	Type	Reset
IC_ECC	[11:10]	I-Cache soft-error protection scheme **Value** / **Meaning**: 0 = No parity/ECC; 1 = Parity; 2 = ECC; 3 = Reserved • When the I-Cache is not configured, this field should be ignored.	RO	IM
ILMB	[14:12]	Number of ILM base registers present **Value** / **Meaning**: 0 = No ILM base register present; 1 = One ILM base register present; 2-7 = Reserved • When ILM is not configured, this field should be ignored.	RO	IM

Continued on next page. . .

Field Name	Bits	Description	Type	Reset
ILMSZ	[19:15]	ILM Size **Value** / **Meaning** 0 — 0 Byte 1 — 1 KiB 2 — 2 KiB 3 — 4 KiB 4 — 8 KiB 5 — 16 KiB 6 — 32 KiB 7 — 64 KiB 8 — 128 KiB 9 — 256 KiB 10 — 512 KiB 11 — 1 MiB 12 — 2 MiB 13 — 4 MiB 14 — 8 MiB 15 — 16 MiB 16-31 — Reserved • When ILM is not configured, this field should be ignored.	RO	IM
ILM_ECC	[22:21]	ILM soft-error protection scheme **Value** / **Meaning** 0 — No parity/ECC 1 — Parity 2 — ECC 3 — Reserved • When ILM is not configured, this field should be ignored.	RO	IM
SETH	[24]	This bit extends the `ISET` field. • When the I-Cache is not configured, this field should be ignored.	RO	IM
IC_REPL	[26:25]	This field is hardwired to zero.	RO	0

16.5.2 Data Cache/Memory Configuration Register

Mnemonic Name: mdcm_cfg
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xfc1 (non-standard read only)



This register provides information about the configurations of the data cache and data local memory.

Field Name	Bits	Description	Type	Reset
DSET	[2:0]	D-Cache settings (# of cache lines per way): When `mdcm_cfg.SETH` = 0:	RO	IM

Value	Meaning
0	64
1	128
2	256
3	512
4	1024
5	2048
6	4096
7	Reserved

When `mdcm_cfg.SETH` = 1:

Value	Meaning
0	32
1	16
2	8
3~7	Reserved

- When the D-Cache is not configured, this field should be ignored.

Continued on next page...

Field Name	Bits	Description	Type	Reset
DWAY	[5:3]	Associativity of D-Cache	RO	IM

Value	Meaning
0	Direct-mapped
1	2-way
2	3-way
3	4-way
4	5-way
5	6-way
6	7-way
7	8-way

- When the D-Cache is not configured, this field should be ignored.

Field Name	Bits	Description	Type	Reset
DSZ	[8:6]	D-Cache block (line) size	RO	IM

Value	Meaning
0	No D-Cache
1	8 bytes
2	16 bytes
3	32 bytes
4	64 bytes
5	128 bytes
6,7	Reserved

- When the D-Cache is not configured, this field should be ignored.

Field Name	Bits	Description	Type	Reset
DLCK	[9]	D-Cache locking support	RO	IM

Value	Meaning
0	No locking support
1	With locking support

- When D-Cache is not configured, this field should be ignored.

Continued on next page...

Field Name	Bits	Description	Type	Reset
DC_ECC	[11:10]	D-Cache soft-error protection scheme **Value** / **Meaning** 0 — No parity/ECC support 1 — Has parity support 2 — Has ECC support 3 — Reserved • When the D-Cache is not configured, this field should be ignored.	RO	IM
DLMB	[14:12]	Number of DLM base registers present **Value** / **Meaning** 0 — No DLM base register present 1 — One DLM base register present 2-7 — Reserved • When DLM is not configured, this field should be ignored.	RO	IM

Continued on next page...

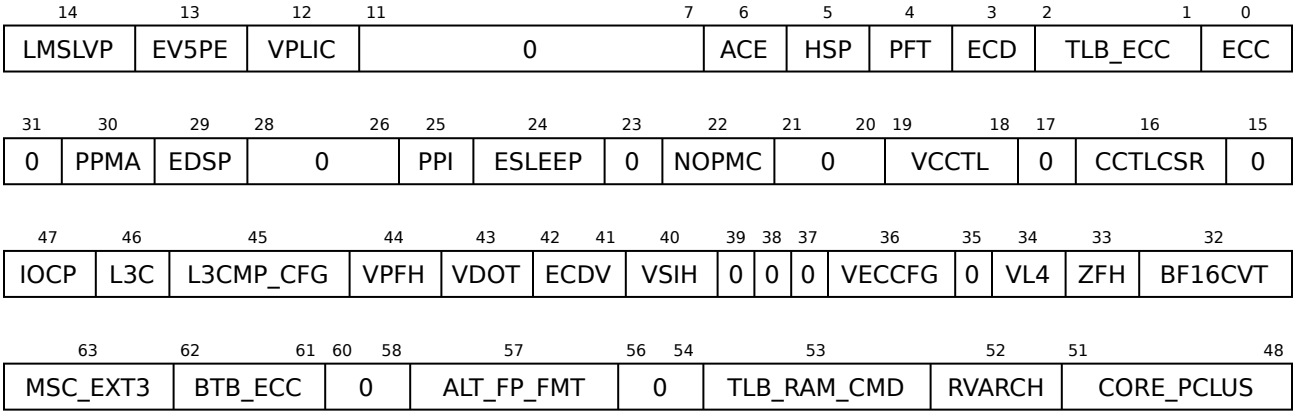
Field Name	Bits	Description	Type	Reset
DLMSZ	[19:15]	DLM Size **Value** / **Meaning** 0 — 0 Byte 1 — 1 KiB 2 — 2 KiB 3 — 4 KiB 4 — 8 KiB 5 — 16 KiB 6 — 32 KiB 7 — 64 KiB 8 — 128 KiB 9 — 256 KiB 10 — 512 KiB 11 — 1 MiB 12 — 2 MiB 13 — 4 MiB 14 — 8 MiB 15 — 16 MiB 16-31 — Reserved • When DLM is not configured, this field should be ignored.	RO	IM
DLM_ECC	[22:21]	DLM soft-error protection scheme **Value** / **Meaning** 0 — No parity/ECC 1 — Parity 2 — ECC 3 — Reserved • When DLM is not configured, this field should be ignored.	RO	IM
SETH	[24]	This bit extends the `DSET` field. • When the D-Cache is not configured, this field should be ignored.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
DC_REPL	[26:25]	Indicates D-Cache replacement policy **Value** / **Meaning** 0 — Reserved 1 — **D-Cache Replacement Policy** is pseudo-lru. 2 — Reserved 3 — Reserved • When the D-Cache is not configured, this field should be ignored.	RO	IM

16.5.3 Misc. Configuration Register

Mnemonic Name: mmsc_cfg
IM Requirement: Required
Access Mode: Machine
CSR Address: 0xfc2 (non-standard read only)



This register provides information regarding miscellaneous processor configurations.

Field Name	Bits	Description	Type	Reset
ECC	[0]	Indicates whether the parity/ECC soft-error protection is implemented or not. **Value** / **Meaning**: 0 = Not implemented; 1 = Implemented The specific parity/ECC scheme used for each protected RAM is specified by the control bits in the following list: • `micm_cfg.IC_ECC` • `micm_cfg.ILM_ECC` • `mdcm_cfg.DC_ECC` • `mdcm_cfg.DLM_ECC` • `mmsc_cfg.BTB_ECC` • `mmsc_cfg.TLB_ECC`	RO	IM
TLB_ECC	[2:1]	TLB parity/ECC support configuration. **Value** / **Meaning**: 0 = No parity/ECC support; 1 = Reserved; 2 = Has ECC support; 3 = Reserved When Shared TLB Soft Error Protection is configured to “none”, the field is 0. When Shared TLB Soft Error Protection is configured to “ecc”, the field is 2.	RO	IM
ECD	[3]	Indicates whether the Andes CoDense Extension is implemented or not. **Value** / **Meaning**: 0 = Not implemented; 1 = Implemented	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
PFT	[4]	Indicates whether the Andes PowerBrake (Performance Throttling) power/performance scaling extension is implemented or not. **Value** / **Meaning**: 0 — Not implemented; 1 — Implemented	RO	1
HSP	[5]	Indicates whether the Andes StackSafe hardware stack protection extension is implemented or not. **Value** / **Meaning**: 0 — Not implemented; 1 — Implemented	RO	1
ACE	[6]	Indicates whether Andes Custom Extension is implemented or not. **Value** / **Meaning**: 0 — Not implemented; 1 — Implemented	RO	IM
VPLIC	[12]	Indicates whether the Andes Vectored PLIC Extension is implemented or not. **Value** / **Meaning**: 0 — Not implemented; 1 — Implemented The value is determined by the Andes Vectored PLIC Extension configuration option.	RO	IM
EV5PE	[13]	Indicates whether AndeStar V5 Performance Extension is implemented or not. AX46MPV always implements AndeStar V5 Performance Extension.	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
LMSLVP	[14]	Indicates if local memory subordinate port is present or not. **Value** / **Meaning** 0 — Local memory subordinate port is not present. 1 — Local memory subordinate port is implemented. The value is determined by the Local Memory Subordinate Port Support configuration option.	RO	IM
CCTLCSR	[16]	Indicates the presence of CSRs for CCTL operations. **Value** / **Meaning** 0 — Feature of CSRs for CCTL operations is not supported. 1 — Feature of CSRs for CCTL operations is supported.	RO	IM
VCCTL	[19:18]	Indicates the version number of CCTL command operation scheme supported by an implementation. **Value** / **Meaning** 0 — Instruction cache, data cache, CCTL TLB, and CCTL BTB operations are not configured. 1 — Instruction cache, data cache, CCTL TLB, or CCTL BTB operations are configured.	RO	IM
NOPMC	[22]	Indicates if Performance Monitoring Counters are implemented or not. **Value** / **Meaning** 0 — Performance monitoring counters are implemented. 1 — All performance monitoring counters CSRs are hardwired to 0.	RO	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
ESLEEP	[24]	Indicates if `wfe`, `sleepvalue`, and `txevt` CSRs are implemented or not. **Value** / **Meaning**: 0 — Not implemented; 1 — Implemented The value is determined by the Wait for Event configuration option.	RO	IM
PPI	[25]	Indicates if the PPI region is supported or not. **Value** / **Meaning**: 0 — PPI region is not supported.; 1 — PPI region is supported. The value is determined by the Size of Private Peripheral Region configuration option.	RO	IM
EDSP	[29]	Indicates if the DSP extension is supported or not. **Value** / **Meaning**: 0 — The DSP extension is not supported.; 1 — The DSP extension is supported. The value is determined by the RISC-V P-extension (draft) DSP/SIMD ISA configuration option.	RO	IM
PPMA	[30]	Indicates if programmable PMA setup with PMA region CSRs is supported or not. **Value** / **Meaning**: 0 — Programmable PMA setup is not supported.; 1 — Programmable PMA setup is supported.	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
BF16CVT	[32]	Indicates if the BFLOAT16 conversion extension is supported or not. **Value** / **Meaning** 0 — The BFLOAT16 conversion extension is not supported. 1 — The BFLOAT16 conversion extension is supported.	RO	1
ZFH	[33]	Indicates if the FP16 half-precision floating-point extension (Zfh) is supported or not. **Value** / **Meaning** 0 — The FP16 extension is not supported. 1 — The FP16 extension is supported.	RO	1
VL4	[34]	Indicates if vector INT4 load extension is supported or not. **Value** / **Meaning** 0 — Vector INT4 load extension is not supported. 1 — Vector INT4 load extension is supported.	RO	1
VECCFG	[36]	Indicates if the `mvec_cfg` CSR is present or not. **Value** / **Meaning** 0 — `mvec_cfg` does not exist. When `misa.V` = 1, the vector processor configuration is implied. 1 — `mvec_cfg` exists. It explicitly indicates the vector processor configuration.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
VSIH	[40]	Indicates if vector small INT handling extension is supported or not. **Value** / **Meaning** 0 — Vector small INT handling extension is not supported. 1 — Vector small INT handling extension is supported.	RO	1
ECDV	[42:41]	Version of Code Dense Extension. **Value** / **Meaning** 0 — The EXEC.IT encoding is 1010...10. 1 — The NEXEC.IT encoding is 1001...00. 2 — Reserved 3 — Reserved The value is determined by the RISC-V Zce Extension configuration option.	RO	IM
VDOT	[43]	Indicates if vector dot product extension is supported or not. **Value** / **Meaning** 0 — Vector dot product extension is not supported. 1 — Vector dot product extension is supported.	RO	1
VPFH	[44]	Indicates if vector packed FP16 extension is supported or not. **Value** / **Meaning** 0 — Vector packed FP16 extension is not supported. 1 — Vector packed FP16 extension is supported.	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
L3CMP_CFG	[45]	Indicates whether cluster configuration information is implemented or not. **Value** / **Meaning** 0 — L3-Cache, IOCP, and CORE_PCLUS are not implemented. 1 — L3-Cache, IOCP, and CORE_PCLUS are implemented.	RO	IM
L3C	[46]	Indicates whether the L3-Cache is present or not. **Value** / **Meaning** 0 — The L3-Cache is not present. 1 — The L3-Cache is present.	RO	IM
IOCP	[47]	Indicates whether the I/O Coherence Port is present or not. **Value** / **Meaning** 0 — IOCP is not present. 1 — IOCP is present. The value is determined by the I/O Coherence Port (IOCP) Support configuration option.	RO	IM
CORE_PCLUS	[51:48]	Indicates the number of cores in a cluster. **Value** / **Meaning** 0 — 1 core 1 — 2 cores 2 — 3 cores 3 — 4 cores 4 — 5 cores 5 — 6 cores 6 — 7 cores 7 — 8 cores	RO	IM

Continued on next page...

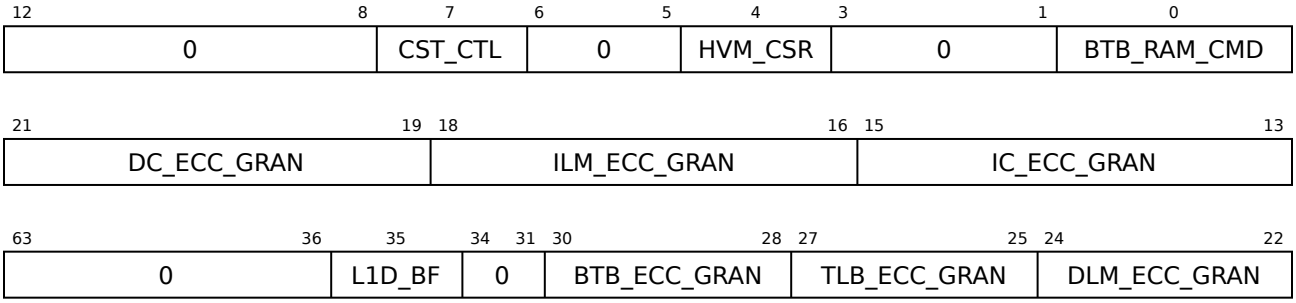
Field Name	Bits	Description	Type	Reset
RVARCH	[52]	Indicates if the `mrvarch_cfg` CSR is present or not. **Value** / **Meaning** 0 — `mrvarch_cfg` is not present. 1 — `mrvarch_cfg` is present.	RO	IM
TLB_RAM_CMD	[53]	Indicates the presence of TLB RAM command for index type of CCTL TLB operations. **Value** / **Meaning** 0 — Index type of CCTL TLB operations is not supported. 1 — Index type of CCTL TLB operations is supported. When Shared TLB Soft Error Protection is configured to “none”, the field is 0. When Shared TLB Soft Error Protection is configured to “ecc”, the field is 1.	RO	IM
ALT_FP_FMT	[57]	Indicates if alternative floating-point format is supported or not. **Value** / **Meaning** 0 — Alternative floating-point format is not supported. 1 — Alternative floating-point format is supported. When Andes BFLOAT16 Computation Support is configured to “no”, the field is 0. When Andes BFLOAT16 Computation Support is configured to “yes”, the field is 1.	RO	IM
BTB_ECC	[62:61]	Indicates whether BTB ECC is supported. **Value** / **Meaning** 0 — No parity/ECC support 1 — Reserved 2 — Has ECC support 3 — Reserved	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
MSC_EXT3	[63]	Indicates if the `mmsc_cfg3` CSR is present or not. 0: `mmsc_cfg3` is not present. 1: `mmsc_cfg3` is present.	RO	IM

16.5.4 Misc. Configuration 3 Register

Mnemonic Name: mmsc_cfg3
IM Requirement: mmsc_cfg.msc_ext3==1
Access Mode: Machine
CSR Address: 0xfc4 (non-standard read only)



Field Name	Bits	Description	Type	Reset
BTB_RAM_CMD	[0]	Indicates whether BTB RAM commands are supported or not. 0: BTB RAM commands are not supported. 1: BTB RAM commands are supported.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
HVM_CSR	[4]	Indicates existence of HVM CSRs. **Value** / **Meaning** 0: `mhvmb` and `mhvm_cfg` do not exist. 1: `mhvmb` and `mhvm_cfg` are present.	RO	IM
CST_CTL	[7]	Indicates whether customer control for ACE (`umisc_ctl.CMR_CTL`) is supported or not. **Value** / **Meaning** 0: Customer control for ACE is not supported. 1: Customer control for ACE is supported.	RO	IM
IC_ECC_GRAN	[15:13]	Indicates the memory protection granularity of I-Cache data RAMs. **Value** / **Meaning** 0: XLEN 1-2: Reserved 3: 32-bit 4: 64-bit 5-7: Reserved	RO	IM
ILM_ECC_GRAN	[18:16]	Indicates the memory protection granularity of ILM data RAMs. **Value** / **Meaning** 0: XLEN 1-2: Reserved 3: 32-bit 4: 64-bit 5-7: Reserved	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
DC_ECC_GRAN	[21:19]	Indicates the memory protection granularity of D-Cache data RAMs.	RO	IM

Value	Meaning
0	XLEN
1-2	Reserved
3	32-bit
4	64-bit
5-7	Reserved

Field Name	Bits	Description	Type	Reset
DLM_ECC_GRAN	[24:22]	Indicates the memory protection granularity of DLM data RAMs.	RO	IM

Value	Meaning
0	XLEN
1-2	Reserved
3	32-bit
4	64-bit
5-7	Reserved

Field Name	Bits	Description	Type	Reset
TLB_ECC_GRAN	[27:25]	Indicates the memory protection granularity of TLB RAMs.	RO	IM

Value	Meaning
0	XLEN
1-2	Reserved
3	32-bit
4	64-bit
5-7	Reserved

Field Name	Bits	Description	Type	Reset
BTB_ECC_GRAN	[30:28]	Indicates the memory protection granularity of BTB RAMs.	RO	IM

Value	Meaning
0	XLEN
1-2	Reserved
3	32-bit
4	64-bit
5-7	Reserved

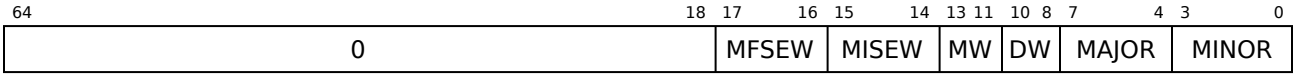
Continued on next page...

Field Name	Bits	Description	Type	Reset
L1D_BF	[35]		RO	IM

Value	Meaning
0	The L1D counting bloom filter is not implemented.
1	The L1D counting bloom filter is implemented.

16.5.5 Vector Configuration Register

Mnemonic Name: mvec_cfg
IM Requirement: misa.V==1 && mmsc_cfg.veccfg==1
Access Mode: Machine
CSR Address: 0xfc7 (non-standard read only)



This register provides information regarding vector processor configurations. The table below lists all valid field values, but some of them may not be implemented. Please see AndesCore AX46MPV Integration Guide (IG094) for details.

The MAJOR and MINOR fields determine which RVV version is implemented. For example, if the RVV 1.0 specification is implemented, MAJOR will be 1 and MINOR will be 0.

Field Name	Bits	Description	Type	Reset
MINOR	[3:0]	Indicates the minor version of vector processor related to architecture specification and implementation versions.	RO	IM
MAJOR	[7:4]	Indicates the major version of vector processor related to architecture specification and implementation versions.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
DW	[10:8]	Indicates the width of the ALU data computation path. **Value** / **Meaning** 0 / VLEN 1 / VLEN/2 2 / VLEN/4 3 / VLEN/8 4~7 / Reserved	RO	IM
MW	[13:11]	Indicates the width of the memory data path. **Value** / **Meaning** 0 / VLEN 1 / VLEN/2 2 / VLEN/4 3 / VLEN/8 4~7 / Reserved	RO	IM
MISEW	[15:14]	Indicates the maximum integer SEW supported. **Value** / **Meaning** 0 / Reserved 1 / 32 2 / 64 3 / Reserved	RO	IM
MFSEW	[17:16]	Indicates the maximum floating-point SEW supported. **Value** / **Meaning** 0 / Vector floating-point instructions are not supported. 1 / 32 2 / 64 3 / Reserved	RO	IM

16.5.6 RISC-V Architecture Configuration Register

Mnemonic Name: mrvarch_cfg
IM Requirement: mmsc_cfg.RVARCH==1
Access Mode: Machine
CSR Address: 0xfca (non-standard read only)

This register provides information about the implemented RISC-V Architecture. The table below lists all valid values, but some of them may not be implemented.

Field Name	Bits	Description	Type	Reset
Zba	[0]	Indicates whether the RISC-V Zba ISA extension is implemented or not. **Value** / **Meaning** 0 — Zba is not implemented. 1 — Zba is implemented.	RO	IM
Zbb	[1]	Indicates whether the RISC-V Zbb ISA extension is implemented or not. **Value** / **Meaning** 0 — Zbb is not implemented. 1 — Zbb is implemented.	RO	IM
Zbc	[2]	Indicates whether the RISC-V Zbc ISA extension is implemented or not. **Value** / **Meaning** 0 — Zbc is not implemented. 1 — Zbc is implemented.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
Zbs	[3]	Indicates whether the RISC-V Zbs ISA extension is implemented or not. **Value** / **Meaning** 0 – Zbs is not implemented. 1 – Zbs is implemented.	RO	IM
Sstc	[8]	Indicates if the RISC-V Sstc extension is implemented. **Value** / **Meaning** 0 – Sstc is not implemented. 1 – Sstc is implemented.	RO	1
Zicbom	[9]	Indicates if the RISC-V Zicbom extension is implemented. **Value** / **Meaning** 0 – Zicbom is not implemented. 1 – Zicbom is implemented.	RO	1
Zicbop	[10]	Indicates if the RISC-V Zicbop extension is implemented. **Value** / **Meaning** 0 – Zicbop is not implemented. 1 – Zicbop is implemented.	RO	1
Zicboz	[11]	Indicates if the RISC-V Zicboz extension is implemented. **Value** / **Meaning** 0 – Zicboz is not implemented. 1 – Zicboz is implemented.	RO	1
Zca	[26]	Indicates whether the RISC-V Zca ISA extension is implemented or not. **Value** / **Meaning** 0 – Zca is not implemented. 1 – Zca is implemented.	RO	IM

Continued on next page...

Field Name	Bits	Description	Type	Reset
Zcb	[27]	Indicates whether the RISC-V Zcb ISA extension is implemented or not. **Value** / **Meaning** 0 – Zcb is not implemented. 1 – Zcb is implemented.	RO	IM
Zcd	[28]	Indicates whether the RISC-V Zcd ISA extension is implemented or not. **Value** / **Meaning** 0 – Zcd is not implemented. 1 – Zcd is implemented.	RO	IM
Zcf	[29]	Indicates whether the RISC-V Zcf ISA extension is implemented or not. **Value** / **Meaning** 0 – Zcf is not implemented. 1 – Zcf is implemented.	RO	IM
Zcmp	[30]	Indicates whether the RISC-V Zcmp ISA extension is implemented or not. **Value** / **Meaning** 0 – Zcmp is not implemented. 1 – Zcmp is implemented.	RO	IM
Zcmt	[31]	Indicates whether the RISC-V Zcmt ISA extension is implemented or not. **Value** / **Meaning** 0 – Zcmt is not implemented. 1 – Zcmt is implemented.	RO	IM
Zfbfmin	[32]	Indicates whether the RISC-V Zfbfmin ISA extension is implemented or not. **Value** / **Meaning** 0 – Zfbfmin is not implemented. 1 – Zfbfmin is implemented.	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
Zvfbfmin	[33]	Indicates whether the RISC-V Zvfbfmin ISA extension is implemented or not. **Value** / **Meaning** 0 — Zvfbfmin is not implemented. 1 — Zvfbfmin is implemented.	RO	1
Zvfbfwma	[34]	Indicates whether the RISC-V Zvfbfwma ISA extension is implemented or not. **Value** / **Meaning** 0 — Zvfbfwma is not implemented. 1 — Zvfbfwma is implemented.	RO	1
Zvqmac	[35]	Indicates whether the RISC-V Zvqmac ISA extension is implemented or not. **Value** / **Meaning** 0 — Zvqmac is not implemented. 1 — Zvqmac is implemented.	RO	1
Zvlsseg	[36]	Indicates whether the RISC-V Zvlsseg ISA extension is implemented or not. **Value** / **Meaning** 0 — Zvlsseg is not implemented. 1 — Zvlsseg is implemented.	RO	1
MRVARCH_EXT3	[63]	Indicates if `mrvarch_cfg3` CSR is present or not. **Value** / **Meaning** 0 — The `mrvarch_cfg3` CSR is not present. 1 — The `mrvarch_cfg3` CSR is present.	RO	1

16.5.7 L3-Cache Control Base Register

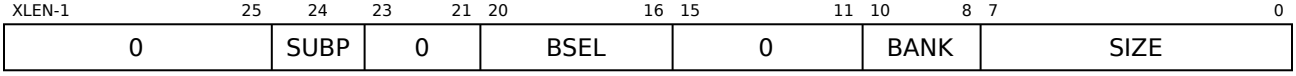
Mnemonic Name: mccache_ctl_base
IM Requirement: mmsc_cfg.L3C
Access Mode: Machine
CSR Address: 0xfcf (non-standard read only)

This register indicates the base address of the memory-mapped L3-Cache control registers.

Field Name	Bits	Description	Type	Reset
L3C_CTL_BASE	[63:0]	Indicates **L3C Register Base**	RO	IM

16.5.8 HVM Configuration Register

Mnemonic Name: mhvm_cfg
IM Requirement: mmsc_cfg3.HVM_CSR
Access Mode: Machine
CSR Address: 0xfd0 (non-standard read only)



Field Name	Bits	Description	Type	Reset
SIZE	[7:0]	Indicates **Size of HVM region**.	RO	IM

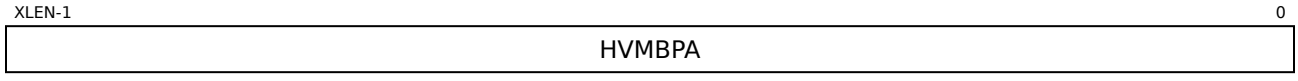
Value	Meaning
0	0 KiB
12	4 KiB
13	8 KiB
14	16 KiB
15	32 KiB
16	64 KiB
17	128 KiB
18	256 KiB
19	512 KiB
20	1 MiB
21	2 MiB
22	4 MiB
23	8 MiB
24	16 MiB
25	32 MiB
26	64 MiB
27	128 MiB
28	256 MiB
29	512 MiB
30	1 GiB
31	2 GiB
32	4 GiB
33~63	Reserved

Continued on next page. . .

Field Name	Bits	Description	Type	Reset
BANK	[10:8]	Indicates **Number of HVM Interface Banks**. 0: 1 bank 1: 2 banks 2: 4 banks 3: 8 banks 4: 16 banks 5: 32 banks 6: 64 banks 7: Reserved	RO	IM
BSEL	[20:16]	Indicates the bit of physical address for HVM banking. This field is only meaningful in dual bank configuration, that is, `mhvm_cfg.BANK` is 1.	RO	IM
SUBP	[24]	Indicates the existence of HVM subordinate port configured by **HVM Subordinate Port Support**. 0: HVM subordinate port is not configured. 1: HVM subordinate port is configured.	RO	IM

16.5.9 HVM Base Address Register

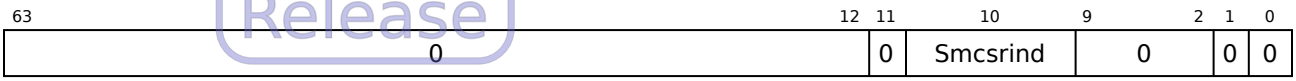
Mnemonic Name: mhvmb
IM Requirement: mmsc_cfg3.HVM_CSR
Access Mode: Machine
CSR Address: 0xfd1 (non-standard read only)



Field Name	Bits	Description	Type	Reset
HVMBPA	[XLEN-1:0]	Indicates **Base Address of HVM Region**.	RO	IM

16.5.10 RISC-V Architecture Configuration Register 3

Mnemonic Name: mrvarch_cfg3
IM Requirement: mrvarch_cfg.MRVARCH_EXT3==1
Access Mode: Machine
CSR Address: 0xfcc (non-standard, read-only)

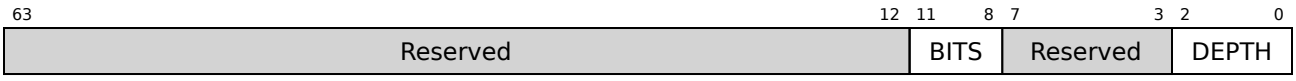


This register provides information about the implemented RISC-V Architecture. The table below lists all valid values, but some of them may not be implemented.

Field Name	Bits	Description	Type	Reset
Smcsrind	[10]	Indicates if the Smcsrind extension is implemented or not. Value 0: Smcsrind is not implemented. Value 1: Smcsrind is implemented.	RO	1

16.5.11 D-Cache Counting Bloom Filter Configuration Register

Mnemonic Name: ml1dbf_cfg
IM Requirement: mmsc_cfg3.L1D_BF==1
Access Mode: Machine
CSR Address: 0xfcd (non-standard, read-only)

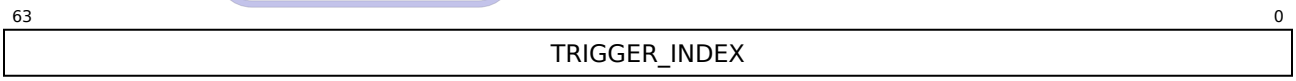


Field Name	Bits	Description	Type	Reset
DEPTH	[2:0]	Each counter bit width of the counting bloom filter. **Value** / **Meaning** 2 — 3 bits 3 — 4 bits 4 — 5 bits	RO	IM
BITS	[11:8]	Bit array bit width of the counting bloom filter. **Value** / **Meaning** 9 — 1024 bits 10 — 2048 bits	RO	IM

16.6 Trigger Registers

16.6.1 Trigger Select

Mnemonic Name: tselect
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a0 (standard read/write)

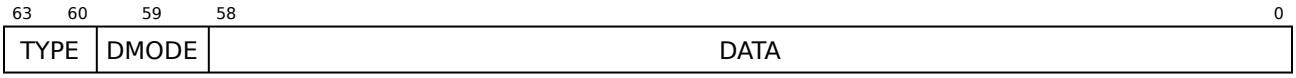


This register determines which trigger is accessible through other trigger registers. Accessible triggers must be set starting from index 0 and must be contiguous. Writing a value greater than or equal to the number of supported triggers may result in this register holding a different value. Debuggers should read back the value to verify that the written index is valid.

As triggers can be used in both Debug mode and Machine mode, a debugger must restore the value of this register after modification.

16.6.2 Trigger Data 1

Mnemonic Name: tdata1
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a1 (standard read/write)



This register provides accesses to the trigger entry in `tdata1` register as specified by the `tselect` register.

Field Name	Bits	Description	Type	Reset
DATA	[58:0]	Trigger-specific data	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
DMODE	[59]	Setting this field to indicate the trigger is used by Debug Mode. **Value** / **Meaning** 0 — Both Debug-mode and M-mode can write the currently selected trigger registers. 1 — Only Debug Mode can write the currently selected trigger registers. Writes from M-mode is ignored.	RW	0
TYPE	[63:60]	Indicates the trigger type. **Value** / **Meaning** 0 — The selected trigger is invalid. 2 — The selected trigger is an address/data match trigger. 3 — The selected trigger is an instruction count trigger. 4 — The selected trigger is an interrupt trigger. 5 — The selected trigger is an exception trigger.	RW	2

16.6.3 Trigger Data 2

Mnemonic Name: tdata2
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a2 (standard read/write)

This register provides accesses to the trigger entry in `tdata2` as specified by the `tselect` register, and it holds trigger-specific data.

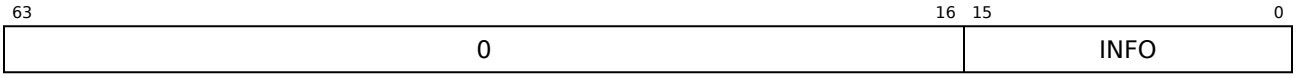
16.6.4 Trigger Data 3

Mnemonic Name: tdata3
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a3 (standard read/write)

This register provides accesses to the trigger entry in `tdata3` as specified by the `tselect` register, and it holds trigger-specific data.

16.6.5 Trigger Info

Mnemonic Name: tinfo
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a4 (standard read/write)



Field Name	Bits	Description	Type	Reset
INFO	[15:0]	One bit for each possible type in `tdata1`. Bit *N* corresponds to type *N*. If the bit is set, then that type is supported by the currently selected trigger. If the currently selected trigger does not exist, this field holds 1.	RO	IM

Bit *N*	Descriptions
0	When this bit is set, there is no trigger at this `tselect`.
1	Reserved and hardwired to 0.
2	When this bit is set, the selected trigger supports the type of address/data match trigger.
3	When this bit is set, the selected trigger supports the type of instruction count trigger.
4	When this bit is set, the selected trigger supports the type of interrupt trigger.
5	When this bit is set, the selected trigger supports the type of exception trigger.
15	When this bit is set, the selected trigger exists (so enumeration should not terminate), but is not currently available.
Others	Reserved for future use.

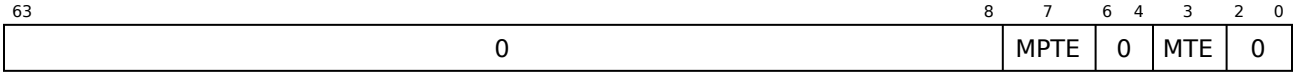
The detailed correlation between trigger types and Number of Trigger is as follows:

- INFO[2] = 1, all trigger types are supported.
- INFO[3] = 1, trigger 0 or 1 (`tselect` = 0 or 1) is supported.
- INFO[4] = 1, trigger 0 (`tselect` = 0) or trigger 4 (`tselect` = 4) is supported when Number of Triggers is 8.
- INFO[5] = 1,
  - trigger 1 (`tselect` = 1) is supported when Number of Triggers is 2,

– trigger 3 (`tselect` = 3) is supported when Number of Triggers is 4, or
– trigger 3 or 7 (`tselect` = 3 or 7) is supported when Number of Triggers is 8.

16.6.6 Trigger Control

Mnemonic Name: tcontrol
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a5 (standard read/write)

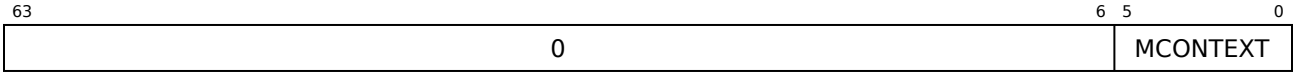


Field Name	Bits	Description	Type	Reset
MTE	[3]	M-mode trigger enable field. When a trap into M-mode is taken, `MTE` is set to 0. When an MRET instruction is executed, `MTE` is set to the value of `MPTE`.	RW	0
MPTE	[7]	M-mode previous trigger enable field. When a trap into M-mode is taken, `MPTE` is set to the value of `MTE`.	RW	0

Value	Meaning
0	Triggers do not match/fire while the hart is in M-mode.
1	Triggers do match/fire while the hart is in M-mode.

16.6.7 Machine Context

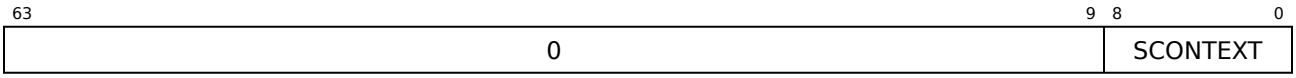
Mnemonic Name: mcontext
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a8 (standard read/write)



Field Name	Bits	Description	Type	Reset
MCONTEXT	[5:0]	Machine mode software can write a context number to this register, which can be used to set triggers that only fire in that specific context.	RW	0

16.6.8 Supervisor Context

Mnemonic Name: scontext
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug, Machine, Supervisor
CSR Address: 0x7aa (standard read/write)

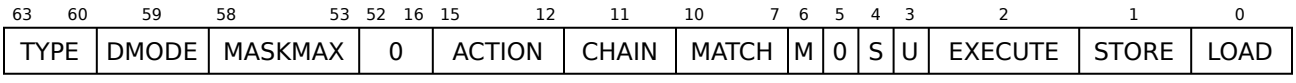


Field Name	Bits	Description	Type	Reset
SCONTEXT	[8:0]	Machine mode software can write a context number to this register, which can be used to set triggers that only fire in that specific context.	RW	0

16.6.9 Match Control

Mnemonic Name: mcontrol
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a1 (standard read/write)

This register is accessed as `tdata1` when `TYPE` is 0 or 2.



Field Name	Bits	Description	Type	Reset
LOAD	[0]	Setting this field to enable this trigger to compare virtual address of a load.	RW*	0
STORE	[1]	Setting this field to enable this trigger to compare virtual address of a store.	RW*	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
EXECUTE	[2]	Setting this field to enable this trigger to compare virtual address of an instruction.	RW	0
U	[3]	Setting this field to enable this trigger in U-mode.	RW	0
S	[4]	Setting this field to enable this trigger in S-mode.	RW	0
M	[6]	Setting this field to enable this trigger in M-mode.	RW	0
MATCH	[10:7]	Setting this field to select the matching scheme.	RW	0

Value	Meaning
0	Matches when the value equals `tdata2`.
1	Matches when the top *M* bits of the value match the top *M* bits of `tdata2`. *M* is 63 minus the index of the least-significant bit containing 0 in `tdata2`.
2	Matches when the value is greater than (unsigned) or equal to `tdata2`.
3	Matches when the value is less than (unsigned) `tdata2`.

CHAIN	[11]	Setting this field to enable trigger chain.	RW	0

Value	Meaning
0	When this trigger matches, the configured action is taken.
1	While this trigger does not match, it prevents the trigger with the next index from matching.

If **Number of Triggers** is 2, this field is hardwired to 0 on trigger 1 (`tselect` = 1).
If **Number of Triggers** is 4, this field is hardwired to 0 on trigger 3 (`tselect` = 3).
If **Number of Triggers** is 8, this field is hardwired to 0 on trigger 3 and trigger 7 (`tselect` = 3 or 7).

Continued on next page...

Field Name	Bits	Description	Type	Reset
ACTION	[15:12]	Setting this field to select what happens when this trigger matches. **Value** / **Meaning** 0: Raise a breakpoint exception. 1: Enter Debug Mode. The value is legal only when DMODE is 1. 2: Trace-on. The value is legal only when the trace interface is configured. 3: Trace-off. The value is legal only when the trace interface is configured. 4: Trace-notify. The value is legal only when the trace interface is configured.	RW	0
MASKMAX	[58:53]	Indicates the largest naturally aligned range supported by the hardware is 2^12 bytes.	RO	12
DMODE	[59]	Setting this field to indicate the trigger is used by Debug Mode. **Value** / **Meaning** 0: Both Debug-mode and M-mode can write the currently selected trigger registers. 1: Only Debug Mode can write the currently selected trigger registers. Writes from M-mode is ignored.	RW	0
TYPE	[63:60]	Indicates the trigger type. **Value** / **Meaning** 0: The selected trigger is invalid. 2: The selected trigger is an address/data match trigger.	RW	2

Note

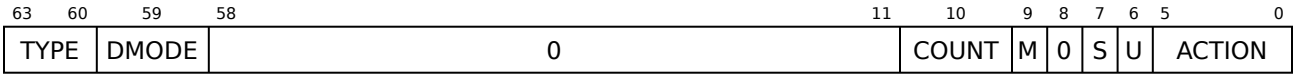
The LOAD/STORE fields take no effect and are cleared if the EXECUTE field is set at the same time.

16.6.10 Instruction Count

Mnemonic Name: icount
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a1 (standard read/write)

This register is accessed as tdata1 when TYPE is 3.

This register exists just for single-stepping support, so COUNT is hardwired to 1. After this trigger fires, the mode bits (M, S, and U) will be cleared instead of causing the COUNT bit to be decremented.



Field Name	Bits	Description	Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches. **Value** / **Meaning** 0 — Raise a breakpoint exception. 1 — Enter Debug Mode. (Only supported when DMODE is 1.)	RW	0
U	[6]	Setting this field to enable this trigger in U-mode.	RW	0
S	[7]	Setting this field to enable this trigger in S-mode.	RW	0
M	[9]	Setting this field to enable this trigger in M-mode.	RW	0
COUNT	[10]	This field is hardwired to 1 for single-stepping support.	RO	1

Continued on next page...

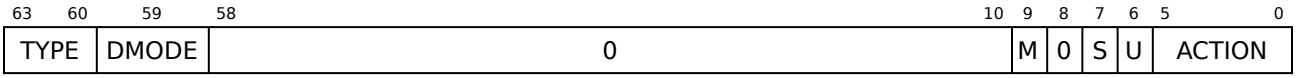
Field Name	Bits	Description	Type	Reset
DMODE	[59]	Setting this field to indicate the trigger is used by Debug Mode. **Value** / **Meaning** 0 — Both Debug-mode and M-mode can write the currently selected trigger registers. 1 — Only Debug Mode can write the currently selected trigger registers. Writes from M-mode is ignored.	RW	0
TYPE	[63:60]	The selected trigger is an instruction count trigger.	RW	2

16.6.11 Interrupt Trigger

Mnemonic Name: itrigger
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a1 (standard read/write)

This register is accessible as `tdata1` when `TYPE` is 4.

This trigger can fire on any interrupt that is configurable in `mie`. To specify which interrupts activate this trigger, set the corresponding bit in `tdata2` to match those used in `mie` to enable the interrupts.



Field Name	Bits	Description	Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches. **Value** / **Meaning** 0 — Raise a breakpoint exception. 1 — Enter Debug Mode (Only supported when `DMODE` is 1).	RW	0

Continued on next page...

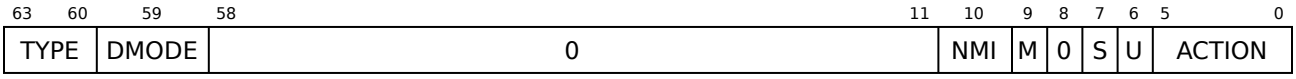
Field Name	Bits	Description	Type	Reset
U	[6]	Setting this field to enable this trigger in U-mode.	RW	0
S	[7]	Setting this field to enable this trigger in S-mode.	RW	0
M	[9]	Setting this field to enable this trigger in M-mode.	RW	0
DMODE	[59]	Setting this field to indicate the trigger is used by Debug Mode. **Value** / **Meaning** 0 — Both Debug-mode and M-mode can write the currently selected trigger registers. 1 — Only Debug Mode can write the currently selected trigger registers. Writes from M-mode is ignored.	RW	0
TYPE	[63:60]	The selected trigger is an interrupt trigger.	RW	2

16.6.12 Exception Trigger

Mnemonic Name: etrigger
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a1 (standard read/write)

This register is accessed as `tdata1` when `TYPE` is 5.

This trigger can fire on up to XLEN of the Exception Codes defined in `mcause` (with Interrupt = 0). These causes are configured by setting the corresponding bit in `tdata2`. For example, to trap on an illegal instruction, the debugger sets bit 2 in `tdata2`.



Field Name	Bits	Description	Type	Reset
ACTION	[5:0]	Setting this field to select what happens when this trigger matches. **Value** / **Meaning** 0 — Raise a breakpoint exception. 1 — Enter Debug Mode (Only supported when `DMODE` is 1).	RW	0
U	[6]	Setting this field to enable this trigger in U-mode.	RW	0
S	[7]	Setting this field to enable this trigger in S-mode.	RW	0
M	[9]	Setting this field to enable this trigger in M-mode.	RW	0
NMI	[10]	Setting this field to enable this trigger in non-maskable interrupts, regardless of the values in the `S`, `U`, and `M` fields.	RW	0

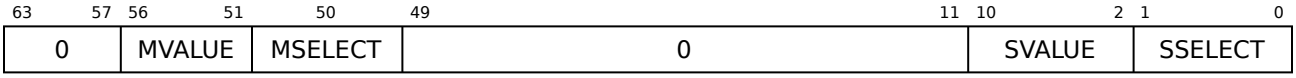
Continued on next page...

Field Name	Bits	Description	Type	Reset
DMODE	[59]	Setting this field to indicate the trigger is used by Debug Mode. **Value** / **Meaning** 0 — Both Debug-mode and M-mode can write the currently selected trigger registers. 1 — Only Debug Mode can write the currently selected trigger registers. Writes from M-mode is ignored.	RW	0
TYPE	[63:60]	The selected trigger is an exception trigger.	RW	2

16.6.13 Trigger Extra

Mnemonic Name: textra
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug and Machine
CSR Address: 0x7a3 (standard read/write)

This register is accessed as `tdata3` when `TYPE` is 2, 3, 4, or 5 of the currently selected trigger registers, as determined by the `tselect` register. It also indicates the context matching scheme of the currently selected triggers.



Field Name	Bits	Description	Type	Reset
SSELECT	[1:0]		RW	0

Value	Meaning
0	Ignore `MVALUE`.
1	This trigger will only match if the lower bits of `scontext` equal `SVALUE`.
2	This trigger will only match if `satp.ASID` equals `SVALUE`.

SVALUE	[10:2]	Data used together with `SSELECT`.	RW	0
MSELECT	[50]		RW	0

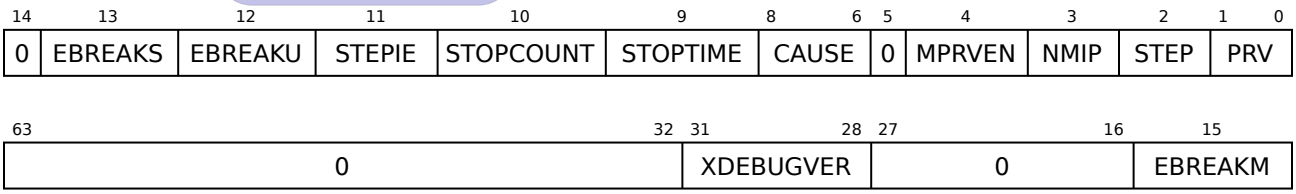
Value	Meaning
0	Ignore `MVALUE`.
1	This trigger will only match if the lower bits of `mcontext` equal `MVALUE`.

MVALUE	[56:51]	Data used together with `MSELECT`.	RW	0

16.7 Debug Registers

16.7.1 Debug Control and Status Register

Mnemonic Name: dcsr
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7b0 (debug-mode-only)



Field Name	Bits	Description	Type	Reset
PRV	[1:0]	The privilege level at which the hart operates upon entering Debug Mode. The external debugger can modify this value to change the privilege level of the hart upon exiting Debug Mode.	WARL	3
STEP	[2]	This bit determines whether instruction execution outside of Debug Mode operates in the Single Step Mode. When set, the hart returns to Debug Mode after executing a single instruction. If the instruction is interrupted by an exception, the hart will immediately enter Debug Mode before executing the trap handler, with appropriate exception registers set.	RW	0

Value	Meaning
0	User/Application
1	Supervisor
2	Reserved
3	Machine

Value	Meaning
0	Single Step Mode is off.
1	Single Step Mode is on.

Continued on next page...

Field Name	Bits	Description	Type	Reset
NMIP	[3]	When this bit is set, a Non-Maskable Interrupt (NMI) is pending for the hart. Since an NMI can indicate a hardware error condition, reliable debugging may become challenging once this bit is set.	RO	0
MPRVEN	[4]	This bit controls whether `mstatus.MPRV` takes effect in Debug Mode. **Value** / **Meaning** 0 — `mstatus.MPRV` is ignored in Debug Mode. 1 — `mstatus.MPRV` takes effect in Debug Mode.	RW	0
CAUSE	[8:6]	Indicates the reason for entering Debug Mode. When there are multiple reasons for entering Debug Mode, the priority to determine the `CAUSE` value is: trigger module > `EBREAK` > halt-on-reset > halt request > single step. Halt requests are issued by the external debugger. **Value** / **Meaning** 0 — Reserved 1 — EBREAK 2 — Trigger module 3 — Halt request 4 — Single step 5 — Halt-on-reset 6–7 — Reserved	RO	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
STOPTIME	[9]	This bit controls whether timers are halted in Debug Mode. The processor only drives its `stoptime` output pin to 1 when it is in Debug Mode and this bit is set. Integration efforts are required to ensure that timers in the platform observe this pin and really stop them.	RW	1

Value	Meaning
0	Do not stop timers in Debug Mode.
1	Stop timers in Debug Mode.

Field Name	Bits	Description	Type	Reset
STOPCOUNT	[10]	This bit controls whether performance counters are stopped in Debug Mode.	RW	1

Value	Meaning
0	Do not stop counters in Debug Mode.
1	Stop counters in Debug Mode.

Field Name	Bits	Description	Type	Reset
STEPIE	[11]	This bit controls whether interrupts are enabled during single stepping.	RW	0

Value	Meaning
0	Disable interrupts during single stepping.
1	Allow interrupts during single stepping.

Field Name	Bits	Description	Type	Reset
EBREAKU	[12]	This bit controls the behavior of `EBREAK` instructions in User/Application Mode.	RW	0

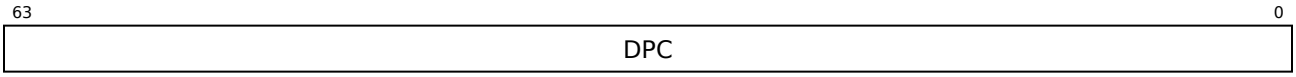
Value	Meaning
0	Generate a regular breakpoint exception
1	Enter Debug Mode

Continued on next page...

Field Name	Bits	Description	Type	Reset
EBREAKS	[13]	This bit controls the behavior of EBREAK instructions in Supervisor Mode. 0: Generate a regular breakpoint exception 1: Enter Debug Mode	RW	0
EBREAKM	[15]	This bit controls the behavior of EBREAK instructions in Machine Mode. 0: Generate a regular breakpoint exception 1: Enter Debug Mode	RW	0
XDEBUGVER	[31:28]	Version of the external debugger. 0 indicates that no external debugger exists, and 4 indicates that the external debugger conforms to the *RISC-V External Debug Support (TD003) V0.13*.	RO	4

16.7.2 Debug Program Counter

Mnemonic Name: dpc
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7b1 (debug-mode-only)



When entering Debug Mode, the dpc CSR is updated with the virtual address of the next instruction to be executed. The behavior is described in more detail in Table 126. When exiting Debug Mode, the PC of the hart is updated to the value stored in this register. The external debugger can modify this register to determine where the hart resumes execution.

Field Name	Bits	Description	Type	Reset
DPC	[63:0]	Debug Program Counter. Bit 0 is hardwired to 0.	RW	0

Table 126: Virtual Address in DPC upon Debug Mode Entry

Cause	Virtual Address in DPC
EBREAK	Address of the `EBREAK` instruction
Single step	Address of the instruction to be executed next if no debugging is active
Trigger module	Address of the instruction which will cause the trigger module to fire.
Halt request	Address of the next instruction to be executed upon entering Debug Mode

16.7.3 Debug Scratch Register 0

Mnemonic Name: dscratch0
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7b2 (debug-mode-only)

A scratch register that is reserved for use by Debug Module.

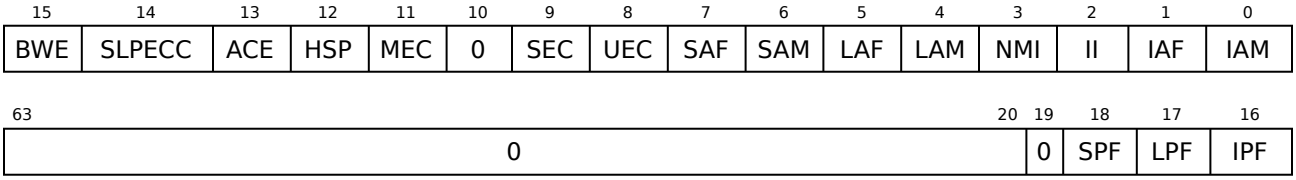
16.7.4 Debug Scratch Register 1

Mnemonic Name: dscratch1
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7b3 (debug-mode-only)

A scratch register that is reserved for use by Debug Module.

16.7.5 Exception Redirection Register

Mnemonic Name: dexc2dbg
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7e0 (non-standard read/write)



This register redirects selected exceptions to cause the hart to enter Debug Mode instead of performing the standard trap handling.

When an exception is redirected to Debug Mode, the `dpc` CSR is updated with the virtual address of the instruction causing the exception. The `dcsr.CAUSE` field is updated to 1 (= EBREAK), and the actual cause of the exception is saved to the `ddcause` CSR. The required updates to `mepc`, `mcause`, `mtval`, `mstatus`, and `mxstatus` CSRs are not affected by this redirection; these CSRs continue to provide information associated with their corresponding exceptions.

Field Name	Bits	Description	Type	Reset
IAM	[0]	Indicates whether Instruction Access Misaligned exceptions are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
IAF	[1]	Indicates whether Instruction Access Fault exceptions are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
II	[2]	Indicates whether Illegal Instruction exceptions are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
NMI	[3]	Indicates whether Non-Maskable Interrupt exceptions are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
LAM	[4]	Indicates whether Load Access Misaligned exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0
LAF	[5]	Indicates whether Load Access Fault exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0
SAM	[6]	Indicates whether Store Access Misaligned exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0
SAF	[7]	Indicates whether Store Access Fault exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0
UEC	[8]	Indicates whether U-mode Environment Call exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0
SEC	[9]	Indicates whether S-mode Environment Call exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 – Not redirected; 1 – Redirected	RW	0

Continued on next page...

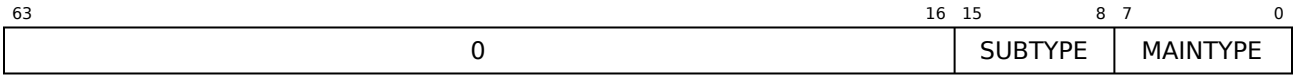
Field Name	Bits	Description	Type	Reset
MEC	[11]	Indicates whether M-mode Environment Call exceptions are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
HSP	[12]	Indicates whether Stack Protection exceptions are redirected to Debug Mode. This bit is present only when `mmsc_cfg.HSP` is set. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
ACE	[13]	Indicates whether ACE-related exceptions are redirected to Debug Mode. This bit is present only when `mmsc_cfg.ACE` is set. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
SLPECC	[14]	Indicates whether local memory subordinate port ECC Error local interrupts are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0
BWE	[15]	Indicates whether Bus-write Transaction Error local interrupts are redirected to Debug Mode. **Value** / **Meaning** 0 – Not redirected 1 – Redirected	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
IPF	[16]	Indicates whether Instruction Page Fault exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 = Not redirected; 1 = Redirected	RW	0
LPF	[17]	Indicates whether Load Page Fault exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 = Not redirected; 1 = Redirected	RW	0
SPF	[18]	Indicates whether Store Page Fault exceptions are redirected to Debug Mode. **Value** / **Meaning**: 0 = Not redirected; 1 = Redirected	RW	0

16.7.6 Debug Detailed Cause

Mnemonic Name: ddcause
IM Requirement: DEBUG_SUPPORT
Access Mode: Debug
CSR Address: 0x7e1 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
MAINTYPE	[7:0]	Cause for redirection to Debug Mode.	RO	0

Value	Meaning
0	Software Breakpoint (EBREAK)
1	Instruction Access Misaligned (IAM)
2	Instruction Access Fault (IAF)
3	Illegal Instruction (II)
4	Non-Maskable Interrupt (NMI)
5	Load Access Misaligned (LAM)
6	Load Access Fault (LAF)
7	Store Access Misaligned (SAM)
8	Store Access Fault (SAF)
9	U-mode Environment Call (UEC)
10	S-mode Environment Call (SEC)
11	Instruction Page Fault
12	M-mode Environment Call (MEC)
13	Load Page Fault
14	Reserved
15	Store/AMO Page Fault
16	Imprecise ECC Error
17	Bus Write Transaction Error
18	Performance Counter Overflow
19–31	Reserved
32	Stack Overflow Exception
33	Stack Underflow Exception
34	ACE Disabled Exception
35–39	Reserved
40–47	ACE Exception
≥48	Reserved

Continued on next page...

Field Name	Bits	Description	Type	Reset
SUBTYPE	[15:8]	Subtypes for `MAINTYPE`. The table below lists the subtypes for DCSR.CAUSE = 1 and DDCAUSE.MAINTYPE = 3.	RO	0

Value	Meaning
0	Illegal instructions
1	Privileged instructions
2	Non-existent CSR
3	Privilege CSR access
4	Read-only CSR update

16.8 Supervisor Trap Related CSRs

16.8.1 Supervisor Status

Mnemonic Name: sstatus
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x100 (standard read/write)

63	62 34	33 32	31 20	19	18	17	16 15	14 13	12 11	10 9	8	7 6	5	4	3 2	1	0
SD	0	UXL	0	MXR	SUM	0	XS	FS	0	VS	SPP	0	SPIE	UPIE	0	SIE	UIE

Field Name	Bits	Description	Type	Reset
UIE	[0]	U-mode interrupt enable bit. **Value** / **Meaning**: 0 Disabled; 1 Enabled	RW	0
SIE	[1]	S-mode interrupt enable bit. **Value** / **Meaning**: 0 Disabled; 1 Enabled	RW	0
UPIE	[4]	`UPIE` holds the value of the `UIE` bit prior to a trap.	RW	0
SPIE	[5]	`SPIE` holds the value of the `SIE` bit prior to a trap.	RW	0
SPP	[8]	`SPP` holds the privilege mode prior to a trap. Encoding is 1 for S-mode and 0 for U-mode.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
VS	[10:9]	The VS bits are shared between both mstatus and sstatus. Normally, S-Mode privileged software uses the VS bits to manage deferred context switches of vector states. M-Mode software should be more conservative in managing context switches using VS bits. Please see the descriptions of the VS field in Section 16.3.1 for more information.	RW	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Field Name	Bits	Description	Type	Reset
FS	[14:13]	The FS bits are shared between both mstatus and sstatus. Normally, S-Mode privileged software uses the FS bits to manage deferred context switches of FPU states. M-Mode software should be more conservative in managing context switches using FS bits. Please see the descriptions of the FS field in Section 16.3.1 for more information.	WLRL	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Continued on next page...

Field Name	Bits	Description	Type	Reset
XS	[16:15]	The XS bits are shared between both mstatus and sstatus. Normally, S-Mode privileged software uses the XS bits to manage deferred context switches of ACE states. M-Mode software should be more conservative in managing context switches using XS bits. Please see the descriptions of the XS field in Section 16.3.1 for more information.	RO	0

Value	Meaning
0	Off
1	Initial
2	Clean
3	Dirty

Field Name	Bits	Description	Type	Reset
SUM	[18]	SUM controls whether an S-mode load/store instruction to a user accessible page is allowed when page translation is enabled. It is in effect in two scenarios: (a) M-mode with MPRV = 1 and MPP = S, and (b) in S-mode. It has no effect when page-based virtual memory is not in effect. A page is user accessible when the U bit of the corresponding PTE entry is 1.	RW	0

Value	Meaning
0	Not Allowed
1	Allowed

Field Name	Bits	Description	Type	Reset
MXR	[19]	MXR controls whether execute-only pages are readable. It has no effect when page-based virtual memory is not in effect.	RW	0

Value	Meaning
0	Execute-only pages are not readable.
1	Execute-only pages are readable.

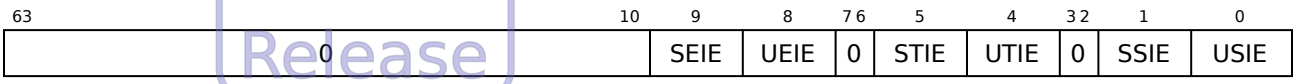
Continued on next page...

Field Name	Bits	Description	Type	Reset
UXL	[33:32]	UXL controls the value of XLEN for U-mode. When U-mode is not available, this field is hardwired to 0. Value 1: 32 Value 2: 64	RO	2/0
SD	[63]	SD summarizes whether either the FS, XS, or VS is dirty.	RO	0

When N extension is not supported, the corresponding bits in sstatus are hardwired to zero.

16.8.2 Supervisor Interrupt Enable

Mnemonic Name: sie
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x104 (standard read/write)



Field Name	Bits	Description	Type	Reset
USIE	[0]	U-mode software interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
SSIE	[1]	S-mode software interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
UTIE	[4]	U-mode timer interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
STIE	[5]	S-mode timer interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

Field Name	Bits	Description	Type	Reset
UEIE	[8]	U-mode external interrupt enable bit.	RW	0

Value	Meaning
0	Disabled
1	Enabled

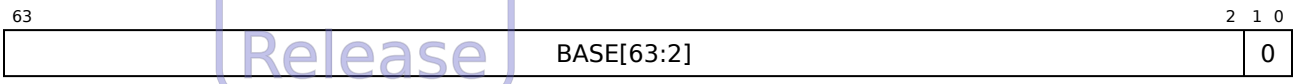
Continued on next page...

Field Name	Bits	Description	Type	Reset
SEIE	[9]	S-mode external interrupt enable bit. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0

When N extension is not supported, the corresponding bits in `sie` are hardwired to zero.

16.8.3 Supervisor Trap Vector Base Address

Mnemonic Name: stvec
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x105 (standard read/write)



This register determines the base address of the trap vector for S-mode trap handling. The least significant 2 bits are hardwired to zeros. When the width of the configured address is less than 64, the upper bits are hardwired to zeros.

When mmisc_ctl.VEC_PLIC is 0 (PLIC is not in the vector mode), this register indicates the entry points for the trap handler and it may point to any 4-byte aligned location in the memory space.

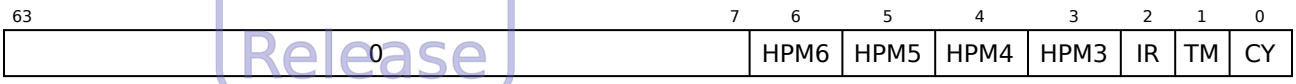
On the other hand, when mmisc_ctl.VEC_PLIC is 1 (PLIC is in the vector mode), this register will be the base address of a vector table with 4-byte entries storing addresses pointing to interrupt service routines. When XLEN = 64, the upper 32-bit address of the interrupt service routines is equal to mtvec [63:32].

- This register should be aligned to a $2^{\log_2 N+2}$-byte boundary for PLIC with N interrupt sources. For example, if N is 1023, the minimum alignment requirement is 4096 bytes (4 KiB).
- stvec[0] is for exceptions and non-external local interrupts.
- stvec[i] is for external PLIC interrupt source i triggered through the sip.SEIP pending condition when mideleg.SEI = 1.
- stvec[1024+i] is for external PLIC interrupt source i triggered through the sip.UEIP pending condition.

Field Name	Bits	Description	Type	Reset
BASE[63:2]	[63:2]	Base address for interrupt and exception handlers. See descriptions above for alignment requirements when PLIC is in the vector mode.	RW	0

16.8.4 Supervisor Counter Enable Register

Mnemonic Name: scounteren
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x106 (standard read/write)



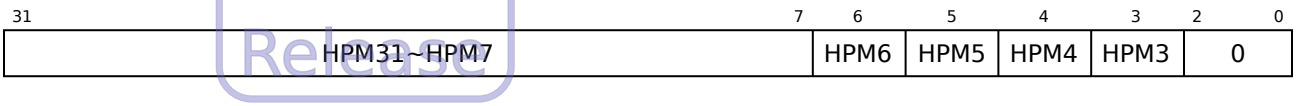
This register controls the availability of the hardware performance monitoring counters to U-mode.

If S-mode is not permitted to access a counter register, or when the corresponding bit in this register is zero, attempts to read the corresponding register from U-mode will cause an illegal instruction exception. If S-mode is permitted to access a counter register and the corresponding bit is set, access to the counter register from U-mode is permitted.

In summary, a counter can be accessed from U-mode only when both `mcounteren.counter` and `scounteren.counter` are set to 1.

16.8.5 Supervisor Count Overflow

Mnemonic Name: scountovf
IM Requirement: mrvarch_cfg.sscofpmf==1 && misa[18]==1
Access Mode: Supervisor
CSR Address: 0xda0 (standard read only)



This register enables supervisor-level overflow interrupt handler software to quickly determine which counter(s) have overflowed, without requiring an execution environment call or series of calls ultimately up to M-mode. Read access to bit *X* is subject to the same `mcounteren` CSRs that mediate access to the `hpmcounter` CSRs by S-mode . In M and S modes, `scountovf` bit *X* is readable when `mcounteren` bit *X* is set; otherwise, it reads as zero.

16.8.6 Supervisor Scratch Register

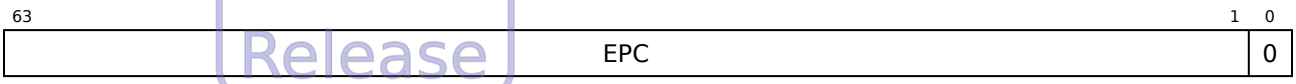
Mnemonic Name: sscratch
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x140 (standard read/write)

A scratch register for temporary data storage, which is typically used by the S-mode trap handler.

Field Name	Bits	Description	Type	Reset
SSCRATCH	[63:0]	Scratch register storage	RW	0

16.8.7 Supervisor Exception Program Counter

Mnemonic Name: sepc
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x141 (standard read/write)



This register is written with the virtual address of the instruction that raises a trap and the trap is taken to S-mode. The interpretation of this address depends on the configured **Page-Based Virtual Memory** mode:

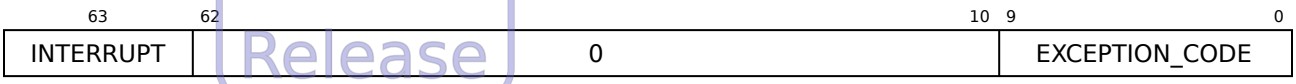
- "sv39": bits 63–40 are hardwired to bit 39.
- "sv48": bits 63–49 are hardwired to bit 48.
- "sv57": bits 63–57 are hardwired to bit 56.
- "bare": bits 63–BIU_ADDR_WIDTH are hardwired to 0.

In all cases, AX46MPV ignores the corresponding hardwired bits of any written values.

Field Name	Bits	Description	Type	Reset
EPC	[63:1]	Exception program counter	RW	0

16.8.8 Supervisor Cause Register

Mnemonic Name: scause
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x142 (standard read/write)



This register indicates the cause of traps when they are taken to S-mode.

Field Name	Bits	Description	Type	Reset
EXCEPTION_CODE	[9:0]	Exception Code	RW	0
INTERRUPT	[63]	Interrupt	RW	0

Each local interrupt can be configured with a local interrupt number. For S-mode local interrupts, cause numbers will be (local interrupt number + 256). Cause numbers below show the default cause numbers of local interrupts.

Table 127: AX46MPV scause Value After Trap

Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	8	User external interrupt
1	9	Supervisor external interrupt
1	256+16	Local memory subordinate port ECC error interrupt (S-mode)
1	256+17	Bus write transaction error interrupt (S-mode)
1	256+18	Performance monitor overflow interrupt(S-mode)
1	256+24	ACE error interrupt (S-mode)
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned

Continued on next page...

Table 127: (continued)

Interrupt	Exception Code	Description
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	11:10	Reserved
0	12	Instruction page fault
0	13	Load page fault
0	14	Reserved
0	15	Store/AMO page fault
0	32	Stack overflow exception
0	33	Stack underflow exception
0	40-47	Andes Custom Extension exception (see *Andes Custom Extension Specification* for more details)

16.8.9 Supervisor Trap Value

Mnemonic Name: stval
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x143 (standard read/write)

This register is updated when a trap is taken to S-mode. The updated value is dependent on the cause of traps:

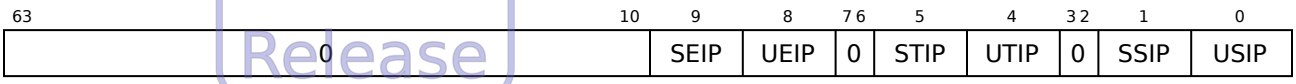
- The updated value is the effective faulting addresses for Hardware Breakpoint exceptions, Address Misaligned exceptions, Access Fault exceptions, or Page Fault exceptions.
- For illegal instruction exceptions, the updated value is the faulting instruction. If the length of the instruction is less than XLEN bits, the upper bits of `stval` are cleared to zero.
- For other exceptions, `stval` is set to zero.
- For instruction-fetch access faults, this register will be updated with the address pointing to the portion of the instruction that caused the fault, while the `sepc` register will be updated with the address pointing to the beginning of the instruction.

When the configured address width is less than 64, the upper bits are hardwired to zeros.

Field Name	Bits	Description	Type	Reset
STVAL	[63:0]	Exception-specific information for software trap handling	RW	0

16.8.10 Supervisor Interrupt Pending

Mnemonic Name: sip
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x144 (standard read/write)



Field Name	Bits	Description	Type	Reset
USIP	[0]	U-mode software interrupt pending bit.	RW	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
SSIP	[1]	S-mode software interrupt pending bit.	RW	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
UTIP	[4]	U-mode timer interrupt pending bit.	RO	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
STIP	[5]	S-mode timer interrupt pending bit.	RO	0

Value	Meaning
0	Not pending
1	Pending

Field Name	Bits	Description	Type	Reset
UEIP	[8]	U-mode external interrupt pending bit.	RW	0

Value	Meaning
0	Not pending
1	Pending

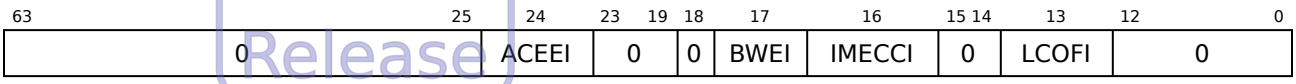
Continued on next page...

Field Name	Bits	Description	Type	Reset
SEIP	[9]	S-mode external interrupt pending bit. 0: Not pending 1: Pending	RO	0

When N extension is not supported, the corresponding bits in `sip` are hardwired to zero.

16.8.11 Supervisor Local Interrupt Enable

Mnemonic Name: slie
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x9C4 (non-standard read/write)



If a supervisor local interrupt is enabled, the supervisor local interrupt can be taken in the mode depending on the `mslideleg` CSR. When `mslideleg` for a local interrupt is set, the supervisor local interrupt will be served in S-mode. Otherwise, it will be served in M-mode.

The privileged mode of `LCOFI`, `ACEEI`, `IMECCI`, and `BWEI` are determined by the current privileged mode. For M/S/U configuration, if the current privileged mode is User or Supervisor, the local interrupt generated is a supervisor local interrupt. For M/U configuration, if the current privileged mode is User, the local interrupt generated is a machine local interrupt.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields below show the default bit location.

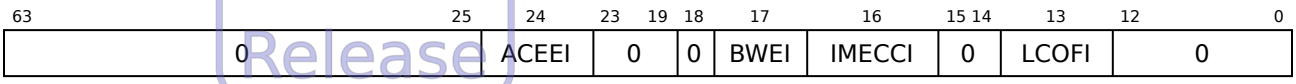
Field Name	Bits	Description	Type	Reset
LCOFI	[13]	Enable S-mode performance monitor overflow local interrupt. **Value** / **Meaning** 0 — Local interrupt is not enabled. 1 — Local interrupt is enabled.	RW	0
IMECCI	[16]	Enable S-mode imprecise ECC error local interrupt. The processor may receive imprecise ECC errors on LM subordinate port accesses or cache writebacks. **Value** / **Meaning** 0 — Local interrupt is not enabled. 1 — Local interrupt is enabled.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
BWEI	[17]	Enable S-mode bus read/write transaction error local interrupt. The processor may receive bus errors on load/store instructions or cache writebacks. **Value** / **Meaning** 0 — Local interrupt is not enabled. 1 — Local interrupt is enabled.	RW	0
ACEEI	[24]	Enable S-mode ACE error local interrupt. **Value** / **Meaning** 0 — Local interrupt is not enabled. 1 — Local interrupt is enabled.	RW	0

16.8.12 Supervisor Local Interrupt Pending

Mnemonic Name: slip
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x9C5 (non-standard read/write)



This register indicates whether a supervisor local interrupt is pending or not. If an enabled supervisor local interrupt is pending, the supervisor local interrupt will be taken in the mode depending on the `mslideleg` CSR. When `mslideleg` for a local interrupt is set, the supervisor local interrupt will be served in S-mode. Otherwise, it will be served in M-mode.

The privileged mode of `LCOFI`, `ACEEI`, `IMECCI`, and `BWEI` are determined by the current privileged mode. For M/S/U configuration, if the current privileged mode is User or Supervisor, the local interrupt generated is a supervisor local interrupt. For M/U configuration, if the current privileged mode is User, the local interrupt generated is a machine local interrupt.

Each local interrupt can be configured with a local interrupt number. Bit location of interrupts are the same as their interrupt numbers. Register fields below show the default bit location.

Field Name	Bits	Description	Type	Reset
LCOFI	[13]	Enable S-mode performance monitor overflow local interrupt. **Value** / **Meaning** 0 — Local interrupt is not enabled. 1 — Local interrupt is enabled.	RW	0
IMECCI	[16]	Pending status of S-mode imprecise ECC error local interrupt. The processor may receive imprecise ECC errors on LM subordinate port accesses or cache writebacks. **Value** / **Meaning** 0 — Local interrupt is not pending. 1 — Local interrupt is pending.	RW	0

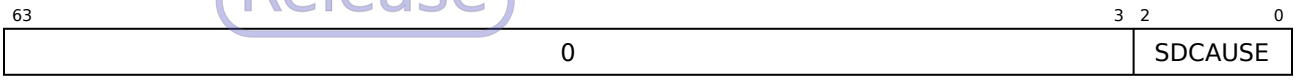
Continued on next page...

Field Name	Bits	Description	Type	Reset
BWEI	[17]	Pending status of S-mode bus read/write transaction error local interrupt. The processor may receive bus errors on load/store instructions or cache writebacks. **Value** / **Meaning** 0 — Local interrupt is not pending. 1 — Local interrupt is pending.	RW	0
ACEEI	[24]	Pending status of S-mode ACE error local interrupt. **Value** / **Meaning** 0 — Local interrupt is not pending. 1 — Local interrupt is pending.	RO	0

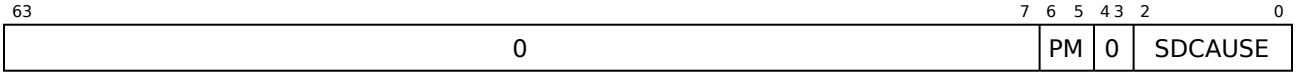
16.8.13 Supervisor Detailed Trap Cause

Mnemonic Name: sdcause
IM Requirement: Required if supervisor mode is implemented
Access Mode: Supervisor
CSR Address: 0x9c9 (non-standard read/write)

For precise exceptions:



For imprecise exceptions (local interrupts):



When multiple events cause traps to be taken with the same scause value, this register helps to further disambiguate them. Certain events can cause either precise exceptions or imprecise exceptions (local interrupts), depending on when they are detected. Consequently, they can appear in multiple tables below.

Imprecise exceptions are triggered as local interrupts. Thus, the tables below for `scause` = Local Interrupt *n* summarizes imprecise exceptions delivered as local interrupt *n*. The `scause` = Local Interrupt *n* notation standing for (`INTERRUPT`, `EXCEPTION_CODE`) fields of `scause` is (1, *n*).

Field Name	Bits	Description	Type	Reset
SDCAUSE	[2:0]	This register further disambiguates causes of traps recorded in the `scause` register. See the list below for details.	RW	0
PM	[6:5]	When `scause` is imprecise exception (in the form of an interrupt), this field records the privileged mode of the instruction that caused the imprecise exception. The `PM` field encoding is defined as follows:	RW	0

Value	Meaning
0	User mode
1	Supervisor mode
2	Reserved
3	Machine mode

The value of SDCAUSE for precise exception:

- When scause = 1 (Instruction access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP instruction access violation
3	Bus error
4	PMA empty hole access

- When scause = 2 (Illegal instruction)

Value	Meaning
0	Please parse the stval CSR
1	FP disabled exception
2	ACE disabled exception

- When scause = 5 (Load access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP load access violation
3	Bus error
4	Misaligned address
5	PMA empty hole access
6	PMA attribute inconsistency
7	PMA NAMO exception

- When scause = 7 (Store access fault)

Value	Meaning
0	Reserved
1	ECC/Parity error
2	PMP store access violation

Continued on next page...

Value	Meaning
3	Bus error
4	Misaligned address
5	PMA empty hole access
6	PMA attribute inconsistency
7	PMA NAMO exception

The value of SDCAUSE for imprecise exceptions:

- When scause = *Local Interrupt 272 (16 + 256)* (ECC error local interrupt)

Value	Meaning
0	Reserved
1	LM subordinate port ECC/Parity error
2	Imprecise store ECC/Parity error
3	Imprecise load ECC/Parity error

- When scause = *Local Interrupt 273 (17 + 256)* (Bus read/write transaction error local interrupt)

Value	Meaning
0	Reserved
1	Reserved
2	Bus error
3	PMP error caused by load instructions
4	PMP error caused by store instructions
5	PMA error caused by load instructions"
6	PMA error caused by store instructions

- For other exceptions and interrupts, this register will not be updated.

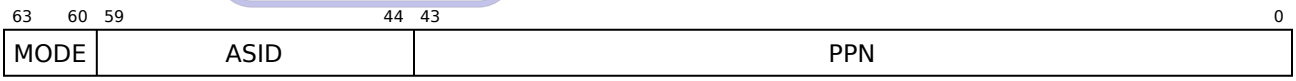
16.8.13.1 Detailed Exception Priority

For instruction, load, store access fault exceptions, a PMP exception has higher priority than a PMA exception when both occur on the same instruction.

16.9 Supervisor Translation Related CSRs

16.9.1 Supervisor Address Translation and Protection

Mnemonic Name: satp
IM Requirement: misa[18]==1
Access Mode: Supervisor
CSR Address: 0x180 (standard read/write)

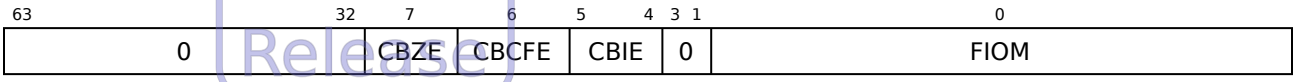


Field Name	Bits	Description	Type	Reset
PPN	[43:0]	PPN holds the physical page number of the root page table.	RW	0
ASID	[59:44]	ASID holds the address space identifier.	RW	0
MODE	[63:60]	MODE holds the page translation mode. When MODE is Bare, virtual addresses are equal to physical addresses in S-mode. When MMU is not supported in the product, this CSR will be hardwired to 0.	RW	0

Value	Name	Meaning
0	Bare	No page translation
8	Sv39	Page-based 39-bit virtual addressing
9	Sv48	Page-based 48-bit virtual addressing
10	Sv57	Page-based 57-bit virtual addressing

16.9.2 Supervisor Environment Configuration Register

Mnemonic Name: senvcfg
IM Requirement: misa.S==1
Access Mode: Supervisor
CSR Address: 0x10a (standard read/write)



This register controls certain characteristics of the execution environment for U-mode when executing under the supervision of S-mode.

Field Name	Bits	Description	Type	Reset
FIOM	[0]	Fence of I/O implies Memory.	RW	0

Value	Meaning
0	FENCE instructions used to order accesses to device I/O are not modified.
1	FENCE instructions used to order accesses to device I/O also order accesses to memory when executed in U-mode (PI implies PR, PO implies PW, SI implies SR, and SO implies SW). If an AMO instruction that accesses an I/O region has its aq and/or rl bit set, it is ordered as though it accesses both device I/O and memory.

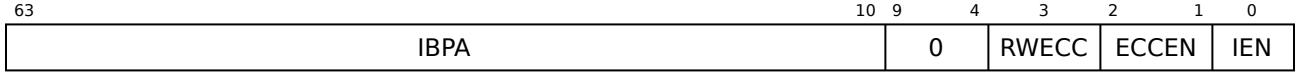
Continued on next page...

Field Name	Bits	Description	Type	Reset
CBIE	[5:4]	Cache Block Invalidate instruction Enable. **Value** / **Meaning** 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed and performs a flush operation. 2: Reserved 3: The instruction is executed and performs an invalidate operation.	RW	0
CBCFE	[6]	Cache Block Clean and Flush instruction Enable. **Value** / **Meaning** 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed.	RW	0
CBZE	[7]	Cache Block Zero instruction Enable. **Value** / **Meaning** 0: The instruction raises an illegal instruction or virtual instruction exception. 1: The instruction is executed.	RW	0

16.10 Memory and Miscellaneous Registers

16.10.1 Instruction Local Memory Base Register

Mnemonic Name: milmb
IM Requirement: ILM_SIZE_KB > 0
Access Mode: Machine
CSR Address: 0x7c0 (non-standard read/write)

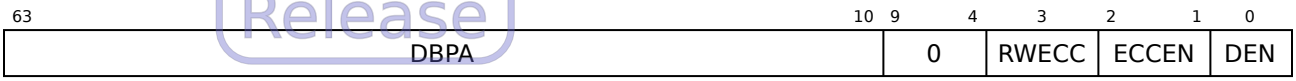


This register controls instruction local memory.

Field Name	Bits	Description	Type	Reset
IEN	[0]	ILM enable control: **Value** / **Meaning** 0 — ILM is disabled. 1 — ILM is enabled. The reset value is controlled by the `ilm_boot` input signal of AX46MPV.	RW	IM
ECCEN	[2:1]	Parity/ECC enable control: **Value** / **Meaning** 0 — Disable parity/ECC 1 — Reserved 2 — Generate exceptions only on uncorrectable parity/ECC errors 3 — Generate exceptions on any type of parity/ECC errors	RW	0
RWECC	[3]	Controls diagnostic accesses of ECC codes of the ILM RAMs. When set, load/store to ILM reads/writes ECC codes to the `mecc_code` register. This bit can be set for injecting ECC errors to test the ECC handler. **Value** / **Meaning** 0 — Disable diagnostic accesses of ECC codes 1 — Enable diagnostic accesses of ECC codes	RW	0
IBPA	[63:10]	Indicates the base physical address of ILM. It has to be an integer multiple of the ILM size.	RO	ILM_BASE[63:10]

16.10.2 Data Local Memory Base Register

Mnemonic Name: mdlmb
IM Requirement: DLM_SIZE_KB > 0
Access Mode: Machine
CSR Address: 0x7c1 (non-standard read/write)



This register controls data local memory.

Field Name	Bits	Description	Type	Reset
DEN	[0]	DLM enable control: **Value** / **Meaning** 0 — DLM is disabled. 1 — DLM is enabled. The reset value is controlled by the `dlm_boot` input signal of AX46MPV.	RW	IM
ECCEN	[2:1]	Parity/ECC enable control: **Value** / **Meaning** 0 — Disable parity/ECC 1 — Reserved 2 — Generate exceptions only on uncorrectable parity/ECC errors 3 — Generate exceptions on any type of parity/ECC errors	RW	0

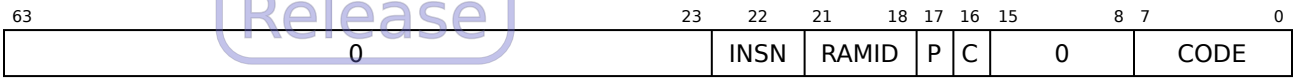
Continued on next page...

Field Name	Bits	Description	Type	Reset
RWECC	[3]	Controls diagnostic accesses of ECC codes of the DLM RAMs. When set, load/store to DLM reads/writes ECC codes to the `mecc_code` register. This bit can be set for injecting ECC errors to test the ECC handler.	RW	0
DBPA	[63:10]	Indicates the base physical address of DLM. It has to be an integer multiple of the DLM size.	RO	DLM_BASE[63:10]

Value	Meaning
0	Disable diagnostic accesses of ECC codes
1	Enable diagnostic accesses of ECC codes

16.10.3 ECC Code Register

Mnemonic Name: mecc_code
IM Requirement: mmsc_cfg.ECC == 1
Access Mode: Machine
CSR Address: 0x7c2 (non-standard read/write)



This register is used for accessing ECC array.

Field Name	Bits	Description	Type	Reset
CODE	[7:0]	This field records the ECC value on ECC error exceptions. It also reads/writes the ECC codes when diagnostic access of ECC codes are enabled (`milmb.RWECC`, `mdlmb.RWECC`, `mcache_ctl.IC_RWECC`, `mcache_ctl.DC_RWECC`, `mcache_ctl.TLB_RWECC`, or `mcache_ctl.BTB_RWECC` is 1). The bit width of the `CODE` field is determined by the maximum ECC granularity. **Max Protection Granularity** / **`mecc_code.CODE`** 32 / 7 bits 64 / 8 bits	RW	0
C	[16]	Correctable error. This bit is updated on parity/ECC error exceptions. **Value** / **Meaning** 0 / Uncorrectable error 1 / Correctable error	RO	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
P	[17]	Indicates whether the error is precise or not. This bit is updated on parity/ECC error exceptions. **Value** / **Meaning** 0 – Imprecise error 1 – Precise error	RO	1
RAMID	[21:18]	The RAM ID that caused parity/ECC errors. This bit is updated on parity/ECC error exceptions. **Value** / **Meaning** 0–1 – Reserved 2 – Tag RAM of I-Cache 3 – Data RAM of I-Cache 4 – Tag RAM of D-Cache 5 – Data RAM of D-Cache 6 – Tag RAM of TLB 7 – Data RAM of TLB 8 – ILM 9 – DLM 10 – BTB 11–15 – Reserved	RO	0
INSN	[22]	Indicates if the parity/ECC error is caused by instruction fetch or data access. **Value** / **Meaning** 0 – Data access 1 – Instruction fetch	RO	0

16.10.4 NMI Vector Base Address Register

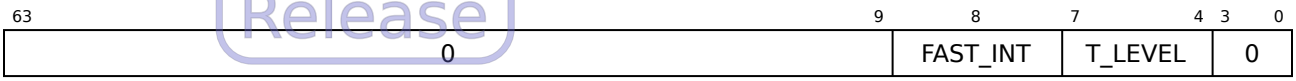
Mnemonic Name: mnvec
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x7c3 (non-standard read/write)

This register indicates the entry point when an NMI occurs.

Field Name	Bits	Description	Type	Reset
MNVEC	[63:0]	Base address of the NMI handler. Its value is the zero-extended value of the `coreN_reset_vector[VALEN-1:0]` input signal to this core (Core N).	RO	Pin Configured

16.10.5 Performance Throttling Control Register

Mnemonic Name: mpft_ctl
IM Requirement: POWERBRAKE_SUPPORT = "yes"
Access Mode: Machine
CSR Address: 0x7c5 (non-standard read/write)

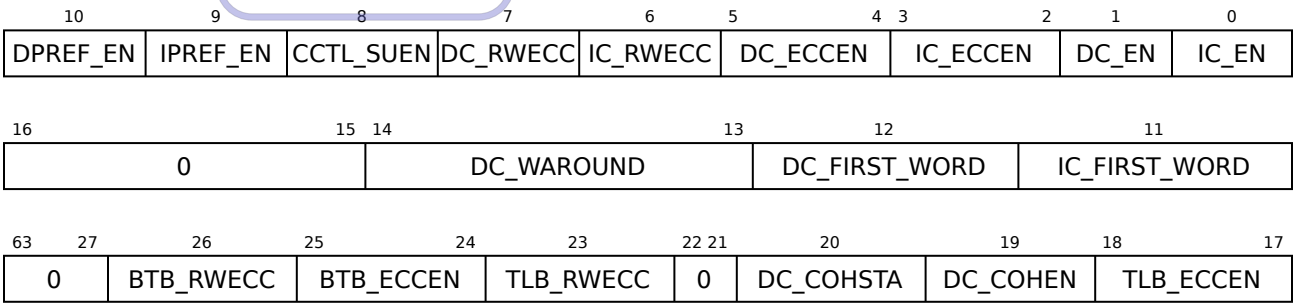


Field Name	Bits	Description	Type	Reset
T_LEVEL	[7:4]	Throttling Level. The processor has the highest performance at throttling level 0 and the lowest performance at throttling level 15.	RW	0
FAST_INT	[8]	Fast interrupt response. If this field is set, `mxstatus.PFT_EN` will be automatically cleared when the processor enters an interrupt handler.	RW	0

Value	Meaning
0	Level 0 (the highest performance)
1-14	Level 1-14
15	Level 15 (the lowest performance)

16.10.6 Cache Control Register

Mnemonic Name: mcache_ctl
IM Requirement: Cache optional (micm_cfg.ISZ != 0 || mdcm_cfg.DSZ != 0 || mmsc_cfg.TLB_ECC != 0 || mmsc_cfg.BTB_ECC != 0)
Access Mode: Machine
CSR Address: 0x7ca (non-standard read/write)



Field Name	Bits	Description	Type	Reset
IC_EN	[0]	Controls if the instruction cache is enabled. **Value** / **Meaning** 0 — I-Cache is disabled. 1 — I-Cache is enabled.	RW	0
DC_EN	[1]	Controls if the data cache is enabled. **Value** / **Meaning** 0 — D-Cache is disabled. 1 — D-Cache is enabled.	RW	0
IC_ECCEN	[3:2]	Parity/ECC error checking enable control for the instruction cache. **Value** / **Meaning** 0 — Disable parity/ECC 1 — Reserved 2 — Generate exceptions only on uncorrectable parity/ECC errors 3 — Generate exceptions on any type of parity/ECC errors This field is hardwired to zeros when **I-Cache Soft Error Protection** is none.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
DC_ECCEN	[5:4]	Parity/ECC error checking enable control for the data cache. **Value** / **Meaning** 0 — Disable parity/ECC 1 — Reserved 2 — Generate exceptions only on uncorrectable parity/ECC errors 3 — Generate exceptions on any type of parity/ECC errors This field is hardwired to zeros when **D-Cache Soft Error Protection** is none.	RW	0
IC_RWECC	[6]	Controls diagnostic accesses of ECC codes of the instruction cache RAMs. It is set to enable CCTL operations to access the ECC codes (see Section 7.5). This bit can be set for injecting ECC errors to test the ECC handler. **Value** / **Meaning** 0 — Disable diagnostic accesses of ECC codes 1 — Enable diagnostic accesses of ECC codes This field is hardwired to zero when **I-Cache Soft Error Protection** is none.	RW	0
DC_RWECC	[7]	Controls diagnostic accesses of ECC codes of the data cache RAMs. It is set to enable CCTL operations to access the ECC codes (see Section 7.5). This bit can be set for injecting ECC errors to test the ECC handler. **Value** / **Meaning** 0 — Disable diagnostic accesses of ECC codes 1 — Enable diagnostic accesses of ECC codes This field is hardwired to zero when **D-Cache Soft Error Protection** is none.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
CCTL_SUEN	[8]	Enable bit for Superuser-mode and User-mode software to access `ucctlbeginaddr` and `ucctlcommand` CSRs.	RW	0

Value	Meaning
0	Disable `ucctlbeginaddr` and `ucctlcommand` accesses in S/U mode
1	Enable `ucctlbeginaddr` and `ucctlcommand` accesses in S/U mode

Field Name	Bits	Description	Type	Reset
IPREF_EN	[9]	This bit controls hardware prefetch for instruction fetches to cacheable memory regions when I-Cache size is not 0.	RW	0

Value	Meaning
0	Disable hardware prefetch on instruction fetches
1	Enable hardware prefetch on instruction fetches

Field Name	Bits	Description	Type	Reset
DPREF_EN	[10]	This bit controls hardware prefetch for load/store accesses to cacheable memory regions when D-Cache size is not 0.	RW	0

Value	Meaning
0	Disable hardware prefetch on load/store memory accesses
1	Enable hardware prefetch on load/store memory accesses

Field Name	Bits	Description	Type	Reset
IC_FIRST_WORD	[11]	I-Cache miss allocation filling policy.	RO	1

Value	Meaning
1	Cache line data returns the lowest address (double) word first

Continued on next page...

Field Name	Bits	Description	Type	Reset
DC_FIRST_WORD	[12]	D-Cache miss allocation filling policy. **Value** / **Meaning** 1 — Cache line data returns the lowest address (double) word first	RO	1
DC_WAROUND	[14:13]	D-Cache Write-Around threshold **Value** / **Meaning** 0 — Disable streaming. All cacheable write misses allocate a cache line according to the PMA settings. 1 — Stop allocating D-Cache entries regardless of the PMA settings after consecutive stores to 4 cache lines. 2 — Stop allocating D-Cache entries regardless of the PMA settings after consecutive stores to 64 cache lines. 3 — Stop allocating D-Cache entries regardless of the PMA settings after consecutive stores to 128 cache lines.	RW	0
TLB_ECCEN	[18:17]	ECC error checking enable control for the STLB tag and data SRAMs. **Value** / **Meaning** 0 — Disable ECC 1 — Reserved 2 — Generate exceptions only on unrepairable ECC errors 3 — Generate exceptions on any type of ECC errors This field is hardwired to 0 when **Shared TLB Soft Error Protection** is configured to “none”.	RW	0

Continued on next page...

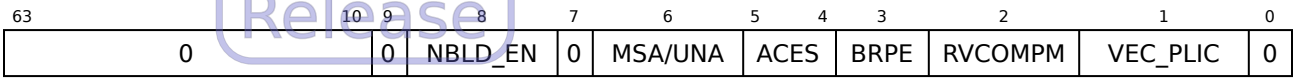
Field Name	Bits	Description	Type	Reset
DC_COHEN	[19]	Enable data cache to participate in the coherence management. **Value** / **Meaning** 0: Disable Data cache to participate in the coherence management 1: Enable Data cache to participate in the coherence management	RW	0
DC_COHSTA	[20]	Indicate if data cache is participated in the coherence management **Value** / **Meaning** 0: Data cache is not participated in the coherence management 1: Data cache is participated in the coherence management	RO	0
TLB_RWECC	[23]	Controls diagnostic accesses of ECC codes of the STLB tag and data RAMs. It is set to enable CCTL operations to access the ECC codes (see Section 7.5). This bit can be set for injecting ECC errors to test the ECC handler. **Value** / **Meaning** 0: Disable diagnostic accesses of ECC codes 1: Enable diagnostic accesses of ECC codes This field is hardwired to 0 when **Shared TLB Soft Error Protection** is configured to “none”.	RW	0

Continued on next page. . .

Field Name	Bits	Description	Type	Reset
BTB_ECCEN	[25:24]	ECC error checking enable control for the BTB RAMs. **Value** / **Meaning** 0 – Disable ECC 1 – Reserved 2 – Generate exceptions only on unrepairable ECC errors 3 – Generate exceptions on any type of ECC errors This field is hardwired to 0 when **BTB Soft Error Protection** is configured to “none”.	RW	0
BTB_RWECC	[26]	Controls diagnostic accesses of ECC codes of the BTB RAMs. It is set to enable CCTL operations to access the ECC codes (see Section 3.1). This bit can be set for injecting ECC errors to test the ECC handler. **Value** / **Meaning** 0 – Disable diagnostic accesses of ECC codes 1 – Enable diagnostic accesses of ECC codes This field is hardwired to 0 when **BTB Soft Error Protection** is configured to “none”.	RW	0

16.10.7 Machine Miscellaneous Control Register

Mnemonic Name: mmisc_ctl
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x7d0 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
VEC_PLIC	[1]	Selects the operation mode of PLIC: **Value** / **Meaning** 0 — Regular mode 1 — Vector mode Note that both this bit and the vector mode enable bit (VECTORED) of the Feature Enable Register in NCEPLIC100 should be turned on for the vectored interrupt support to work correctly. See AndesCore AX46MPV Integration Guide (IG094). This bit is hardwired to 0 if the vectored PLIC feature is not supported.	RW	0
RVCOMPM	[2]	RISC-V compatibility mode enable bit. If the compatibility mode is turned on, all Andes-specific instructions become reserved instructions, including ACE instructions. **Value** / **Meaning** 0 — Disabled 1 — Enabled	RW	0

Continued on next page...

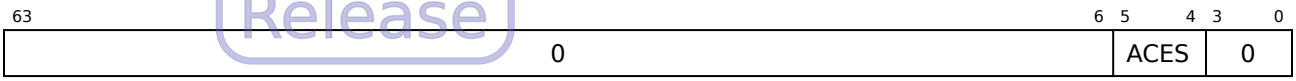
Field Name	Bits	Description	Type	Reset
BRPE	[3]	Branch prediction enable bit. This bit controls all branch prediction structures. **Value** / **Meaning**: 0 = Disabled; 1 = Enabled This bit is hardwired to 0 if branch prediction structure is not supported.	RW	1
ACES	[5:4]	Andes Custom Extension (ACE) extension context status field: **Value** / **Meaning**: 0 = Off; 1 = Initial; 2 = Clean; 3 = Dirty • This field should not be Off (0) for ACE instructions to execute normally. The ACE unit enters Off state by software program through `CSRW` instructions. • A normal flow to turn on the ACE unit is as follows: – `ACES` is in the Off state. – An ACE instruction executed in the Off state triggers an illegal instruction with `sdcause/mdcause` = 2 (ACE disabled exception). – The exception handler initializes all ACE register states, changes this field to the Initial state, and then returns from exception. – The ACE instruction is executed again. Since `ACES` is not in the Off state this time, it shall execute correctly. If any ACE register states are modified, `ACES` will be updated to the Dirty state automatically by hardware. This field is hardwired to 0 if ACE extension is not configured.	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
MSA/UNA	[6]	This field controls whether load/store instructions can access misaligned memory locations without generating Address Misaligned exceptions. Supported instructions: LW/LH/LHU/SW/SH/LD/LWU/SD **Value** / **Meaning** 0 — Misaligned accesses generate Address Misaligned exceptions. 1 — Misaligned accesses are allowed.	RW	1
NBLD_EN	[8]	This field controls the blocking behavior of load instructions. When this bit is clear, load instructions are blocking. When this bit is set, load instructions will not be blocking on such occasions and bus errors will no longer be reported synchronously. See Section 10.1 for details. **Value** / **Meaning** 0 — Load instructions are blocking. 1 — Load instructions are non-blocking.	RW	0

16.10.8 Supervisor Miscellaneous Control Register

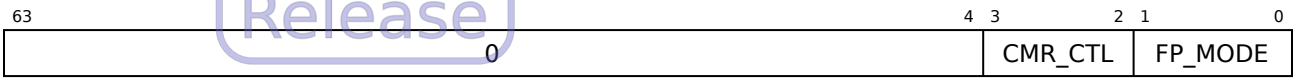
Mnemonic Name: smisc_ctl
IM Requirement: (misa[18]==1) && (mmsc_cfg.ACE==1)
Access Mode: Supervisor
CSR Address: 0x9d0 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
ACES	[5:4]	Andes Custom Extension (ACE) extension context status field: **Value** / **Meaning** 0 — Off 1 — Initial 2 — Clean 3 — Dirty • This field should not be Off (0) for ACE instructions to execute normally. The ACE unit enters Off state by software program through `CSRW` instructions. • A normal flow to turn on the ACE unit is as follows: – `ACES` is in the Off state. – An ACE instruction executed in the Off state triggers an illegal instruction with `sdcause/mdcause` = 2 (ACE disabled exception). – The exception handler initializes all ACE register states, changes this field to the Initial state, and then returns from the exception. – The ACE instruction is executed again. Since `ACES` is not in the Off state this time, it shall execute correctly. If any ACE register states are modified, `ACES` will be updated to the Dirty state automatically by hardware. This field is hardwired to 0 if ACE extension is not configured.	RW	0

16.10.9 User Miscellaneous Control Register

Mnemonic Name: umisc_ctl
IM Requirement: (mmsc_cfg.ALT_FP_FMT==1) || (mmsc_cfg3.CST_CTL==1)
Access Mode: User and above
CSR Address: 0x813 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
FP_MODE	[1:0]		RW	0
CMR_CTL	[3:2]	Customer control field for ACE logic usages	RW	0

Value	Meaning
0	FP16 format is used to represent 16-bit floating-point values.
1	BFLOAT16 format is used to represent 16-bit floating-point values.
2–3	Reserved

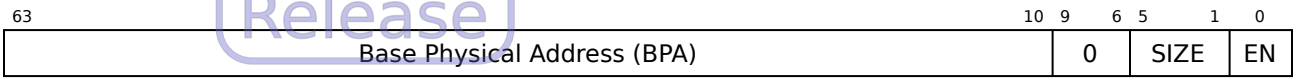
16.10.10 Clock Control Register

Mnemonic Name: mclk_ctl
IM Requirement: (misa.V == 1) | (mmsc_cfg.BF16CVT == 1)
Access Mode: Machine
CSR Address: 0x7df (non-standard read/write)

This register is hardwired to zeros.

16.10.11 PPI (Private Peripheral Interface) Base Register

Mnemonic Name: mppib
IM Requirement: mmsc_cfg.PPI==1
Access Mode: Machine
SR Encoding: 0x7f0 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
EN	[0]	PPI enable control: **Value** / **Meaning** 0 / PPI is disabled. 1 / PPI is enabled. • When **PPI Size** is not 0 KiB, PPI is enabled.	RO	1

Continued on next page...

Field Name	Bits	Description	Type	Reset
SIZE	[5:1]	PPI Size:	RO	IM

Value	Meaning
0	0 Byte
1	1 KiB
2	2 KiB
3	4 KiB
4	8 KiB
5	16 KiB
6	32 KiB
7	64 KiB
8	128 KiB
9	256 KiB
10	512 KiB
11	1 MiB
12	2 MiB
13	4 MiB
14	8 MiB
15	16 MiB
16	32 MiB
17	64 MiB
18	128 MiB
19	256 MiB
20	512 MiB
21	1 GiB
22	2 GiB
23-31	Reserved

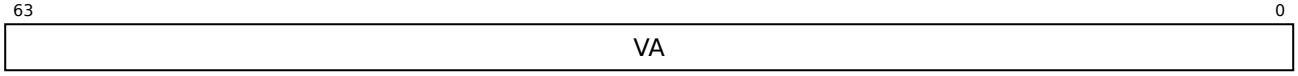
BPA	[63:10]	The base physical address of PPI. It has to be an integer multiple of the PPI size.	RO	IM

16.10.12 Machine CCTL Begin Address

Mnemonic Name: mcctlbeginaddr
IM Requirement: Cache optional
Access Mode: Machine
CSR Address: 0x7cb (non-standard read/write)

This register holds the address information required by CCTL operations. It is only present when (`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0) and (`mmsc_cfg.CCTLCSR` = 1).

- For "VA" type of CCTL operations:



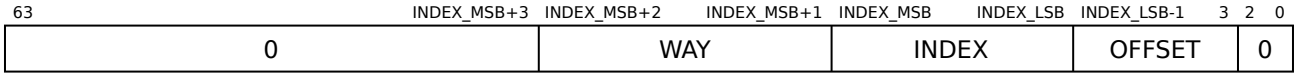
  The `mcctlbeginaddr` register contains the starting virtual address for CCTL operations triggered by writes to the `mcctlcommand` register. For CCTL lock operations, the `mcctldata` register will be updated with a status value (0: fail; 1: success) when the operations complete.

  After an update to the `mcctlcommand` register with a VA-type command, the value of this register will be incremented with the byte size of the corresponding cache line.

- For "Index" type of CCTL operations:

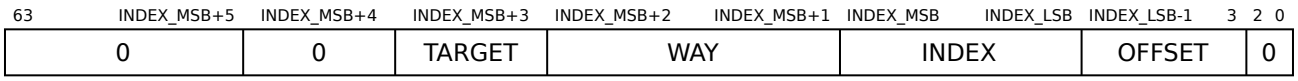
  The `mcctlbeginaddr` register contains the cache index for CCTL operations triggered by writes to the `mcctlcommand` register.

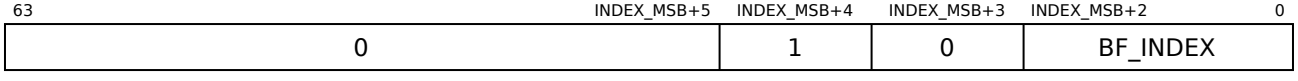
  – For "L1I_IX_INVAL", "L1I_IX_RTAG", "L1I_IX_WTAG", "L1D_IX_INVAL", "L1D_IX_WB" or "L1D_IX_WBINVAL":



    The `WAY` field in this register will be incremented to the value of the next way. If the incremented `WAY` wraps to 0 (i.e., the first way of a set), then the `INDEX` field in this register will be incremented.

  – For "L1D_IX_RTAG" or "L1D_IX_WTAG":





Field Name	Bits	Description	Type	Reset
BF_INDEX	[BITS-1:0]	Specifies the target counter index of a bloom filter counter. See Section 16.5.11 for the details of the bit width, `mmldbf_cfg.BITS`.	RW	0
TARGET	[INDEX_MSB+4:INDEX_MSB+3]	Specifies the target tag RAM for L1D_IX_RTAG and L1D_IX_WTAG.	RW	0

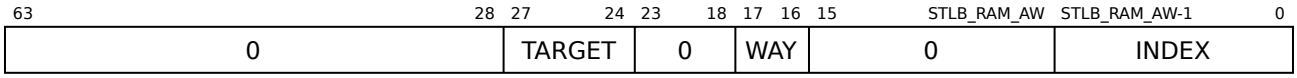
Value	Meaning
b00	Primary D-Cache Tag RAM
b01	Duplicate D-Cache Tag RAM
b10	Counting Bloom Filter RAM
b11	Reserved

The `WAY` field in this register will be incremented to the value of the next way. If the incremented `WAY` wraps to 0 (i.e., the first way of a set), then the `INDEX` field in this register will be incremented. The `TARGET` field in this register specifies the target of D-Cache primary tag RAM, duplicate tag RAM, or counting bloom filter RAM.

– For "L1*_IX_RDATA" and "L1*_IX_WDATA" commands:

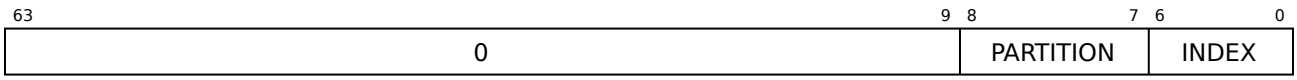
The `OFFSET` field in this register will be incremented first to the next `OFFSET` value. If the incremented `OFFSET` field wraps around to 0 (i.e., the first data of a cache line), then the `WAY` field in this register will be incremented. If the incremented `WAY` field wraps around to 0 (i.e., the first way of a set), then the `INDEX` field in this register will be incremented. The `TARGET` field in this register is ignored.

– For "TLB_IX_*" commands:



After an update to the `mcctlcommand` CSR with a TLB type command, the value of this register will not be changed. The `TARGET` field should be set to 3 for accessing the STLB RAMs. Otherwise, the CCTL command will be treated like NOP.

– For "BTB_IX_*" commands:



The `INDEX` field in this register will be incremented first to the next `INDEX` value. If the incremented `INDEX` field wraps around to 0, then `PARTITION` in this register will be incremented.

16.10.13 Machine CCTL Command

Mnemonic Name: mcctlcommand
IM Requirement: Cache optional
Access Mode: Machine
CSR Address: 0x7cc (non-standard read/write)

Writing to this register will trigger a CCTL operation, with the type of the operation specified by the value written. Valid CCTL operations are defined in Table 128. See Section 3.1 and Section 7.5 for more information. CCTL operations are inherently not atomic; see notes below for usage limitations.

This register is only present when (`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0) and (`mmsc_cfg.CCTLCSR` = 1).

Table 128: CCTL Command Definition

	Value	Command	Type
0	0b00000_000	L1D_VA_INVAL	VA
1	0b00000_001	L1D_VA_WB	VA
2	0b00000_010	L1D_VA_WBINVAL	VA
3	0b00000_011	L1D_VA_LOCK	VA
4	0b00000_100	L1D_VA_UNLOCK	VA
6	0b00000_110	L1D_WBINVAL_ALL	-
7	0b00000_111	L1D_WB_ALL	-
8	0b00001_000	L1I_VA_INVAL	VA
11	0b00001_011	L1I_VA_LOCK	VA
12	0b00001_100	L1I_VA_UNLOCK	VA
16	0b00010_000	L1D_IX_INVAL	Index
17	0b00010_001	L1D_IX_WB	Index
18	0b00010_010	L1D_IX_WBINVAL	Index
19	0b00010_011	L1D_IX_RTAG	Index
20	0b00010_100	L1D_IX_RDATA	Index
21	0b00010_101	L1D_IX_WTAG	Index
22	0b00010_110	L1D_IX_WDATA	Index
23	0b00010_111	L1D_INVAL_ALL	-
24	0b00011_000	L1I_IX_INVAL	Index
27	0b00011_011	L1I_IX_RTAG	Index

Continued on next page...

Table 128: (continued)

	Value	Command	Type
28	0b00011_100	L1I_IX_RDATA	Index
29	0b00011_101	L1I_IX_WTAG	Index
30	0b00011_110	L1I_IX_WDATA	Index
147	0b10010_011	TLB_IX_RTAG	Index
148	0b10010_100	TLB_IX_RDATA	Index
149	0b10010_101	TLB_IX_WTAG	Index
150	0b10010_110	TLB_IX_WDATA	Index
156	0b10011_100	BTB_IX_RDATA	Index
158	0b10011_110	BTB_IX_WDATA	Index

Note

CCTL operations take parameters from multiple CSR registers, thus they are inherently not atomic. In addition, those CSRs share common storage across privilege levels, making them vulnerable to be overwritten across context switches. This implies that software with higher privilege level should back up the content of CCTL CSR registers with interrupts disabled before using them, and restore their values afterwards if CCTL operations will be invoked in multiple privilege levels. The same is true if CCTL operations will be used in both non-interrupt codes and interrupt handlers.

16.10.14 Machine CCTL Data

Mnemonic Name: mcctldata
IM Requirement: Cache optional
Access Mode: Machine
CSR Address: 0x7cd (non-standard read/write)

This register holds data required/returned by certain CCTL operations. It is only present when (`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0) and (`mmsc_cfg.CCTLCSR` = 1). The complete list of CCTL operations are summarized in Table 129 and described below.

Table 129: CCTL Commands Which Access `mcctldata`

Value of `mcctlcommand`		Command	Type
3	0b00000_011	L1D_VA_LOCK	VA
11	0b00001_011	L1I_VA_LOCK	VA
19	0b00010_011	L1D_IX_RTAG	Index
20	0b00010_100	L1D_IX_RDATA	Index
21	0b00010_101	L1D_IX_WTAG	Index
22	0b00010_110	L1D_IX_WDATA	Index
27	0b00011_011	L1I_IX_RTAG	Index
28	0b00011_100	L1I_IX_RDATA	Index
29	0b00011_101	L1I_IX_WTAG	Index
30	0b00011_110	L1I_IX_WDATA	Index
147	0b10010_011	TLB_IX_RTAG	Index
148	0b10010_100	TLB_IX_RDATA	Index
149	0b10010_101	TLB_IX_WTAG	Index
150	0b10010_110	TLB_IX_WDATA	Index
156	0b10011_100	BTB_IX_RDATA	Index
158	0b10011_110	BTB_IX_WDATA	Index

- For CCTL lock operations: The `mcctldata` register will be updated with a status value (0: fail; 1: success) when the operations complete.
- For CCTL index read/write-data operations: `mcctldata[63:0]` holds the cache data for the operations.

- For CCTL index read/write-tag operations, the `mcctldata` register holds the cache tag for the operations.
  - The bit positions of the fields for I-Cache CCTL index read/write-tag is as follows:
    - The TAG field does not hold every bit of PA[(PALEN-1):10] for the cache line. The cache tag RAMs do not hold all physical addresses down to bit 10. They only hold enough bits for tag look-ups. The unused lower order TAG bits will be reported as *Don't Care* value for tag-read operations and ignored on tag-write operations. The full PA value should be constructed using the corresponding index bits.
    - I-Cache TAG RAM holds PA[(PA_LEN-1):A].
      - A = min(12:(log_2(`micm_cfg.ISET`)+6))
    - If A is 11, bit 0 is unused. If A is 12, bit 0 and bit 1 are unused.
    - Bit XLEN-3 holds the duplicated lock bit for the I-Cache to better tolerate soft errors.
    - Bit XLEN-2 holds the lock bit.
    - Bit XLEN-1 holds the valid bit.

Table 130: I-Cache CCTL Index Read/Write TAG Bit Fields

Field	Bit Position
TAG	[0+:PALEN-11]
LOCK_DUP	[XLEN-3]
LOCK	[XLEN-2]
VALID	[XLEN-1]

- For D-Cache CCTL index type read/write-tag operations
  - The bit positions of the primary tag or duplicated tag RAM fields:
    - The `MESI` field holds the cache line status.
    - Bit 3 holds the lock bit.
    - The `TAG` field holds the tag data for the cache line. DTW = PALEN-12.

Table 131: D-Cache TAG RAM Bit Fields

Field	Bit Position
MESI	[0+:2]
LOCK	[3]

Continued on next page...

Table 131: (continued)

Field	Bit Position
Reserved	[4]
TAG	[5+:DTW]

– The bit positions of the counting bloom filter RAM fields:

  * The CNT field holds the counter value of the counting bloom filter. See Section 16.5.11 for the details of the bit width, mmu_mldbf_cfg.DEPTH.

Table 132: D-Cache Counting Bloom Filter RAM Bit Fields

Field	Bit Position
CNT	[0+:DEPTH]

· For index type of CCTL TLB operations:

- The bit position of the fields for TLB CCTL index read/write-tag is as follows:

  – The TAG field holds Partial Virtual Page Number (Partial-VPN), Address Space ID (ASID), and VALID.

Table 133: TLB CCTL Index Read/Write TAG Bit Fields

Field	Bit Position
VALID	[0]
ASID	[1+:ASIDLEN]
Partial-VPN	[(1+ASIDLEN)+:VALEN-STLB_RAM_AW-12]

- The bit position of the fields for TLB CCTL index read/write-data is as follows:

  – The DATA field holds Physical Page Number (PPN), PTE information (V, R, W, X, U, G, A, and D).

Table 134: TLB CCTL Index Read/Write DATA Bit Fields

Field	Bit Position
V	[0]
R	[1]
W	[2]
X	[3]
U	[4]

Continued on next page...

Table 134: (continued)

Field	Bit Position
G	[5]
A	[6]
D	[7]
PPN	[8+:(PALEN-12)]
PBMT	[(8+(PALEN-12))+:2]

  * For index type of BTB CCTL operations:

- When BTB SRAM data width is less than 64, bits higher than BTB SRAM data width in `mcctldata` are updated as 0 and ignored.

Table 135: BTB CCTL Index Read/Write DATA Bit Fields

Field	Bit Position
BTB_RAM_DATA	[XLEN-1:0]

16.10.15 Supervisor CCTL Data

Mnemonic Name: scctldata
IM Requirement: Cache optional
Access Mode: Supervisor and above
CSR Address: 0x9cd (non-standard read/write)

This register is only present when ((`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0), `mmsc_cfg.CCTLCSR` = 1, and `misa[18]` = 1). It is an alias to the `mcctldata` register and is only accessible to Supervisor-mode software when `mcache_ctl.CCTL_SUEN` is 1. Otherwise, illegal instruction exceptions will be triggered.

Note

- S-mode software triggers CCTL operations through writing the `ucctlcommand` register, and the associated addresses for the CCTL operations are specified in the `ucctlbeginaddr` register.
- Due to the sharing of storage with `mcctldata`, the interrupt-handler and M-mode software should back up `mcctldata` before use. See notes in Section 16.10.13 for usage limitations.

16.10.16 User CCTL Begin Address

Mnemonic Name: ucctlbeginaddr
IM Requirement: Cache optional
Access Mode: User and above
CSR Address: 0x80b (non-standard read/write)

This register is only present when ((`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0), `mmsc_cfg.CCTLCSR` = 1, and `misa[20]` = 1). It is an alias to the `mcctlbeginaddr` register and is only accessible to Supervisor-mode and User-mode software when `mcache_ctl.CCTL_SUEN` is 1. Otherwise, illegal instruction exceptions will be triggered.

Note

- Both S-mode and U-mode software trigger CCTL operations through writing the `ucctlcommand` register; the associated addresses for CCTL operations are specified in the `ucctlbeginaddr` register.
- Due to the sharing of storage with `mcctlbeginaddr`, the interrupt handler and privileged-mode software should back up `mcctlbeginaddr` before use. See notes in Section 16.10.13 for usage limitations.

16.10.17 User CCTL Command

Mnemonic Name: ucctlcommand
IM Requirement: Cache optional
Access Mode: User and above
CSR Address: 0x80c (non-standard read/write)

Writing to this register will trigger a CCTL operation, with the type of the operation specified by the value written. Valid User CCTL commands are defined in Table 136. Both S-mode and U-mode software trigger CCTL operations through this register. However, if `ucctlcommand` is written by U-mode software with a command whose "U-Mode Allowed" field is 0 in Table 136, an illegal instruction exception will still be generated.

This register is only present when ((`micm_cfg.ISZ` != 0, `mdcm_cfg.DSZ` != 0, `mmsc_cfg.TLB_RAM_CMD` != 0, or `mmsc_cfg3.BTB_RAM_CMD` != 0), `mmsc_cfg.CCTLCSR` = 1, and `misa[20]` = 1), and it is an alias to the `mcctlcommand` register. This register is only accessible to Supervisor-mode and User-mode software when `mcache_ctl.CCTL_SUEN` is 1. Otherwise, illegal instruction exceptions will be triggered.

Table 136: User CCTL Command Definition

Value of `ucctlcommand`		Command	Type	U-Mode Allowed
0	0b0000_000	L1D_VA_INVAL	VA	1
1	0b0000_001	L1D_VA_WB	VA	1
2	0b0000_010	L1D_VA_WBINVAL	VA	1
3	0b0000_011	L1D_VA_LOCK	VA	0
4	0b0000_100	L1D_VA_UNLOCK	VA	0
6	0b0000_110	L1D_WBINVAL_ALL	-	0
7	0b0000_111	L1D_WB_ALL	-	0
8	0b0001_000	L1I_VA_INVAL	VA	1
11	0b0001_011	L1I_VA_LOCK	VA	0
12	0b0001_100	L1I_VA_UNLOCK	VA	0
16	0b0010_000	L1D_IX_INVAL	Index	0
17	0b0010_001	L1D_IX_WB	Index	0
18	0b0010_010	L1D_IX_WBINVAL	Index	0
19	0b0010_011	L1D_IX_RTAG	Index	0
20	0b0010_100	L1D_IX_RDATA	Index	0

Continued on next page...

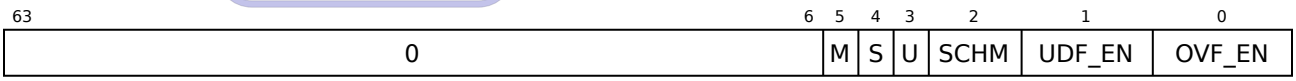
Table 136: (continued)

Value of `ucctlcommand`		Command	Type	U-Mode Allowed
21	0b0010_101	L1D_IX_WTAG	Index	0
22	0b0010_110	L1D_IX_WDATA	Index	0
23	0b0010_111	L1D_INVAL_ALL	-	0
24	0b0011_000	L1I_IX_INVAL	Index	0
27	0b0011_011	L1I_IX_RTAG	Index	0
28	0b0011_100	L1I_IX_RDATA	Index	0
29	0b0011_101	L1I_IX_WTAG	Index	0
30	0b0011_110	L1I_IX_WDATA	Index	0
147	0b10010_011	TLB_IX_RTAG	Index	0
148	0b10010_100	TLB_IX_RDATA	Index	0
149	0b10010_101	TLB_IX_WTAG	Index	0
150	0b10010_110	TLB_IX_WDATA	Index	0
156	0b10011_100	BTB_IX_RDATA	Index	0
158	0b10011_110	BTB_IX_WDATA	Index	0

16.11 Hardware Stack Protection and Recording Registers

16.11.1 Machine Hardware Stack Protection Control

Mnemonic Name: mhsp_ctl
IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)
Access Mode: Machine
CSR Address: 0x7C6 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
OVF_EN	[0]	Enable bit for both stack overflow protection and recording mechanisms. This bit is cleared to 0 automatically by hardware when a stack protection (overflow or underflow) exception is taken.	RW	0
UDF_EN	[1]	Enable bit for the stack underflow protection mechanism. This bit is cleared to 0 automatically by hardware when a stack protection (overflow or underflow) exception is taken.	RW	0

OVF_EN

Value	Meaning
0	Stack overflow protection and recording mechanisms are disabled.
1	Stack overflow protection and recording mechanisms are enabled.

UDF_EN

Value	Meaning
0	The stack underflow protection is disabled.
1	The stack underflow protection is enabled.

Continued on next page...

Field Name	Bits	Description	Type	Reset
SCHM	[2]	Selects the operating scheme of the stack protection and recording mechanism. **Value** / **Meaning** 0 — Stack overflow/underflow detection 1 — Top-of-stack recording	RW	0
U	[3]	Enables both SP protection and recording mechanisms in User mode. **Value** / **Meaning** 0 — Both mechanisms are disabled in User mode. 1 — Both mechanisms are enabled in User mode.	RW	0
S	[4]	Enables both SP protection and recording mechanisms in Supervisor mode. **Value** / **Meaning** 0 — Both mechanisms are disabled in Supervisor mode. 1 — Both mechanisms are enabled in Supervisor mode.	RW	0
M	[5]	Enables both SP protection and recording mechanisms in Machine mode. **Value** / **Meaning** 0 — Both mechanisms are disabled in Machine mode. 1 — Both mechanisms are enabled in Machine mode.	RW	0

Note: ALU, MUL, DIV, JAL, JALR, LOAD, C.PUSH, C.POP, C.POPRET, and C.POPRETZ instructions support both stack protection and recording.

16.11.2 Machine SP Bound Register

Mnemonic Name: msp_bound
IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)
Access Mode: Machine
CSR Address: 0x7c7 (non-standard read/write)

When the `SP` overflow detection mechanism is properly selected and enabled, any updated value to the `SP` register (via any instruction) is compared with the `msp_bound` register. If the updated value to the `SP` register is smaller than the `msp_bound` register, a stack overflow exception is generated. The stack overflow exception has an exception code of 32 in the `mcause` register.

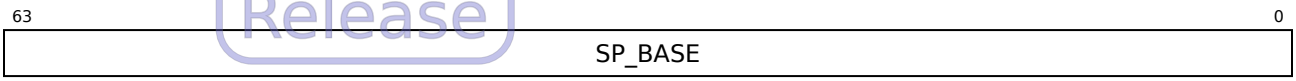
When the top of stack recording mechanism is properly selected and enabled, any updated value to the `SP` register on any instruction is compared with the `msp_bound` register. If the updated value to the `SP` register is smaller than the `msp_bound` register, the `msp_bound` register is updated with this updated value. It is an RW-type register with the all-one reset value (0xFFFFFFFF for RV32 and 0xFFFFFFFFFFFFFFFF for RV64).

Programming Note:

- The "`CSRRW sp, msp_bound, rs`" instruction updates both `sp` and `msp_bound` registers at the same time. When the stack overflow detection mechanism is enabled, using this instruction may generate unpredictable exception behavior.

16.11.3 Machine SP Base Register

Mnemonic Name: msp_base
IM Requirement: STACKSAFE_SUPPORT = "yes" (mmsc_cfg.HSP == 1)
Access Mode: Machine
CSR Address: 0x7c8 (non-standard read/write)



When the `SP` underflow detection mechanism is properly selected and enabled, any updated value to the `SP` register (via any instruction) is compared with the `msp_base` register. If the updated value to the `SP` register is greater than the `msp_base` register, a stack underflow exception is generated. The stack underflow exception has an exception code of 33 in the `mcause` register.

It is an RW-type register with the all-one reset value (0xFFFFFFFF for RV32 and 0xFFFFFFFFFFFFFFFF for RV64).

Programming Note:

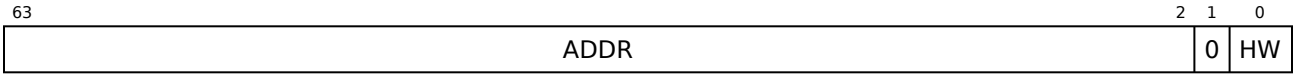
- The "`CSRRW sp, msp_base, rs`" instruction updates both `sp` and `msp_base` registers at the same time. When the stack underflow detection mechanism is enabled, using this instruction may generate unpredictable exception behavior.

16.12 CoDense Registers

16.12.1 Instruction Table Base Address Register

Mnemonic Name: uitb
IM Requirement: CODENSE_SUPPORT = "yes"
Access Mode: User and above
CSR Address: 0x800 (non-standard read/write)

This register defines the base address of the CoDense instruction table. Each entry in the table contains a 32-bit instruction, accessible for lookup and execution by a CoDense instruction. The table is typically generated by the compiler to replace 32-bit instructions with shorter 16-bit Andes CoDense instructions, hence reducing the overall code size.

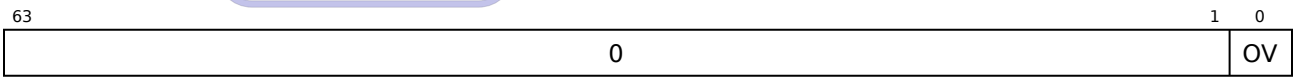


Field Name	Bits	Description	Type	Reset
HW	[0]	This bit specifies if the CoDense instruction table is hardwired. **Value** / **Meaning** 0 — The instruction table is located in memory. `uitb.ADDR` should be initialized to point to the table before using CoDense instructions. 1 — The instruction table is hardwired. Initialization of `uitb.ADDR` is not needed before using CoDense instructions.	RO	0
ADDR	[63:2]	The base address of the CoDense instruction table. This field is reserved if `uitb.HW` = 1.	RW	0

16.13 DSP Registers

16.13.1 Code Register

Mnemonic Name: ucode
IM Requirement: DSP_SUPPORT = "yes" (mmsc_cfg.EDSP == 1)
Access Mode: User
CSR Address: 0x801 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
OV	[0]	Overflow flag. It will be set by DSP instructions with a saturated result.	RW	0

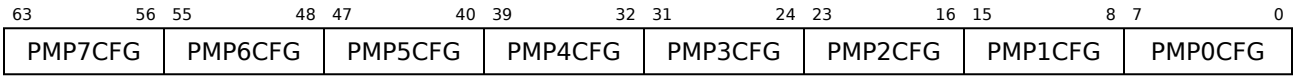
Value	Meaning
0	A saturated result is not generated.
1	A saturated result is generated.

16.14 Physical Memory Protection Unit Configuration & Address Registers

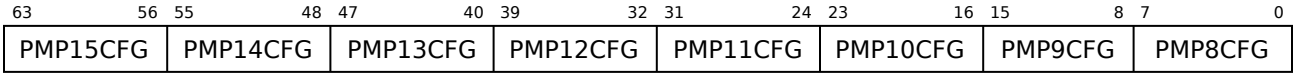
16.14.1 PMP Configuration Registers

Mnemonic Name: pmpcfg0, pmpcfg2, pmpcfg4, pmpcfg6, pmpcfg8, pmpcfg10, pmpcfg12, and pmpcfg14
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x3a0, 0x3a2, 0x3a4, 0x3a6, 0x3a8, 0x3aa, 0x3ac, and 0x3ae (standard read/write)

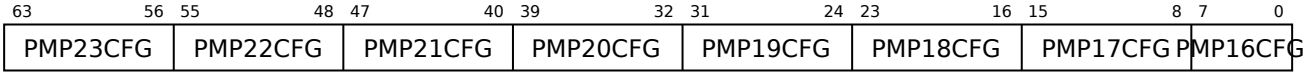
0x3A0



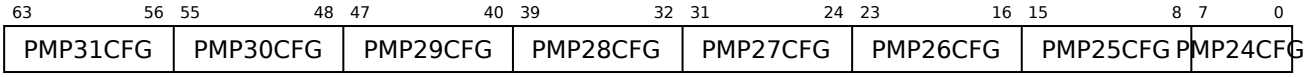
0x3A2



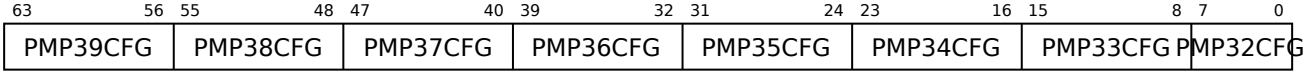
0x3A4



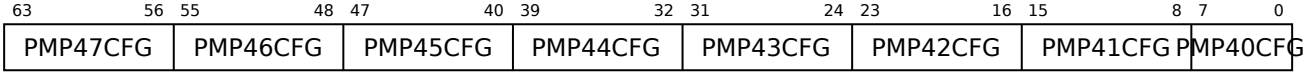
0x3A6



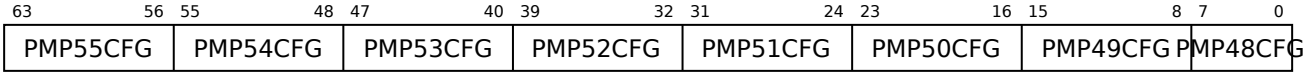
0x3A8



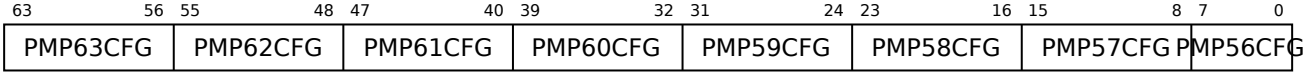
0x3AA



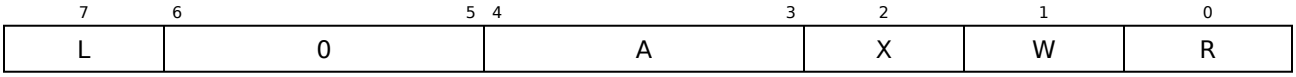
0x3AC



0x3AE



PMP Configuration Format (for PMP$_i$CFG)



Field Name	Bits	Description	Type	Reset
R	[0]	Read access control. **Value** / **Meaning** 0 — Read accesses are not allowed. 1 — Read accesses are allowed.	WARL	0
W	[1]	Write access control. **Value** / **Meaning** 0 — Write accesses are not allowed. 1 — Write accesses are allowed.	WARL	0
X	[2]	Instruction execution control. **Value** / **Meaning** 0 — Instruction execution is not allowed. 1 — Instruction execution is allowed.	WARL	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
A	[4:3]	Address matching mode.	WARL	0

Value	Meaning
0	OFF: Null region
1	TOR: Top of range. For PMP entry 0, it matches any address *A* < `pmpaddr0`. For PMP entry *i*, it matches any address *A*, such that $\texttt{pmpaddr}_i > A >= \texttt{pmpaddr}_{i-1}$. 4-byte ranges are not supported.
2	Reserved. Hardware converts the written value to “OFF”.
3	NAPOT: Naturally aligned power-of-2 region. This mode uses the low-order bits of the associated address register to encode the size of the range. See Table 137 for range encoding from the value of a PMP address register. The minimal size of NAPOT regions must be 4096 bytes.

Continued on next page. . .

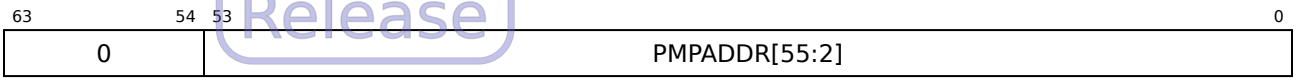
Field Name	Bits	Description	Type	Reset
L	[7]	Write lock and permission enforcement bit for Machine mode. **Value** / **Meaning** 0 — Machine mode writes to PMP entry registers are allowed. R/W/X permissions apply to S and U modes. 1 — For PMP entry *i*, writes to $\texttt{PMP}_i\texttt{CFG}$ and $\texttt{PMPADDR}_i$ are ignored. Additionally, if $\texttt{PMP}_i\texttt{CFG.A}$ is set to TOR, writes to $\texttt{pmpaddr}_{i-1}$ are ignored as well. As for permission enforcement, R/W/X permissions apply to all modes. This bit can only be cleared to 0 with a system reset.	W1S*	0

Note

1. Use the WARL technique to detect the number of PMP entries.
2. The register type of the L field is W1S because only a system reset can clear this bit.
3. The permission combination of R = 0 and W = 1 is reserved.
   a. Writing R and W with this combination results in hardware converting the written value to R = 0 and W = 0.

16.14.2 PMP Address Register

Mnemonic Name: pmpaddr0–pmpaddr63
IM Requirement: Required
Access Mode: Machine
CSR Address: 0x3b0–0x3ef (standard read/write)



Each PMP address register encodes bits 55–2 of a 56-bit physical address, as shown in the register format. Not all physical address bits may be implemented, and so the `pmpaddr` registers are WARL. Different address matching modes also determine the PMP address and memory size.

If PMP configuration field A is set as 1, the address matching mode becomes TOR (Top of range) mode. In TOR mode, PMP entry 0 matches any address of A < pmpaddr0. PMP entry i matches any address of pmpaddr(i-1) <= A < pmpaddr(i), where i ranges from 1 and 63. When i is 1, PMP entry 1 matches any address of pmpaddr0 <= A < pmpaddr1. The start address of PMP entry 1 is bit[55:2] of pmpaddr0, the end address is (bit[55:2]-1) of pmpaddr1. The memory size is (end address - start address + 1) * 4 bytes.

If PMP configuration field A is set as 3, the address matching mode becomes NAPOT (Naturally aligned power-of-2 region) mode. In NAPOT mode, not all physical address bits may be implemented. The encoding is described in Table 137. “a” is an arbitrary value representing one bit value of the physical address.

Table 137: AX46MPV NAPOT Range Encoding in PMP Address and Configuration Registers

Register Content	Match Size(Byte)
aaaa...aaa0	8
aaaa...aa01	16
aaaa...a011	32
...	...
aa01...1111	2^{XLEN}
a011...1111	2^{XLEN+1}
0111...1111	2^{XLEN+2}
1111...1111	2^{XLEN+3} *1

Note

In Andes implementation, the behavior of this register content is the same as that of 0111...1111, and hence the match size is equivalent to 2^{XLEN+2}.

The smallest PMP entry granularity is 4096 bytes and $pmpaddr_i[9:0]$ is hardwired to zero when the mode is OFF or TOR.

Note that under RVV configuration, the minimum granularity will be 4096 bytes, and match sizes smaller than this will be reserved values.

Note

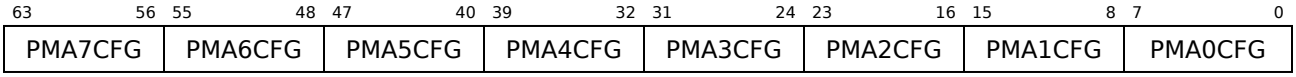
When PMP is used in cacheable memory space, each PMP region must also be naturally aligned to the cache line size. Otherwise, a deliberate load to a cache line may be partially covered by the PMP region. This could bring the full cache line to the Data Cache and allow CCTL operations to access the reset of the cache line that may have different access restrictions.

16.15 Physical Memory Attribute Unit Configuration & Address Registers

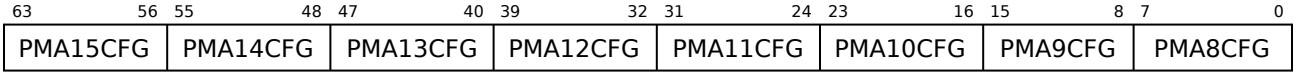
16.15.1 PMA Configuration Registers

Mnemonic Name: pmacfg0 and pmacfg2
IM Requirement: PPMA_SUPPORT
Access Mode: Machine
CSR Address: 0xBC0 and 0xBC2 (standard read/write)

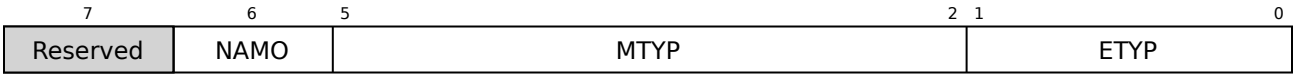
0xBC0



0xBC2



PMA Configuration Format (for PMA$_i$CFG)



Field Name	Bits	Description	Type	Reset
ETYP	[1:0]	Entry address matching mode.	RW	0

Value	Meaning
0	OFF: This PMA entry is disabled.
1	Reserved
2	Reserved
3	NAPOT: Naturally aligned power-of-2 region. The granularity is 4K bytes. This mode uses the low-order bits of the associated address register to encode the size of the range. See Table 138 for range encoding from the value of a PMA address register.

This field is 0 when it is set to 1 or 2.

Continued on next page...

Field Name	Bits	Description	Type	Reset
MTYP	[5:2]	Memory type attribute. This field defines the cacheability and idempotency of memory regions. In the table below, “Device” regions are non-idempotent regions and “Memory” regions are idempotent. Non-cacheable memory regions (type 2 and 3) are also referred to as uncached regions in this document.	RW	0

Value	Meaning
0	Device, Non-bufferable
1	Device, bufferable
2	Memory, Non-cacheable, Non-bufferable
3	Memory, Non-cacheable, Bufferable
4	Reserved. Hardware converts the written value to 3
5	Reserved. Hardware converts the written value to 3
6	Reserved. Hardware converts the written value to 3
7	Reserved. Hardware converts the written value to 3
8	Memory, Write-back, No-allocate
9	Memory, Write-back, Read-allocate
10	Memory, Write-back, Write-allocate
11	Memory, Write-back, Read and Write-allocate
12 – 14	Reserved. Hardware converts the written value to 15.
15	Empty hole, nothing exists. An instruction fetch will generate an instruction access fault. Load instruction accesses will generate load access faults. Store instruction accesses will generate store access faults.

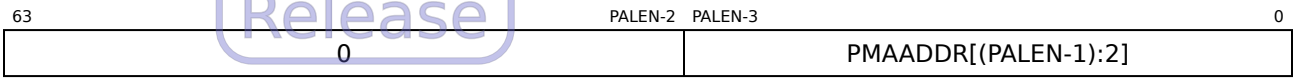
Continued on next page...

Field Name	Bits	Description	Type	Reset
NAMO	[6]	Indicates whether Atomic Memory Operation instructions (including LR/SC) are supported in this region. When this bit is set and an AMO instruction is encountered in this region, an access fault PMA NAMO exception will be generated. Atomic Memory Operation instructions (including LR/SC) to read-no-allocate memory (MTYP = 8 or 10) are not supported. Hardware automatically sets NAMO when writing 8 or 10 to MTYP.	RW	0

Value	Meaning
0	AMO instructions (including LR/SC) are supported in the region.
1	AMO instructions (including LR/SC) are not supported in the region.

16.15.2 PMA Address Register

Mnemonic Name: pmaaddr0–pmaaddr15
IM Requirement: PPMA_SUPPORT
Access Mode: Machine
CSR Address: 0xBD0 to 0xBDf (standard read/write)



Each PMA address register encodes bits (PALEN-1)–2 of a physical address, as shown in the register format. Not all physical address bits may be implemented. The encoding is described in Table 138. "a" in the table represents a one-bit address, with arbitrary values.

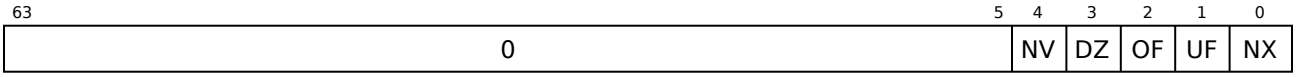
Table 138: AX46MPV NAPOT Range Encoding in PMA Address and Configuration Registers

Register Content	Match Size(Byte)
aaaa...aaaaaaaaaaa	Reserved
...	Reserved
aaaa...aa011111111	Reserved
aaaa...a0111111111	2^{12}
aaaa...01111111111	2^{13}
...	...
aa01...11111111111	2^{XLEN}
a011...11111111111	2^{XLEN+1}
0111...11111111111	2^{XLEN+2}
1111...11111111111	Reserved

16.16 Floating-Point CSRs

16.16.1 Floating-Point Accrued Exception Flags

Mnemonic Name: fflags
IM Requirement: F or D ISA extension (optional)
Access Mode: User
CSR Address: 0x001 (standard read/write)

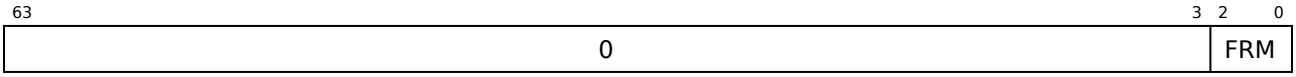


This register is an aliased portion of the `fcsr` register.

Field Name	Bits	Description	Type	Reset
NX	[0]	Inexact exception flag	RW	0
UF	[1]	Underflow exception flag	RW	0
OF	[2]	Overflow exception flag	RW	0
DZ	[3]	Divided by zero flag	RW	0
NV	[4]	Invalid operation flag	RW	0

16.16.2 Floating-Point Rounding Mode

Mnemonic Name: frm
IM Requirement: F or D ISA extension (optional)
Access Mode: User
CSR Address: 0x002 (standard read/write)



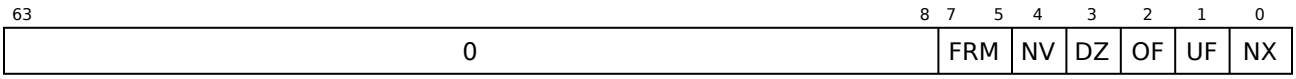
This register is an aliased portion of the `fcsr` register, with the field position shifted to LSB.

Field Name	Bits	Description	Type	Reset
FRM	[2:0]	Floating-point instruction dynamic round mode control:	RW	0

Value	Meaning
0	RNE: Round to nearest, ties to even
1	RTZ: Round towards zero
2	RDN: Round down (towards -infinity)
3	RUP: Round up (towards +infinity)
4	RMM: Round to nearest, ties to max magnitude
5	Reserved
6	Reserved
7	Reserved

16.16.3 Floating-Point Control and Status

Mnemonic Name: fcsr
IM Requirement: F or D ISA extension (optional)
Access Mode: User
CSR Address: 0x003 (standard read/write)



Field Name	Bits	Description	Type	Reset
NX	[0]	Inexact exception flag	RW	0
UF	[1]	Underflow exception flag	RW	0
OF	[2]	Overflow exception flag	RW	0
DZ	[3]	Divided by zero flag	RW	0
NV	[4]	Invalid operation flag	RW	0

Continued on next page...

Field Name	Bits	Description	Type	Reset
FRM	[7:5]	Floating-point instruction dynamic round mode control:	RW	0

Value	Meaning
0	RNE: Round to nearest, ties to even
1	RTZ: Round towards zero
2	RDN: Round down (towards -infinity)
3	RUP: Round up (towards +infinity)
4	RMM: Round to nearest, ties to max magnitude
5	Reserved
6	Reserved
7	Reserved

16.17 User Counter Related CSRs

16.17.1 Cycle Counter

Mnemonic Name: cycle
IM Requirement: misa.U==1
Access Mode: User
CSR Address: 0xc00 (standard read only)

This register is the read-only shadow of `mcycle` register. Any attempt to write to this register in any mode will cause an illegal instruction exception (`mcause` = 2). Additionally, when the `mcounteren.CY` bit is cleared, accessing this register in User mode will trigger an illegal instruction exception (`mcause` = 2).

16.17.2 User Time Register

Mnemonic Name: time
IM Requirement: misa.U==1
Access Mode: User
CSR Address: 0xc01 (software emulation)

This is the register for RDTIME instructions. The `time` CSR is a read-only shadow of the `mtime` register of NCEPLMT210.

16.17.3 Instruction-Retired Counter

Mnemonic Name: instret
IM Requirement: misa.U==1
Access Mode: User
CSR Address: 0xc02 (standard read-only)

This register is the read-only shadow of `minstret` register. Attempts to write to this register in any mode will cause an illegal instruction exception (`mcause` = 2). Additionally, when the `mcounteren.IR` bit is cleared, accessing this register in User mode will trigger an illegal instruction exception (`mcause` = 2).

16.17.4 Performance Monitoring Counter

Mnemonic Name: hpmcounter3–hpmcounter31
IM Requirement: misa.U==1
Access Mode: User
CSR Address: 0xc03 to 0xc1f (standard read only)

These registers are the read-only shadow of `mhpmcounter3-31` registers. Writing to these registers in any mode will cause an illegal instruction exception (`mcause` = 2). Additionally, when the `mcounteren.HPM3-31` bits are cleared, attempts to read these registers in User mode will cause an illegal instruction exception (`mcause` = 2).

16.18 Vector CSRs

16.18.1 Vector Start Position

Mnemonic Name: vstart
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0x008 (standard read write)

The `vstart` register indicates the index of the first element to be executed by a vector instruction. Normally, `vstart` is only written by hardware when a trap occurs during the execution of a vector instruction. The written value indicates the index of the element that caused the trap. The execution can be resumed from the index after the trap is handled. Executing a vector instruction resets `vstart` to zero.

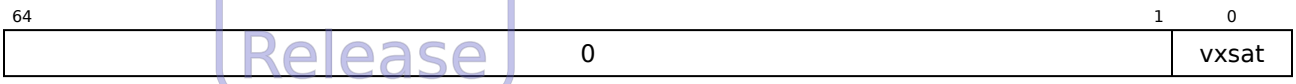
The bit width of `vstart` depends on the configuration of VLEN. Upper bits of `vstart` are hardwired to zero.

Table 139: Bit Width of `vstart`

VLEN	Bit Width of `vstart`
128	7
256	8
512	9
1024	10

16.18.2 Fixed-Point Saturate Flag

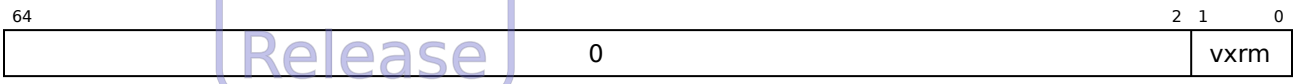
Mnemonic Name: vxsat
IM Requirement: misa.V=1
Access Mode: User
CSR Address: 0x009 (standard read write)



Field Name	Bits	Description	Type	Reset
vxsat	[0]	Indicates a fixed-point instruction has had to saturate an output value	RW	IM

16.18.3 Fixed-Point Rounding Mode

Mnemonic Name: vxrm
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0x00A (standard read-write)



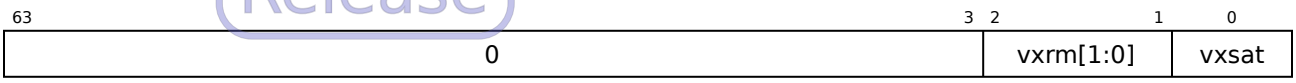
Field Name	Bits	Description	Type	Reset
vxrm	[1:0]	Indicates fixed-point rounding-mode.	RW	IM

Value	Meaning
0	rnu: round-to-nearest-up (add +0.5 LSB)
1	rne: round-to-nearest-even
2	rdn: round-down
3	rod: round-to-odd

16.18.4 Vector Control and Status Register

Mnemonic Name: vcsr
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0x00F (standard read write)

The `vxsat` and `vxrm` registers are mirrored in fields within the `vcsr` register.



16.18.5 Vector Length

Mnemonic Name: vl
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0xC20 (standard read only)

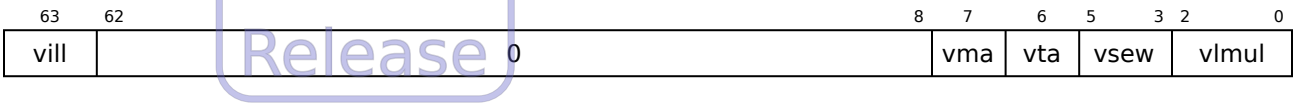
The `vl` register holds an unsigned integer specifying the number of elements to be updated with results from a vector instruction. `vl` can be updated by vsetvli, vsetivli, or vsetvl instructions.

Table 140: Bit Width of `vl`

VLEN	Bit Width of `vl`
128	8
256	9
512	10
1024	11

16.18.6 Vector Data Type Register

Mnemonic Name: vtype
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0xC21 (standard read only)



Field Name	Bits	Description	Type	Reset
vlmul	[2:0]	Vector register group multiplier (LMUL)	RO	0
vsew	[5:3]	Selected element width (SEW)	RO	0
vta	[6]	Vector tail agnostic	RO	0
vma	[7]	Vector mask agnostic	RO	0
vill	[63]	illegal	RO	1

16.18.7 Vector Register Length in Bytes

Mnemonic Name: vlenb
IM Requirement: misa.V==1
Access Mode: User
CSR Address: 0xC22 (standard read only)

This register indicates the length of vector registers in bytes, i.e., VLEN/8.

16.19 Supervisor Timer Related CSRs

16.19.1 Supervisor Timer Compare

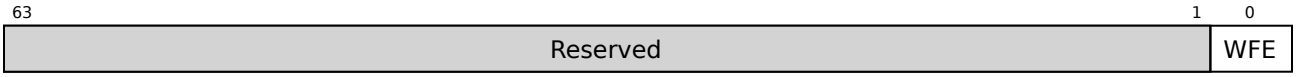
Mnemonic Name: stimecmp
IM Requirement: misa.S==1 && mrvarch_cfg3.Sstc==1
Access Mode: Supervisor
CSR Address: 0x14D (standard read/write)

Field Name	Bits	Description	Type	Reset
STIMECMP	[XLEN-1:0]	Supervisor-mode timer compare value. When the timer value equals this value, a supervisor timer interrupt is generated.	RW	0

16.20 Wait for Event Related CSRs

16.20.1 Wait For Event Control Register

Mnemonic Name: wfe
IM Requirement: mmsc_cfg.ESLEEP == 1
Access Mode: User and above
CSR Address: 0x810 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
WFE	[0]	Enable bit to change the WFI mode into the WFE mode. 0: WFI instructions behave the same as what described in the RISC-V specification. 1: WFI mode is changed to WFE mode, but WFI instructions are still being used.	RW	0

Similar to WFI instructions, executing a WFE instruction will trigger an illegal instruction fault if mstatus.TW is set to 1.

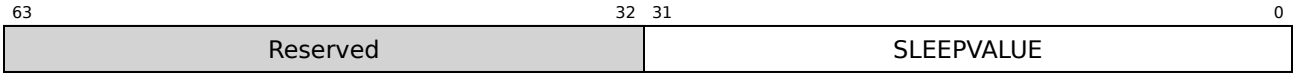
When a WFE instruction is executed and the processor enters sleep mode, it can only be woken up under the following conditions. Maskable interrupts cannot wake up the processor under this situation.

- Receiving an event signal from the “rx_evt” input:
  - If mstatus.MIE is 0, the processor will wake up and resume execution from the instruction after the WFE instruction.
  - If mstatus.MIE is 1, the processor will wake up and either:
    - Start executing from an enabled and pending interrupt service routine, or
    - Resume from the instruction after the WFE instruction.

- Receiving an NMI interrupt:
  - The processor will wake up and start to execute from the first instruction of the NMI handler.
- Receiving a debug signal from the debug module:
  - The processor will wake up and enter the debug mode.

16.20.2 Sleep Value

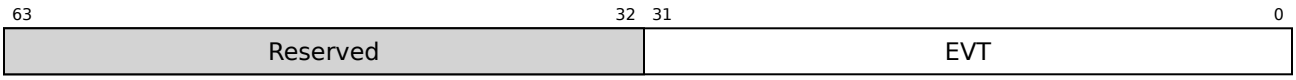
Mnemonic Name: sleepvalue
IM Requirement: mmsc_cfg.ESLEEP == 1
Access Mode: Machine
CSR Address: 0x811 (non-standard read/write)



Field Name	Bits	Description	Type	Reset
SLEEPVALUE	[31:0]	Specifies the sleep duration in clock cycles. Used with the WFE instruction to implement power-saving sleep modes.	RW	0

16.20.3 Transmit Event

Mnemonic Name: txevt
IM Requirement: mmsc_cfg.ESLEEP == 1
Access Mode: Machine
CSR Address: 0x812 (non-standard read/write)



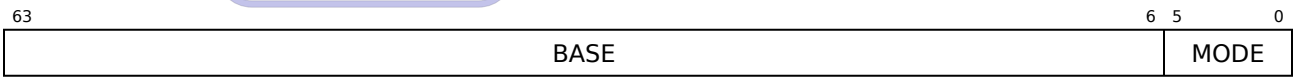
Field Name	Bits	Description	Type	Reset
EVT	[31:0]	Event transmission bits. Setting any bit will wake up cores in WFE mode that are will be self-cleared in the next clock cycle.	RW	0

Value	Meaning
0	No event
1	Send event

16.21 RISC-V Zc* Extension Registers

16.21.1 Table Jump Base Vector and Control Register

Mnemonic Name: jvt
IM Requirement: mrvarch_cfg.ZCMT == 1
Access Mode: User
CSR Address: 0x017 (standard read/write)



Field Name	Bits	Description	Type	Reset
MODE	[5:0]	Indicates the behavior of table jump instructions. **Value** / **Meaning** 0 — Jump table mode. The table jump instruction (`CM.JT` or `CM.JALT`) reads an entry from the jump table and jumps to the address by that entry. 1–63 — Reserved.	RW	0
BASE[63:6]	[63:6]	The base address of the jump table for **Zcmt** table jump instructions. This is a virtual address and must be aligned to a 64-byte boundary. The memory pointed to by `jvt.BASE` is treated as instruction memory for executing table jump instructions, which implies the execution permission.	RW	0

The jump table can only be placed in instruction memories. In AX46MPV, the instruction memories include ILM and the main memory which allows instruction fetches.

16.22 Machine Indirect Register Access

16.22.1 Machine Indirect Register Select

Mnemonic Name: miselect
Access Mode: Machine
CSR Address: 0x350 (standard read/write)

This register selects which `indirect register` is accessed through the `mireg` alias registers.

Field Name	Bits	Description	Type	Reset
INDEX	[XLEN-1:0]	Index of the `indirect register` to be accessed.	RW	0

16.22.2 Machine Indirect Register Alias 2

Mnemonic Name: mireg2
Access Mode: Machine
CSR Address: 0x352 (standard read/write)

This register provides access to the `indirect register` selected by `miselect`.

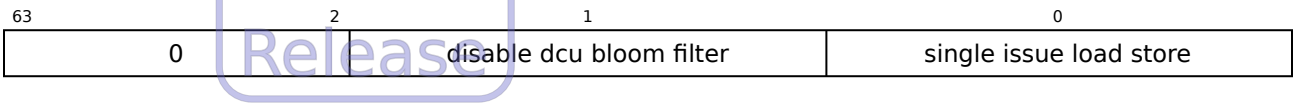
16.22.3 Machine Indirect Register Alias 3

Mnemonic Name: mireg3
Access Mode: Machine
CSR Address: 0x353 (standard read/write)

This register provides access to the `indirect register` selected by `miselect`.

16.22.4 Implementation Defined 0

Mnemonic Name: impd0
Access Mode: Machine
miselect Index: 0x2000 (standard read/write)
Access indirect register: mireg2



Field Name	Bits	Description	Type	Reset
single issue load store	[0]	When this bit is set, a load or store instruction will not be dual issued with another load or store instruction.	RW	0
disable dcu bloom filter	[1]	When this bit is set, the DCU counting bloom filter is disabled and will always return positive results.	RW	0

17 Vector Instruction Implementation

The result of vector instructions may vary depending on the implementation. This chapter introduces detailed implementation of these instructions.

17.1 Vector Tail Agnostic (VTA) and Vector Mask Agnostic (VMA)

AX46MPV ignores the setting of VTA and VMA and executes all vector instructions using the undisturbed policy.

17.2 Vector Start Index CSR (Vstart)

AX46MPV raises an illegal instruction exception when any of the following instructions are executed with a non-zero `vstart` value:

- Streaming port instructions
- Instructions with specific `vstart` behavior in the RISC-V V vector extension, including:
  - Vector reduction instructions
  - vcpop, vfirst, vmsbf, vmsif, vmsof, viota
  - vcompress

17.3 Unordered Floating-Point Sum Reduction

The following pseudo-code shows how the unordered floating-point sum reduction is calculated.

```
unordered_floating_point_sum_reduction (
    vd,    // destination operand
    vs1,   // source operand
    vs2,   // source operand
    vl,    // vector length
    mask,  // mask[i] indicates the element i is inactive (0) or active (1)
    frm,   // rounding mode
    EEW    // effective element width
) {
// temp: temp operands
// DLEN: data path width

    if (vl == 0) return;

    temp[0] = vs1[0];
    for (i=1; i<(DLEN/EEW); i++) {
        temp[i] = (frm == Round Down) ? +0.0 : -0.0;
    }
    for (i=0; i<vl; i++){
        if(mask[i]) {
            temp[i%(DLEN/EEW)] = temp[i%(DLEN/EEW)] + vs2[i];
        }
    }
    for (half=DLEN/2; half>=64; half=half/2) {
        for (i=0; i<half/EEW; i++) {
            temp[i] = temp[int(i/(64/EEW))*(64*2/EEW)+(i%(64/EEW))]
                    + temp[int(i/(64/EEW))*(64*2/EEW)+(i%(64/EEW))+(64/EEW)]
        }
    }

    for (half=32; half>=EEW; half=half/2) {
        for (i=0; i<half/EEW; i++) {
            temp[i] = temp[i*2] + temp[i*2+1];
        }
    }
    vd[0] = temp[0];
}
```

17.4 Vector Load and Store Instructions

The following pseudo-code shows the exception handling of vector load/store instructions.

```
vector_load_store (
    instruction,
    translate_en, // enable for virtual address translation
    vstart,       // the index of the first element to be executed
    evl,          // effective vector length
    mask,         // mask[i]=0 indicates the element i is inactive
                  // mask[i]=1 indicates the element i is active
    ff,           // indicates fault-only-first or not
    NFIELDS       // number of fields:  1 for regular load/store
                  //                    2-8 for segment load/store
) {
    if (instruction == "NDS.VLE4.V") {
        // For NDS.VLE4.V, each operation handles two elements
        for (i=vstart; i<evl; i=i+2) {
            i_lower_element = (i & ~1); // Align i to the nearest lower even number
            i_higher_element = i_lower_element + 1;
            va = calculate_address(i_lower_element, 0);
            if (i == vstart) {
                // If the first element encountered a fault, the vstart will be updated to the index of the first element.
                // For example, if vstart = 1 and it encountered a fault, the vstart will be updated to 1.
                check_abort(translate_en, va, i, ff);
            } else {
                check_abort(translate_en, va, i_lower_element, ff);
            }
        }
    } else {
        for (i=vstart; i<evl; i=i++) {
        if (mask[i] == 0) continue;
        for (f=0; f<NFIELDS; f++) {
            va = calculate_address(i, f);
            check_abort(translate_en, va, i, ff);
        }
    }
    }
}
```

```
check_abort (
    translate_en, // enable for virtual address translation
    va,           // virtual address
    i,            // element index
    ff,           // indicates fault-only-first or not
) {
    if (PMA(va) == ILM || PMA(va) == DLM) {
        abort(ACCESS_FAULT, PMA_ATTRIBUTE_INCONSISTENCY, i, ff);
    } else {
        pa = translate_address(translate_en, va, i, ff);
        // pa is a physical address
        if (pa is not naturally aligned to the size of the element) {
            // use BYTE as PMA lookup size regardless of the size of the ↩
               element
            if (PMA(pa) == DEVICE) {
                abort(ACCESS_FAULT, MISALIGNED_ADDRESS, i, ff);
            }
            else {
                abort(ADDRESS_MISALIGNED, RESERVED, i, ff);
            }
        }
        if (pa has a PMP violation) {
                abort(ACCESS_FAULT, PMP_ACCESS_VIOLATION, i, ff);
        }
        if (PMA(pa) == DEVICE) {
            abort(ACCESS_FAULT, PMA_ATTRIBUTE_INCONSISTENCY, i, ff);
        }
        if ((PMA(pa) != PMA(the first pa)) ||
            (HIT_HVM(pa) != HIT_HVM(the first pa))) {
            // the first pa of the first access of the instruction
            abort(ACCESS_FAULT, PMA_ATTRIBUTE_INCONSISTENCY, i, ff);
        }
        if (PMA(pa) == EMPTY_HOLE) {
            abort(ACCESS_FAULT, PMA_EMPTY_HOLE_ACCESS, i, ff);
        }
    }
}

PMA (
    address
) {
```

```
    if (inside_ilm(address)) {
        return ILM;
    } else if (inside_dlm(address)) {
        return DLM;
    } else if (inside_hvm(address)) {
        return HVM;
    } else if (hit_ppma(address)) {
        return ppma(address);
    } else if (hit_spma(address)) {
        return spma(address);
    } else {
        return default_pma();
    }
}

default_pma {
    if (mcache_ctl.dc_en) {
        return MEMORY_CACHEABLE_WRITE_BACK_READ_WRITE_ALLOCATE;
    } else {
        return MEMORY_NON_CACHEABLE_BUFFERABLE;
    }
}

translate_address (
    translate_en, // enable for address translation and protection
    va,           // virtual address
    i,            // element index
    ff            // fault-only-first
) {
    if (translate_en) {
        if (invalid va) {
            // the bits of va[XLEN-1:VALEN] are not equal to va[VALEN-1]
            abort(PAGE_FAULT, RESERVED, i, ff);
        }
        // fault at virtual address translation process
        if (accessing PTE violates a PMP check) {
            abort(ACCESS_FAULT, PMP_ACCESS_VIOLATION, i, ff);
        }
        if (accessing PTE violates a PMA check) {
            abort(ACCESS_FAULT, PMA_EMPTY_HOLE_ACCESS, i, ff);
        }
        if (accessing PTE encounters a ECC/Parity error) {
```

```
            abort(ACCESS_FAULT, ECC_PARITY_ERROR, i, ff);
        }
        if (accessing PTE encounters a bus error) {
            abort(ACCESS_FAULT, BUS_ERROR, i, ff);
        }
        if (violating PTE permission or page alignment scheme) {
            abort(PAGE_FAULT, RESERVED, i, ff);
        }
        return the pa;
    }
    else {
        return the va;
    }
}

abort (
    cause,         // cause
    detail_cause, // detail cause
    i,             // element index
    ff             // indicates fault-only-first or not
) {
    if (ff && (i != 0)) {
        vl = i;
    }
    else {
        vstart = i;
        raise_exception(cause, detail_cause);
    }
    abort the instruction execution;
}
```

17.5 ACE Streaming Port Instructions

17.5.1 VRF Func6 Definition

Func6[5] indicates whether masked (0) or unmasked (1) vector execution is assumed based on Vector Mask Register v0.

Table 141: VRF Func6 Definition

Func6[4:0]	Type	Description
'b0_0000	Load	Data from an external device is read into the register (Element size is byte).
	Store	Data in the register is written to an external device (Element size is byte).
'b0_0101	Load	Data from an external device is read into the register (Element size is halfword).
	Store	Data in the register is written to an external device (Element size is halfword).
'b0_0110	Load	Data from an external device is read into the register (Element size is word).
	Store	Data in the register is written to an external device (Element size is word).
'b0_0111	Load	Data from an external device is read into the register (Element size is doubleword). Illegal instruction exceptions occur when ELEN is 32.
	Store	Data in the register is written to an external device (Element size is doubleword). Illegal instruction exceptions occur when ELEN is 32.
'b0_1000	Load	Nibbles are zero-extended to 8-bit width in the register.
	Store	Illegal instruction exception (Nibbles cannot be written).
'b1_1000	Load	Nibbles are sign-extended to 8-bit width in the register.
	Store	Illegal instruction exception (Nibbles cannot be written).
Others	Any	Illegal instruction exception

Note

- The VRF transfer type has an EEW encoded directly in the instruction. The corresponding EMUL is calculated as EMUL = (EEW/SEW) * LMUL.

17.5.2 FRF Func6 Definition

Func6[5] is fixed to 1.

Table 142: FRF Func6 Definition

Func6[4:0]	Type	Description
'b0_0001	Load	Scalar 16-bit floating point – FP16 value is read into the register. Illegal instruction exceptions occur when both FP32 and FP64 are not supported.
	Store	Scalar 16-bit floating point – FP16 value is written from the register. Illegal instruction exceptions occur when both FP32 and FP64 are not supported.
'b0_0010	Load	Scalar 32-bit floating point – FP32 value is read into the register. Illegal instruction exceptions occur when both FP32 and FP64 are not supported.
	Store	Scalar 32-bit floating point – FP32 value is written from the register. Illegal instruction exception occur when both FP32 and FP64 are not supported.
'b0_0011	Load	Scalar 64-bit floating point – FP64 value is read into the register. Illegal instruction exceptions occur when ELEN is 32 or FP64 is not supported.
	Store	Scalar 64-bit floating point – FP64 value is written from the register. Illegal instruction exceptions occur when ELEN is 32 or FP64 is not supported.
Others	Any	Illegal instruction exception

17.5.3 XRF Func6 Definition

Func6[5] is fixed to 1.

Table 143: XRF Func6 Definition

Func6[4:0]	Type	Description
'b0_0000	Load	Bytes are zero-extended to XLEN in the register.
	Store	Lower bytes of the register are written.
'b0_0101	Load	Halfwords are zero-extended to XLEN in the register.
	Store	Lower halfwords of the register is written.

Continued on next page...

Table 143: (continued)

Func6[4:0]	Type	Description
'b0_0110	Load	Words are zero-extended to XLEN in the register.
	Store	Lower words of the register is written.
'b0_0111	Load	Doublewords are read into the register. Illegal instruction exceptions occur when XLEN or ELEN is 32.
	Store	Doublewords of the register is written. Illegal instruction exceptions occur when XLEN or ELEN is 32.
'b0_1000	Load	Nibbles (4-bit) are zero-extended to XLEN in the register.
	Store	Illegal instruction exception (nibbles cannot be written to memory).
'b1_0000	Load	Bytes are sign-extended to XLEN in the register.
	Store	Illegal instruction exception (sign extension is meaningless for writes).
'b1_0101	Load	Halfwords are sign-extended to XLEN in the register.
	Store	Illegal instruction exception (sign extension is meaningless for writes).
'b1_0110	Load	Words are sign-extended to XLEN in the register.
	Store	Illegal instruction exception (sign extension is meaningless for writes).
'b1_0111	Load	Doublewords are read into the register. Illegal instruction exceptions occur when XLEN or ELEN is 32.
	Store	Doublewords of the register are written. Illegal instruction exceptions occur when XLEN or ELEN is 32.
'b1_1000	Load	Nibbles (4-bit) are sign-extended to XLEN in the register.
	Store	Illegal instruction exception (nibbles cannot be written to memory).
Others	Any	Illegal instruction exception

17.5.4 Illegal Cases of ACE Streaming Port Instructions

When executing streaming port instructions, AX46MPV raises an illegal instruction exception under one of the following conditions:

- `mmisc_ctl.RVCOMPM` is 1.
- `mmisc_ctl.ACES` is off.
- Undefined values of Func6 definition or illegal instruction defined in Func6 definition.
- Defined values of Func6 definition, and
  - `mstatus.VS` is off.
  - `mstatus.FS` is off, and the transfer type is FRF.

- The transfer type is VRF, and
  - The destination vector register group overlaps the source mask register (v0) when Func6[5] is 0.
  - The source and destination vector register numbers are misaligned appropriately for the vector register group size.
  - EMUL is out of range (EMUL > 8 or EMUL < 1/8)
  - `vstart` with a non-zero value.
  - `vtype.vill` is 1.
  - **The maximum vector element size (ELEN)** is 32, and Func6[4:0] is 0b00111.
- The transfer type is FRF, and
  - **RISC-V Floating-Point Extension** is “none”.
  - **RISC-V Floating-Point Extension** is “single precision”, and Func6[4:0] is 0b00011.
  - **The maximum vector element size (ELEN)** is 32, , and Func6[4:0] is 0b00011.
- The transfer type is XRF, and
  - **The maximum vector element size (ELEN)** is 32, , and Func6[4:0] is 0b00111 or 0b10111.

17.6 MDCAUSE of Illegal Instruction Exception

When a vector instruction is decoded as illegal, the `mdcause` register may be updated to 0, 1, 2, or 3 depending on the following factors:

- When the instruction type is one of the following:
  - Integer or Floating-Point instructions in RISC-V V Standard Extension
  - Integer or Floating-Point instructions in AndeStar V5 Vector Instruction Extensions
  - Integer or Floating-Point instructions in Andes Custom Extension
- Configuration of the following CSR fields:
  - `mstatus.VS`
  - `mstatus.FS`
  - `mmisc_ctl.ACES`
  - `mmisc_ctl.RVCOMPM`

The following pseudo-code shows the updated value of the `mdcause` CSR.

```
// NOTE1: AndeStar V5 Vector Instruction Extensions includes
//            - Andes V5 INT4 Vector Load Extension
//            - Andes V5 Vector BFLOAT16 Conversion Extension
//            - Andes V5 Vector Packed FP16 Extension
//            - Andes V5 Vector Dot Product Extension
//            - Andes V5 Vector Small INT Handling Extension
//            - Andes V5 Vector Quad-Widening Integer Multiply-Add Extension
// NOTE2: For Andes Custom Extension instructions,
//        please contact Andes Technology for further information.
get_mdcause (
    inst,
    vs,         // indicates VS field of mstatus CSR
    fs,         // indicates FS field of mstatus CSR
    rvcompm     // indicates RVCOMPM field of mmisc_ctl CSR
) {
    if (is_vector_fp_inst(inst, rvcompm) && (fs == OFF)) {
        mdcause = 1; // FP disabled exception
    } else if (is_vector_inst(inst, rvcompm) && (vs == OFF)) {
        mdcause = 3; // RVV disabled exception
    } else {
        mdcause = 0; // The actual faulting instruction
                     // is stored in mtval CSR.
```

```
    }
}

is_vector_inst(
    inst,
    rvcompm
) {
   if (inst is defined in RISC-V V Standard Extension) {
        return true;
   } else if (rvcompm == 1) {
        return false;
   } else if (inst is defined in AndeStar V5 Vector Instruction Extensions) {
        return true;
   } else {
        return false;
   }
}

is_vector_fp_inst(
    inst,
    mmisc_ctl_rvcompm
) {
    if (inst is a Vector Floating-Point Instruction
        defined in RISC-V V Standard Extension) {
        return true;
    } else if (rvcompm == 1) {
        return false;
    } else if (inst is a Vector Floating-Point Instruction
                defined in AndeStar V5 Vector Instruction Extensions) {
        return true;
    } else {
        return false;
    }
}
```