Ascalon Architecture Manual

Version 0.10, 04/25/2025: Beta Release r0.5

Table of Contents

Preamble

This document describes the feature implemented in Ascalon CPU Core and Ascalon CPU Cluster.

Chapter 1. Revision History

Revision	Date	Descriptions
0.1a	03/28/2024	Alpha Release
0.1b	08/05/2024	Beta Release
0.1c	10/25/2024	r0.2g Release
0.9a	12/17/2024	LAC Release Candidate a
0.9b	01/12/2025	LAC Release Candidate b
0.9c	01/30/2025	LAC Release Candidate c
0.10	04/25/2022	Beta r0.5 Release

Chapter 2. Overview

Ascalon CPU Cluster is the smallest atomic functional unit for the RISC-V CPU subsystem developed by Tenstorrent. Ascalon CPU Cluster contains the following features to support a full set of CPU functionality.

2.1. Main Component in Ascalon Cluster

Ascalon CPU Cluster contains the following components and interfaces. Ascalon CPU also has a couple of verilog parameters for flexible IP integration.

2.1.1. Components

Ascalon Cluster contains the following components:

- Eight RISC-V CPU Cores (Ascalon)
- Shared Cache
- Cluster Internal Network
- External Interface (CHI.E, AXI5-Lite)
- ACLINT
- Trace Unit (N-Trace)
- Debug Unit
- JTAG Debug Transport Module (DTM) and Debug Module
- Power Management Unit (CPL) and PLL

The details of each component is available in Chapter 3.

2.1.2. Interface

Ascalon Cluster has the following interfaces:

- Reset (cold reset, warm reset, hold signals)
- Clock (Reference Clock, SOC Management Clock, System Fabric Clock, Ascalon Cluster Clock, JTAG)
- Memory Interface (CHI-E, AXI5-Lite)
- RAS Error Interrupt
- Debug (JTAG)
- Configuration Pins (Cluster ID)

Details available in Chapter 4.

2.1.3. Verilog Parameter

Ascalon Cluster IP has the following verilog parameters to help integration:

- Base Address for MMR Register
- Security Source ID Parameters
- Reset Value for Implementation Defined Registers

Detail available in Section 7.1.

2.2. Ascalon Cluster Block Diagram

Figure 1 shows the major components and the connection. This also describes what protocol is used for the connection.

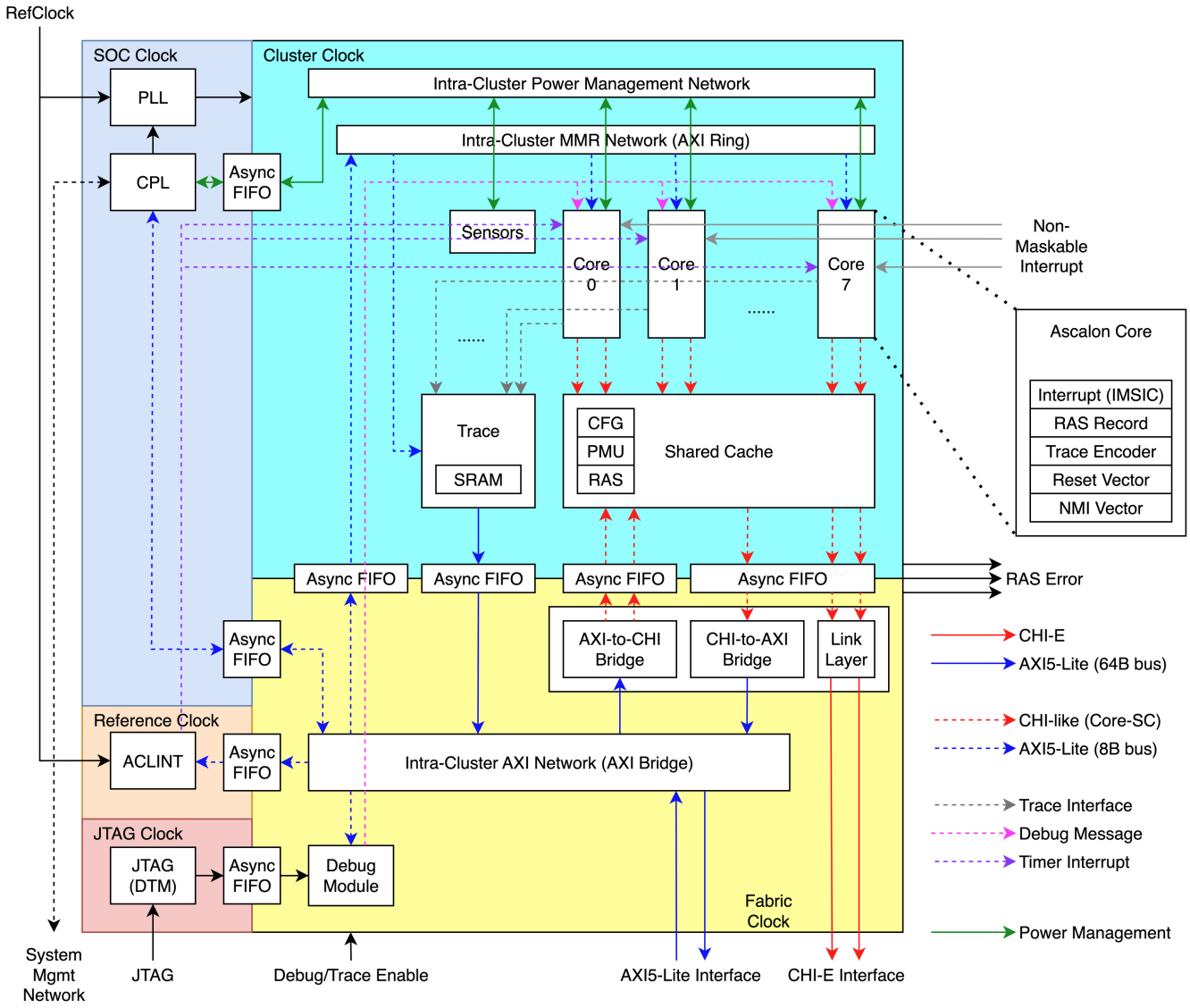

Figure 1. Ascalon Cluster

Chapter 7 discusses how Ascalon IP should be integrated in SoC.

2.3. Reference

This specification refers to the following external specifications:

- RISC-V Profiles (github.com/riscv/riscv-profiles)
- Advanced Interrupt Architecture (github.com/riscv/riscv-aia)
- Trace Specification (github.com/riscv-non-isa/tg-nexus-trace)
- Debug Specification (github.com/riscv/riscv-debug-spec)
- Server SOC spec (github.com/riscv-non-isa/server-soc)
- AMBA CHI (developer.arm.com/documentation/ihi0050/latest/)
- AMBA AXI (developer.arm.com/documentation/ihi0022/e/)

we should archive certain versions when we freeze the specification.

Chapter 3. System Component

3.1. Ascalon CPU (High-Performance RISC-V CPU)

3.1.1. Overview

Ascalon Cluster IP contains up to 8 high-performance RISC-V CPUs. These CPUs support all of the RVA23U64 and RVA23S64 mandatory extensions. For more details, see Chapter 5 and Section 3.9.1.

The feature defined by Tenstorrent is described in the rest of this section.

3.1.2. Core Enable Signal

The CPU core takes 1-bit signal to disable itself. When the core is disabled, the CPU core must not make any external memory requests. Shared cache does not make a snoop message for disabled CPU cores. The MMR address is adjusted to prepare a transparent software view. This signal must be stable before the first instruction fetch from the CPU cores.

The signal is defined as a configuration register in CPL Table 2.

This feature is to manage yeild. The provider can sell SoC by disabling a couple of cores.

3.1.3. Reset Vector

After a reset of the Ascalon CPU, it will start to fetch from the PC specified by the reset vector register. The reset vector register is a Memory Mapped Register which can be modified by a security processor. To prevent overwriting this register, bit-63 of reset vector register is treated as a "Lock" bit.

This Memory Mapped Register is part of a regular Memory Mapped Register (M-level Core MMR). [56:2] of that register represents the reset vector. The MSB is treated as a lock bit. When the lock bit is 1, the hardware ignores the write operation for the corresponding register.

This MMR is defined for each of the CPU cores.

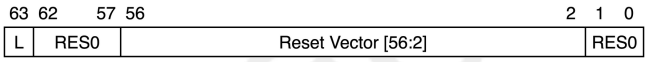

Figure 2. Reset Vector Register

3.1.4. Non-Maskable Interrupt

Each Ascalon CPU core has an external pin to receive a non-maskable interrupt. Ascalon CPU supports Resumable Non-Maskable Interrupt (RNMI) defined in RISC-V specification.

When the pin is asserted to be high, the corresponding Ascalon CPU takes the non-maskable interrupt regardless of its execution status. If the interrupt is reported by other means (e.g., active low), the external logic should convert the incoming interrupt to be active high.

After non-maskable interrupt is taken, the CPU jumps to the non-maskable interrupt vector.

The non-maskable interrupt vector and exception vector are defined in the M-level Core MMR space. [56:2] of these registers represent the instruction addresses after the interrupts are taken. The MSB is treated as a lock bit. When the lock bit is 1, the hardware ignores the write operation for the corresponding register.

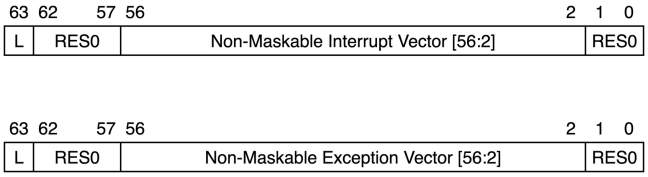

Figure 3. Non-Maskable Interrupt/Exception Vector Register

Non-Maskable Interrupt can be masked under following situations:

- mnstatus disables the interrupt
- RISC-V harts are in Debug Mode
- RISC-V harts are in Patch Mode

This MMR is defined for each of the CPU cores.

3.1.5. Wakeup Timer Register

Each Ascalon CPU core has an MMR to specify the **mtime** or **time** value which will make an interrupt. This information is used by the power management unit to determine the wakeup time.

```
uint64_t get_wtime() // read-only, MMR for each core
{
    if (CPU is not WFI) return UNDEFINED_VALUE;

    // S-level wakeup timer
    swtime = 0xFFF...F (all1)
    if (S-level timer interrupt is enabled)
    {
        swtime = stimecmp;
    }

    // VS-level wakeup timer
    vswtime = 0xFFF...F (all1)
    if (VS-level timer interrupt is enabled)
    {
        vswtime = vstimecmp - htimedelta;
        if(htimedelta > vstimecmp) vswtime = 0;
    }

    // wakeup timer register value
    wtime = min(vswtime, swtime);
    return wtime;
}
```

The power management unit captures the earliest wakeup time when it is determining the sleep level. If the wakeup is close enough, the power management unit does not navigate entering deep sleep mode (e.g., C4). If the wakeup is far away enough, the power management unit will set the earliest wakeup timing to the dedicated timer compare register, so that it can wakeup the cluster in-time.

This MMR is defined for each of the CPU cores. This **wtime** register is a read-only M-level MMR.

3.1.6. Cache Flush Register

The Ascalon CPU has the MMR flush contents from the L1 data cache.

TODO: We need an MMR interface for this feature.

3.1.7. Restriction for Instruction Fetch

Non-Idempotent Memory Access

For instruction fetch for a non-idempotent memory, such as IO memory, the Ascalon CPU may access an entire 64B aligned block. This behavior is explicitly permitted from RISC-V architecture. On top of that, the Ascalon CPU may access the same instruction address multiple times if an unexpected event happens (e.g., interrupt right after the cache fill request went out). If a memory location is sensitive for the number of memory accesses, system software should not permit to execute from such a memory region. If the region is not marked as executable, Ascalon CPU does not generate external memory accesses due to instruction fetch.

3.1.8. Restrictions for Explicit Memory Access (Load/Store operations)

Exception Handling for Page Crossing

A misaligned memory access may cross a page boundary. In such a case, the CPU checks both pages before completing a corresponding operation. If a page identifies a fault (e.g., PMP check failure, permission fault from MMU), the translation for the other page may not happen. If both pieces detect an error, the CPU may report either one of those as the result of corresponding memory access.

in RISC-V architecture terminology, a page crossing memory access makes two accesses. Since these accesses are treated as individual access, the check is performed for each access. For example, the PMP specification requests the CPU to take fault if one "access" hits multiple regions. This "one access" is applicable for both halves of memory accesses generated by a single misaligned operation.

Atomic Memory Operation (AMO)

Ascalon does not support Atomic Memory Operation for an address which is not naturally aligned with memory access size. If an application makes a memory access with a misaligned memory address, the CPU takes an access fault.

IO Memory Access

Ascalon does not support IO memory access for an address which is not naturally aligned with memory access size. When explicit memory requests (e.g., load / store operation) are making misaligned IO memory accesses, the CPU takes an access fault.

Vector Load / Store operation

Vector load / store operation makes memory requests on a per element basis. This means that based on width field or vtype, the CPU may create different access sizes. This may take effect for an IO memory alignment check. Software should take care of the vector element size if the target IO memory is sensitive to the memory access size.

Vector segment load may ignore non-idempotency if the load opration is interrupted by a fault condition. This means that one memory address can be accessed multiple times, even if the address is marked as non-idempotent.

3.1.9. Speculative Attack Mitigation Control

Ascalon supports two hardware-based speculative attack mitigations: Branch Predictor Protection (BPP) and Restricted Speculative Store Bypassing (RSSB). These are for complicating a security context from controlling the branch prediction and the memory dependency prediction of another security context. BPP is controlled by the side-channel predictor protection control CSRs `c_msppc` (at address `0xBF0`) which is accessed from M-mode. RSSB is controlled by 6 side-channel memory protection control CSRs: `c_mspmc` (at address `0xBF1`, by M-mode), `c_sspmc` (at address `0x9F1`, by S/HS-mode), `c_usspmc` (at address `0x8F1`, by U-mode), `c_hsspmc` (at address `0xAF0`, by HS-mode), `c_vsspmc` (at address `0xAF1`, by VS-mode), and `c_vuspmc` (at address `0x8F0`, by VU-mode).

Branch Predictor Protection (BPP)

Ascalon only allows M-mode software to access the side-channel predictor protection CSR `c_msppc`. This CSR is defined for each CPU core as follows:

Figure 4. Side-channel Predictor Protection Register

The entire BPP feature is enabled by default, and disabled only if `c_msppc.DISABLE` is set. By enabling the feature, branch predictor content is scrambled based on the hardware-managed, CPU context-aware (e.g., privilege mode) secret key. This feature prevents a process from consuming potentially maliciously injected branch predictor content controlled by other context. Setting this bit also results in the flush of other branch predictor structures, such as return address stack. The actual details are described in the micro-architecture specification.

This feature also allows the CPU to scrap branch predictor contents periodically. `c_msppc.PERIOD` controls the interval for scrapping the branch predictor context. Periodical refresh of the random number is enablled by setting `c_msppc.REKEY`, which is unset by default.

`c_msppc.FLBHR` is used against speculative side-channel attacks exploiting the branch history register (`ghist`). This bit is by default unset, thus the protection for `ghist` is disabled.

The rest of the bits in `c_msppc[43:3]` are reserved.

Restricted Speculative Store Bypassing (RSSB)

In Ascalon, every privilege mode can control RSSB by using the specific side-channel memory protection configuration CSR that is within each privilege mode. As shown below, there are six software CSRs (assuming with virtualization support): `c_mspmc` for M-mode, `c_sspmc` for S-mode, `c_uspmc` for U-mode, `c_hsspmc` for HS-mode, `c_vsspmc` for VS-mode, and `c_usspmc` for VU-mode. The CSR that is actually accessed is determined collectively by the CSR specified in the software, and the current privilege mode (e.g., `c_vuspmc` is used when a software runs within VU-mode and tries to access `c_uspmc`).

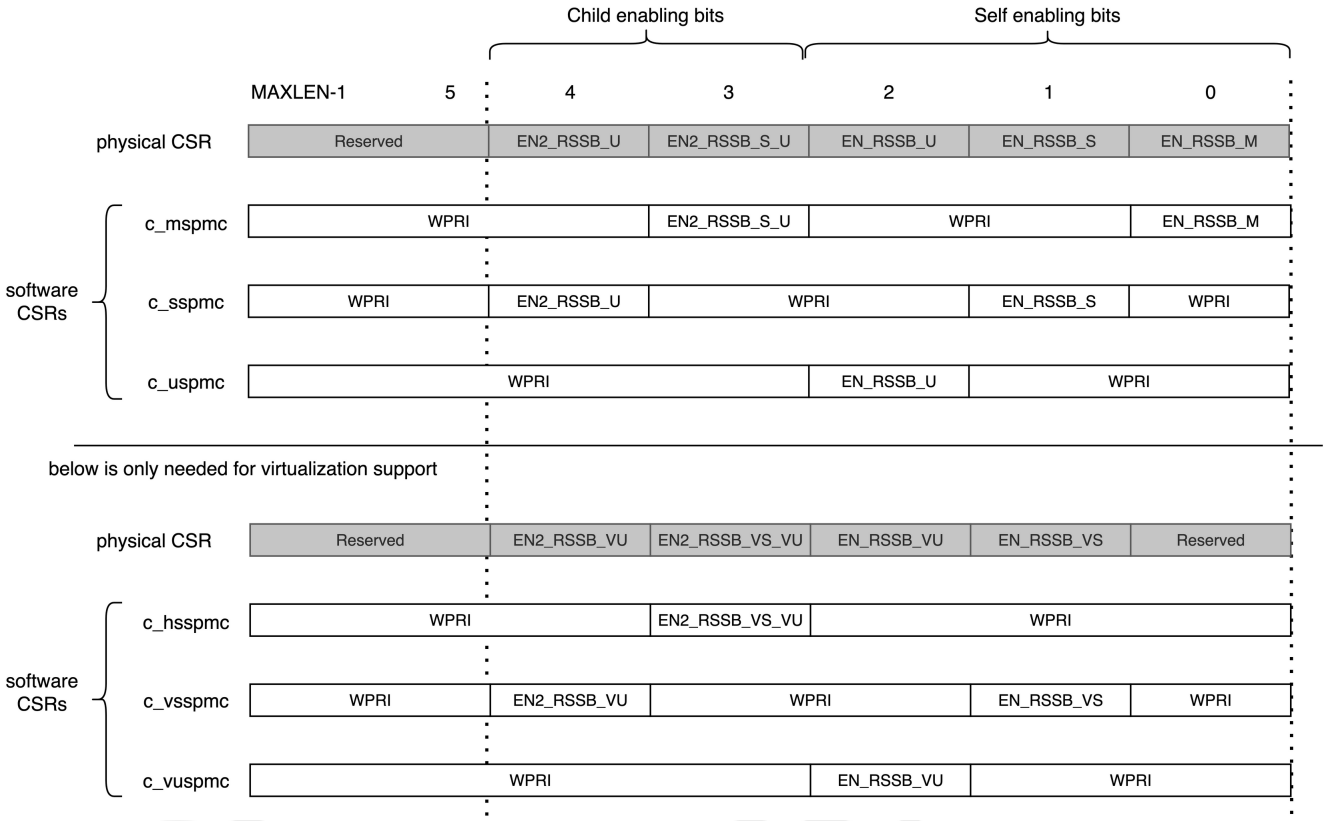

Figure 5. Side-channel Memory Protection CSRs

Each software has a maximum of two useful bits. One is the "self enabling bit" (`EN_RSSB_*`), which enables the RSSB for the current privilege mode (note that since S and HS-mode are the same, `EN_S` in `c_sspmc` controls the RSSB self-enabling of S/HS mode). The other is the "child enabling bit" (`EN2_RSSB_*`), which additionally enables/disables RSSB for some privilege levels lower than the current privilege. Therefore, the RSSB of a given privilege mode is enabled if the self enabling bit is set in that privilege mode's CSR, or the child enabling bit of a higher privilege mode which can control the RSSB of the current privilege mode is also set. This allows higher-level privileges to enforce RSSB in lower privilege modes without the lower level's consent. Additionally, higher-privilege levels can allow lower levels to make their own decisions for using RSSB by setting the child enabling bit to 0.

- M-mode enables RSSB = `c_mspmc.EN_RSSB_M`
- S-mode enables RSSB = `c_sspmc.EN_RSSB_S` || `c_mspmc.EN2_RSSB_S_U`
- U-mode enables RSSB = `c_uspmc.EN_RSSB_U` || `c_mspmc.EN2_RSSB_S_U` || `c_spmc.EN2_RSSB_U`
- VS-mode enables RSSB = `c_vsspmc.EN_RSSB_VS` || `c_hsspmc.EN2_RSSB_VS_VU`
- VU-mode enables RSSB = `c_vuspmc.EN_RSSB_VU` || `c_hsspmc.EN2_RSSB_VS_VU` || `c_vspmc.EN2_RSSB_VU`

The reset values of all `EN_RSSB_*` and `EN2_RSSB_*` are 0, except `EN_RSSB_M`, meaning RSSB is enabled in machine mode by default.

Despite 6 software CSRs, Ascalon implements 2 physical CSRs, as shown in the above figure. One CSR covers all configuration bits in M, S, and U mode, and the other covers the configurations in HS, VS, and VU mode. Each software CSR is therefore a restricted view of the hardware CSR.

3.1.10. Instruction Patching

CPU can take trap on the specified instruction opcode and enter special execution mode named patch mode. In patch mode, CPU may execute trapped instruction with slightly different manner to mitigate bug severity.

3.1.11. Physical Memory Attribute (PMA)

Physical Memory Attribute is an Implementation Defined structure in RISC-V specification. Ascalon CPU supports per-CPU PMA CSR registers to support PMA features.

The details of this are described in [physical_memory_attribute]

3.1.12. Static TEE Support (PMP-base security feature)

Ascalon supports a memory isolation technology to mitigate the security risk from incorrect HW / SW implementation. `matp` is a CSR to specify the security world ID.

The details of this are described in Section 9.2.

3.1.13. RAS Register

Ascalon CPU tracks correctable / uncorrectable errors for the following structures. The error is reported in the RAS register compatible with RERI specification.

- External Source
- Instruction Cache Tag Array
- Instruction Cache Data Array
- Instruction Cache Compressed Array
- Data Cache Tag Array
- Data Cache Data Array
- Level-2 Translation Buffer / Translation Cache

A complete list of the error events and the structures associated with them is provided in the Ascalon uArch specification.

The RERI specification allows each error category (i.e. CE, UED, UEC) to be configured to generate one of the following signals:

- None
- Low Priority
- High Priority
- Platform Specific

The RERI specification has no description of the meaning and intent of these signals beyond what is implied by the name. Also, these are configured on a per error record basis and may be either unique to each error record or grouped according to similarities between error record events and error categories.

These signals are asserted when the RAS error record is valid. When the error information is cleared by the RAS Agent(s), the signal is deasserted within a cycle. The TT RAS Unit/Error Bank will have the OR reduce these signals according to their names, in order to produce a final Low, High, and Platform signal output. Additional OR reduction is done at various IPs to produce one or a few sets of Low, High, and Platform signals. These signals are then connected to an APLIC to support hart based RAS agents, and then exported to the SoP interface to support ucontroller or system qualified and determined RAS agents.

These signals may be connected to a core for hart local interrupts (e.g. CE delivered to the hart associated with the error). This is in addition to the error event related traps/faults that may be generated.

The details of this are described in Section 3.7.

3.1.14. IMSIC

Incoming MSI Controller (IMSIC) is an external interrupt controller for MSI. IMSIC is attached to each Ascalon CPU.

M-level interrupt files and S-level interrupt files support 255 interrupt identifiers.

Ascalon CPU supports 5 Guest Supervisor Interrupt files (VS-level interrupt files). Each VS-level interrupt file supports 63 interrupt identifiers.

3.1.15. Trace Support

Ascalon Cluster supports N-Trace as the trace mechanism.

To support N-Trace, Ascalon CPU core employs one trace encoder. The trace encoder is controlled by the Memory Mapped Registers defined in the N-Trace specification.

The detail of the entire trace mechanism are described in Section 3.4.

3.2. Shared Cache

Ascalon CPU Cluster deploys a 12MB second level cache which is shared from all Ascalon CPU cores.

The shared cache has 2 different master interfaces and 1 slave port from intra-cluster network.

One master interface is for the coherent network connected via CHI protocol, and the other is for the non-coherent network connected via AXI protocol. Both are used by the software running on Ascalon CPU cores. All cacheable memory traffics are coming from the CHI interface. All non-cacheable memory traffics are coming from the AXI interface.

Shared cache has a subordinate port for RAS, Performance Counter, and Control / Configuration.

3.2.1. Cache Flush

When the Ascalon Cluster enters a C4 power state, all cache lines within the Ascalon Cluster are expected to be invalidated. Shared cache has the MMR to flush the cache structure.

This cache flush mechanism is controlled by the `sc_flush_cfg` MMR register.

Field	Description	RW	Reset	Description
[0]	Cache Flush	RW	0	Start Shared Cache Flush (CBO.FLUSH for all cache lines)
[3:1]	Cache Status	R	0	Shared Cache Status (Normal, Flushing, Idle, Suspended, Error)

Table 1. Shared Cache Flush Register (`sc_flush_cfg`)

3.2.2. Cache Way Partitioning (QoS)

RISC-V CBQRI defines capacity QOS registers.

- RCID: 16
- NCBLK : 24
- only allocation support (no mon support)

Based on CBQRI specification, shared cache selects a victim from a subset of physical ways. To select corresponding QoS information, CPU core should attach `Effective RCID` in the RSVDC[3:0] field (Section 3.6).

When shared cache receives any allocation request, it picks a victim from the enabled way. If shared cache failed to find a victim (e.g., the available way is disabled), shared cache does not allocate the incoming cache lines.

Each shared cache slice has 32-entry cache partition tables. The resource allocation table holds the assigned cache partition information for each QoS ID. Shared cache can be configured to ignore some part of the QoS ID to support more RCID from the partition table.

3.2.3. RAS Register

Shared cache tracks correctable / uncorrectable errors for the following components:

- External Errors
- Shared Cache Control Structures (Cache Tag Array, Snoop Filters)
- Shared Cache Data Structures (Cache Data Array)

A complete list of the error events and the structures associated with them is provided in the Ascalon uArch specification.

The error events are recorded in MMR error record registers as defined by the RERI Error Bank. There is one RAS Unit/Error Bank per shared cache slice, and each RAS Unit/Error Bank has one or more error records as determined by the RAS Unit/Error Bank N_RECORDS parameter.

The RERI specification allows each error category (i.e. CE, UED, UEC) to be configured to generate on of the following signals:

- None
- Low Priority
- High Priority
- Platform Specific

The RERI specification has no description of the meaning and intent of these signals beyond what is implied by the name. Also, these are configured on a per error record basis and may be either unique to each error record or grouped according to similarities between error record events and error categories.

These signals are asserted when the RAS error record is valid. When the error information is cleared by the RAS Agent(s), the signal is deasserted within a cycle. The TT RAS Unit/Error Bank will have the OR reduce these signals according to their names, in order to produce a final Low, High, and Platform signal output. Additional OR reduction is done at various IPs to produce one or a few sets of Low, High, and Platform signals. These signals are then connected to an APLIC to support hart based RAS agents, and then exported to the SoP interface to support ucontroller or system qualified and determined RAS agents.

Section 3.7 covers more detail.

3.2.4. Performance Counter

Shared cache deploys a total of 128 performance counters. 128 counters are distributed to each shared cache slice. Each shared cache slice counts up to 8 events at one time.

Each performance counter has 2 MMR registers. One is an event ID register (`sc_hpm_event*`) and the other is an event count register (`sc_hpm_count*`). Each event count register counts the event specified by the ID register.

The address of performance counters are listed in Section 6.2.5.2.3. The event ID register is shown in Table 36.

3.2.5. Static TEE Support (PMP-base security feature)

The system treats the MSB (bit-55) of Physical Address as the secure world ID. Shared cache receives this information from the NS bit of CHI interface.

3.3. ACLINT

3.3.1. Timer Interrupt Support

ACLINT is a client level interrupt controller. Ascalon Cluster implements one ACLINT for each cluster. ACLINT only supports machine timer interrupt, since Ascalon Cluster process software is interrupted by MSI (external interrupt).

Each ACLINT employs one **`mtime`** and nine **`mtimecmp*`**. The **`mtime`** register is running at a 100MHz reference clock. ACLINT has dedicated machine timer interrupt signals for each CPU core in the cluster.

Although Ascalon Cluster employs only eight Ascalon CPUs, the cluster supports a 9th **`mtimecmp`** for CPL. CPL sets **`mtimecmp`** when it decides to shut-off the cluster PLL. By setting **`mtimecmp`**, ACLINT makes an interrupt when the cluster is taking a timer interrupt. The CPU's **`wtime`** register is used to get wakeup time efficiently.

The reference clock is mostly used for this unit.

Since clock frequency is always 100MHz, the system can choose another clock source.

3.3.2. Timer Sync

When **`mtime`** is updated or **`time`** CSRs are likely not synchronized, CPL firmware may write a dummy value to the **`timesync`** register to enforce a timer sync between the **`mtime`** and **`time`** CSRs. This process enforces a copy of **`mtime`** to **`time`** in order to put these timers in sync.

3.4. Trace

3.4.1. Overview

Ascalon Cluster supports N-Trace. Each cluster has one Trace Funnel and one Trace Sink. Each Ascalon CPU core has a Trace Encoder.

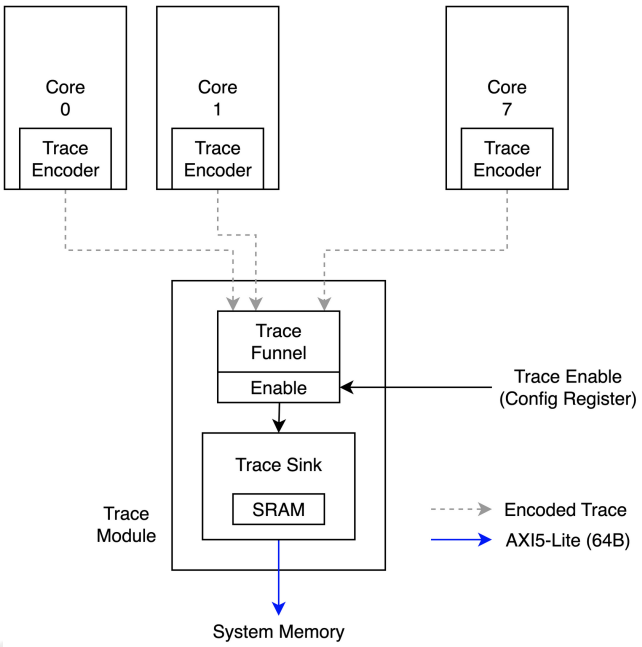

Figure 6. Trace Topology

3.4.2. Trace Disable

For security reason, trace logic has disabling logic driven by a `config1_trace_disable` register in the CPL. When this bit is asserted, trace funnel output is made to be disabled. Internally, `trFunnelEnable` is made to be 0.

This signal is expected to be controlled by a security processor in a production system.

3.4.3. Trace Encoder (in CPU Core)

Each Ascalon CPU core has a Trace Encoder to create trace data saved in the system memory.

3.4.4. Trace Module (Trace Funnel and Trace Sink)

Trace module employs Trace Funnel and Trace Sink.

Trace Funnel combines multiple trace streams coming from the CPU cores in the cluster. Trace Sink dumps the incoming trace to the SRAM in the trace module (SRAM mode) or the system memory (SMEM mode). These components are programmed by Memory Mapped Registers, defined in the N-trace specification.

Trace Sink has 32KB SRAM for SRAM mode.

3.5. Debug

Ascalon follows Debug Specification.

The Debug Module communicates with the JTAG Debug Transport Module (DTM) to realize external debug capability for Ascalon CPUs. The Debug Module also has MMR debug the software without using JTAG interface.

3.5.1. JTAG Debug Transport Module (DTM)

JTAG DTM will pass commands from JTAG to the Debug Module in the cluster. As described in the RISC-V Debug Specification, JTAG DTM converts JTAG access to internal DMI access, then sends it to a connected Debug Module. When the DM has no clock, and DTM attempts to access the DM, all access request from JTAG interface to DM (opcode = 0x11) will result in `failed`.

bit-15 of `dtmcs` is treated as debug active information. This is a WARL field to activate the Debug Module. While this signal is 1, the cluster does not shut-off PLL.

3.5.2. Debug Module (DM)

Debug Module receives debug abstract commands from MMR or JTAG via DMI, then sends the appropriate debug message to CPU.

One Debug Module is shared from all CPU cores in the corresponding Ascalon Cluster. The Debug Module has a set of internal registers that are defined in RISC-V Debug Specification. All internal registers have their own memory addresses so that other RISC-V harts can have access to the Debug Module.

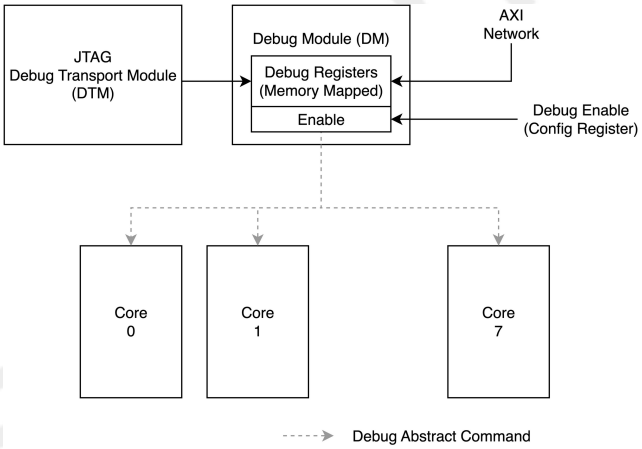

Figure 7. Debug Module Topology

Debug Enable

Debug Module has a debug enable signal supplied from the CPL config register. (`config1_debug_disable[1:0]`). The DM receives 2 bits from the CPL register. Each bit is associated with a DMI from the DTM (JTAG) and also a DMI from the system network (AXI Bridge). When the corresponding bit is asserted, DMI access is suppressed and returns an error response without taking any transaction to DM. The register definition is available in Table 2.

[1:0]	Description
`00`	Support both enabled at same time. Same cycle collision cases error packet.
`01`	AXI based debug only
`10`	JTAG based debug only
`11`	Feature disabled

Table 2. Debug Disable Signal (`config1_debug_disable[1:0]`)

Supported Messages

Ascalon Debug Module supports the following features. This satisfies the minimum set of debug features:

- Allow any individual hart to be halted and resumed.
- Provide status on which harts are halted.
- Provide abstract read and write access to a halted hart's GPRs.
- Provide access to a reset signal that allows debugging from the very first instruction after reset.
- Provide a Program Buffer to force the hart to execute arbitrary instructions.
- Support at least one halt group and at least one resume group, besides group 0.
- Hardwire `relaxedpriv` to be 0.
- Support `stopcount` control.

Debug PMA (Debug Execution Memory Protection)

The CPU supports a hardwired PMA register to permit debug as the very first instruction after reset. Debug PMA protects the region associated with Debug Mode Execution Memory (4KB) in non-Debug Mode. Debug PMA is flipped to permit access only for the Debug Execution Memory region in Debug Mode.

Debug Execution Memory Protection from Intra-Cluster Network

Intra-Cluster AXI Network embeds request source information in AXI ID. DM uses AXI ID to identify the source of the memory access request. If the request is NOT coming from a local CPU (the request comes from Trace, APLIC, AXI subordinate port, etc...), DM ignores the request (e.g., return 0 for read, ignore write operation) for data transfer memory access.

Instruction Sequence in Debug ROM

After a CPU enters Debug Mode, the CPU starts communicating with the debug module by executing instructions saved in Debug ROM. This communication is mainly done by mailbox MMR register. CPU enters park loop to check instructions from the debug module. The mailbox is assiged to the offset 0x100, so that HW can easily prevent access from non-Debug Mode software.

Abstract Command

Abstract command is realized by regular instruction fetch. If DM is programmed to execute an abstract command, CPU jumps to address offset X to execute the instruction sequence prepared by DM. Since Ascalon CPU supports only GPR accesses, load/store operations from/to DM data memory space are executed. This instruction sequence must be executed in Debug ROM Mode.

The base address is specified by cluster MMR base address Verilog parameters and Cluster ID. Data transfer for GPR read / write is copied to / from the data registers that are mapped to address range 0x000 - 0x3FF.

Program Buffer

Program Buffer access is done by Memory Mapped Registers. The base address is specified by cluster MMR base address Verilog parameters and Cluster ID. Since the Program Buffer is followed by c.ebreak space, CPU will execute c.ebreak operations after it completes the execution from Program Buffer.

Memory Mapped Registers in Debug Module

All registers in the Debug Module are accessible via both DMI from JTAG DTM and AXI network (standard non-coherent network). Each register has a dedicated 4B address space. Registers can be accessed by any naturally aligned power of two size.

If the registers are accessed by anything larger than 4B, the internal logic cracks the requests into multiple 4B operations. The order of the processing is UNPREDICTABLE, the implementation may process it from any location covered by the request. If the order is important, the MMR access needs to be done by an explicit 4B operation.

Since there are 0x0 - 0x7f register address spaces, a total of 512B is assigned. When access is initiated by AXI network, DMI response is mapped to corresponding AXI response.

DMI Response	AXI Response	Comment
success	OKAY	Read / Write operation accepted appropriately
fail	DECERR	Read / Write operation failed
busy	DECERR	Response should be blocked by BUSY Signal

Table 3. DMI Response and AXI Response Mapping

This interface returns DECERR if DM is still processing previously submitted commands (this is required to avoid potential deadlock). Software can check DM status by accessing the `custom_dm_busy` register, to avoid receiving DECERR from DM. The bottom 8-bit of this register covers implementation specific reasons why the module is busy. If `custom_dm_busy` returns 0, DM is not processing any pending operations.

Unless debugging processes submit operations to DM, DM won't enter a busy state. This means that DM access is done by single CPU. Checking `custom_dm_busy` *followed by an actual DMI command won't cause a problem.*

The custom register returns a value regardless of DM status. Address mapping of this register is specified in Section 3.9.1.

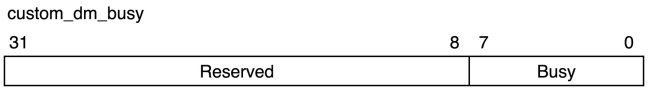

Figure 8. Debug Module Custom Register

This feature was requested from multiple people, including software.

NDM Reset

The Debug Module has the capability to reset the logic in the system. This reset mechanism applies a warm reset for all Ascalon Clusters in the system. This reset signal is also propagated to other components in the system.

To communicate with external system debug components, DM supports a handshake mechanism. For handshake between DM in the cluster and the rest of the world, Ascalon Cluster has two interface signals (`ndmdebug_request`, `ndmdebug_process`). `ndmreset_request` is the output signal, `ndmdebug_process` is the input signal for the cluster.

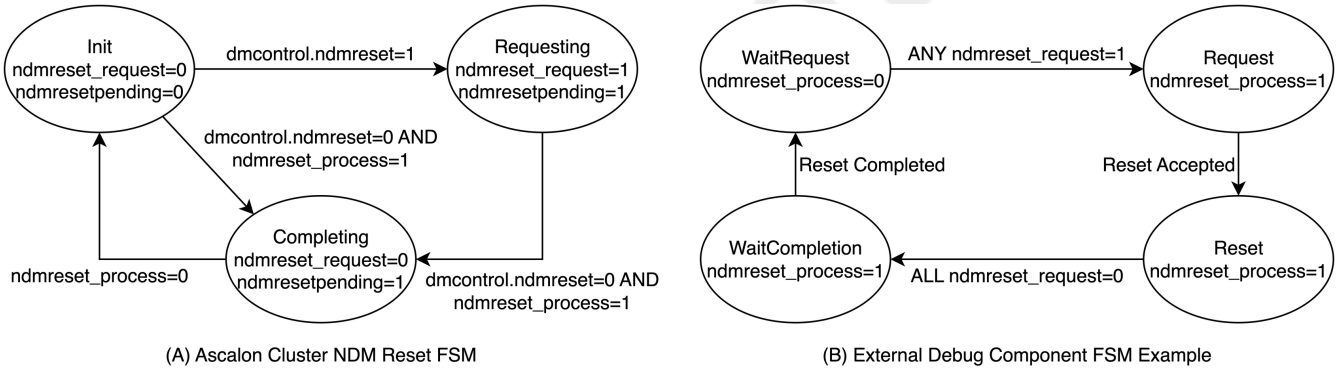

Figure 9. NDM Reset FSM

DM maintains FSM to process NDM reset (Figure 9). After this reset, the FSM is initialized to init state. When `(dmcontrol.ndmreset && !ndmreset_process)` is 1, `ndmreset_request` is asserted to be 1. DM keeps the request signal at 1 until the system asserts `(ndmreset_process && !dmcontrol.ndmreset)` to be 1. An external system debug component should keep `ndmreset_process` at 1 until `ndmreset_request` is deasserted.

Once the system decides to apply ndmreset to the entire system, the external agent resets the rest of the system by asserting reset pins or reset MMR in CPL. To apply the reset appropriately, the external agent enforces an activation of PLLs in the cluster with the lowest clock frequency. While `ndmreset_process` is asserted, all CPU cores in the cluster stop fetching instructions.

`ndmresetpending` signal in `dmstatus` is kept high when either `ndmreset_request` or `ndmreset_process` is high.

DM Reset (`dmactive`)

Debug Module has the capability of resetting DM. This is supported by writing 0 to the `dmactive` register.

DTM Reset (`dtmhardreset` and `dmireset`)

DTM has its own reset mechanism. This feature is supported by `dtmhardreset` and `dmireset`. These reset signals reset DTM or DMI (Debug Module Interface), respectively.

Section 4.1 describes other reset operations supported in the Ascalon Cluster.

3.6. QoS

Ascalon Cluster supports RISC-V CBQRI, in order to support QoS features for shared resources. To specify QoS behavior, memory access requests from the CPU core attach a QoS ID for the RSVDC field of the REQ channel.

3.6.1. QoS Identifier

The Ascalon CPU core deploys the `srmcfg` register as defined in SSQOSID, which consists of two fields: RCID and MCID. According to the specification, each of these fields has a 12-bit field.

To specify the Quality of Service (QoS) group in the system, the CPU is expected to send both RCID and MCID for shared resource accesses. However, the additional sideband for MCID transfer is not negligible. To mitigate the hardware overhead from MCID

support, Ascalon applies the RCID-prefixed mode with P=0. QoS ID is equivalent to RCID[3:0] in Ascalon. The ID is attached to RSVDC[3:0] of CHI REQ packet for each memory request, specifying the corresponding QoS group.

Figure 10 shows how it is mapped to a 4-bit field.

Figure 10. Quality of Service ID (QOSID)

QoS ID (RCID) is not attached to each cache line. When cache eviction happens (e.g., writeback of dirty line), the RCID of the request that triggered the eviction is attached to the eviction command (e.g., WriteBackFull). If the RCID of the triggered transaction is not available (e.g., cache flush operation for cluster shutdown), RCID=0 is attached to a memory request.

3.6.2. Transferring QoS ID

In coherent network (CHI network in cluster), the QoS ID from the CPU core is attached to RSVDC[3:0] of REQ channel. This QoS ID is propagated up to a cluster interface if the memory request is coherent access. For non-coherent network (AXI network in cluster), the RSVDC field is remapped to AxUSER bits. Table 4 shows how QoS ID is mapped to AxUSER bit.

Source	Destination	Attached Field
Intra-Cluster AXI Network	Cluster AXI Manager Port	AxUSER[7:4]
Cluster AXI Subordinate Port	Intra-Cluster AXI Network	AxUSER[7:4]
CHI-to-AXI bridge	Intra-Cluster AXI Network	AxUSER[7:4]
Intra-Cluster AXI Network	AXI-to-CHI bridge	AxUSER[7:4]
Other Manager Agent	Intra-Cluster AXI Network	N/A (0)
Intra-Cluster AXI Network	Other Subordinate Agents	N/A (0)

Table 4. QoS ID (RCID[3:0]) Mapping for Various Interface

For an external interface, `AxUSER[3:0]` is used for source ID information.

3.6.3. QoS Support in Shared Cache

Shared cache in Ascalon Cluster supports QoS. Shared cache slice has a partition table to specify cache partitioning information. The system can select one index scheme by using configuration registers in shared cache slices. RCID and AT are used to specify partition information.

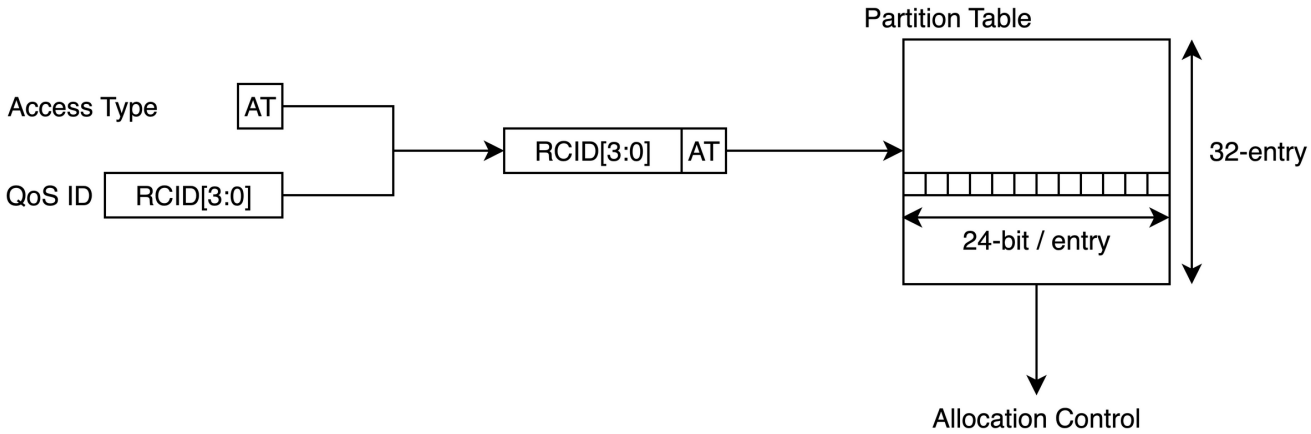

Figure 11. Ascalon Cluster Shared Cache QoS Scheme

3.7. RAS

3.7.1. RAS Register

In Ascalon Cluster, all components employing SRAM (e.g., CPU core, Shared Cache, and CPL) support the RAS error reporting register complying with RERI specification. Each RAS event (e.g., parity error from instruction cache tag) has its own dedicated error register to record the error information.

When specified RAS event is detected, the register notify the interrupt to external components. The RAS interrupt signal is

connected to external pins. Each RAS register supports a dummy error reporting mechanism as specified in RERI.

3.7.2. Supported Feature in RERI

Each RAS register supports the following features from the RERI specification.

Register Name	Field	Detail
`valid_summary`	`sv`	Supported (should be 1)
`control_i`	`ces`	Supported for all structures
`control_i`	`uecs`	Supported for SECDED structures
`control_i`	`ueds`	Optionally supported
`control_i`	`cece`	Supported
`control_i`	`eid`	Supported, update counter every 65536 cycles
`addr_i`	-	Physical address associated with corresponding error
`info_i`	-	Physical location (e.g., cache index, way). Enable bit for **`addr_i`**.
`suppl_info_i`	-	Optional
`timestamp_i`	-	Optional, **`time`** CSR is used as a timestamp for CPU cores

Table 5. Feature Support List for RERI

When the error register satisfies certain conditions specified by **`control_i`**, the RAS register reports the error to the system. Errors reported in the CPU core can be reported by local RAS errors or by wired interrupt signal. Wired interrupt signals are asserted when **`ces`**, **`ueds`**, or **`uecs`** are marked as 11. Errors reported outside of the CPU core (e.g., Shared Cache) are reported by a wired interrupt signal. Wired interrupt signal is kept high until the RAS register is reset by software.

Each register bank has a 3-bit wired interrupt signal depending on the detected error type (High-Priority, Low-Priority, Platform-Specific). The wired interrupts from register banks of the same category (e.g., Core, Shared Cache, CPL) are aggregated, and 3-bit signals are reported by external interrupt pins. This means that a total of 9-bit pins (3-bit from Cores, 3-bit from Shared Caches, 3-bit from CPL) are reported to external pins. All signals associated with RAS register are synchronized with the source clock.

Figure 12 shows a high-level diagram for RAS interrupt signal connection. Each RAS unit can report up to one signal for each error category and for each error record. These bits are OR reduced within the RAS Unit to produce 3 error signals. Further OR reduction among various other RAS Units within the chiplet and from other chiplets may be done to produce a set of signals that are external pins and/or connected to the APLIC.

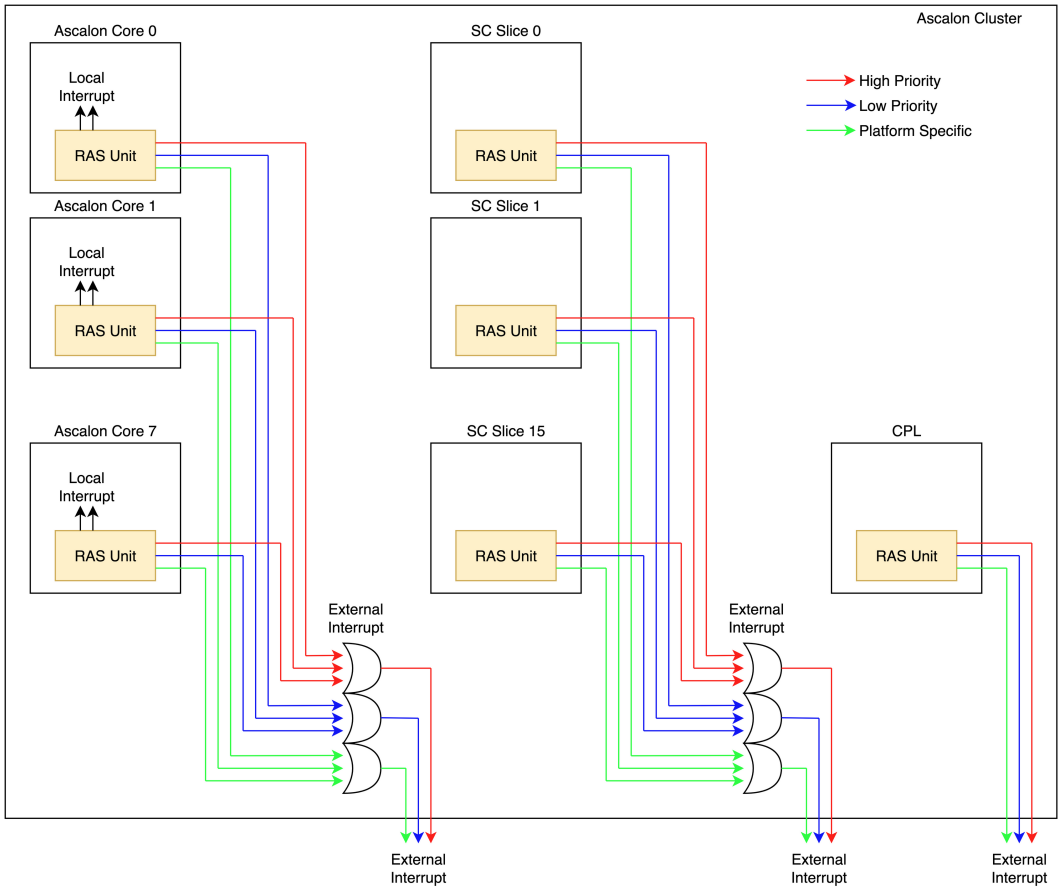

Figure 12. Ascalon Cluster RAS Interrupt Connection

`control_i.eid` cycles according to the RAS unit timestamp tick which is typically at 100 Hz. This field is 16b allowing, for a maximum of 65535 counts before firing. If this field is not 0, it counts down and sets valid field of record status reg to go high.

3.7.3. Supported Event Lists

The Ascalon uArch document lists the error events that are supported by each unit and are associated with each RAS Unit. The list includes RERI status and register values in detail. These values are provided at either the error event or error element level and are uniquely set for each error event.

The RAS Unit error records are treated as a pool of error records available to record the most critical error event presented to the RAS unit. The number of error records per RAS Unit is determined by the RAS Unit parameter N_RECORDS. The RAS Unit may be configured, via the N_RECORDS_RESERVED parameter, to exclude a number of error records from any hardware error events. This reserves them for software based error spoofing.

3.8. Power / Clock Management

3.8.1. Cluster Power Management Logic (CPL)

Cluster Power Management Logic (CPL) is the system management controller implemented in the cluster. Figure 13 shows high-level block diagram of the CPL module in Ascalon Cluster.

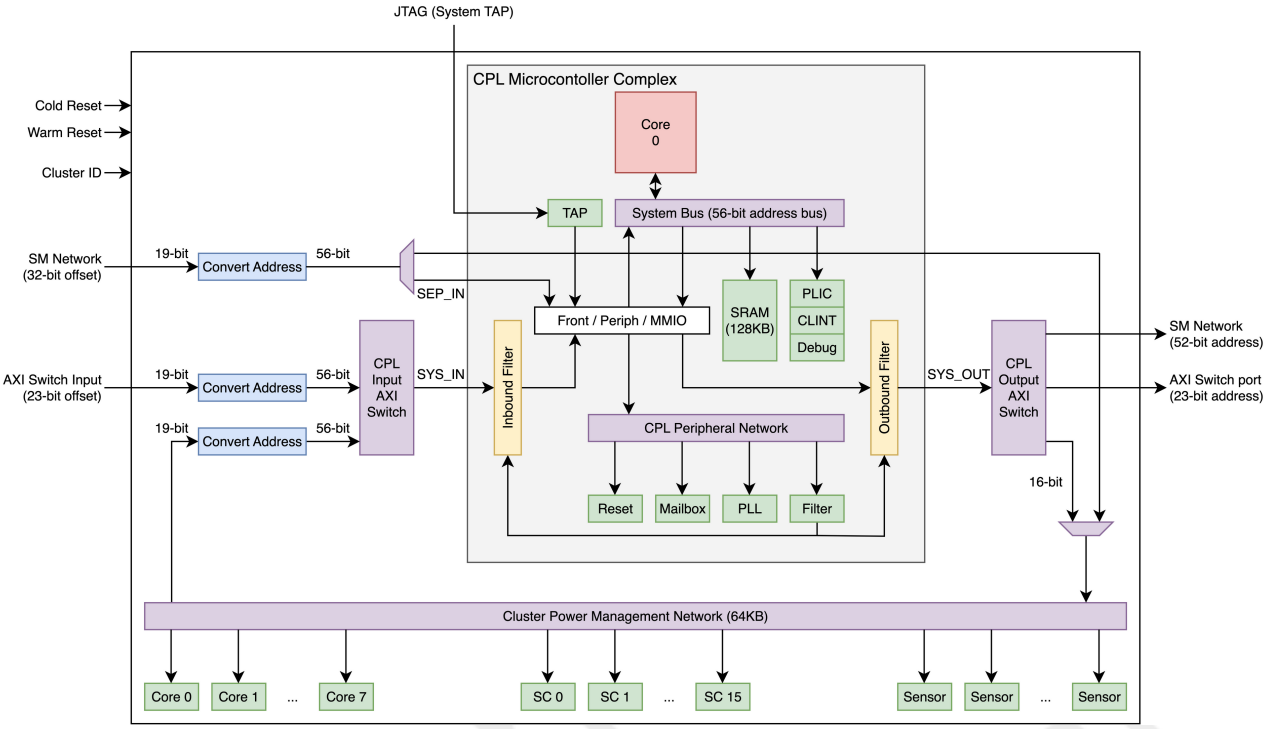

Figure 13. CPL Block Diagram

This component has the dedicated operation path to the PLL to activate / deactivate the Ascalon Cluster clock. To manage the Ascalon Cluster clock during PLL shut-off, the system management clock (SoC clock) is used in this module. This component populates power management information from the dedicated internal network and distributes a power management message to each CPU core. This unit is connected with a dedicated external network interface to communicate with a higher-level power management unit. The network is synchronized with the SoC clock.

3.8.2. CPL Microcontroller and Scratch Pad Memory

CPL has a microcontroller to run firmware. CPL also employs interrupt controller and debug functionalities. CPL has local memory protected by SECDED ECC. When ECC corruption is detected, a RAS error (e.g., parity errors) is reported / recorded by the RERI interface.

3.8.3. CPL Address Conversion

CPL has dedicated address space and a dedicated network to connect various components. To adjust the address for the CPL network, CPL deploys the address conversion unit for external input. The conversion logic remaps the address to local alias region.

```
// this function returns converted physical address (56-bit)
uint64_t cpl_convert_address(uint64_t physical_address)
{
    // RTL trim physical_address[51:19]
    bool in_pm_network = physical_address[18:16] == 3'b111; // check if the address is landing to PM network
with in 512KB region
    if (in_pm_network) return {4'h0, CLUSTER_MMR_BASE[51:27], 2'b01, i_cluster_id[3:0], 2'b10,
physical_address[18:0]};
    return {24'h0, 13'h1800, physical_address[18:0]}; // convert to local alias
}
```

This function is applied to the input to CPL (shown as "Convert Address" in Figure 13)

3.8.4. CPL Fabric

CPL takes 2 inputs from external agents. One is connected to an external system management network. The other is connected to the internal cluster network. The external system management network is directly connected to the CPL internal network. The internal cluster network is connected to the CPL internal network after the request is checked by an inbound filter. The input port takes the address offset to feed the address into the CPL microcontroller.

Table 6 shows address map in CPL. The internal address is mapped to 512KB address space in the cluster address.

Resource Name	Internal Address	Size	Output Port Selection
WatchDog Timer	0x00000000 - 0x00000FFF	4KB	Internal
Debug Module	0x00001000 - 0x00001FFF	4KB	Internal
CPL Reset Unit	0x00002000 - 0x00002FFF	4KB	Internal
PLL Interface	0x00003000 - 0x00003FFF	4KB	Internal
CPL External Register	0x00010000 - 0x00011FFF	8KB	Internal
Inbound Filter Config	0x00015000 - 0x00015FFF	4KB	Internal
Outbound Filter Config	0x00016000 - 0x00016FFF	4KB	Internal
Scratch Pad Memory	0x00040000 - 0x0005FFFF	128KB	Internal
Power Management Network	0x00070000 - 0x0007FFFF	64KB	MMIO (not routed to CPL)

Table 6. CPL Internal Address Map

Figure 13 has multiple connections between different networks. Table 7 shows the AXI ID width for each connection.

Manager	Subordinate	AXI ID Width
SM Network	CPL SEP Input	7
AXI Switch	CPL Input Switch	14
PM Network	CPL Input Switch	4
CPL Input Switch	CPL SYS Input	15
CPL SYS Output	CPL Output Switch	7
CPL Output Switch	SM Network	7
CPL Output Switch	AXI Switch	7
CPL Output Switch	PM Network	7

Table 7. CPL AXI Width for Each Connection

The CPL switch routes packets based on the global address coming from the CPL microcontroller. The CPL switch takes cluster ID (`i_cluster_id[3:0]`) and `CLUSTER_MMR_BASE[51:27]` to determine the output port.

```
// this function returns enum {SYS_NETWORK, PM_NETWORK, LOCAL_CLUSTER}
output_port_t cpl_switch_routing(uint64_t physical_address)
{
    bool in_cluster_mmr = physical_address[51:27] == CLUSTER_MMR_BASE[51:27];
    bool in_local_cluster = physical_address[24:21] == i_cluster_id[3:0];
    bool in_cpl_region = physical_address[26:25] == 2'b01 && physical_address[20:19] == 2'b10;
    if (in_cluster_mmr && in_local_cluster &&  in_cpl_region) return PM_NETWORK;
    if (in_cluster_mmr && in_local_cluster && !in_cpl_region) return LOCAL_CLUSTER;
    return SYS_NETWORK;
}
```

3.8.5. CPL Peripheral

CPL has its own peripheral bus and it is connected to various registers. The registers are mapped to an external register field.

Mailbox

CPL employs a mailbox to communicate with other power management components. When the power management components detect an unusual state, the information is transferred to the mailbox so that the system can take care of an abnormal state.

PLL (or PLL Interface)

Control clock frequency for CPU cores and shared caches. Power Management Unit has a dedicated interface to PLL to control frequency based on the power budget.

The IP customer needs to integrate a PLL interface. Detailed integration guidance is described in the IP integration guide.

3.8.6. CPL External Register

Config MMR (`cluster_config`)

Fuse information programmed in config register is distributed by CPL firmware. Each potential receiver has MMR register to hold configuration information in 64-bit format.

Field	Functionality	Lock	Reset Value	Description
[63:48]	Reserved	N/A	N/A	reserved field
[47:45]	core_mhartid[7][2:0]	Yes	0	mhartid[2:0] for physical core 7
[44]	core_enable[7]	Yes	0	When this field is 0, physical core 7 is disabled
[43:41]	core_mhartid[6][2:0]	Yes	0	mhartid[2:0] for physical core 6
[40]	core_enable[6]	Yes	0	When this field is 0, physical core 6 is disabled
[39:37]	core_mhartid[5][2:0]	Yes	0	mhartid[2:0] for physical core 5
[36]	core_enable[5]	Yes	0	When this field is 0, physical core 5 is disabled
[35:33]	core_mhartid[4][2:0]	Yes	0	mhartid[2:0] for physical core 4
[32]	core_enable[4]	Yes	0	When this field is 0, physical core 4 is disabled
[31:29]	core_mhartid[3][2:0]	Yes	0	mhartid[2:0] for physical core 3
[28]	core_enable[3]	Yes	0	When this field is 0, physical core 3 is disabled
[27:25]	core_mhartid[2][2:0]	Yes	0	mhartid[2:0] for physical core 2
[24]	core_enable[2]	Yes	0	When this field is 0, physical core 2 is disabled
[23:21]	core_mhartid[1][2:0]	Yes	0	mhartid[2:0] for physical core 1
[20]	core_enable[1]	Yes	0	When this field is 0, physical core 1 is disabled
[19:17]	core_mhartid[0][2:0]	Yes	0	mhartid[2:0] for physical core 0
[16]	core_enable[0]	Yes	0	when this field is 0, physical core 0 is disabled
[15]	lock	Yes	0	when this field is set, write to `lockable` entry is ignored
[14:12]	Reserved	N/A	N/A	reserved field
[11]	export_restriction_enabled	Yes	0	reduced FP throughput for export control restriction
[10:9]	debug_enable	Yes	0	DM accept command form JTAG only, AXI only, both
[8]	trace enable	Yes	0	Trace is enabled
[7:0]	sc_harvest_strap[7:0]	Yes	0	Shared cache way-enable signal (`config0_sc_disable_way`)

Table 8. Cluster Configuration Register Format

to avoid an unexpected transaction process, the default value of this MMR register is 0 (all core / features are disabled).

MMR is implemented in all corresponding components (e.g., DM, ACLINT, AXI Switch, NMI Interrupt, CPU Core). Each component calculates the mapping based on the following pseudo code and changes its behavior. For example, a halt request from DM is adjusted based on the virtual core to physical core mapping.

```
// input / output
input        core_enable[7:0]
input  [2:0] core_mhartid[7:0]
output       Core_Enable[7:0]; // Physical Core Enable
output [2:0] Core_HartID[7:0]; // Physical Core HartID
output       Hart_Enable[7:0]; // Hart (Logical Core) Enabled
output [2:0] Hart_CoreID[7:0]; // Hart (LOgical Core) Physical Core ID

// mapping physical ID to logical ID
always_comb begin
    Hart_Enable[7:0] = {false};
    for (i=0; i<8; i++) { // 8 is # of Core in Cluster
        Physical_Core_Enable[i] = false;
        if(core_enable[i]) {
            if(Hart_Enable[core_mhartid[i][2:0]]) {
                Physical_Core_Enable[i] = true;
                Physical_Core_HartID[i] = core_mhartid[i];
                Hart_Enable[core_mhartid[i][2:0]] = true;
```

```
                Hart_CoreID[core_mhartid[i][2:0]] = i;
            }
        }
    }
end
```

These config registers are not initialized if `config_hold` is asserted.

Reset Unit (0x040 - 0x7F)

CPL owns the reset logic for the entire cluster logic. External pins in cluster only reset the CPL registers. CPL firmware needs to initialize reset of cluster logic. CPL internal logic is initialized by CPL warm reset.

	Signal Name	Register Name	Description	Reset	Reset Value
[0]	`cpl_mmr_reset_n`	Cluster Reset	Warm reset signal for Ascalon Cluster	Cluster Cold Reset	0
[63:1]	Reserved	N/A	N/A	N/A	N/A

Table 9. CPL Reset Register 0 (address: 0x040)

	Signal Name	Register Name	Description	Reset	Reset Value
[0]	`cpl_mmr_ras_hold`	Cluster RAS Hold	Reset hold signal for RAS registers	Cluster Cold Reset	0
[1]	`cpl_mmr_dm_hold`	Cluster Debug Hold	Reset hold signal for DM registers	Cluster Cold Reset	0
[63:2]	Reserved	N/A	N/A	N/A	N/A

Table 10. CPL Reset Register 1 (address: 0x048)

	Signal Name	Register Name	Description	Reset	Reset Value
[7:0]	`cpl_mmr_nofetch[7:0]`	Cluster No Fetch	RISC-V CPU Does Not Start Fetch after reset	Cluster Warm Reset	all1
[63:8]	Reserved	N/A	N/A	N/A	N/A

Table 11. CPL Reset Register 2 (address: 0x050)

`cpl_nofetch` *functionality is supported from reset vector register for the CPL microcontroller. CPL has a reset vector register which is programmable from an external agent. Until this register is updated, CPL will not start fetching the instructions from memory.*

3.8.7. Power Management Network

Power Management Network is the network between the CPL and cluster internal components including sensors.

The sensor satellites within the Power Management Network check the operation conditions in the cluster. Typically, communication is performed by Memory Mapped Register accesses. Based on observed sensor status, the CPL may change the operating condition for the cluster. However, when satellites observe unusual operation conditions, the Ascalon Cluster may take special action.

High Temperature (Tj Shutdown)

When sensor satellites observed abnormal operation conditions like high temperature, they can report a warning to the CPL by interrupt. When the CPL takes the interrupt, the CPL firmware is expected to set a low clock frequency to reduce the temperature. This interrupt signal is kept high for as long as the temperature exceeds the threshold.

Abnormal Temperature (Tj Max)

When the sensor satellites observe a more fatal operation condition like a higher temperature, they may report fatal status to an external system management logic. This signal is routed to other agents to reduce the power consumption from each agent. When other agents observe fatal thermal condition and want to reduce the activity of Ascalon Cluster, an external agent can assert an abnormal environment pin to reduce the clock frequency. When the pin is asserted, the PLL controller sets the lowest frequency possible to minimize the power consumption from the cluster.

Ascalon cluster shutdowns the cluster clock (technically setting 0 Hz as the operational clock) to achieve this requirement.

3.9. Instruction Patching

Instruction patching is the feature to mitigate the severity of a hardware bug.

3.9.1. Software Visible Registers

Name	Type	Descrption
`pversion`	MMR	Version information (Read-only)
`pcontrol0`	MMR	Control of each `preg*` [15:00] control information for `preg0` [31:16] control information for `preg1` [47:32] control information for `preg2` [63:48] control information for `preg3` In each 16-bit representaion, [0] enable [8:1] prv + v multihot encoding [1] : U-mode (v=0, prv=0) [2] : HS-mode (v=0, prv=1) [3] : reserved [4] : M-mode (v=0, prv=3) [5] : VU-mode (v=1, prv=0) [6] : VS-mode (v=1, prv=1) [7] : reserved [8] : reserved [9] vector conservative mode [14:10] reserved [15] lock bit
`pcontrolu`	MMR	Control of ucode sequence.
`preg0`	MMR	Patch register 0, [31:0] opcode, [63:32] mask
`preg1`	MMR	Patch register 1, [31:0] opcode, [63:32] mask
`preg2`	MMR	Patch register 2, [31:0] opcode, [63:32] mask
`preg3`	MMR	Patch register 3, [31:0] opcode, [63:32] mask
`pinst`	CSR	Patched instruction
`pstatus`	CSR	[0] trans: translation enable / disable [1] size: instruction size (2B or 4B)
`dpc`	CSR	Reuse debug mode register to track patched instruction address
`dcsr`	CSR	Use `dcsr.prv` and `dcsr.v` to apply translation
`dscratch0`	CSR	Saving context information to make space for regular manipulation
`dscratch1`	CSR	Saving context information to make space for regular manipulation

When `pcontrol0` locks the patch register, the corresponding `preg*` and `pcontrol*` contents ignore any write operations.

3.9.2. Patch Mode and Patch PMA

When RISC-V CPU is executing an instruction specified by patch registers, the CPU is taking special interrupt. This interrupt takes the CPU to Patch Mode. Similar to Debug Mode, the Patch Mode is fully owned by RISC-V hardware. In the Patch Mode, CPU fetches instructions from a dedicated memory region and the CPU may have access to the same region via load / store operations. This region is not accessible from regular execution mode. This protection is enforced by a special hardwired PMA entry (Patch PMA) The patch RAM is mapped on CPL SRAM space. Therefore, Patch PMA prevents access to this region unless the CPU is in Patch Mode.

Status	Instruction Fetch	Load / Store, Translation
`pstatus.trans` = 0	Bare Translation, PMA / PMP are disabled	
`pstatus.trans` = 1	Same with `pstatus.trans` = 0	Translation region specified by `dcsr.prv` and `dcsr.v`

Table 12. Address Translation / Memory Access Protection in Patch Mode

Similar to Debug Mode, address translation is always disabled for instruction fetch. For load store operation, address translation is performed by the context specified in `dcsr.prv` and `dcsr.v` if `pstatus.trans` is set to 1. When `pstatus.trans` is 0, the address translation is disabled and PMP check is also disabled.

3.10. Intra-Cluster Network

3.10.1. Shared Cache Crossbar

Shared cache has a large crossbar to process packets coming from Ascalon CPU cores. The shared cache crossbar is divided into two networks. The network is selected by PA[6] of the memory request. Each Ascalon CPU core injects up to 1 CHI request packet into each network.

Shared cache crossbar determines the destination of each packet based on TgtId of incoming packet. If the packet is cacheable memory access, the packet is routed to a coherent network (CHI network). If the packet is non-cacheable memory access, the packet is routed from a coherent network to a non-cohenrent network (AXI network). Unlike coherent network, non-coherent network does not have a separated network.

In the rest of this section, CHI network among CPU cores and shared cache in cluster is called internal CHI network.

3.10.2. Shared Cache Bridge

To support both cacheable / non-cacheable memory accesses, shared cache has the capability of converting protocol between AXI and CHI. CHI-to-AXI bridge (CAB) converts CHI requests to AXI requests. This is behaving as HN-I and SN-I of the internal CHI network. AXI-to-CHI bridge (ACB) converts AXI requests to CHI requests. This is behaving as RN-I for the internal CHI network.

Shared cache bridge also has the capability of converting internal CHI protocol to CHI compliant protocol before CHI packets go out to an external interface. This converts Ascalon-special CHI protocol features (e.g., valid / ready handshake) to standard CHI protocol (e.g., credit-base handshake).

CHI network has a 32B data bus. AXI network has a 64B data bus.

3.10.3. Intra-Cluster AXI Network (AXI-Bridge)

AXI Intra-Cluster Network (AXI-Bridge) is the switch structure in the Ascalon CPU Cluster to route non-coherent memory access to either an external interface or to internal components.

This network supports high memory bandwidth (64B / cycle) to help give external DMA access to shared cache array. To support high bandwidth, this network can process up to 64B / cycle / direction bandwidth. This network belongs to the System Fabric clock domain.

This network supports AXI5-Lite as the communication protocal. This protocol is also exposed to the Ascalon Cluster IP external interface.

Intra-Cluster AXI network has various input / output ports. Each of them has slightly different user defined bits (AxUSER). All incoming packets are converted to 8-bit common representation. When the request is going out from a switch network, the request is adjusted to different value.

	Port Name	AxUSER[7:4]	AxUSER[3:0]	Internal Representation
input	AXI Subordinate Port	QoSID[3:0]	SrcID[3:0]	AxUSER[7:0] = {QoSID[3:0], SrcID[3:0]}
input	CHI-to-AXI Bridge	QoSID[3:0]	N/A	AxUSER[7:0] = {QoSID[3:0], CLUSTER_LOCAL_SRCID}
input	Trace Output	N/A	N/A	AxUSER[7:0] = {4'b0000, CLUSTER_REMOTE_SRCID}
input	CPL Input Switch	N/A	SrcID[3:0]	AxUSER[7:0] = {4'b0000, CLUSTER_CPL_SRCID}
output	AXI Manager Port	QoSID[3:0]	ConvertSrcID(SrcID[3:0])	AxUSER[7:0] = {QoSID[3:0], SrcID[3:0]}
output	AXI-to-CHI Bridge	QoSID[3:0]	SrcID[3:0]	AxUSER[7:0] = {QoSID[3:0], SrcID[3:0]}
output	CPL Output Switch	N/A	SrcID[3:0]	AxUSER[7:0] = {Don't Care, CLUSTER_CPL_SRCID}
output	Intra-Cluster MMR Network	N/A	SrcID[3:0]	AxUSER[7:0] = {Don't Care, SrcID[3:0]}

Table 13. AxUSER Assignment

`ConvertSrcID` in the table stands for converting outgoing CLUSTER_LOCAL_SRCID to CLUSTER_REMOTE_SRCID. This conversion does not happen for other Src ID encoding.

```
uint8_t ConvertSrcID(uint8_t src_id) {
    if (src_id == CLUSTER_LOCAL_SRCID) {
        return CLUSTER_REMOTE_SRCID;
    } else {
        return src_id;
    }
}
```

`Attribute(AxADDR[51:27])` in the table stands for "attaching accessing memory" attribute. `AxUSER[2:0]` is `3'b000` and `AxUSER[3]` is marked `1'b1` when `AxADDR[51:27]` is equal to `CLUSTER_MMR_BASE`.

3.10.4. Intra-Cluster MMR Network (AXI-Ring)

AXI MMR Network (AXI-Ring) is the network structure in the Ascalon CPU Cluster to be connected with MMRs and internal AXI network.

This network is the configuration / management network for the cluster. Therefore, the design does not expect high memory bandwidth. Since there is no bandwidth requirement, this network supports up to 8B / cycle access bandwidth. This network belongs to the Ascalon Cluster clock domain.

When the CPU core is disabled, AXI MMR Network adjusts the MMR address for disabled CPU cores.

3.10.5. Intra-Cluster Power Management Network (PM Network)

Intra-Cluster Power Management Network (PM Network) is the network structure in the Ascalon CPU Cluster that is to be connected with the power management / control components (CPU cores, Thermal Sensors) to manage power / thermal consumption. The network uses custom low bandwidth protocol since the network needs minimum bandwidth requirements. This network belongs to the Ascalon Cluster clock domain.

3.10.6. Memory Subsystem Detected Error Handling

Ascalon Cluster support various different memory interfaces and each of them has slightly different definitions of memory access error. Within the Ascalon Cluster, AXI error, CHI error, and RISC-V exceptions are converted in the following manner:

AXI Error Type	CHI Error Type	When Core consumed the data	Comment
DECERR	NDERR	Instruction / Load / Store Access Fault (Exception Code = 1, 5, 7)	Memory request failed because there are no memory, permission check failure, etc...
SLVERR	DERR	Hardware Error (Exception Code = 19) and reporting RAS error (UUE)	Memory request accessed to ECC corrupted data.

Table 14. AXI Error Type, CHI Error Type Conversion, and CPU Action

AXI-CHI conversion is following the AMBA standard way (e.g., how 3rd party converts the error). CHI-RISCV conversion is based on the peer review of specification.

3.10.7. Async FIFO

Async FIFOs are inserted between any clock domain crossing point except for interrupt wire signals. The FIFO contains flop buffers and multiplexing logic to read out the data.

Chapter 4. External Interface

This section describes how the Ascalon Cluster is connected with external modules. For completeness, some internal signals (e.g., Ascalon Cluster clock) is also listed in this section.

4.1. Reset

Table 15 shows the list of reset signals for components in the Ascalon Cluster.

Signal Name	Implementation	Description
`cluster_cold_reset_n`	External Interface	Cold reset signal for all cluster internal resources
`cluster_warm_reset_n`	External Interface	Warm reset signal for all cluster internal resources
`cluster_config_hold`	External Interface	When this signal is asserted, configuration related registers are not initialized by `cluster_warmreset_n`
`cluster_ras_hold`	External Interface	When this signal is asserted, RAS related registers are not initialized by `cluster_warmreset_n`
`cluster_debug_hold`	External Interface	When this signal is asserted, Debug related registers are not initialized by `cluster_warmreset_n`
`ndmreset_request`	External Interface	When DM decides to apply NDM reset, this signal is asserted
`ndmreset_process`	External Interface	When this signal is asserted, the CPU does not start fetching instructions after reset deassertion
`cpl_mmr_reset_n`	Internal Register	Reset signal for cluster internal logic
`cpl_mmr_config_hold`	Internal Register	When this signal is asserted, configuration related registers are not initizelized by `cpl_mmr_reset_n`
`cpl_mmr_ras_hold`	Internal Register	When this signal is asserted, RAS related registers are not initialized by `cpl_mmr_reset_n`
`cpl_mmr_dm_hold`	Internal Register	When this signal is asserted, Debug related registers are not initialized by `cpl_mmr_reset_n`
`cpl_mmr_nofetch`	Internal Register	CPU does not start fetching instructions after `cpl_mmr_reset_n` deassertion while this signal is asserted

Table 15. Ascalon Cluster Reset Signal List

Detailed description of reset registers are shown in Table 9.

`cluster_cold_reset_n` initializes all cluster logic including reset register. `cluster_warm_reset_n` initializes cluster logic excluding some critical components. The components are specified by `cluster_config_hold`, `cluster_ras_hold`, and `cluster_dm_hold`.

CPL has its internal reset registers apply a warm reset for internal logic. These registers are set / reset by external pins (e.g., `cluster_warm_reset_n`).

When a warm reset signal (`cluster_warm_reset_n`) is asserted (set from 1 to 0), synchronized reset signal (`internal_warm_reset_n`) captures the hold signal (e.g., `cluster_ras_hold`) before applying the reset. To satisfy this requirement, the reset signal is delayed by 8-cycle in internal logic. This reset signal delay is counted by SOC clock. If the reset domain is slower than the SOC clock, firmware must set the hold signal by MMR write before asserting the reset. Expected signal manipulation is done following pseudo code.

```
always @(posedge i_sc_clk or negedge internal_cold_reset_n) // SOC clock
begin
  // synchronizer for external signal to internal logic
  reset_synchronizer(internal_cold_reset_n, cluster_cold_reset_n);
  reset_synchronizer(internal_warm_reset_n, cluster_warm_reset_n);
  synchronizer(internal_ras_hold, cluster_ras_hold);
  synchronizer(internal_dm_hold,  cluster_dm_hold );

  // create delayed version of reset
  if (internal_cold_reset_n == 0)
    warm_reset_delay[7:0] <= 8'h00;
  else
    warm_reset_delay[7:0] <= {warm_reset_delay[6:0], internal_warm_reset_n};
```

```
    // reset register
    if (internal_cold_reset_n == 0)
      cpl_mmr_reset_n <= 1'b0;
    else if (warm_reset_delay[7] == 0)
      cpl_mmr_reset_n <= warm_reset_delay[6];
    else if (mmr_write_for_reset_n)
      cpl_mmr_reset_n <= mmr_write_data;

    // Config hold signal
    if (internal_cold_reset_n == 0)
      cpl_mmr_config_hold <= 1'b0;
    else if (internal_warm_reset_n == 0)
      cpl_mmr_config_hold <= internal_config_hold;
    else if (mmr_write_for_ras_hold)
      cpl_mmr_config_hold <= mmr_write_data;

    // RAS hold signal
    if (internal_cold_reset_n == 0)
      cpl_mmr_ras_hold <= 1'b0;
    else if (internal_warm_reset_n == 0)
      cpl_mmr_ras_hold <= internal_ras_hold;
    else if (mmr_write_for_ras_hold)
      cpl_mmr_ras_hold <= mmr_write_data;

    // Debug module hold signal
    if (internal_cold_reset_n == 0)
      cpl_mmr_dm_hold <= 1'b0;
    else if (internal_warm_reset_n == 0)
      cpl_mmr_dm_hold <= internal_dm_hold;
    else if (mmr_write_for_dm_hold)
      cpl_mmr_dm_hold <= mmr_write_data;
  end
```

`ndmreset_request`, `ndmreset_process` are defined to support NDM reset. This reset can reset outside of the cluster to initialize the entire system. Section 3.5.2.9 describes how these interface signals work.

After reset deassertion, CPU cores do not start fetching instructions unless both `ndmreset_process` and `cpl_mmr_nofetch` are deasserted.

All reset signals are synchronized with their source. This means that external signals are treated as asynchronous signals and internal signals coming from CPL are synchronized with the SOC clock.

Figure 14 shows how reset signals are generated inside Ascalon Cluster.

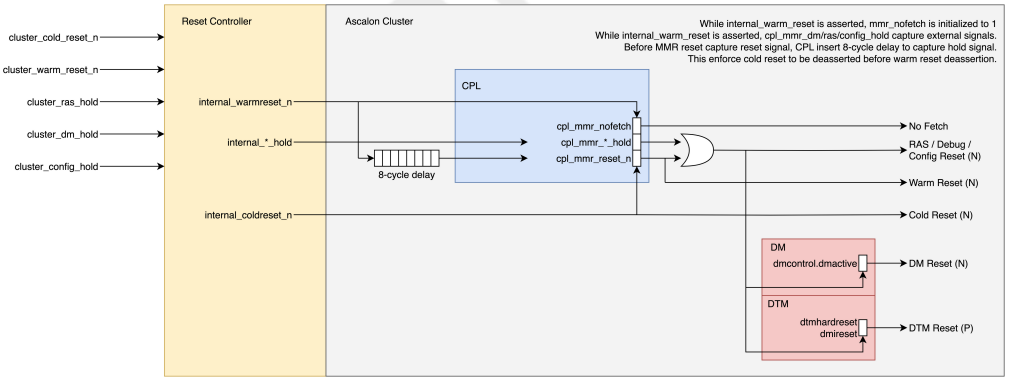

Figure 14. Ascalon Cluster Reset Signal Structure

4.1.1. Power-on Reset / Cold Reset

The external reset signal (`cluster_coldreset_n`) is asserted when the system power is turned on. This reset signal initializes CPL so that CPL can reset the remaining cluster logic. The reset signal is asynchronous (Ascalon Cluster RTL has synch flop.) The internal logic synchronizes the reset signal to the local clock domain. This reset signal also asserts the warm reset signal to initialize the remaining logic in the cluster.

4.1.2. Warm Reset

When either `cluster_coldreset_n`, `cluster_warmreset_n`, or `cpl_mmr_reset_n` is asserted, Ascalon Cluster initializes internal logic. Depending on hold signals, the reset may not initialize what the register contains. These hold signals must be asserted 16-cycles before the reset signal is asserted.

4.1.3. No Fetch

After reset is deasserted, CPU will not start the first instruction fetch while either `ndmreset_process` or `cpl_mmr_nofetch` is 1.

4.1.4. Timing Constraint

Cold reset, warm reset, SRAM hold, and external configuration pins (Section 4.6) have timing constraints.

- The clock of resetting logic needs to be active (cycle count in this section is counted by the clock of resetting logic.)
- While cold reset is asserted, warm reset must be asserted.
- When cold reset is asserted both cold and warm resets need to be asserted more than 16 cycles before deasserting the resets.
- When warm reset is asserted, the warm reset needs to be asserted more than 16 cycles before deasserting the warm reset.
- All external configuration pins must be stable for more than 16 cycles before warm reset deassertion.
- Reset signal deassertion is synchronized with the clock's rising edge.
- To avoid resetting critical logic not to be reset, the critical logic hold signal must be asserted more than 16 cycles before the corresponding reset signal is asserted.
- The SRAM hold signal needs to be stable for more than 16 cycles before corresponding reset deassertion.

Figure 15 shows the timing constraint for cold reset (top) and warm reset (bottom). More detail sequence is in Section 8.1.

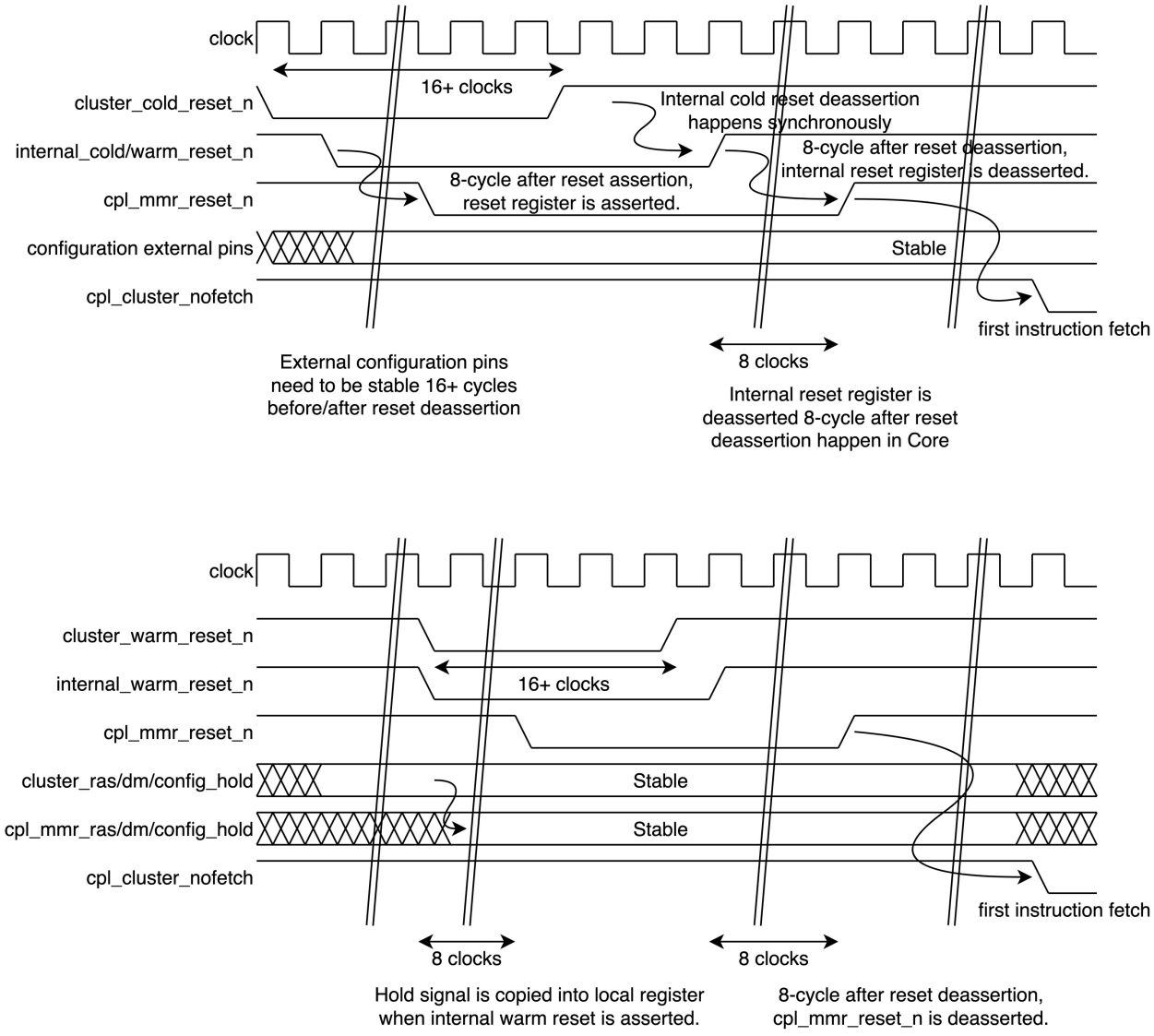

Figure 15. Reset Signal Timing Constraint

4.2. Clock

Ascalon Cluster takes 5 clocks. 4 clocks are supplied from external pins. 1 clock is generated from PLL inside the cluster.

4.2.1. Reference Clock

Ascalon Cluster receives a 100MHz Reference Clock to update `mtime` register and other purposes.

RISC-V SOC spec requests the CPU to update `mtime` *every 10ns (100MHz)*

4.2.2. JTAG Clock

Clock for JTAG interface.

4.2.3. Power Management Clock

Ascalon Cluster receives a 400MHz Power Management clock for power / system management components. This clock must be stable (no dynamic frequency scaling) after the operation begins so that power management firmware can calculate the real time.

4.2.4. System Fabric Clock

Ascalon Cluster receives a System Fabric Clock so that it can make requests synchronized with the Fabric Clock.

4.2.5. Ascalon Cluster Clock (internal signal)

Ascalon Cluster has a dedicated PLL to generate its clock for one Ascalon Cluster. Since PLL is a part of Ascalon Cluster, Ascalon Cluster clock does not have external pins.

The PLL receive a power management message and interrupts signals to switch its operating clock frequency.

4.3. Memory Interface

Ascalon Cluster has multiple memory ports for various purposes. For all interfaces, the address specifies the system physical address (e.g., no hashing).

4.3.1. Coherent Network : CHI Manager Port

Ascalon Cluster supports CHI-E interface as the protocol for the coherent network. Each cluster has 2 CHI ports. The port is selected by PA[6]. Each cluster uses single SrcID. TgtID is specified by the system fabric for REQ channel. Up to 12-bit TxnID can be used to differentiate packets. The data bus width is 256-bit. These memory interface signals are synchronized with the System Fabric Clock. Section 9.3 lists CHI messages supported by Ascalon Cluster.

4.3.2. Non-Coherent Network : AXI Manager Port

Ascalon Cluster adopts AXI5-Lite for a non-coherent network manager port. The bus data width is 512-bit. Ascalon Cluster uses up to 4096 AXI ID (12-bit) for read / write accesses. These memory interface signals are synchronized with the System Fabric Clock.

4.3.3. Non-Coherent Network : AXI Subordinate Port

Ascalon Cluster adopts AXI5-Lite for a non-coherent network subordinate port for Scratch pad access and MMR access. The bus data width is 512-bit. Ascalon Cluster supports up to 1024 AXI ID (10-bit) for read / write accesses.

The Ascalon Cluster AXI subordinate port does not have AXI ID base ordering tracking mechanism. Therefore, the incoming request is expected to be using a unique ID in the system.

Incoming MMR access size is expected to be naturally aligned 4B or 8B. If the access size is larger than 8B for MMR access, the intra-cluster network may crack the access and it may limit inflight MMR accesses in the cluster. If the access size is neither naturally aligned 4B nor 8B, the MMR access may fail.

4.3.4. Chiplet System Management Network

Ascalon Cluster supports a dedicated system management network. This is a AXI4 network with a 64-bit data bus. This network is running on the SOC clock to avoid DVFS effects from the Fabric Clock.

Unlike other application memory interfaces, this bus can specify up to 32-bit as its address. If the request flows down to the memory network for application processors, any address above 32-bit is expected to be filled by 0.

Cluster has both a manager port and a subordinate port for this network. The AXI ID for this network is 7-bit.

4.4. Wired Interrupt

4.4.1. Non-Maskable Interrupt (`i_nmi_valid[7:0]`)

One Non-Maskable Interrupt (NMI) signal is connected to each CPU core. Each NMI signal is connected to a corresponding logical CPU core.

4.4.2. Power Management Interrupt (Abnormal Temperature)

When the power management unit observes an unexpected condition, the power management unit can report this status to an external agent. This is reported by an abnormal environment out interface (`o_abnormal_environment[1:0]`). This is an active high signal and is kept asserted while the temperature sensor is observing an abnormal temperature. This output signal is not synchronized with any clocks. The receiver needs to insert an appropriate synchronizer to avoid metastability. "bit-0" indicates that the internal logic or sensor is requesting minimizing the power consumption. "bit-1" indicates that the internal logic or sensor is requesting immediate shutdown.

When other agents in the chip observe a similar abnormal condition, these components can request emergency shutdown (`i_emergency_shutdown`). This signal is connected to the PLL controller to make the cluster clock become the lowest frequency. The signal is asynchronous. The cluster logic is inserting synchronizer flops to avoid metastability.

Emergency Thermal Shutdown

If this pin is asserted, Ascalon Cluster will immediately stop the PLL to get the thermal condition under control.

4.5. RAS Error Reporting

When Ascalon CPU detects a correctable / uncorrectable error, the error can be reported by local interrupt to the corresponding core or the interrupt signal to a Platform-Level Interrupt Controller like APLIC. Ascalon Cluster can generate a total of 9-bit wired interrupt signals from internal RAS registers.

4.5.1. Core RAS Error report (`o_ras_error_core[2:0]`)

CPU core may detect RAS errors from Core internal SRAMs or external memory access. When a RAS error is reported and the RAS registers are programmed appropriately, a RAS error can be reported by a wired interrupt signal. The RAS error signal will stay high until the software explicitly flushes the RAS register. Ascalon Cluster has 3 wired interrupt signals (0:High-Priority, 1:Low-Priority, 2:Platform-Specific) for CPU Cores.

4.5.2. Shared Cache RAS Error report (`o_ras_error_sc[2:0]`)

Shared cache may detect RAS errors from the L2 cache system and snoop filter. Similar to a RAS error from the CPU, shared cache drives one RAS error interrupt signal to the system. Ascalon Cluster has 3 wired interrupt signals (0:High-Priority, 1:Low-Priority, 2:Platform-Specific) for shared cache.

4.5.3. Other RAS Error report (`o_ras_error_other[2:0]`)

The power management unit may detect RAS errors from RAM structures in the unit. Similar to a RAS error from the CPU, the power management unit drives one RAS error interrupt signal to the system. Ascalon Cluster has 3 wired interrupt signals (0:High-Priority, 1:Low-Priority, 2:Platform-Specific) for other components in cluster.

4.6. Configuration Pin

4.6.1. Cluster ID (`i_cluster_id[3:0]`)

Based on RISC-V Privileged Specification, each CPU core needs its unique ID. For Ascalon Cluster, the CPU ID (hart ID) is determined by the concatenation of the Cluster ID and the Core ID in Cluster. The Ascalon system may support up to 128 CPU cores in the system. Regardless of the number of cores in a cluster, hart ID is assiged in the following manner:

Hart ID [2:0]	Core ID in Ascalon Cluster
Hart ID [6:3]	Connected to Cluster ID [3:0]

Table 16. Hart ID Assignment

Cluster ID must start from 0 in the system.

4.7. Other

4.7.1. JTAG (Debug)

JTAG interface for external debugger and silicon debug.

4.7.2. MBIST

MBIST management interface. This interface covers redundancy information programming.

Chapter 5. RISC-V Architecture / Extensions for Ascalon CPU

5.1. RISC-V Server Platform

While RISC-V International has not yet produced an official Server Platform Architecture Specification, there are several documents that will be used to define this specification. These include:

- Server SoC Requirements github.com/riscv-non-isa/server-soc
- Boot and Runtime services (BRS) specification github.com/riscv-non-isa/riscv-brs
- ISA Profiles (e.g., RVA23)

The Server SoC Requirements document provides additional requirements beyond those in the ISA specifications. In some cases, this mandates some features that were otherwise specified as optional. Ascalon will comply with this specification as much as possible, bearing in mind that the specification might not be frozen before Ascalon is. At this point, any newly specified features will need to meet a high bar to make it into Ascalon.

Note: RISC-V has an out-of-date Server Platform Specification (riscv-platform-spec). We have asked this to be removed. In any case, this old specification should be ignored.

5.2. RVA23 Profile

All of the RVA23U64 and RVA23S64 extensions (and features) are listed below. These are based on the *non-frozen* RISC-V International specification: rva23-profile

While the following list of extensions (RVA23 and otherwise) is complete, we are using the ASC Architecture / Extensions Requirements Confluence page as the official reference for signoff purposes.

- U - indicates that the specification is part of the RISC-V Unprivileged Architecture
- P - indicates that the specification is part of the RISC-V Privileged Architecture

Ascalon supports the following base ISA:

Name	Version	Description
RV64I	2.1 U	64-bit base ISA; ecall traps to execution environment
Supervisor	1.13	ISA
Hypervisor	1.0	ISA
Machine	1.13	ISA
misa fields	-	A C D F H M I S U V

Table 17. Base Instruction Set

5.3. RVA23U64

Ascalon supports all of the RVA23U64 Mandatory Extensions.

Name	Type	Version	Description
A	Extension	2.1 U	Atomic instructions
B	Extension	1.0 U	Bitmanip
C	Extension	2.0 U	Compressed Instructions
D	Extension	2.2 U	Double-precision floating-point instructions
F	Extension	2.2 U	Single-precision floating-point instructions
M	Extension	2.0 U	Integer multiplication and division
V	Extension	1.0 U	Vector
Za64rs	Feature	-	Reservation sets are contiguous, naturally aligned, and a maximum of 64 bytes

Name	Type	Version	Description
Zawrs	Extension	1.01 U	Wait on reservation set
Zcb	Extension	1.0 U	(Zc) Additional 16b compressed instructions (ld, st, extend)
Zcmop	Extension	1.0 U	Compressed Maybe Operations
Zfa	Extension	1.0 U	Additional scalar FP instructions
Zfhmin	Extension	1.0 U	Half-Precision Floating-point transfer and convert (subsumed by optional Zfh)
Zic64b	Feature	-	Cache blocks must be 64 bytes in size, naturally aligned in the address space
Zicbom	Extension	1.0 U	CMO: Cache-Block Management Operations
Zicbop	Extension	1.0 U	CMO: Cache-Block Prefetch Operations
Zicboz	Extension	1.0 U	CMO: Cache-Block Zero Operations
Ziccamoa	Feature	-	Main memory regions with both the cacheability and coherence PMAs must support AMOArithmetic [PMA]
Ziccif	Feature	-	Main memory regions with both the cacheability and coherence PMAs must support instruction fetch [PMA]. Any instruction fetches of naturally aligned power-of-2 sizes up to min(ILEN,XLEN) are atomic (i.e., 32 bits for RVA23).
Zicclsm	Feature	-	Main memory regions with both the cacheability and coherence PMAs must be supporting misaligned loads and stores [PMA]
Ziccrse	Feature	-	Main memory regions with both the cacheability and coherence PMAs must support RsrvEventual [PMA]
Zicntr	Extension	2.0 U	Base counters and timers (3 x 64 bit; shadow)
Zicond	Extension	1.0 U	Conditional Zeroing instructions
Zicsr	Extension	2.0 U	CSR instructions
Zihintntl	Extension	1.0 U	Non-temporal locality hints.
Zihintpause	Extension	2.0 U	Pause instruction
Zihpm	Extension	2.0 U	(with Zicntr) Hardware performance counters (29 x 64-bit)
Zimop	Extension	1.0 U	Maybe Operations
Ssnpm	Extension	x.x	Pointer masking (U/VU)
Zkt	Extension	1.0 U	Zk: Data-independent execution time.
Zvbb	Extension	1.0 U	Vector Bitmanip
Zvfhmin	Extension	x.x	(under V) Vector FP16 conversion instructions (subsumed by optional Zvfh)
Zvkt	Extension	1.0 U	Vector data-independent execution time (subsumed by optional Zvkng)

Table 18. RVA23U64 Mandatory Extensions

The B extension subsumes the Zba, Zbb, and Zbs extensions. "B" has also been added as a bit to the misa register.

Ascalon also supports several of the optional RVA23U64 extensions.

Name	Type	x.x	Description
Zfbfmin	Extension	1.0	Scalar BF16 FP conversions
Zfh	Extension	1.0 U	Scalar Half-Precision Floating-Point (FP16)
Zvbc	Extension	1.0 U	Vector carryless multiply
Zvfbfmin	Extension	1.0	Vector BF16 FP conversions
Zvfbfwma	Extension	1.0	Vector BF16 widening mul-add
Zvfh	Extension	x.x	(V1.0) Vector half-precision floating-point (FP16)
Zvkng	Extension	1.0 U	(Zvkg) Vector Crypto NIST Algorithms with GHASH

Table 19. RVA23U64 Optional Extensions (Supported)

Ascalon does not support a few of the optional RVA23U64 extensions.

Name	Type	Description
Zacas	Extension	Atomic compare and swap

Name	Type	Description
Zbc	Extension	Scalar carryless multiply
Zvksg	Extension	Vector Crypto ShangMi Algorithms with GHASH

Table 20. RVA23U64 Optional Extensions (Not Supported)

5.4. RVA23S64

Ascalon supports all of the RVA23S64 Mandatory Extensions.

Name	Type	Version	Description
H	Extension	1.0 S	Hypervisor
Shcounterenw	Feature	-	For any hpmcounter that is not read-only zero, the corresponding bit in hcounteren must be writable
Shgatpa	Feature	-	For each supported virtual memory scheme SvNN supported in satp, the corresponding hgatp SvNNx4 mode must be supported. The hgatp mode Bare must also be supported.
Shtvala	Feature	-	htval must be written with the faulting guest physical address in all circumstances permitted by the ISA
Shvsatpa	Feature	-	All translation modes supported in satp must be supported in vsatp
Shvstvala	Feature	-	vstval must be written in all cases described above for stval
Shvstvecd	Feature	-	vstvec.MODE must be capable of holding the value 0 (Direct). When vstvec.MODE=Direct, vstvec.BASE must be capable of holding any valid four-byte-aligned address
Ss1p13	Extension	1.13 S	Privileged Architecture version 1.13. (currently a draft)
Ssccptr	Feature	-	Main memory regions with both the cacheability and coherence PMAs must support hardware page-table reads [PMA]
Sscofpmf	Extension	1.0 S	Count Overflow and Mode-Based Filtering
Sscounterenw	Feature	-	For any hpmcounter that is not read-only zero, the corresponding bit in scounteren must be writable
Ssstateen	Extension	1.0 S	Supervisor-mode view of the state-enable extension. The supervisor-mode (sstateen0-3) and hypervisor-mode (hstateen0-3) state-enable registers must be provided.
Sstc	Extension	1.0.0 S	supervisor-mode timer interrupts
Sstvala	Feature	-	stval must be written with the faulting virtual address for load, store, and instruction page-fault, access-fault, and misaligned exceptions, and for breakpoint exceptions other than those caused by execution of the EBREAK or C.EBREAK instructions. For illegal-instruction exceptions, stval must be written with the faulting instruction.
Sstvecd	Feature	-	stvec.MODE must be capable of holding the value 0 (Direct). When stvec.MODE=Direct, stvec.BASE must be capable of holding any valid four-byte-aligned address.
Ssu64xl	Feature	-	sstatus.UXL must be capable of holding the value 2 (i.e., UXLEN=64 must be supported).
Sv39	Extension	x.x	(part of S) Page-Based 39-bit Virtual-Memory System
Svade	Feature	-	Page-fault exceptions are raised when a page is accessed when A bit is clear, or written when D bit is clear. (Superseded by implementing Svadu)
Svbare	Feature	-	The satp mode "Bare" must be supported
Svinval	Extension	1.0 S	Fine-Grained Address-Translation Cache Invalidation
Svnapot	Extension	1.0 S	NAPOT Translation Contiguity (special PTE encoding for 64KiB regions)
Svpbmt	Extension	1.0 S	Page-Based Memory Types
Zifencei	Extension	2.0 U	Instruction-Fetch Fence

Table 21. RVA23S64 Mandatory Extensions

Ascalon also supports *all* of the optional RVA23S64 extensions.

Name	Type	Version	Description
sdext	Extension	1.0	(Debug) Debug mode (required for external debug)
Ssstrict	Feature	-	No non-conforming extensions are present. Attempts to execute unimplemented opcodes or access unimplemented CSRs in the standard or reserved encoding spaces raises an illegal instruction exception that results in a contained trap to the supervisor-mode trap handler.
Sv48	Extension	- S	(Part of S) Page-Based 48-bit Virtual-Memory System
Sv57	Extension	- S	(Part of S) Page-Based 57-bit Virtual-Memory System
Svadu	Extension	1.0 S	Hardware A/D bit updates
Zkr	Extension	1.0 U	(Part of Zk) Entropy CSR

Table 22. RVA23S64 Optional Extensions (Supported)

5.5. Other Features (not in the profile)

Ascalon supports some extensions and features outside of the RVA23 profile.

Name	Supported?	Version	Description
RVTSO	No	-	TSO Memory model
BE/LE	No	-	CSR selectable - default is LE
misalgned AMO	No	-	AMOs to misaligned addresses will take an exception
misaligned IO	No	-	Misaligned IO accesses will take an access-fault exception
misa writable	Yes	-	Mask = 0x341029
mtvec writable	Yes	-	Mask = 0xfffffffffffffffd
AIA	Yes	1.0	Advanced Interrupt Architecture. smaia, ssaia Only MSI is supported for interrupt signalling by Ascalon CPU. Details are available in Section 3.1.14 and [_aplic].
RNMI	Yes	1.0.0 S	(Smrnmi) Resumable non-maskable interrupt will be implemnted instead of **NMI**
RAS	Yes	1.0	Reliability, Availablility & Servicability. This will comply with the evolving RISC-V RAS Error-record Register Interface riscv-reri.pdf specification
TEE	Yes	-	TrustZone like solution - Static TEE
E-Trace	No	-	We have opted for N-trace
N-Trace	Yes	1.0	Trace Encoder, Trace Funnel, Trace Sink
Debug	Yes	1.0	Only supports Program Buffer.
Ssqosid	Yes	1.0	Quality-of-Serivce (QoS) Identifiers
Cbqri	Yes	1.0	RISC-V Capacity and Bandwidth QoS register interface
Sdtrig	No	-	(Debug) Debug trigger module
Smnpm	Yes	1.0	Pointer masking (S/HS)
Smmpm	Yes	1.0	Pointer masking (M)
Smstateen	Yes	1.0 S	Machine-level extension state enable
xttzvqdot	Yes	-	Quad byte-wise dot-product instructions in custom opcode space

Table 23. Other Extensions not in RVA23

In addition, see below for the page which contains local links to the ISA specification documents: RISCV Architecture Specifications

NB: These specifications may be out of date. It is always good to check for the latest specs at: github.com/riscv

5.6. Implementation Defined parameters for Ascalon

Parameter	value	Comment
`mvendorid`	TBD	Tenstorrent JEDEC Vendor ID
`marchid`	TBD	This number represents the Ascalon 8-wide decode CPU

Parameter	value	Comment
`mimpid`	TBD	Please refer to release note
ASIDLEN	16	width of ASID
VMIDLEN	14	width of VMID
GEILEN	5	hardware supported guest VMs
PMP entries	16	only NAPOT regions are supported. The minimum region size is 4KB.
M-level interrupt identifier	255	-
S-level interrupt identifier	255	-
VS-level interrupt identifier	63	-

Table 24. Implementation Specified / Implementation Defined Parameters

Chapter 6. Software Visible Registers

This section describes address the mapping and details of the Control and Status Register (CSR) and the Memory Mapped Register (MMR). The CSR and MMR covered by RISC-V specification is out-of-scope for this section.

6.1. CSR (Control and Status Register)

Ascalon CPU has the following custom CSR registers. All other registers are following CSR address mapping specified in RISC-V specification.

Name	Address	Description
matp	7C7	Machine address translation and protection.
pmacfg0	7E0	PMA Config Register 0
pmacfg1	7E1	PMA Config Register 1
pmacfg2	7E2	PMA Config Register 2
pmacfg3	7E3	PMA Config Register 3
pmacfg4	7E4	PMA Config Register 4
pmacfg5	7E5	PMA Config Register 5
pmacfg6	7E6	PMA Config Register 6
pmacfg7	7E7	PMA Config Register 7
pmacfg8	7E8	PMA Config Register 8
pmacfg9	7E9	PMA Config Register 9
pmacfg10	7EA	PMA Config Register 10
pmacfg11	7EB	PMA Config Register 11
pmacfg12	7EC	PMA Config Register 12
pmacfg13	7ED	PMA Config Register 13
pmacfg14	7EE	PMA Config Register 14
pmacfg15	7EF	PMA Config Register 15
fecfg	BC0	Front End - CFG register
fecfg1	BC1	Front End - CFG register 1
fecfg2	BC2	Front End - CFG register 1
fecfg3	BC3	Front End - CFG register 3
fecfg4	BC4	Front End - CFG register 4
mccfg	BC5	MC config register
mcthrcfg0	BC6	MC - Power Throttle Config 0
mcthrcfg1	BC7	MC - Power Throttle Config 1
wfitimer	BC8	Timer control register
tracecfg	BC9	Trace config register
lscfg0	BD0	Load Store - CFG register 0
lscfg1	BD1	Load Store - CFG register 1
lscfg2	BD2	Load Store - CFG register 2
lscfg3	BD3	Load Store - CFG register 3
lscfg4	BD4	Load Store - CFG register 4
lscfg5	BD5	Load Store - CFG register 5
lscfg6	BD6	Load Store - CFG register 6
lscfg7	BD7	Load Store - CFG register 7
lscfg8	BD8	Load Store - CFG register 8
lscfg9	BD9	Load Store - CFG register 9
lscfg10	BDA	Load Store - CFG register 10
lscfg11	BDB	Load Store - CFG register 11
lscfg12	BDC	Load Store - CFG register 12

Name	Address	Description
lscfg13	BDD	Load Store - CFG register 13
lscfg14	BDE	Load Store - CFG register 14
lscfg15	BDF	Load Store - CFG register 15

Table 25. Custom CSR register for Ascalon CPU Core

6.2. MMR (Memory Mapped Register)

Ascalon IP supports a system to deploy up to 16 Ascalon clusters. Each Ascalon cluster defines 4 different MMR spaces to manage the system. The cluster ID is specified by external pins. The base address is specified by the verilog parameter so that APLIC can route MSI appropriately. To support 16 Ascalon Clusters, the system should assign 128MB address space for the Memory Mapped Register.

Each cluster has 4 different Memory Mapped Register sections:

- Machine-level interrupt file
- Machine-level Memory Mapped Registers
- Supervisor-level interrupt file
- Supervisor-level Memory Mapped Registers

Each Memory Mapped Register region needs 2MB of address space. Machine-level MMRs are expected to be protected by the PMP mechanism so that S-mode / U-mode cannot access to Machine-level resources. S-mode should manage access to Machine-level MMR by using the regular virtual memory system.

This table shows system level MMR address mapping:

MMR offset	MMR size	MMR region description
0x00000000	2MB	Cluster-0 Machine-level Interrupt File
0x00200000	2MB	Cluster-1 Machine-level Interrupt File
0x00400000	2MB	Cluster-2 Machine-level Interrupt File
0x00600000	2MB	Cluster-3 Machine-level Interrupt File
0x00800000	2MB	Cluster-4 Machine-level Interrupt File
0x00A00000	2MB	Cluster-5 Machine-level Interrupt File
0x00C00000	2MB	Cluster-6 Machine-level Interrupt File
0x00E00000	2MB	Cluster-7 Machine-level Interrupt File
0x01000000	2MB	Cluster-8 Machine-level Interrupt File
0x01200000	2MB	Cluster-9 Machine-level Interrupt File
0x01400000	2MB	Cluster-10 Machine-level Interrupt File
0x01600000	2MB	Cluster-11 Machine-level Interrupt File
0x01800000	2MB	Cluster-12 Machine-level Interrupt File
0x01A00000	2MB	Cluster-13 Machine-level Interrupt File
0x01C00000	2MB	Cluster-14 Machine-level Interrupt File
0x01E00000	2MB	Cluster-15 Machine-level Interrupt File
0x02000000	2MB	Cluster-0 M-level MMR Region
0x02200000	2MB	Cluster-1 M-level MMR Region
0x02400000	2MB	Cluster-2 M-level MMR Region
0x02600000	2MB	Cluster-3 M-level MMR Region
0x02800000	2MB	Cluster-4 M-level MMR Region
0x02A00000	2MB	Cluster-5 M-level MMR Region
0x02C00000	2MB	Cluster-6 M-level MMR Region
0x02E00000	2MB	Cluster-7 M-level MMR Region
0x03000000	2MB	Cluster-8 M-level MMR Region
0x03200000	2MB	Cluster-9 M-level MMR Region
0x03400000	2MB	Cluster-10 M-level MMR Region

MMR offset	MMR size	MMR region description
0x03600000	2MB	Cluster-11 M-level MMR Region
0x03800000	2MB	Cluster-12 M-level MMR Region
0x03A00000	2MB	Cluster-13 M-level MMR Region
0x03C00000	2MB	Cluster-14 M-level MMR Region
0x03E00000	2MB	Cluster-15 M-level MMR Region
0x04000000	2MB	Cluster-0 Supervisor-level Interrupt File
0x04200000	2MB	Cluster-1 Supervisor-level Interrupt File
0x04400000	2MB	Cluster-2 Supervisor-level Interrupt File
0x04600000	2MB	Cluster-3 Supervisor-level Interrupt File
0x04800000	2MB	Cluster-4 Supervisor-level Interrupt File
0x04A00000	2MB	Cluster-5 Supervisor-level Interrupt File
0x04C00000	2MB	Cluster-6 Supervisor-level Interrupt File
0x04E00000	2MB	Cluster-7 Supervisor-level Interrupt File
0x05000000	2MB	Cluster-8 Supervisor-level Interrupt File
0x05200000	2MB	Cluster-9 Supervisor-level Interrupt File
0x05400000	2MB	Cluster-10 Supervisor-level Interrupt File
0x05600000	2MB	Cluster-11 Supervisor-level Interrupt File
0x05800000	2MB	Cluster-12 Supervisor-level Interrupt File
0x05A00000	2MB	Cluster-13 Supervisor-level Interrupt File
0x05C00000	2MB	Cluster-14 Supervisor-level Interrupt File
0x05E00000	2MB	Cluster-15 Supervisor-level Interrupt File
0x06000000	2MB	Cluster-0 S-level MMR Region
0x06200000	2MB	Cluster-1 S-level MMR Region
0x06400000	2MB	Cluster-2 S-level MMR Region
0x06600000	2MB	Cluster-3 S-level MMR Region
0x06800000	2MB	Cluster-4 S-level MMR Region
0x06A00000	2MB	Cluster-5 S-level MMR Region
0x06C00000	2MB	Cluster-6 S-level MMR Region
0x06E00000	2MB	Cluster-7 S-level MMR Region
0x07000000	2MB	Cluster-8 S-level MMR Region
0x07200000	2MB	Cluster-9 S-level MMR Region
0x07400000	2MB	Cluster-10 S-level MMR Region
0x07600000	2MB	Cluster-11 S-level MMR Region
0x07800000	2MB	Cluster-12 S-level MMR Region
0x07A00000	2MB	Cluster-13 S-level MMR Region
0x07C00000	2MB	Cluster-14 S-level MMR Region
0x07E00000	2MB	Cluster-15 S-level MMR Region

Table 26. Address Mapping for Memory Mapped Registers in Ascalon Cluster

If a system has less than 16 clusters, MMR access to unimplemented MMR registers should be ignored. This means that the read operation returns 0, and the write operation does not take effect. Implementation may return an error response.

System must not hang up even if memory access for unimplemented MMR happens.

6.2.1. Machine-level Interrupt File

Each cluster has up to 8 RISC-V harts. This means that up to 8 machine interrupt files are defined in the cluster MMR. The access to this region is expected to be prohibited by the PMP, so that only M-mode software can access to this region.

MMR address	MMR size	Description
0x000000	256KB	Core-0 machine-level interrupt file

MMR address	MMR size	Description
0x040000	256KB	Core-1 machine-level interrupt file
0x080000	256KB	Core-2 machine-level interrupt file
0x0C0000	256KB	Core-3 machine-level interrupt file
0x100000	256KB	Core-4 machine-level interrupt file
0x140000	256KB	Core-5 machine-level interrupt file
0x180000	256KB	Core-6 machine-level interrupt file
0x1C0000	256KB	Core-7 machine-level interrupt file

Table 27. Address Mapping for Machine-level Interrupt Domain

For each 256KB space, only write operations for first 8B can take effect to the system. Write opearations for remaing addresses space are ignored. Any read operation in this domain returns 0.

6.2.2. Supervisor-level Interrupt File

Each cluster has up to 8 RISC-V harts. This means that up to 8 supervisor-level interrupt files are defined in cluster MMR.

MMR address	MMR size	Description
0x000000	256KB	Core-0 supervisor-level interrupt file
0x040000	256KB	Core-1 supervisor-level interrupt file
0x080000	256KB	Core-2 supervisor-level interrupt file
0x0C0000	256KB	Core-3 supervisor-level interrupt file
0x100000	256KB	Core-4 supervisor-level interrupt file
0x140000	256KB	Core-5 supervisor-level interrupt file
0x180000	256KB	Core-6 supervisor-level interrupt file
0x1C0000	256KB	Core-7 supervisor-level interrupt file

Table 28. Address Mapping for Supervisor-level Interrupt Domain

Each supervisor-level interrupt file supports up to 5 guest supervisor-level interrupt files. Each guest interrupt file occupies 4KB memory address space. The 256KB region can cover up to 63 guest interrupt files, which is the maximum guest interrupt file support specified in the Advanced Interrupt Architecture specification.

6.2.3. Cluster M-level MMR Region

The Cluster MMR Region has 32 subdomains (64KB for each space). Each subdomain has 64KB of address space. The access to this region is expected to be prohibited by the PMP.

Memory access requests for address range 0x180000-0x1FFFFF are forwarded to an external interface, if IP is configured to forward the packet to external ports. The details on how to configure the RTL is described in Section 7.1.

MMR address	MMR size	Location	Description
0x000000	64KB	AXI-Ring (Cluster Clock Domain)	Core-0 MMR
0x010000	64KB	AXI-Ring (Cluster Clock Domain)	Core-1 MMR
0x020000	64KB	AXI-Ring (Cluster Clock Domain)	Core-2 MMR
0x030000	64KB	AXI-Ring (Cluster Clock Domain)	Core-3 MMR
0x040000	64KB	AXI-Ring (Cluster Clock Domain)	Core-4 MMR
0x050000	64KB	AXI-Ring (Cluster Clock Domain)	Core-5 MMR
0x060000	64KB	AXI-Ring (Cluster Clock Domain)	Core-6 MMR
0x070000	64KB	AXI-Ring (Cluster Clock Domain)	Core-7 MMR
0x080000	64KB	AXI-Ring (Cluster Clock Domain)	Trace Funnel, Trace Sink
0x090000	64KB	AXI-Ring (Cluster Clock Domain)	
0x0A0000	64KB	AXI-Ring (Cluster Clock Domain)	
0x0B0000	64KB	AXI-Ring (Cluster Clock Domain)	
0x0C0000	64KB	AXI-Ring (Cluster Clock Domain)	
0x0D0000	64KB	AXI-Ring (Cluster Clock Domain)	

MMR address	MMR size	Location	Description
0x0E0000	64KB	AXI-Ring (Cluster Clock Domain)	
0x0F0000	64KB	AXI-Ring (Cluster Clock Domain)	
0x100000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x110000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x120000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x130000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x140000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x150000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x160000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL
0x170000	64KB	AXI-Bridge (Fabric Clock Domain)	CPL (Power Management Network)
0x180000	64KB	AXI-Bridge (Fabric Clock Domain)	ACLINT
0x190000	64KB	AXI-Bridge (Fabric Clock Domain)	Debug Module
0x1A0000	64KB	AXI-Bridge (Fabric Clock Domain)	Shared Cache MMR
0x1B0000	64KB	AXI-Bridge (Fabric Clock Domain)	Cluster Config MMR
0x1C0000	64KB	AXI-Bridge (Fabric Clock Domain)	
0x1D0000	64KB	AXI-Bridge (Fabric Clock Domain)	
0x1E0000	64KB	AXI-Bridge (Fabric Clock Domain)	
0x1F0000	64KB	AXI-Bridge (Fabric Clock Domain)	

Table 29. Address Offset in M-level MMR Region

6.2.4. Cluster S-level MMR Region

No register is defined for this region.

After internal discussion, we concluded we have no use case for an S-mode MMR region. Thus, there is no MMR belonging to this region.

6.2.5. Memory Mapped Register Detail

Section 9.6 shows the list of MMR in Ascalon cluster.

Core MMR

Reset Vector Register and Non-Maskable Interrupt Vector Register tells the 64-bit register to hold the instruction address. Bit-63 is treated as "Lock" bit. When lock bit is 1, the write to the register is ignored by the hardware.

Shared Cache MMR

Shared Cache has 64KB of address space for the MMR register. The MMR register covers (1) configuration of shared cache, (2) RAS error recording registers, (3) QoS registers, and (4) Performance Counters. The shared cache address map is distributed by PA[15:12]. Each 64B behaves as a single endpoint. Any access to this region should be done by strongly ordered IO memory access.

All MMR register access supports naturally aligned 4B or 8B access only. The other memory's access size may fail to read / write the resource.

Shared Cache is reponsible for returning a response for the entire 64KB space.

Shared Cache RAS MMR

Shared Cache provides a RERI Compliant RAS Unit/Error Bank per slice, supporting 3 error records.

Shared Cache QoS MMR

Shared cache has one MMR register for QoS support.

Shared Cache Performance Counter MMR

Shared Cache supports total 32 of 8B performance counters. 32 performance counters are distributed to 4 register banks. Each register bank is distributed to 16 slices.

ACLINT MMR

ACLINT covers the machine timer (`mtime`) and corresponding compare registers (`mtimecmp*`). This unit also covers revisions of Ascalon Cluster. Revision registers are read-only registers for Ascalon Cluster RTL.

Trace Unit MMR

Trace unit has MMR space for Trace Sink, Trace Funnel, and TRACE SRAM Buffer. Trace SRAM Buffer space is typically readable from Trace Sink MMR. Therefore, Trace Sink SRAM space is for hardware debug purposes.

Debug Module MMR

All debug registers are mapped to an M-level MMR domain.

Debug Module Execution Memory

This space contains instructions / data for communication between the CPU and the DM. This region is accessible only from local CPU cores in the cluster.

Debug Module AXI DMI Interface

This space is to inject DMI from the AXI interface. Read / write requests follow the definition in DMI. A custom register holds the permission if the DM accepts internal debugger access.

CPL

CPL resource is mapped from multiple agents. The external address mapping is listed in Table 6.

Chapter 7. IP Integration

This Ascalon CPU Cluster is integrated in Tenstorrent's SoCs for next generation AI acceleration infrastructure.

7.1. Verilog Parameters

Ascalon Cluster takes the following verilog parameters to configure RTL appropriately:

Parameter Name	Description
`CLUSTER_MMR_BASE[51:27]`	Base address for Ascalon Cluster MMR (Default:0x8, Addr[51:0]=0x4000_0000)
`CLUSTER_SP_BASE[51:28]`	Base address for Ascalon Scratch Pad Memory (Default:0x6, Addr[51:0]=0x6000_0000)
`CLUSTER_LOCAL_SRCID[3:0]`	Source ID for Local Resource (Default:0001)
`CLUSTER_REMOTE_SRCID[3:0]`	Source ID for Remote Resource (Default:0000)
`CLUSTER_CPL_SRCID[3:0]`	Source ID for CPL Resource (Default:0011)
`DEFAULT_RESET_VECTOR[55:2]`	Out of reset value of reset vector (Default:0)
`DEFAULT_NMI_INTERRUPT_VECTOR[55:2]`	Out of reset value of NMI interrupt vector (Default:0)
`DEFAULT_NMI_EXCEPTION_VECTOR[55:2]`	Out of reset value of NMI exception vector (Default:0)
`DEFAULT_PMA_REGISTER[15:0][63:0]`	Out of reset value of PMA register (default 0xFC00_0000_0000_0017, IO channel-0 for entire space)

Table 30. Verilog Parameter for Ascalon Cluster

The detail is described in the rest of this section.

7.1.1. Base Address for Memory Mapped Register

Ascalon Cluster has the parameter to specify the base address for the Ascalon memory mapped registers. This information is required to differentiate the memory access requests for internal register / external registers.

This region specifies 128MB of address space. PA[24:21] is the cluster ID.

7.1.2. Base Address for Scratch Pad Memory

Ascalon Cluster has the parameters to specify the base address for scratch pad memory space. This information is required to differentiate the memory access requests for internal register / external registers.

This region specifies 256MB address space. PA[27:24] is the cluster ID.

7.1.3. Source ID Assignment

AXI network has a 4-bit sideband to specify the origin of memory access. Ascalon Cluster can assign the following three source IDs for outgoing requests:

- CLUSTER_LOCAL_SRCID: traffic generated from local CPU cores
- CLUSTER_REMOTE_SRCID: traffic generated from other CPU cores
- CLUSTER_CPL_SRCID: traffic generated from CPL

7.1.4. Out of Reset Value

After reset, some implementation-defined registers are initialized to platform specific values. These are the list of the registers:

- Reset Vector Register
- NMI Interrupt Vector Register
- NMI Exception Vector Register
- PMA Register

These registers are initialized to the value specified by the verilog parameter.

If the initial value is marked as `locked`*, software cannot update the registers.*

Chapter 8. Software Sequence

This section describe basic operations for Ascalon Cluster IP. Before allowing agents to access internal resources, a reset of all other chiplets must be completed.

8.1. Boot / Reset Sequence

This section describes the expected reset sequence. This sequence expects an external agent to drive reset signals and configurations.

In this section, SMC is treated as a SOC-level (chiplet-level for Ascalon) management controller. CPL will communicate with SMC to finish initializing the entire Ascalon Cluster.

Unless clarified in the text, at least 16 SOC cycles are expected from one step to another step. For example, if step 1 is "SMC deasserts reset pin" and step 2 is "SMC sends command to CPL", CPL can expect at least 16 idle cycles after reset deassertion before the first command from SMC.

8.1.1. Boot Sequence (Cold Reset)

This sequence is used after boot sequence.

- SMC activates SOC clock and Fabric Clock (reference clock is always stable)
- SMC asserts `cluster_cold_reset_n` by resetting unit in SMC
- `cluster_cold_reset_n` asserts `internal_warm_reset_n` to initialize the system
- SMC programs configuration registers (e.g., chiplet ID)
- SMC deasserts `cluster_cold_reset_n`
- SMC initiates SRAM repair if required (this step could be earlier than this point)
- SMC programs configuration registers in CPL based on fuse information (`cpl_config0` and `cpl_config1`)
- SMC writes CPL firmware into CPL SRAM from System Management Network
- SMC unsets `cpl_mmr_nofetch` (reset vector register for CPL) so that CPL can start fetching instructiona (CPL boot completion)
- CPL activates cluster PLL (via programming the registers) and waits until the PLL is locked (to lowest frequency)
- CPL notify "PLL initialization completion" to SMC. This is done by interrupt wire. SMC may take interrupt or polling the register to check the PLL initialization.
- CPL deasserts `cpl_mmr_reset_n`
- CPL programs MMRs in the cluster
  - Reset Vector for each core
  - NMI Interrupt / Exception Vector for each core
  - Configuration Registers for various components
- CPL sends "Reset Completion" message to SMC. This is done by interrupt wire. SMC may take interrupt or polling the register to check the PLL initialization.
- After SMC waits for the all ready signal, SMC sends "Start Execution" to CPL. This can be done by programming mailbox in CPL or setting certain MMR register in CPL.
- CPL deasserts No Fetch signal (CPU boot completion)
- Each hart starts instruction execution
- Eventually, M-mode software enables NMI in boot sequence

8.1.2. Warm Reset Sequence

This sequence is used after unexpected system events (e.g., System Hang). If the system management is still healthy, the system management components populate available information from available registers and perform a shutdown (or reboot) of the system.

- SMC detects the critical system events, like double bit errors in some structures
- SMC communicates with other SMCs in SOPs to reset the system

- SMC programs `cpl_mmr_sramhold` to 1 if it wants to populate SRAM contents
- SMC requests CPLs to assert `cpl_mmr_reset_n`
- CPL asserts both `cpl_mmr_reset_n` and `cpl_mmr_nofetch`
- CPL sends "Reset Completion" message to SMC
- After SMC waits for the all ready signal, SMC sends "Start Execution" to CPL
- CPL deasserts No Fetch signal (CPU boot completion)
- Each hart starts instruction execution

8.2. CPL Fatal Event (CPL Warm Reset)

When CPL detects fatal errors (e.g., double bit error in SRAM, watchdog timeout, emergency shutdown requests), CPL immediately stops the cluster clock. SMC in the system will try to recover the state by performing the following steps:

- SMC asserts `cpl_mmr_reset_n` in SMC's reset unit
- CPL enforces minimum frequency because of warm reset
- SMC deasserts `cpl_mmr_reset_n` in SMC's reset unit
- SMC collects error information if required
- SMC downloads firmware
- SMC sets CPL's reset vector so that CPL restart the management
- CPL updates the PLL setting

CPL warm reset flow will not initialize critical registers (e.g., configuration registers).

8.3. NDM Reset Sequence

- DM receives NDM reset request
- DM asserts `ndmreset_request` signal to SMC
- SMC forwards the request to other chiplets (e.g., other computeifndef::release[] chiplet, IO chiplet, etc...)
- SOP DM (FW on SMC) receives NDM reset request
- SMC receives `ndmreset_process` signal
- SMC starts asserting `cpl_mmr_reset_n` with `cpl_mmr_dm_hold` and `fabric_warm_reset_n`
- SMC communicates to each other to complete the reset sequence
- SMC deasserts reset signals
- SOP communicates to each other to confirm reset completion
- SMC deasserts `ndmreset_process` to DM
- Local DM finish NDM reset process

Chapter 9. Appendix

This chapter includes Tenstorrent's specifications associated with the Ascalon Cluster specifications.

9.1. Physical Memory Attribute

The PMA structure is the set of registers that define physical memory attributes. An attribute may specify the memory type (I/O or DRAM), access permission, etc... For Ascalon, the CPU creates new sets of implementations defined as CSR registers to support a PMA structure.

9.1.1. Physical Memory Attribute

The following table presents available memory attributes. The table summarizes available physical memory attributes per physical memory type. When software programs have an unsupported value, the hardware may convert the line to be a permitted legal value.

	Normal Memory Cacheable	Normal Memory Non-cacheable	I/O Memory
Access Size	Any	Any	Any
Access Permission	RWX, None	RWX, None	R, X, RW, RX, RWX, None
Supported Atomic	AMOArithmetic	AMONone	AMONone
LR/SC Reservability	RsrvEventual	RsrvNone	RsrvNone
Misaligned Atomic	No	No	No
Cacheability, Coherency	Cacheable, Coherent	Non-Cacheable	Non-Cacheable
Routing	Coherent Network	Coherent Network, Non-Coherent Network	Non-Coherent Network
I/O access ordering	N/A	N/A	Relaxed, Ordered Channel 0, Ordered Channel 1
Idempotency	Idempotent	Idempotent	Non-Idempotent
I/O Request Combining	N/A	N/A	Yes or No

Table 31. PMA Attribute

Ascalon CPU does not support misaligned memory access for I/O memories. When a CPU accesses to I/O memory by misaligned address, the CPU raises an access fault. **None** stands for the no access permission. If memory access hits in the region with **None**, the access will result in an access fault.

Idempotent memory access must not have side-effects. Non-Idempotent memory access may have side effects.

This table is for the Ascalon CPU, future designs may support misaligned atomic operation and AMO for non-cacheable memories.

"AMOArithmetic" supports all AMO operations (please refer to RISC-V Privileged Spec for further detailed definition.)

When cacheability is overwritten by PBMT (Page-Based Memory Type), the supported access type from PMA will be ingored. For the **NC** type, the memory access is treated as "Normal Non-Cacheable Memory with Non-Coherent Network routing."" For the **IO** type, the memory access is treated as "Stronly Ordered Channel 0 without Combining Support and RWX access permission."

PBMT does not give new access permission for the overwritten memory space. For eaxmple, PBMT cannot give read permission if the memory has no read permission in PMA. Vacant memory (no memory behind the address) is treated as "Access Permission is None." This means that PBMT cannot make a vacant page accessible for vacant memory.

Scratch Pad Support (Routing Information)

Scratchpad memory is defined for efficient communication with AI/ML accelerators. This memory domain is mapped to a data array in the shared cache. This PMA gives a hint to the CPU on how to route the request.

Read/Write Combining Support (I/O Attribute Consideration)

Unlike x86 or Arm, RISC-V default PMA does not specify if the write operation can be “combined (in x86)” or “gathered (in Arm).” This operation is often important to support certain devices (e.g., PCIe devices) in a high performant manner.

If write combining is enabled, the CPU may initiate combined read/write operations. For example, combining the CPU crack unit-stride load/store operation from vector load/store operation into the element level. If the combining is allowed, the CPU treats the unit-stride operation as a single large load/store operation (e.g., a 32B consecutive load/store operation.)

PCI Express prefetchable memory regio permits write combining, while non-prefetchable memory region does not permit it.

9.1.2. CSR for PMA Structure

For Ascalon CPU, PMA is specified by a set of CSR registers. The CSR registers are expected to be initialized during boot timing by machine mode. After the machine mode sets up the PMA, the CPU keeps the PMA region as a constant region unless the machine mode finds new hardware in the system (e.g., hotplug).

PMA CSR registers have similar formats with Physical Memory Protection (PMP) CSRs. Each register specifies the region of memory. The region must be a power-of-2 address region. The corresponding attribute is also specified in the CSR register.

CSR Name and Address.

To organize PMA structure, Ascalon CPU defines up to 32 PMA registers. CSR is mapped to machine-level custom CSR. The registers are WARL (Write Any Value, Read Legal Value).

The Ascalon CPU implements 16 PMA registers (`pmacfg00`, `pmacfg01`, `pmacfg02`, ..., `pmacfg15`).

When illegal data is written into `pmacfg*`, the Ascalon CPU retains the previous value.

PMA name: `pmacfg00`, `pmacfg01`, `pmacfg02`, ..., `pmacfg31` CSR address : 0x7E0 - 0x7FF (`pmacfg00` - `pmacfg31`)

CSR address	CSR name	Definition
0x7E0	pmacfg00	CSR register defines PMA region 0
0x7E1	pmacfg01	CSR register defines PMA region 1
0x7E2	pmacfg02	CSR register defines PMA region 2
...	...	...
0x7FF	pmacfg31	CSR register defines PMA region 31

Table 32. CSR Address for PMA registers

To check PMA register availability, machine mode should write dummy data into the corresponding PMA and read the value. For example, software may write 0xFFFFFFFF00000000, followed by reading the same register to check the size. If the return value is 0, there is no PMA register.

CSR Format

pmacfg* has a 64-bit field to specify the attribute, the hart interprets PMA attributes as shown below.

	Ascalon Definition	Permitted behavior
[2:0]	Permission	[0] Read permission, [1] Write permission, [2] Execute permission
[4:3]	Memory Type	00: Main Memory, 01: I/O memory (relaxed ordering), 10: I/O memory (channel 0), 11: I/O memory (channel 1)
[6:5]	AMO Type	00: AMONone, 01: AMOSwap, 10: AMOLogical, 11: AMOArithmetic
[7] (Main Memory)	Cacheability	1: Cacheable, 0: Non-Cacheable
[7] (I/O Memory)	Combining Capability	1: Request Combining Allowed, 0: Combining Not Allowed
[8]	Routing (Coherency)	1: Routing to Coherent Network, 0: Routing to Non-Coherent Network
[11:9]	Reserved	Reserved for Future Use
[51:12]	PA: Physical Address [51:12]	Address contained by this PMA region
[57:52]	Reserved	Reserved for Future Use
[63:58]	SIZE: Physical Address Size	How many LSB should be masked. If the value is 16, bottom 16-bit is masked (structure specify 64KB of memory region). When SIZE=0, PMA entry is treated as invalid (no PMA entry match).

Table 33. Bit field for pmacfg registers*

The `pmacfg*` is WARL. If an unsupported value (e.g., AMONone for Cacheable Normal Memory) in Table 31 is written, the written value is IMPLEMENTATION defined. For the Ascalon CPU design, RTL treats the entire PMA CSR as a collective WARL field. Any

write with non-supported value in Table 31 prevents the update to the `pmacfg*` CSR.

interpretation of [7] is affected by the value of Memory Type ([4:3])

Address Matching

The PMA address matching process is done by the following computation. If the SIZE field is programmed a value larger than 56, then any address will match with the corresponding PMA entry. If the SIZE field is programmed as 0, the entry will never make a match. When the size field is programmed 0 < SIZE < 12, the implementation treat the write as the illegal value. Based on the WARL nature of this register, implementation update register with some other legal value or the write may be ignored.

```
const uint64_t PA_WIDTH = 52
bool PMA_address_match (pmacfg_t pma, address_t pa) // pa is checking physical address
{
  if ((pa >> PA_WIDTH) & ((1 << (56-PA_WIDTH-STEE_ENABLED))-1)) return false; // PA[54:52] != 0 cause PMA
mismatch
  if (pma.SIZE == 0) return false; // SIZE=0 entry won't match any entry
  if (pma.SIZE < 12) return IMPLEMENTATION_DEFINED_VALUE; // behavior for 0 < SIZE < 12 is not specified.
  uint64_t mask = (~0ULL << pma.SIZE) & ((1ULL << PA_WIDTH) - 1ULL) // 0x000F_FFFF_FFFF_F000 if SIZE=12
  return ((pma.PA << 12) & mask) == (pa & mask)
}
```

Ascalon CPU does not update register for WARL register with illegal value

Constraint

If the PMA register is programmed to cover unsupported behavior in this section, the implementation treats illegal value. For example, PMA specifies normal cacheable memory as AMONone is illegal value since normal cacheable memory must be AMOArithmetic (Table 31).

Ascalon CPU does not update register for WARL register with illegal value

9.1.3. PMA Check

When the CPU accesses the memory subsystem, it accesses the PMA structure by PA, in order to get the corresponding memory attribute. The CPU will check if the operation is permitted by using memory attribute.

Update Sequence

PMA attributes can be cached in address translation cache structures (e.g., TLB). This means that software must initiate appropriate translation cache invalidation operations (e.g., TLB invalidation operations) to squash stale PMA information cached in the translaiton caches.

Misbehavior

If none of the PMA entry is matched for a given memory access, the CPU should report an access fault.

Multi-Hit Behavior

If multiple PMA entries (`pmacfgX0`, `pmacfgX1, …, `pmacfgXn` where X0 < X1 < … < Xn) are matching to the same physical address, a PMA check result should be done by the attribute from the youngest PMA entry (`pmacfgX0`). This restriction follows the multi-hit policy of PMP entry matching policy.

The System PMA is treated as the highest priority PMA. This means that the System PMA can be treated as `pmacfg_minus1,2,3` *so that it can naturally overwrite attribute from* `pmacfg0.`

Reset Value

After reset, PMA structure is initialized to a certain pre-determined attribute (e.g., 0x0000FFFF - 0x00000000, IO memory) so that machine mode can initialize the system. This initial PMA value should be able to be specified by a Verilog parameter.

For the Ascalon CPU, "Ordered IO memory (channel-0) for entire memory address space" is the expected reset value for `pmacfg00` to avoid unnecessary memory accesses. This reset value prevents any speculative memory accesses for the entire address space

until the PMA is appropriately programmed.

9.2. Static Trusted Execution Environment (STEE)

AP STEE offers a trusted execution environment for software to handle security assets that don't trust the security assurance offered by regular operating systems. STEE works in such a way that the resources within it, like memory and address aperture, are isolated from the rest of the system either at design time or boot-up time. ARM TrustZone is the most popularly used STEE.

Ascalon offers STEE support to software. The goal is to emulate ARM TrustZone. The compatibility with ARM TrustZone can smooth the migration from ARM cores to Ascalon. This is because 1) the rest of the system can stay the same - TrustZone aware subsystems can be kept and re-used; 2) Software for TrustZone only needs minimal changes to run on Ascalon.

9.2.1. PMP Based Static TEE for Ascalon

Ascalon's STEE solution is based on PMP, a standard feature in RISC-V. What's described here is a recommended way to emulate TrustZone. Users can use PMP in other ways to obtain other types of STEEs.

Using PMP, Ascalon can emulate TrustZone as depicted in Figure 16.

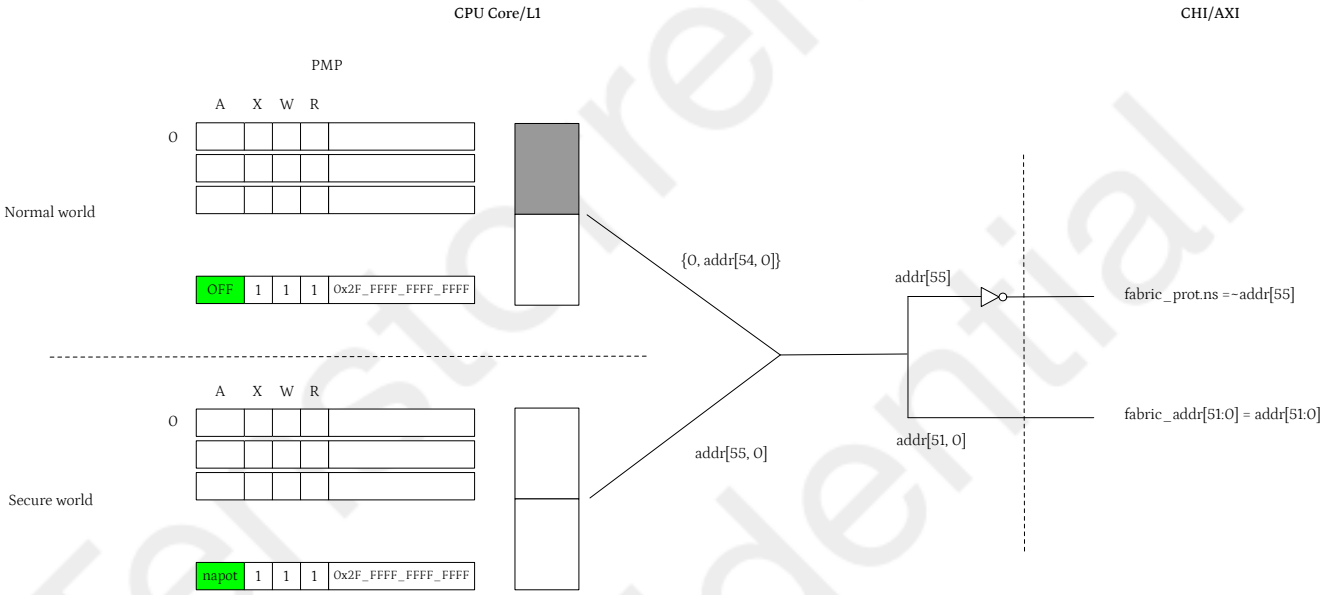

Figure 16. TrustZone Emulation Based on PMP

The Secure Monitor running in M mode is in charge of the world switch by configuring the PMP. According to the PMP setting, the normal world only sees the lower half of the address space, while secure world sees the full address space. The MSB drives the prot.ns bit on fabric, which is further used by the destination to tell if a transaction can access secure world assets.

If both normal world and secure world enable MMU by their own OSes, a TLB flush needs to be done on world switch. ARM introduced NSTID to avoid the need for a TLB flush. Something similar can be done here: a CSR only accessible by M mode is used as the Secure World Identifier (SWID). When an entry is added to the TLB, SWID is added as part of the tag to differentiate secure world entries from normal world entries.

The minimum number of PMP entries needed for TZ emulation is 1, as shown in Figure 16. In a more general case, 2 entries would be needed.

The PMP based emulation occupies the MSB of the CPU physical address (bit 55) to differentiate secure and normal accesses. As shown in Figure 16, the CPU still sends out the full 56-bit address. It is the cluster/SoC integration's responsibility to connect the MSB to prot.ns according to the need of STEE support. If large systems needs all 56 address bits, this PMP based TZ emulation cannot be used. In such a case, X86 is the comparison target, which has no static TEE to emulate.

9.2.2. World Switch Performance Improvement

The secure and normal worlds have their own HS/VS/(V)U privileges. The VMID and ASID for the two worlds may collide as they may not necessarily be managed by one single piece of software. To avoid the need of TLB flushing on world switch, Ascalon introduces a Secure World Identifier (SWID).

The custom Machine Address Translation and Protection (`matp`) CSR is a MXLEN-bit register, formatted as shown in Figure 17. It is readable and writable by M-mode, Debug-mode, and patch-mode. This register holds a SWIDLEN-bit Secure World Identifier (SWID) field. The number of implemented SWID bits may be determined by writing one to every bit position in the SWID field, then reading back the value in `matp` to see which bit positions in the SWID field hold a value of 1. The least-significant bits of

SWID are implemented first — that is, if SWIDLEN > 0, SWID[SWIDLEN-1:0] is writable. The maximal value of SWIDLEN, termed SWIDMAX, is 7. This means the number of worlds can be 2^n, where n can be from 0 to 7. For Ascalon, SWIDLEN equals 1.

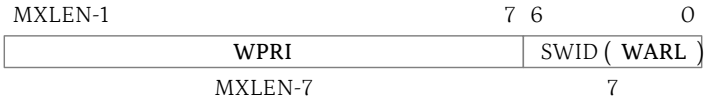

Figure 17. Custom Machine Address Translation and Protection CSR (`matp`)

The address of `matp` CSR is 0x7C7.

SWID impacts ITLB/DTLB/L2TLB in a similar way as VMID/ASID. When creating a TLB entry, the associated SWID of the memory access is copied to the TLB entry's tag. When checking a virtual address against TLB entries, the associated SWID is compared against the SWID field in the TLB entries. A TLB match always requires a SWID match.

Global mapping of TLB entries only applies within their own world. In other words, for a global mapping TLB entry with G bit set to 1, a successful match not only requires the virtual address to match, but also needs the SWID to match.

The associated SWID of a memory access shows which world issued the memory access. There can be different ways to implement the associated SWID. Ascalon chooses an easy way such that the associate SWID matches the SWID field in **`matp`**. This requires M-mode software to wait until all memory accesses in one world are complete before changing the SWID field in **`matp`**.

TLB entry invalidation applies within its own world. This means one world shall not be capable of invalidating TLB entries that belong to another world.

With the above micro-architecture definition, no TLB flushing is needed on a security world switch.

9.2.3. Interrupt Support for STEE

Special items for SoC integration are the interrupts dedicated to the secure world(s), such as, the timer interrupts for secure hypervisor and secure supervisor.

ARM TrustZone's secure interrupt architecture has two properties:

1. Interrupt for the inactive world gets trapped to EL3 (Machine Mode equivalent). The interrupt handler running in EL3 routes the interrupt accordingly.
2. Interrupt for the active world doesn't get trapped to EL3 but instead goes directly to its destination.

To do something similar, ideally the architectural requirements for STEE are as follows:

- Each CPU core shall have dedicated timer comparison and correspondingly dedicated interrupts for Secure Supervisor and Secure Hypervisor per Secure World. The timer setting is done by Machine Mode and the timer interrupt settings shall only be accessible by the corresponding secure world.
- Interrupt controller, like RISC-V Message-Signaled Interrupt module and PLIC, shall be secure-world aware with secure interrupt(s) separated from normal interrupt(s). It shall also make the secure world(s)' interrupt status and control registers only accessible by the secure world(s). The interrupt controller will at least check the expected NS value of the incoming interrupt transaction.

Ascalon, the first generation of TT's RISC-V CPU core, chooses a simpler scheme:

- There is no additional timer comparison and correspondingly dedicated interrupts for Secure Supervisor and Secure Hypervisor per Secure World. During world switch, the Secure Monitor performs a context switch of **`stimecmp`** and **`vstimecmp`**. This means that while one world is active, timer interrupts from other world(s) are disabled.
- The interrupt controllers are not STEE aware. This means that all secure interrupts are routed to Machine Mode.
- During world switch, delegation of the old world's interrupt is disabled. This is mainly for the normal world, as the secure world routes its interrupt to Machine Mode anyway.

9.2.4. Side-Channel Attack Mitigation

In addition to VMID/ASID, SWID needs to be used to derive secret keys to encrypt the prediction table's contents.

On automated flushing of the fetch predictor and return address stack, the context switch detection shall include change of SWID.

9.2.5. Other Micro-Architectural Security Requirements

Besides the architectural operations on TLB, additional proprietary operations on TLB shall not be implemented, except for debug operations under secure debug control.

For a memory access, a cache hit will still be required to pass the PMP check.

Configuring Shared Cache (SC) MMR shall not compromise STEE's data or change ownership of any data in SC. Below are some examples:

- configuring SC MMR cannot modify STEE data with NS tag == 0
- configuring SC MMR cannot change the NS tag of a line while still keeping it valid
- configuring SC MMR cannot invalidate a line with NS tag == 0 without flushing its updated value to M$ or DRAM

With the above requirements, there is no need to do traffic filtering based on NS for the SC MMRs and the ScratchPad.

9.2.6. Integration Requirements

If TrustZone emulation is needed, the Cluster/SoC integrator will create the connection to buses as shown in Figure 16. The NS bit will be carried through by both coherent fabric (e.g. CHI) and non-coherent fabric (e.g. AXI).

Each Ascalon Cluster shall implement the following secure reset input from either the Security Processor (SEP) or SMC (TBD).

Traffic destinations shall check the prot.ns bit of the inbound traffic and will only allow the traffic to access secure world assets when prot.ns equals the expected value.

Intermediate storages for secure world assets shall also check the prot.ns bit of the inbound traffic, and will only allow the traffic to access secure world assets when prot.ns equals the expected value.

Here are some examples:

- Shared Cache (L2) and fabric proxy cache shall include a NS bit in its tag. The NS bit gets the value of the incoming transaction's prot.ns during cache line allocation. It is further compared with the incoming transaction's prot.ns during cache line match. On cache line eviction, the NS bit drives the outbound traffic's prot.ns.
- STEE region in DRAM shall be checked against incoming transactions to the memory cache. This avoids adding the NS bit to memory cache's tags.
- BTW, L1 cache already has PA[55] in the cache line's tag. No change is needed.

Fabric of the CPU cluster and SoC shall both support prot.ns by treating prot.ns as an address bit. The only difference between an address bit and prot.ns is that prot.ns is not used for traffic routing.

Traffic sources shall guarantee the prot.ns output equal to the expected value for all their traffics. Internally, the traffic originator shall have the concept of secure worlds. They shall tie the SWID information to the outbound traffic, either by adding the SWID information to the traffic itself or by introducing a context switch with a global SWID. Such SWID information will be used to select the page table for IOMMU. It will also be used to qualify the MSB of the physical address after IOMMU translation or bypass. Eventually the qualified PA MSB will be driving the prot.ns bit(s) while the actual PA MSB sent out will be tied to 0.

The scope of STEE should be limited to the scope of pure computation. It doesn't aim to cover IO devices. However, it doesn't mean STEE cannot have trusted communication with the outside. STEE will use a higher level crypto protocol to achieve this goal. This actually aligns with the fundamental requirements for STEE software - communication stack and device drivers stay outside of STEE for higher robustness of the secure world.

Following the above guidelines, Tenstorrent's first gen of chiplets see that Ascalon CPU Cores and Tensix Cores are the traffic sources which need to follow the requirements defined above. All other traffic sources should have PMP-like logic to limit them to the non-secure world with their NS tied to 1.

9.2.7. Software Requirements

A Secure Monitor is required to run in M mode to handle world switch routines using an ECALL handler at run time. Different world switch routines should all adhere to the following flow in order to securely transition from one world to another:

- Wait for all memory transactions to complete
- Save the previous context
- Restore the next context

- Update the PMP setting according to Figure 16
- Flip the Machine Mode interrupt delegation setting for the normal world interrupts.
- If `matp` is implemented, update SWID field in `matp` CSR, otherwise run SFENCE.VMA instructions with rs1=x0 and rs2=x0

To clarify more on preemptive multi-tasking, as world switching has to go through the Secure Monitor in Machine Mode, preemptive multi-tasking across multiple worlds needs to be handled by Machine Mode software. An example implementation could be using a Machine Mode timer to alternate between worlds, where interrupts for the inactive world remain enabled but handled by the Machine Mode without delegation. In this case, interrupts for the inactive worlds could bring the HART to the Machine Mode interrupt handler, which further routes the interrupt to the expected privilege. It then switches to the target world, which later receives the interrupt and handles it appropriately.

9.2.8. Security Analysis

Assets for different worlds and the security solution are as follows:

- Architectural states within the CPU core.
  - Enforcing the world switch through Machine Mode leverages the existing isolation mechanism for different privileges to isolate different worlds.
- Micro-architectural states within the CPU core, including cache, TLB, prediction state, request buffer for memory access, etc.
  - Cache lines don't carry the Secure World Identifier. This means that even a cache hit is required to pass the PMP check.
  - For TLB, either a TLB flush on world switch or an introduction of the SWID bit guarantees that address translation for one world won't be mistakenly used by another world. In particular, the global mapping only applies to its own world so that one world cannot use global mapping to skip the check on SWID. Besides the architectural operations on TLB, additional proprietary operations on TLB shall not be implemented, except for debug operations under secure debug control.
  - For memory access request buffers, the Secure Monitor that's responsible for the world switch needs to wait for all pending transactions to complete before initiating the world switch.
  - For predictor state isolation, SWID, together with VMID/ASID, is used to derive TAGE predictor content. Change of SWID is also used to trigger predictor state flush.
- Memory and peripheral IO.
  - CPU's PMP setting ensures the upper half of the address space to be accessible by secure world only. All secure memory or peripheral IO apertures require that the ms-prot bit driven by inversion of MSB must be 0.
  - Other SoC traffic originators leverage IOMMU and SWID information to ensure separation of SWID contexts.

9.3. Supported CHI Messages

Ascalon Cluster supports CHI messages listed in Table 34. "CMO" in the table stands for CMO combined operation, such as `WriteBackFullCleanInv`.

Channel	Message Name	Supported?	Comment and Justification
Request	ReqLCrdReturn	N	
Request	PrefetchTgt	N	
Request	PCrdReturn	N	
Request	DVMOp	N	No DVM operation in system
Request	ReadNoSnp	Y	
Request	ReadNoSnpSep	N	
Request	ReadShared	N	
Request	ReadClean	Y	
Request	ReadOnce	Y	
Request	ReadUnique	Y	
Request	ReadPreferUnique	N	
Request	MakeReadUnique	N	
Request	ReadNotSharedDirty	N	
Request	CleanShared	Y	
Request	CleanSharedPersist	N	
Request	CleanSharedPersistSep	N	
Request	CleanInvalid	Y	
Request	MakeInvalid	Y	
Request	ReadOnceCleanInvalid	N	
Request	ReadOnceMakeInvalid	N	
Request	CleanUnique	Y	
Request	MakeUnique	N	
Request	Evict	Y	
Request	WriteNoSnpPtl(CMO)	Y	No Combined CMO flow support
Request	WriteNoSnpFull(CMO)	Y	No Combined CMO flow support
Request	WriteNoSnpZero	N	
Request	WriteEvictFull	Y	
Request	WriteEvictOrEvict	N	
Request	WriteCleanFull(CMO)	Y	No Combined CMO flow support
Request	WriteBackPtl	N	
Request	WriteBackFull(CMO)	Y	No Combined CMO flow support
Request	WriteUniquePtlStash	N	
Request	WriteUniqueFullStash	N	
Request	WriteUniquePtl(CMO)	N	
Request	WriteUniqueFull(CMO)	Y	No Combined CMO flow support
Request	WriteUniqueZero	N	
Request	StashOnceUnique	N	
Request	StashOnceSepUnique	N	
Request	StashOnceShared	N	
Request	StashOnceSepShared	N	
Request	AtomicLoad	N	
Request	AtomicStore	N	
Request	AtomicCompare	N	
Request	AtomicSwap	N	

Response	RspLCrdReturn	N	No Link powerdown support
Response	SnpResp	Y	
Response	SnpRespFwded	Y	
Response	CompAck	Y	
Response	RetryAck	Y	
Response	Comp	Y	
Response	CompCMO	N	No Combined CMO flow support
Response	Persist	N	No Persist flow
Response	CompPersist	N	No Persist flow
Response	StashDone	N	No Stash Request Support
Response	CompStashDone	N	No Stash Request Support
Response	RespSepData	Y	
Response	CompDBIDResp	Y	
Response	DBIDResp	Y	
Response	DBIDRespOrd	Y	
Response	TagMatch	N	No Memory Tagging
Response	PCrdGrant	Y	
Response	ReadReceipt	Y	
Data	DatLCrdReturn	N	No Link powerdown support
Data	SnpRespData	Y	
Data	SnpRespDataFwded	Y	
Data	CopyBackWrData	Y	
Data	NonCopyBackWrData	Y	
Data	NCBWrDataCompAck	N	
Data	CompData	Y	
Data	DataSepResp	Y	
Data	SnpRespDataPtl	N	
Data	WriteDataCancel	N	
Snoop	SnpLCrdReturn	N	No Link powerdown support
Snoop	SnpShared	Y	
Snoop	SnpClean	Y	
Snoop	SnpOnce	Y	
Snoop	SnpNotSharedDirty	Y	
Snoop	SnpUnique	Y	
Snoop	SnpPreferUnique	Y	
Snoop	SnpCleanShared	Y	
Snoop	SnpCleanInvalid	Y	
Snoop	SnpMakeInvalid	Y	
Snoop	SnpSharedFwd	Y	
Snoop	SnpCleanFwd	Y	
Snoop	SnpOnceFwd	Y	
Snoop	SnpNotSharedDirtyFwd	Y	
Snoop	SnpUniqueFwd	Y	
Snoop	SnpPreferUniqueFwd	Y	
Snoop	SnpUniqueStash	Y	
Snoop	SnpMakeInvalidStash	Y	
Snoop	SnpStashUnique	Y	
Snoop	SnpStashShared	Y	
Snoop	SnpQuery	Y	
Snoop	SnpDVMOp	N	No DVM operation in system

Table 34. Supported CHI Message

9.4. Performance Counter Event Definition

The Table 35 shows the list of hardware performance monitor events defined for the Ascalon CPU core. The events in the CPU core are accessible from CSR access. The Table 36 shows the list of hardware performance monitor events defined for Shared Cache. The events in Shared Cache are accessible from MMR access.

Event ID	Name	Description
0x3000000 00	cpu_cycles	Event for each CPU cycle; implemented via the mcycle CSR
0x3400000 01	instructions	Event for each retired instruction; implemented via the minstret CSR
-1	m_mode_cycles	Simulation only event for each CPU cycle that happens while in m mode
-1	m_mode_instret	Event for each retired instruction that happens while in m mode
-1	s_mode_cycles	Event for each CPU cycle that happens while in s mode
-1	s_mode_instret	Event for each retired instruction that happens while in s mode
-1	u_mode_cycles	Event for each CPU cycle that happens while in u mode
-1	u_mode_instret	Event for each retired instruction that happens while in u mode
0x3000000 02	ref_cpu_cycles	Event for each reference CPU cycle; wall clock
0x8000000 00	stalls_bst_full	Event (speculative) for each cycle BPM1 mux is stalled due to full Branch Status Table
0x8000000 01	stalls_pfx_full	Event (speculative) for each cycle BPM1 mux is stalled due to full Prefix table
0x8000000 02	nfp_early_redirect	Event (speculative) for every instance of disagreement between next-fetch predictor and Branch Direction Predictor
0x8000000 03	nfp_late_redirect	Event (speculative) for every instance of disagreement between next-fetch predictor and Branch Target Predictor
0x1	stalls_indirect_miss	Event (speculative) for each cycle BPM1 mux is stalled due to a miss in indirect prediction via Branch Target Predictor or Return Address Stack
0x200000	stalls_icache_miss	Event (speculative) for each cycle BPM1 mux is stalled due to a miss in instruction cache
0x200001	stalls_itlb_miss	Event (speculative) for each cycle BPM1 mux is stalled due to a miss in the IC
0x200002	stalls_exception	Event (speculative) for each cycle BPM1 mux is stalled due to exceptions within IC including ITLB page access faults
0x8020000 00	stalls_irb_full	Event (speculative) for each cycle BPM1 mux is stalled due to i-side request buffers full IC/I$
0x8020000 01	stalls_ifbuf_full	Event (speculative) for each cycle BPM1 mux is stalled due to full IFBUF
0x8020000 02	page_crossing_fetchblocks	Event (speculative) for every fetch block crossing a page boundary and forced to split
0x8020000 03	ifbuf_full_redirect	Event (speculative) for every instance of redirect caused by IFBUF full
-1	fault_resync	Event (speculative) for every instance of resync due to detectable soft-errors from L2 or LS unit
-1	fault_refetch	Event (speculative) for every instance of fetch redirect due to soft-errors in frontend structures
0x2	cmode_entry	Event (non-speculative) for every instance of entering conservative mode
0x4000003	branch_misses	Event for retiring a branch instruction that mispredicted during its execution. The misprediction could be due to direction predictor or target predictor. This will not include mispredictions due to next-fetch predictor.
0x8400000 04	br_ret_misses	Event for each retired and mispredicted control flow instruction that uses the return address stack for prediction
0x8400000 05	ind_br_misses	Event for each retired and mispredicted indirect control flow instruction that uses the Branch Target Predictor. Return instructions are not included
0x8400000 06	rel_br_misses	Event for each retired and mispredicted conditional control flow instruction that uses the Branch Direction Predictor
0x4	spec_branch_redirect	Event (speculative) for every instance of branch misprediction flush
0x5	spec_lsu_resyncs	Event (speculative) for every instance of load misprediction flush
0x6	total_flushes	Event (speculative) for every flush is emitted from midcore regardless of origin

0x7	total_traps	Event for every trap emitted from midcore
-1	bst_full_on_ex_redirect	Event (speculative) when EX redirect needs extra latency due to BST full
-1	pfx_full_on_ex_redirect	Event (speculative) when EX redirect needs extra latency due to PFX full
0x200003	l1i_read_access	Event (speculative) for an L1I cache access. Only demand accesses are counted. Accesses in the shadow of a demand miss are not counted. Count once for every fetchgroup access
0x200004	l1i_read_miss	Event (speculative) for an L1I cache miss.Misses in the shadow of a demand miss are not counted
0x200005	l1i_prefetch_access	Event (speculative) for every L1I cache access for instruction prefetch
0x200006	l1i_prefetch_miss	Event (speculative) for every L1I cache miss for an instruction prefetch
0x200007	itlb_read_access	Event (speculative) for an L1I TLB demand access. Accesses in the shadow of a demand miss are not counted
0x200008	itlb_read_miss	Event (speculative) for an L1I TLB demand miss. Misses in the shadow of a demand miss are not counted
0x200009	itlb_prefetch_access	Event (speculative) for an L1I TLB access for an instruction prefetch
0x20000a	itlb_prefetch_miss	Event (speculative) for an L1I TLB miss for an instruction prefetch
0x8020000 4	ic_way_mispred	Event(speculative) for each instruction cache way misprediction
0x8000000 7	ras_underflow	Event (speculative) for every underflow event of return address stack
0x8000000 8	ras_overflow	Event (speculative) for every overflow event of return address stack
0x8020000 5	num_fetchgroups	Event(speculative) for every fetchgroup written into the IFBUF. If fetchgroup has 2 writes in the same cycle into the IFBUF it is still counted as 1 fetchgroup
0x8000000 9	bdp_bank_conflicts	Event for every cycle an update to Branch Direction Predictor is delayed due to a bank conflict
0x8000000 0a	btp_bank_conflicts	Event for every cycle an update to Branch Target Predictor is delayed due to bank conflict
0x8000000 0b	bpu_writes	Event for every write into either Branch Direction Predictor or Branch Target Predictor. Count twice if both are written in the same cycle
0x4400000 0	uops_decoded	Event (speculative) for every instruction provided from decode to Mapper
0x4400000 1	decode_serialize_cycles	Event for every CPU cycle that decode is stalling dispatch due to serialization before or after
0x4400000 2	decode_idle_serialize_cycles	Event for every CPU cycle that decode is still stalling dispatch due to idle serialization after serialization before or after has been resolved
0x8040000 00	nonspec_resync	Event for every instance of resync due to CSR access
0x8041000 01	patch_match_m_mode_excepti on	Event for every Patch RAM exception happens in regular M mode execution
0x8041000 02	patch_match_s_mode_exceptio n	Event for every Patch RAM exception happens in regular S mode executio
0x8041000 04	patch_match_u_mode_exceptio n	Event for every Patch RAM exception happens in regular U mode execution
0x8041000 08	patch_match_vs_mode_excepti on	Event for every Patch RAM exception happens in regular VS mode execution
0x8041001 0	patch_match_vu_mode_excepti on	Event for every Patch RAM exception happens in regular VU mode execution
0x8041002 0	patch_match_ucode	Event for every Patch RAM exception happens in micocode sequence
0x8442000 01	patch_match_m_mode_excepti on_cycles	Event for every cycle from when Patch RAM exception is dispatched in M mode to last uop of patch sequence (DRET) is dispatched which finishes the patch
0x8442000 02	patch_match_s_mode_exceptio n_cycles	Event for every cycle from when Patch RAM exception is dispatched in S mode to last uop of patch sequence (DRET) is dispatched which finishes the patch
0x8442000 04	patch_match_u_mode_exceptio n_cycles	Event for every cycle from when Patch RAM exception is dispatched in U mode to last uop of patch sequence (DRET) is dispatched which finishes the patch
0x8442000 08	patch_match_vs_mode_excepti on_cycles	Event for every cycle from when Patch RAM exception is dispatched in VS mode to last uop of patch sequence (DRET) is dispatched which finishes the patch
0x8442001 0	patch_match_vu_mode_excepti on_cycles	Event for every cycle from when Patch RAM exception is dispatched in VU mode to last uop of patch sequence (DRET) is dispatched which finishes the patch
0x8442002 0	patch_match_ucode_cycles	Event for every from when Patch RAM exception is dispatched during a microcode sequence to last uop of patch sequence (DRET) is dispatched which finishes the patch

0x1000000	stalled_cycles_frontend	Event fired on each cycle when backend is ready to accept a fetch bundle but frontend is not able to supply any instructions.
0x1000001	stalled_cycles_backend	Event fired on each where frontend is ready to supply at least one instruction but backend is not able to accept. This should be measured at stage MC0
0x9020000	cycles_no_int_prn	Event (speculative) for each cycle Mapper is stalled due to no integer physical registers
0x9020001	cycles_no_fp_prn	Event (speculative) for each cycle Mapper is stalled due to no floating point physical registers
0x9020002	cycles_no_vec_prn	Event (speculative) for each cycle Mapper is stalled due to no Vector physical registers
0x9020003	cycles_no_vl_prn	Event (speculative) for each cycle Mapper is stalled due to no Vector length physical registers
0x9020004	cycles_no_vm_prn	Event (speculative) for each cycle Mapper is stalled due to no Vector mask physical registers
0x9020005	cycles_no_rob	Event (speculative) for each cycle Mapper is stalled due to no reorder buffer entries
0x1460000	dispatched_nops	Event for every NOP instruction dispatched
0x1461000 1	op_retired_direct_branch	Event for each retired direct control flow instruction
0x1461000 2	op_retired_ret_branch	Event for each retired control flow instruction that uses the return address stack for prediction
0x1461000 4	op_retired_indirect_branch	Event for each retired indirect control flow instruction that uses the Branch Target Predictor for prediction. Return instructions are not included
0x1461000 8	op_retired_cond_branch	Event for each retired conditional control flow instruction that uses the Branch Direction Predictor
0x1461001 0	op_retired_ld	Event for every retired Load operation
0x1461002 0	op_retired_st	Event for every retired Store operation
0x1461004 0	op_retired_int	Event for every retired integer operation. This does not include Load or Store operations
0x1461008 0	op_retired_csr	Event for every retired CSR operation
0x1461010 0	op_retired_fp	Event for every retired floating point operation. This does not include Load or Store operations
0x1461020 0	op_retired_vec	Event for every retired vector operation. This does not include Load or Store operations
0x1462001	op_complete_ld	Event (speculative) for every confirmed load operation. This does not include Load or Store operations
0x1462002	op_complete_st	Event (speculative) for every confirmed store operation. This does not include Load or Store operations
0x1462004	op_complete_int	Event (speculative) for every confirmed integer operation. This does not include Load or Store operations
0x1462008	op_complete_fp	Event (speculative) for every confirmed floating point operation. This does not include Load or Store operations.
0x1462001 0	op_complete_vec	Event (speculative) for every confirmed vector point operation. This does not include Load or Store operations.
0x1440001	op_issued_pipe0	Event (speculative) for pipe 0 issuing. Multiple issues per cycle should be precisely counted
0x1440002	op_issued_pipe1	Event (speculative) for pipe 1 issuing. Multiple issues per cycle should be precisely counted
0x1440004	op_issued_pipe2	Event (speculative) for pipe 2 issuing. Multiple issues per cycle should be precisely counted
0x1440008	op_issued_pipe3	Event (speculative) for pipe 3 issuing. Multiple issues per cycle should be precisely counted
0x1440001 0	op_issued_pipe4	Event (speculative) for pipe 4 issuing. Multiple issues per cycle should be precisely counted
0x1440002 0	op_issued_pipe5	Event (speculative) for pipe 5 issuing. Multiple issues per cycle should be precisely counted
0x1440004 0	op_issued_pipe6	Event (speculative) for pipe 6 issuing. Multiple issues per cycle should be precisely counted

0x14400080	op_issued_pipe7	Event (speculative) for pipe 7 issuing. Multiple issues per cycle should be precisely counted
0x14400100	op_issued_pipe8	Event (speculative) for pipe 8 issuing. Multiple issues per cycle should be precisely counted
0x14400200	op_issued_pipe9	Event (speculative) for pipe 9 issuing. Multiple issues per cycle should be precisely counted
0x14400400	op_issued_pipe10	Event (speculative) for pipe 10 issuing. Multiple issues per cycle should be precisely counted
0x14400800	op_issued_pipe11	Event (speculative) for pipe 11 issuing. Multiple issues per cycle should be precisely counted
0x14401000	op_issued_pipe12	Event (speculative) for pipe 12 issuing. Multiple issues per cycle should be precisely counted
0x14402000	op_issued_pipe13	Event (speculative) for pipe 13 issuing. Multiple issues per cycle should be precisely counted
0x14404000	op_issued_pipe14	Event (speculative) for pipe 14 issuing. Multiple issues per cycle should be precisely counted
0x14408000	op_issued_pipe15	Event (speculative) for pipe 15 issuing. Multiple issues per cycle should be precisely counted
0x94400000	wasted_issue_slots_via_throttling	Event for counting wasted issue slot due to issue throttling
-1	store_uops_rejected_via_stq_advance	Event for counting stores rejected by stq tail ptr check
0x98400001	op_issued_fp64	Event (speculative) for fp64 ops issued. Multiple issues oer cycle should be precisely counted
0x90400002	fp64_export_overflow	Event for counting cycles where we overflowed the export restriction on FP64 Ops
0x34000003	cache_references	The count will represent all the requests made to L1 I and D caches. Includes accesses made by speculatively executed instructions and hardware prefetchers but does not include non-cacheable requests
0x34000004	cache_misses	Event for a request made to cache subsystem that misses in L1 I and D caches. Includes accesses made by speculatively executed instructions and hardware prefetchers. but does not include non-cacheable requests
0x2fc00021	l1d_read_access_non_clc	Event (speculative) for a l1d_cache access caused by a non cacheline crossing demand memory-read operation. Does not include accesses to IO or noncacheable regions
0x2fc00041	l1d_read_access_clc	Event (speculative) for a l1d_cache access caused by a cacheline crossing demand memory-read operation. Does not include accesses to IO or noncacheable regions
0x2fc00081	l1d_read_access_4kx	Event (speculative) for a l1d_cache access caused by a page crossing demand memory-read operation. Does not include accesses to IO or noncacheable regions
0x2fc000e1	l1d_read_access_all	Event (speculative) for a l1d_cache access caused by a all demand memory-read operation. Does not include accesses to IO or noncacheable regions
0x2fc00022	l1d_write_access_non_clc	Event (speculative) for a l1d_cache access caused by a non cacheline crossing demand memory-write operation. Does not include accesses to IO or noncacheable regions
0x2fc00042	l1d_write_access_clc	Event (speculative) for a l1d_cache access caused by a cacheline crossing demand memory-write operation. Does not include accesses to IO or noncacheable regions
0x2fc00082	l1d_write_access_4kx	Event (speculative) for a l1d_cache access caused by a page crossing demand memory-write operation. Does not include accesses to IO or noncacheable regions
0x2fc000e2	l1d_write_access_all	Event (speculative) for a l1d_cache access caused by all demand memory-write operation. Does not include accesses to IO or noncacheable regions
0x2fc00024	l1d_prefetch_access_non_clc	Event (speculative) for a l1d_cache access caused by a non cacheline crossing prefetch operation. Does not include accesses to IO or noncacheable regions
0x2fc00044	l1d_prefetch_access_clc	Event (speculative) for a l1d_cache access caused by a cacheline crossing prefetch operation. Does not include accesses to IO or noncacheable regions
0x2fc000e4	l1d_prefetch_access_all	Event (speculative) for a l1d_cache access caused by all prefetch operation. Does not include accesses to IO or noncacheable regions
0x2fc000e8	l1d_mmu_access	Event (speculative) for a l1d_cache access caused by a MMU operation. Does not include accesses to IO or noncacheable regions
0x2fc000f0	l1d_snoop_access	Event (speculative) for a l1d_cache access caused by a snoop operation. Does not include accesses to IO or noncacheable regions
0x2fc000ff	l1d_access_all	Event (speculative) for a l1d_cache access caused by all operations. Does not include accesses to IO or noncacheable regions
0x25400011	l1d_read_miss	Event (speculative) for a l1d_cache refill that is due to a demand memory-read operation. Does not include accesses to IO or noncacheable regions

0x25400012	l1d_write_miss	Event (speculative) for a l1d_cache refill that is due to a demand memory-write operation. Does not include accesses to IO or noncacheable regions
0x25400014	l1d_prefetch_miss	Event (speculative) for a l1d_cache refill that is due to a prefetch operation. Does not include accesses to IO or noncacheable regions
0x25400018	l1d_mmu_miss	Event (speculative) for a l1d_cache refill that is due to a MMU operation. Does not include accesses to IO or noncacheable regions
0x2540001f	l1d_miss_all	Event (speculative) for a l1d_cache refill that is due to all operation. Does not include accesses to IO or noncacheable regions
0xa5400041	transbuf_or_reqbuf_cannot_alloc_load	Event (speculative) for every instance of a failed TransBuffer or ReqBuffer allocation by a demand memory-read operation
0xa5400042	transbuf_or_reqbuf_cannot_alloc_store	Event (speculative) for every instance of a failed TransBuffer or ReqBuffer allocation by a demand memory-write operation
0xa5400044	transbuf_or_reqbuf_cannot_alloc_prefetch	Event (speculative) for every instance of a failed TransBuffer or ReqBuffer allocation by a prefetch operation
0xa5400048	transbuf_or_reqbuf_cannot_alloc_mmu	Event (speculative) for every instance of a failed TransBuffer or ReqBuffer allocation by a MMU operation
0xa540004f	transbuf_cannot_alloc_all	Event (speculative) for every instance of a failed TransBuffer or ReqBuffer allocation
0xa5400202	l1d_write_upgrade_req	Event for a store allocating the memory request buffer for an upgrade request of an L1D cache line
0x2bc10019	dtlb_read_access	Event (speculative) for an l1 dTLB access caused by a demand memory-read operation
0x2bc1001a	dtlb_write_access	Event (speculative) for an l1 dTLB access caused by a demand memory-write operation
0x2bc1001c	dtlb_prefetch_access	Event (speculative) for an l1 dTLB access caused by a prefetch operation
0x2bc10009	dtlb_read_access_cacheable	Event (speculative) for an l1 dTLB access caused by a cacheable demand memory-read operation
0x2bc10011	dtlb_read_access_noncacheable	Event (speculative) for an l1 dTLB access caused by a non-cacheable demand memory-read operation
0x2bc1000a	dtlb_write_access_cacheable	Event (speculative) for an l1 dTLB access caused by a cacheable demand memory-write operation
0x2bc10012	dtlb_write_access_noncacheable	Event (speculative) for an l1 dTLB access caused by a non-cacheable demand memory-write operation
0x2bc1001f	dtlb_access_all	Event (speculative) for an l1 dTLB access caused by all operation
0x25010019	dtlb_read_miss	Event (speculative) for an l1 dTLB refill caused by a demand memory-read operation
0x2501001a	dtlb_write_miss	Event (speculative) for an l1 dTLB refill caused by a demand memory-write operation
0x2501001c	dtlb_prefetch_miss	Event (speculative) for an l1 dTLB refill caused by a prefetch operation
0x250100Of	dtlb_miss_4k	Event (speculative) for 4k l1 dTLB refill caused by all operation
0x25010017	dtlb_miss_hugepage	Event (speculative) for huge page l1 dTLB refill caused by all operation
0x2501001f	dtlb_miss_all	Event (speculative) for both 4k and huge page l1 dTLB refill caused by all operation
0x21800001	leaf_tlb_access_ls	Event (speculative) for leaf TLB access caused by LS
0x21800002	leaf_tlb_access_fe	Event (speculative) for leaf TLB access caused by FE
0x21800004	leaf_tlb_access_mmu_prefetch	Event (speculative) for leaf TLB access caused by MMU prefetch
0x21800007	leaf_tlb_access_all	Event (speculative) for all leaf TLB access
0x21810001	leaf_tlb_miss_ls	Event (speculative) for leaf TLB miss caused by LS
0x21810002	leaf_tlb_miss_fe	Event (speculative) for leaf TLB miss caused by FE
0x21810004	leaf_tlb_miss_mmu_prefetch	Event (speculative) for leaf TLB miss caused by MMU prefetch

0x218200 01	nonleaf_tlb_access_ls	Event (speculative) for non-leaf TLB access caused by LS
0x218200 02	nonleaf_tlb_access_fe	Event (speculative) for non-leaf TLB access caused by FE
0x218200 04	nonleaf_tlb_access_mmu_prefe tch	Event (speculative) for non-leaf TLB access caused by MMU prefetch
0x218200 07	nonleaf_tlb_access_all	Event (speculative) for all non-leaf TLB access
0x218300 01	nonleaf_tlb_miss_ls	Event (speculative) for non-leaf TLB miss caused by LS
0x218300 02	nonleaf_tlb_miss_fe	Event (speculative) for non-leaf TLB miss caused by FE
0x218300 04	nonleaf_tlb_miss_mmu_prefetc h	Event (speculative) for non-leaf TLB miss caused by MMU prefetch
0x218300 07	nonleaf_tlb_miss_all	Event (speculative) for all non-leaf TLB miss
0x2184000 1	page_table_walks_ls	Event (speculative) for every page walk initiated by LS
0x2184000 2	page_table_walks_fe	Event (speculative) for every page walk initiated by FE
0x2184000 4	page_table_walks_mmu_prefetc h	Event (speculative) for every page walk initiated by MMU Prefetch
0x2184000 7	page_table_walks_all	Event (speculative) for every page walk
0xafc20011	stlf_replay_load	Event (speculative) for any demand memory-read replay caused by STLF
0xafc2001 8	stlf_replay_mmu	Event (speculative) for any MMU operation replay caused by STLF
0xafc2001f	stlf_replay_all	Event (speculative) for any LS replay caused by STLF
0xafc2002 1	data_bank_conflict_replay_load	Event (speculative) for any demand memory-read replay caused by data bank conflict
0xafc2002 2	data_bank_conflict_replay_stor e	Event (speculative) for any demand memory-write replay caused by data bank conflict
0xafc2002 8	data_bank_conflict_replay_mm u	Event (speculative) for any MMU operation replay caused by data bank conflict
0xafc2002 f	data_bank_conflict_replay_all	Event (speculative) for any LS replay caused by data bank conflict
0xafc2004 1	ls_way_predictor_replay_load	Event (speculative) for any demand memory-read replay caused by the DC main way predictor
0xafc2004 2	ls_way_predictor_replay_store	Event (speculative) for any demand memory-write replay caused by the DC main way predictor
0xafc2004 4	ls_way_predictor_replay_prefet ch	Event (speculative) for any prefetch replay caused by the DC main way predictor
0xafc2004 8	ls_way_predictor_replay_mmu	Event (speculative) for any MMU operation replay caused by the DC main way predictor
0xafc2004f	ls_way_predictor_replay_all	Event (speculative) for any LS replay caused by the DC main way predictor
0xafc2008 1	ls_micro_way_predictor_replay _load	Event (speculative) for any demand memory-read replay caused by the DC micro way predictor
0xafc2008 4	ls_micro_way_predictor_replay _prefetch	Event (speculative) for any prefetch replay caused by the DC micro way predictor
0xafc2008 f	ls_micro_way_predictor_replay _all	Event (speculative) for any LS replay caused by the DC micro way predictor
0xafc20101	tag_bank_conflict_replay_load	Event (speculative) for any demand memory-read replay caused by tag bank conflict
0xafc2010 2	tag_bank_conflict_replay_store	Event (speculative) for any demand memory-write replay caused by tag bank conflict
0xafc2010 4	tag_bank_conflict_replay_prefe tch	Event (speculative) for any prefetch replay caused by tag bank conflict
0xafc2010 8	tag_bank_conflict_replay_mmu	Event (speculative) for any MMU operation replay caused by tag bank conflict
0xafc2010f	tag_bank_conflict_replay_all	Event (speculative) for any LS replay caused by tag bank conflict
0xafc2040 1	dtlb_replay_load	Event for any demand memory-read replay caused by DTLB miss

0xafc20402	dtlb_replay_store	Event for any demand memory-write replay caused by DTLB miss
0xafc20404	dtlb_replay_prefetch	Event for any prefetch replay caused by DTLB miss
0xafc2040f	dtlb_replay_all	Event for any LS replay caused by DTLB miss
0xafc20201	sipt_replay_load	Event (speculative) for any demand memory-read replay caused by mismatch between RSTLB and DTLB/UTLB
0xafc20202	sipt_replay_store	Event (speculative) for any demand memory-write replay caused by mismatch between RSTLB and DTLB/UTLB
0xafc2020f	sipt_replay_all	Event (speculative) for any LS replay caused by mismatch between RSTLB and DTLB/UTLB
0xafc21001	reqbuf_hit_replay_load	Event for any demand memory-read replay caused by hitting on a reqbuf entry
0xafc21002	reqbuf_hit_replay_store	Event for any demand memory-write replay caused by hitting on a reqbuf entry
0xafc21008	reqbuf_hit_replay_mmu	Event for any MMU operation replay caused by hitting on a reqbuf entry
0xafc2100f	reqbuf_hit_replay_all	Event for any LS replay caused by hitting on a reqbuf entry
0xafc22001	fillbuf_hit_replay_load	Event for any demand memory-read replay caused by hitting on a fillbuffer entry
0xafc22002	fillbuf_hit_replay_store	Event for any demand memory-write replay caused by hitting on a fillbuffer entry
0xafc22008	fillbuf_hit_replay_mmu	Event for any MMU operation replay caused by hitting on a fillbuffer entry
0xafc2200f	fillbuf_hit_replay_all	Event for any LS replay caused by hitting on a fillbuffer entry
0x25400021	l1d_miss_reqbuf_link_load	Event (speculative) for every instance of a load l1d_cache miss that links in the request buffer
0x25400022	l1d_miss_reqbuf_link_store	Event (speculative) for every instance of a store l1d_cache miss that links in the request buffer
0x25400028	l1d_miss_reqbuf_link_mmu	Event (speculative) for every instance of an mmu request l1d_cache miss that links in the request buffer
0x2540002f	l1d_miss_reqbuf_link_all	Event (speculative) for every instance of any l1d_cache miss that links in the request buffer
0x25400081	l1d_miss_misc_replay_load	Event (speculative) for every instance of a load l1d_cache miss that receives a non-fullness-related replay response fom the DMI
0x25400082	l1d_miss_misc_replay_store	Event (speculative) for every instance of a store l1d_cache miss that receives a non-fullness-related replay response fom the DMI
0x25400084	l1d_miss_misc_replay_prefetch	Event (speculative) for every instance of a prefetch l1d_cache miss that receives a non-fullness-related replay response fom the DMI
0x25400088	l1d_miss_misc_replay_mmu	Event (speculative) for every instance of an mmu request l1d_cache miss that receives a non-fullness-related replay response fom the DMI
0x2540008f	l1d_miss_misc_replay_all	Event (speculative) for every instance of any l1d_cache miss that receives a non-fullness-related replay response fom the DM
0xa4cb003c	l1d_victim_fill_evict	Event for an L1 data cache coupled fill+evict
0xa4cb003d	l1d_victim_early_evict	Event for an L1 data cache decoupled evict (fill in PENDing state)
0xa4cb0037	l1d_victim_demand_req	Event for an L1 data cache eviction caused by a demand memory operation
0xa4cb003b	l1d_victim_prefetch_req	Event for an L1 data cache eviction caused by a prefetch memory operation
0xa4cb001f	l1d_victim_mru_alloc	Event for an L1 data cache eviction where the MRU allocation replacement policy was selected
0xa4cb002f	l1d_victim_lru_alloc	Event for an L1 data cache eviction where the LRU allocation replacement policy was selected
0xa4cb003f	l1d_victim_all	Event for any L1 data cache eviction
0x24c90001	l1d_cache_invalidate_snoop	Event for all l1 data cache invalidates due to snoops from the Shared Cache
0x24c90002	l1d_cache_invalidate_cmo	Event for all l1 data cache invalidates due to cache management operations

0x24c900 04	l1d_cache_invalidate_ras	Event for all l1 data cache invalidates due to RAS errors
0x24c900 07	l1d_cache_invalidate_all	Event for all l1 data cache invalidates
0x274000 01	lsu_resyncs_rar_stpipe	Event for all RAR resyncs raised by Store Pipe
0x274000 02	lsu_resyncs_rar_ldpipe	Event for all RAR resyncs raised by Load Pipe
0x274000 03	lsu_resyncs_rar_all	Event for all RAR resyncs
0xabc300 01	pfc_prefetches_late_l1pend	Event (speculative) for every instance a demand memory operation hits in L1D cache on a prefetched line whose data hasn't filled yet
0xabc300 02	pfc_prefetches_late_reqbuf	Event (speculative) for every instance a demand memory operation misses in the L1D cache but but hits on a prefetch request in ReqBuf
0xabc300 04	pfc_prefetches_late_wasted	Event (speculative) for every instance a prefetch request was later to the ReqBuf than a demand memory operation
0xabc300 07	pfc_prefetches_late_all	Event (speculative) for every instance a prefetch request is too late to satisfy a demand memory operation's ability to hit out of the L1D cache
0xa20000 79	ls_chillout_cycles_relaxed	Event to count chillout cycles in relaxed mode
0xa20000 7a	ls_chillout_cycles_medium	Event to count chillout cycles in medium mode
0xa20000 7c	ls_chillout_cycles_heavy	Event to count chillout cycles in heavy mode
0xa20000 7f	ls_chillout_cycles_all	Event to count chillout cycles in all modes
0xa20000 0f	ls_chillout_cycles_ldc	Event to count chillout cycles due to a request from ldc
0xa20000 17	ls_chillout_cycles_stc	Event to count chillout cycles due to a request from ldc
0xa20000 27	ls_chillout_cycles_mmu	Event to count chillout cycles due to a request from ldc
0xa20000 47	ls_chillout_cycles_cif	Event to count chillout cycles due to a request from ldc
0xa20100 01	ls_chillout_requests_ldc	Event to count each time ldc requests chillout mode
0xa20100 02	ls_chillout_requests_stc	Event to count each time stc requests chillout mode
0xa20100 04	ls_chillout_requests_mmu	Event to count each time mmu requests chillout mode
0xa20100 08	ls_chillout_requests_cif	Event to count each time cif requests chillout mode
0xa20100 0f	ls_chillout_requests_all	Event to count each time anyone requests chillout mode
0xa20200 01	ls_chillout_entrances_ldc	Event to count each time we enter chillout mode due to a request from ldc
0xa20200 02	ls_chillout_entrances_stc	Event to count each time we enter chillout mode due to a request from stc
0xa20200 04	ls_chillout_entrances_mmu	Event to count each time we enter chillout mode due to a request from mmu
0xa20200 08	ls_chillout_entrances_cif	Event to count each time we enter chillout mode due to a request from cif
0xa20200 0f	ls_chillout_entrances_all	Event to count each time we enter chillout mode due to a request from all
0xa40500 01	utlb_hit_load	Event (speculative) for micro-TLB hit caused by a demand read operation
0xa40500 02	utlb_hit_store	Event (speculative) for micro-TLB hit caused by a demand write operation
0xa40500 03	utlb_hit_all	Event (speculative) for micro-TLB hit caused by a demand memory operation
0xa40600 01	utlb_miss_load	Event (speculative) for micro-TLB miss caused by a demand read operation

0xa4060002	utlb_miss_store	Event (speculative) for micro-TLB miss caused by a demand write operation
0xa4060003	utlb_miss_all	Event (speculative) for micro-TLB miss caused by a demand memory operation
0xa4010000	ldq_cannot_alloc	Event (speculative) for every instance of a failed Load Queue allocation to a demand memory-read operation
0xa4020000	mdp_correct_prediction	Event for a correct prediction by MDP
0xa4030000	mdp_false_hit	Event for every instance of prediction by MDP where the load does not get its data from STLF
0xa4040000	mdp_total_prediction	Event for every instance of a load predicted by MDP
0x20400000	stalls_mem_l1d_miss	Event (speculative) for every cycle the instruction picker has ops — but does not pick — and there is a pending L1D demand miss
0x20410000	stalls_mem_l1dtlb_miss	Event (speculative) for every cycle the instruction picker has ops — but does not pick — and there is a pending L1 DTLB demand miss
0xa4420000	rar_cannot_alloc	Event (speculative) for every instance of failed Read-After-Read queue allocation to a demand memory-read operation
0xa4430000	raw_cannot_alloc	Event (speculative) for every instance of a failed Read-After-Write queue allocation to a demand memory-read operation
0xa4440000	pcb_cannot_alloc	Event (speculative) for every instance of a failed Page Cross Buffer allocation
0xa4450000	udb_cannot_alloc	Event (speculative) for every instance of a failed Unaligned Data Buffer allocation
0xa0460000	udb_data_return	Event (speculative) for every instance a memory operation used the Unaligned Data Buffer for data return
0xa0470000	udb_lost	Event (speculative) for every instance an allocated Unaligned Data Buffer entry was lost to another cacheline crossing request
0x24480000	atomics_retired_lr	Event for every retired load reserved operation
0x20490000	lr_stall	Stall cycles due to load reserved operation
0x244a0000	ld_executed_vec_nano	Event (speculative) for every confirmed Load nano-operation
0x244b0000	ld_masked_vec_nano	Event (speculative) for every fully masked, confirmed load operation
0x244c0000	stlf_hits	Event (speculative) for every instance of a Store instruction forwarding data to a demand memory-read operation
0xa4800001	dfp_access_load	Event (speculative) for data pipe access by load
0xa4800002	dfp_access_store	Event (speculative) for data pipe access by store
0xa4800008	dfp_access_mmu	Event (speculative) for data pipe access by mmu
0xa4800010	dfp_access_evict	Event (speculative) for data pipe access by evicts
0xa4800020	dfp_access_fill	Event (speculative) for data pipe access by fills
0xa4800040	dfp_access_snoop	Event (speculative) for data pipe access by snoops
0xa4800007f	dfp_access_all	Event (speculative) for any data pipe access
0x20810000	tlb_invalidates	Event for each tlb invalidation — this will track retired instructions that invalidate the TLB
0x20820000	stalls_mem_stores	Event (speculative) for every cycle the instruction picker has ops — but does not pick — and the STQ can not drain to SMB
0x20830000	lsu_resyncs_raw	Event for all RAW resyncs raised by LSU
0xa4840000	smb_wants_to_alloc	Event (speculative) for every instance of a demand memory-write operation attempting to alloacte the store merge buffer (both successful and failed attempts)
0xa4850000	smb_cannot_alloc	Event (speculative) for every instance of a failed store merge buffer allocation to a demand memory-write operation

0x24860000	atomics_retired_sc	Event for every retired store conditional operation
0x24870000	atomics_retired_sc_fail	Event for every retired failed store conditional operation
0x24880000	atomics_retired_sc_success	Event for every retired successful store conditional operation
0x24890000	atomics_retired_amo	Event for every retired AMO operation
0x248a0000	st_executed_vec_nano	Event (speculative) for every confirmed Store nano-operation
0x248b0000	st_masked_vec_nano	Event (speculative) for every fully masked, confirmed Store nano-operation
0xa4c00001	tap_access_load	Event (speculative) for tag pipe access by load
0xa4c00002	tap_access_store	Event (speculative) for tag pipe access by store
0xa4c00004	tap_access_prefetch	Event (speculative) for tag pipe access by prefetch
0xa4c00008	tap_access_mmu	Event (speculative) for tag pipe access by mmu
0xa4c00010	tap_access_evict	Event (speculative) for tag pipe access by evicts
0xa4c00020	tap_access_fill	Event (speculative) for tag pipe access by fills
0xa4c00040	tap_access_snoop	Event (speculative) for tag pipe access by snoops
0xa4c0007f	tap_access_all	Event (speculative) for any tag pipe access
0xa4c10001	uwp_access_agp	Event (speculative) for micro way predictor access from AGP
0xa4c10002	uwp_access_arb	Event (speculative) for micro way predictor access from ARB
0xa4c10003	uwp_access_all	Event (speculative) for any micro way predictor access
0xa4c20001	uwp_miss_agp	Event (speculative) for an AGP-copy micro-way-predictor refill caused by a demand memory operation
0xa4c20002	uwp_miss_tap_dfp	Event (speculative) for a TAP/DFP-copy micro-way-predictor refill caused by a demand memory operation
0xa4c20003	uwp_miss_all	Event (speculative) for any micro-way-predictor refill caused by a demand memory operation
0xa4c30001	uwp_true_hit_agp	Event (speculative) for every instance of an AGP access uWP hit matching the L1D hit way
0xa4c30002	uwp_true_hit_arb	Event (speculative) for every instance of an ARB access uWP hit matching the L1D hit way
0xa4c30003	uwp_true_hit_all	Event (speculative) for every instance of any uWP hit matching the L1D hit way
0xa4ca0001	uwp_invalidate_agp	Event (speculative) for an AGP-copy micro-way-predictor invalidate
0xa4ca0002	uwp_invalidate_tap_dfp	Event (speculative) for a TAP/DFP-copy micro-way-predictor invalidate
0xa4ca0003	uwp_invalidate_all	Event (speculative) for any micro-way-predictor invalidate
0xa4c40000	wp_access	Event (speculative) for way predictor access
0xa4c50000	wp_miss	Event (speculative) for a WP refill caused by a demand memory operation
0xa4c60000	wp_true_hit	Event (speculative) for every instance of a WP hit matching the L1D hit way
0x24c70000	pfc_prefetches_hit	Event (speculative) for every instance a demand memory operation hits on prefetch-data. Only count the first hit is counted
0x24c80000	pfc_useless_prefetches	Event (speculative) for every L1D eviction which is an unused prefetch

0xa5000031	tlp_access_load	Event (speculative) for tlp pipe access by load
0xa5000032	tlp_access_store	Event (speculative) for tlp pipe access by store
0xa5000034	tlp_access_prefetch	Event (speculative) for tlp pipe access by prefetch
0xa5000017	tlp_access_agp	Event (speculative) for tlp pipe access from AGP
0xa5000027	tlp_access_arb	Event (speculative) for tlp pipe access from ARB
0xa5000037	tlp_access_all	Event (speculative) for tlp pipe access
0xa5410000	fillbuf_cannot_alloc	Event for every instance of a failed FillBuffer allocation
0xa5c00000	pfc_agt_cannot_alloc	Event for every failed instance a new Active Generation Table allocation
0xa5c200fd	pfc_agt_training_alloc	Event for every instance of AGT entry allocation
0xa5c200fe	pfc_agt_training_update	Event for every instance of AGT entry update
0xa5c200e7	pfc_agt_training_tag_miss	Event for every instance of AGT entry allocation or update by a TAG miss instruction
0xa5c200eb	pfc_agt_training_pf_hit	Event for every instance of AGT entry allocation or update by a TAG hit on a prefetched line
0xa5c2003f	pfc_agt_training_load	Event for every instance of AGT entry allocation or update by a load instruction
0xa5c2005f	pfc_agt_training_store	Event for every instance of AGT entry allocation or update by a store instruction
0xa5c200ff	pfc_agt_training_all	Event for every instance of AGT entry allocation or update
0xa1c10000	pfc_agt_evict	Event for every successful instance of Active Generation Table entry evicted to Pattern History Table
0xa1c30000	pfc_pht_tap_lookup	Event for every instance of read-access (lookup) to Pattern History Table
0xa1c40000	pfc_pht_tap_hit	Event for every instance of a hit in the Pattern History Table
0xa1c50000	pfc_pht_agt_alloc	Event for AGT evictions allocating a new PHT entry
0xa1c60000	pfc_pht_agt_update	Event for AGT evictions updating an existing PHT entry
0xa5c80000	pfc_prt_alloc	Event for PHT triggers allocating a new PRT entry
0xa5c90000	pfc_prt_update	Event for PHT triggers updating an existing PRT entry
0xa5ca0000	pfc_prt_cannot_alloc	Event for every failed instance of a new PRT allocation
0xa5cc0000	pfc_no_tlb_credit_stalls	Event for every failed instance a Prefetch TLB request due to no credits
0xa5cd0000	pfc_no_tag_credit_stalls	Event for every failed instance a Prefetch Tag request due to no credit
0x25cb0000	pfc_prefetches_sent	Event for every instance of a Prefetch Tag requests sent to LS
0xa1ce0000	pfc_prt_l1d_evict_hit	Event for L1D eviction CAM hitting on one or more PRT entries
0xa1cf0000	pfc_prt_reqbuf_alloc_hit	Event for reqbuf allocation CAM hitting on one or more PRT entries
-1	ldq_missq_full_delay	Wasted cycles across all LDQ entries between ReqBuf full indication and load allocs or links in the ReqBuf
-1	stq_missq_full_delay	Wasted cycles across all STQ entries between ReqBuf full indication and store allocs or links in the ReqBuf

Table 35. CPU Performance Monitoring Counter Event List

Event ID	Name	Description
0	sc_cache_access	Count each Memory-read operation or Memory-write operation that causes a cache access to SC. Each access to a cache line is counted, including refills of and write-backs from other caches.
1	sc_cache_rd	Count Memory-read operation that causes a cache access to SC. This event is not counted by non-cacheable accesses like ReadNoSnp and WriteNoSnp.
2	sc_cache_miss	Count each Memory-read operation or Memory-write operation that causes a cache access to SC but is not completed by SC.
3	sc_cache_miss_rd	Count Memory-read operation that causes a cache access to SC but is not completed by SC.
4	sc_cache_refill	Counts each access counted by SC_CACHE_ACCESS that causes a refill of the SC from outside of SC.
5	sc_cache_allocate	Counts each Memory-write operation that writes an entire line into SC without fetching data from outside SC.
6	sc_cache_wb_dirty	Counts each write-back of dirty cache lines from SC to outside of SC.
7	sc_cache_wb_clean	Counts each write-back of clean cache lines from SC to outside of SC.
8	sc_cache_inval	Counts each invalidation of a cache line in SC, including: (1) Invalidation of a cache line because of a cache maintenance operation; (2) Transfer of ownership of a cache line to another cache because of a coherency or refill request.
9	sc_snoop	Count external snoops received by SC.
10	sc_scratchpad_rd	Count Memory-read operation to ScratchPad Memory.
11	sc_scratchpad_wr	Count Memory-write operation to ScratchPad Memory.
12	f2sc_rd	Read requests from fabric
13	f2sc_wr	Write requests from fabric
14	mshr_lifetime	Cummulative cycles between MSHR allocation and release
15	mshr_allocations	Number of MSHR allocations
16	sc2f_rd_u	ReadUnique requests to fabric
17	sc2f_rd_c	ReadClean requests to fabric
18	sc2f_rd_o	ReadOnce requests to fabric
19	sc2f_wr	Write requests to fabric
20	c2sc_rd_i	Read requests from instruction fetch
21	c2sc_rd_d	Read requests from data cache
22	c2sc_wb_full	Write requests from cores
23	c2sc_evict	Clean evictions from cores

Table 36. Shared Cache Performance Monitoring Counter Event List

9.5. Ascalon Interface Signal Name

This section covers all interface signal names. Interfaces whose signal name started with **`i_`** are input ports. Interfaces whose signal name started with **`o_`** are output ports.

The Table 37 shows non-memory interface signals. None of the signals are associated with any clock. This means that the receiver requires inserting a synchronizer. The Table 38 is the signal list for creating a coherent network. The output signals are synchronized with the Fabric Clock (**`i_clk_fb`**) and the input signals are required to be synchronized with the Fabric Clock (**`i_clk_fb`**). The Table 39 is the signal list for a non-coherent network. The output signals are synchronized with the Fabric Clock (**`i_clk_fb`**) and the input signals are required to be synchronized with the Fabric Clock (**`i_clk_fb`**). The Table 40 is the signal list for a system management network. The output signals are synchronized with SOC clock (**`i_clk_soc`**) and the input signals are required to be synchronized with SOC clock (**`i_clk_soc`**).

Functionality	Signal Name
Reference Clock	i_rf_clk
SOC Clock	i_sc_clk
Fabric Clock	i_fb_clk
Cold Reset	i_cluster_coldreset_n
Warm Reset / NDM Reset	i_cluster_warmreset_n
SRAM Hold	i_cluster_sram_hold
RAS Hold	i_cluster_ras_hold

DM Hold	i_cluster_debug_hold
Config Hold	i_cluster_critical_signal_hold
Force reference clock	i_force_ss_to_ref_clock_n
PLL output observing port	o_dfx_cl_clk_obs
NDM Reset Progress	i_cluster_ndmreset_process
NDM Reset Request	o_cluster_ndmreset_request
Non Maskable Interrupt	i_nmi_valid[7:0]
RAS Interrupt (Core)	o_ras_error_core[2:0]
RAS Interrupt (Shared Cache)	o_ras_error_sc[2:0]
RAS Interrupt (Other)	o_ras_error_other[2:0]
Cluster ID	i_cluster_id[3:0]
Abnormal Sensor	o_abnormal_environment
Emergency Shutdown	i_emergency_shutdown
JTAG Reset	i_jtag_trstn
JTAG TMS	i_jtag_tms
JTAG TDI	i_jtag_tdi
JTAG TCK	i_jtag_tck
JTAG TDO	o_jtag_tdo
JTAG TOD Enable	o_jtag_tdo_en

Table 37. Ascalon Cluster Interface Signals Excluding Memory Interface

Functionality	Even Network	Odd Network
SYSCOREQ	o_coh_m0_syscoreq	o_coh_m1_syscoreq
SYSCOACK	i_coh_m0_syscoack	i_coh_m1_syscoack
TXSACTIVE	o_coh_m0_txsactive	o_coh_m1_txsactive
RXSACTIVE	i_coh_m0_rxsactive	i_coh_m1_rxsactive
LINKACTIVEREQ	o_coh_m0_txlinkactivereq	o_coh_m1_txlinkactivereq
LINKACTIVEACK	i_coh_m0_txlinkactiveack	i_coh_m1_txlinkactiveack
TXREQFLITPEND	o_coh_m0_txreqflitpend	o_coh_m1_txreqflitpend
TXREQFLITV	o_coh_m0_txreqflitv	o_coh_m1_txreqflitv
TXREQFLIT[]	o_coh_m0_txreqflit[N-1:0]	o_coh_m1_txreqflit[N-1:0]
TXREQLCRDV	i_coh_m0_txreqlcrdv	i_coh_m1_txreqlcrdv
TXRSPFLITPEND	o_coh_m0_txrspflitpend	o_coh_m1_txrspflitpend
TXRSPFLITV	o_coh_m0_txrspflitv	o_coh_m1_txrspflitv
TXRSPFLIT[]	o_coh_m0_txrspflit[N-1:0]	o_coh_m1_txrspflit[N-1:0]
TXRSPLCRDV	i_coh_m0_txrsplcrdv	i_coh_m1_txrsplcrdv
TXDATFLITPEND	o_coh_m0_txdatflitpend	o_coh_m1_txdatflitpend
TXDATFLITV	o_coh_m0_txdatflitv	o_coh_m1_txdatflitv
TXDATFLIT[]	o_coh_m0_txdatflit[N-1:0]	o_coh_m1_txdatflit[N-1:0]
TXDATLCRDV	i_coh_m0_txdatlcrdv	i_coh_m1_txdatlcrdv
RXLINKACTIVEREQ	i_coh_m0_rxlinkactivereq	i_coh_m1_rxlinkactivereq
RXLINKACTIVEACK	o_coh_m0_rxlinkactiveack	o_coh_m1_rxlinkactiveack
RXSNPFLITPEND	i_coh_m0_rxsnpflitpend	i_coh_m1_rxsnpflitpend
RXSNPFLITV	i_coh_m0_rxsnpflitv	i_coh_m1_rxsnpflitv
RXSNPFLIT[]	i_coh_m0_rxsnpflit[N-1:0]	i_coh_m1_rxsnpflit[N-1:0]
RXSNPLCRDV	o_coh_m0_rxsnplcrdv	o_coh_m1_rxsnplcrdv
RXRSPFLITPEND	i_coh_m0_rxrspflitpend	i_coh_m1_rxrspflitpend
RXRSPFLITV	i_coh_m0_rxrspflitv	i_coh_m1_rxrspflitv
RXRSPFLIT[]	i_coh_m0_rxrspflit[N-1:0]	i_coh_m1_rxrspflit[N-1:0]
RXRSPLCRDV	o_coh_m0_rxrsplcrdv	o_coh_m1_rxrsplcrdv

RXDATFLITPEND	i_coh_m0_rxdatflitpend	i_coh_m1_rxdatflitpend
RXDATFLITV	i_coh_m0_rxdatflitv	i_coh_m1_rxdatflitv
RXDATFLIT[]	i_coh_m0_rxdatflit[N-1:0]	i_coh_m1_rxdatflit[N-1:0]
RXDATLCRDV	o_coh_m0_rxdatlcrdv	o_coh_m1_rxdatlcrdv

Table 38. Ascalon Cluster CHI Interface for Coherent Network

Functionality	Subordinate Port	Manager Port
AWVALID	i_noc_s0_awvalid	o_noc_m0_awvalid
AWREADY	o_noc_s0_awready	i_noc_m0_awready
AWID	i_noc_s0_awid[9:0]	o_noc_m0_awid[11:0]
AWADDR	i_noc_s0_awaddr[51:0]	o_noc_m0_awaddr[51:0]
AWSIZE	i_noc_s0_awsize[2:0]	o_noc_m0_awsize[2:0]
AWPROT	i_noc_s0_awprot[2:0]	o_noc_m0_awprot[2:0]
AWUSER	i_noc_s0_awuser[7:0]	o_noc_m0_awuser[7:0]
WVALID	i_noc_s0_wvalid	o_noc_m0_wvalid
WREADY	o_noc_s0_wready	i_noc_m0_wready
WDATA	i_noc_s0_wdata[511:0]	o_noc_m0_wdata[511:0]
WSTRB	i_noc_s0_wstrb[63:0]	o_noc_m0_wstrb[63:0]
BVALID	o_noc_s0_bvalid	i_noc_m0_bvalid
BREADY	i_noc_s0_bready	o_noc_m0_bready
BID	o_noc_s0_bid[9:0]	i_noc_m0_bid[11:0]
BRESP	o_noc_s0_bresp[1:0]	i_noc_m0_bresp[1:0]
ARVALID	i_noc_s0_arvalid	o_noc_m0_arvalid
ARREADY	o_noc_s0_arready	i_noc_m0_arready
ARID	i_noc_s0_arid[9:0]	o_noc_m0_arid[11:0]
ARADDR	i_noc_s0_araddr[51:0]	o_noc_m0_araddr[51:0]
ARSIZE	i_noc_s0_arsize[2:0]	o_noc_m0_arsize[2:0]
ARPROT	i_noc_s0_arprot[2:0]	o_noc_m0_arprot[2:0]
ARUSER	i_noc_s0_aruser[7:0]	o_noc_m0_aruser[7:0]
RVALID	o_noc_s0_rvalid	i_noc_m0_rvalid
RREADY	i_noc_s0_rready	o_noc_m0_rready
RID	o_noc_s0_rid[9:0]	i_noc_m0_rid[11:0]
RDATA	o_noc_s0_rdata[511:0]	i_noc_m0_rdata[511:0]
RRESP	o_noc_s0_rresp[1:0]	i_noc_m0_rresp[1:0]
RUSER		
WUSER		
BUSER		

Table 39. Ascalon Cluster AXI Interface for Non-Coherent Network

Functionality	Subordinate Port	Manager Port
AWVALID	i_cpl_s0_awvalid	o_cpl_m0_awvalid
AWREADY	o_cpl_s0_awready	i_cpl_m0_awready
AWID	i_cpl_s0_awid[6:0]	o_cpl_m0_awid[6:0]
AWADDR	i_cpl_s0_awaddr[31:0]	o_cpl_m0_awaddr[31:0]
AWLEN	i_cpl_s0_awlen[7:0]	o_cpl_m0_awlen[7:0]
AWSIZE	i_cpl_s0_awsize[2:0]	o_cpl_m0_awsize[2:0]
AWBURST	i_cpl_s0_awburst[1:0]	o_cpl_m0_awburst[1:0]
AWPROT	i_cpl_s0_awprot[2:0]	o_cpl_m0_awprot[2:0]
AWUSER	i_cpl_s0_awuser[3:0]	o_cpl_m0_awuser[3:0]
WVALID	i_cpl_s0_wvalid	o_cpl_m0_wvalid
WREADY	o_cpl_s0_wready	i_cpl_m0_wready

WDATA	i_cpl_s0_wdata[63:0]	o_cpl_m0_wdata[63:0]
WSTRB	i_cpl_s0_wstrb[7:0]	o_cpl_m0_wstrb[7:0]
WLAST	i_cpl_s0_wlast	o_cpl_m0_wlast
BVALID	o_cpl_s0_bvalid	i_cpl_m0_bvalid
BREADY	i_cpl_s0_bready	o_cpl_m0_bready
BID	o_cpl_s0_bid[6:0]	i_cpl_m0_bid[6:0]
BRESP	o_cpl_s0_bresp[1:0]	i_cpl_m0_bresp[1:0]
ARVALID	i_cpl_s0_arvalid	o_cpl_m0_arvalid
ARREADY	o_cpl_s0_arready	i_cpl_m0_arready
ARID	i_cpl_s0_arid[6:0]	o_cpl_m0_arid[6:0]
ARADDR	i_cpl_s0_araddr[31:0]	o_cpl_m0_araddr[31:0]
ARLEN	i_cpl_s0_arlen[7:0]	o_cpl_m0_arlen[7:0]
ARSIZE	i_cpl_s0_arsize[2:0]	o_cpl_m0_arsize[2:0]
ARBURST	i_cpl_s0_arburst[1:0]	o_cpl_m0_arburst[1:0]
ARPROT	i_cpl_s0_arprot[2:0]	o_cpl_m0_arprot[2:0]
ARUSER	i_cpl_s0_aruser[3:0]	o_cpl_m0_aruser[3:0]
RVALID	o_cpl_s0_rvalid	i_cpl_m0_rvalid
RREADY	i_cpl_s0_rready	o_cpl_m0_rready
RID	o_cpl_s0_rid[6:0]	i_cpl_m0_rid[6:0]
RDATA	o_cpl_s0_rdata[63:0]	i_cpl_m0_rdata[63:0]
RRESP	o_cpl_s0_rresp[1:0]	i_cpl_m0_rresp[1:0]
RLAST	o_cpl_s0_rlast	i_cpl_m0_rlast
RUSER		
WUSER		
BUSER		

Table 40. Ascalon Cluster AXI Interface for CPL

9.6. Ascalon Cluster Memory Mapped Register

Name	Fields			Address	Description
	Name	Width	Range	22:0	
wtime				0x000000	Wake up time register, min(vswtime, swtime);
	wtime	64	63:0		
ctime				0x000008	Receives mtime broadcast for each core, write value t to WTIME causes time CSR to be updated with value t
	ctime	64	63:0		
c_nmivec				0x000020	base address for Non Maskable Interrupts handler
	lock	1	63:63		
	rsvd_62_57	6	62:57		
	nmivec_56_32	25	56:32		
	nmivec_31_2	30	31:2		
	rsvd_1_0	2	1:0		
c_nmevec				0x000028	Base address for Non Maskable Exception handler
	lock	1	63:63		
	rsvd_62_57	6	62:57		
	nmevec_56_32	25	56:32		
	nmevec_31_2	30	31:2		

Name	Fields			Address	Description
	rsvd_1_0	2	1:0		
trTeControl				0x001000	
	trTeActive	1	0:0		
	trTeEnable	1	1:1		
	trTeInstTracing	1	2:2		
	trTeEmpty	1	3:3		
	trTeInstMode	3	6:4		
	trTeContext	1	9:9		
	trTeInstTriggerEnable	1	11:11		
	trTeInstStallOrOverflow	1	12:12		
	trTeInstStallEna	1	13:13		
	trTeInhibitSrc	1	15:15		
	trTeSyncMode	2	17:16		
	trTeSyncMax	4	23:20		
	trTeFormat	3	26:24		
trTeImpl				0x001004	
	trTeVerMajor	4	3:0		
	trTeVerMinor	4	7:4		
	trTeCompType	4	11:8		
	trTeProtocolMajor	4	19:16		
	trTeProtocolMinor	4	23:20		
	trTeVendorFrameLength	4	27:24		Specify frame length. Frame Length = trTeVendorFrameLength* 64 ; Frame Length should be a multiple of Bank Data Width.
trTeInstFeatures				0x001008	
	trTeInstNoAddrDiff	1	0:0		
	trTeInstNoTrapAddr	1	1:1		
	trTeInstEnRepeatedHistory	1	8:8		
	trTeSrcID	12	27:16		
	trTeSrcBits	4	31:28		
trTeInstFilters				0x00100C	
	trTeInstFilters	16	15:0		
trTeFilter0Control				0x001400	
	trTeFilterEnable	1	0:0		
	trTeFilterMatchPrivilege	1	1:1		
trTeFilter0MatchInst				0x001404	
	trTeFilterMatchChoicePrivilege	8	7:0		
trDstControl				0x002000	
	trDstActive	1	0:0		
	trDstEnable	1	1:1		
	trDstInstTracing	1	2:2		
	trDstEmpty	1	3:3		
	trDstInstMode	3	6:4		
	trDstContext	1	9:9		
	trDstInstTriggerEnable	1	11:11		
	trDstInstStallOrOverflow	1	12:12		
	trDstInstStallEna	1	13:13		
	trDstInhibitSrc	1	15:15		

Name	Fields			Address	Description
	trDstSyncMode	2	17:16		When the field is set tp 2'b10, sent timestamp. All other vlaues not NA.
	trDstSyncMax	4	23:20		When trDstSyncMode is set to 2'b10, timestamp will be sent for every 2^(trDstSyncMax + 4) Cluster clocks.
	trDstFormat	3	26:24		bit[0]: XOR Enable, bit[1]: VLT Enable. Supported values : 2'b3 (XOR+VLT Compression), 2'b1 (XOR Compresion), 2'b0 (No Compression)
trDstImpl				0x002004	
	trDstVerMajor	4	3:0		
	trDstVerMinor	4	7:4		
	trDstCompType	4	11:8		
	trDstProtocolMajor	4	19:16		
	trDstProtocolMinor	4	23:20		
	trDstVendorFrameLength	4	27:24		Specify frame length. Frame Length = trDstVendorFrameLength* 64 ; Frame Length should be a multiple of Bank Data Width.
	trDstVendorStreamLength	3	30:28		Specify Stream length. A stream starts with "no compressed packet". Stream length specifies number of frames before a non-compressed packet is sent.Stream Length = 32* 2^(trDstVendorStreamLength + 1)
trDstInstFeatures				0x002008	
	trDstInstNoAddrDiff	1	0:0		
	trDstInstNoTrapAddr	1	1:1		
	trDstInstEnRepeatedHistory	1	8:8		
	trDstSrcID	12	27:16		
	trDstSrcBits	4	31:28		
c_dbg_cla_counter0_cfg				0x002100	Configure CLA counter0. One of the 4 counters used for counting cycles after a event match, and trigger a action on match.
	rsvd	31	63:33		
	reset_on_target	1	32:32		When Counter = target, reset to 0. With this bit set, counter will provide a periodic tick w/ frequency of (target +1) if the action is set to Auto Incr. The periodic tick will continue till Stop Auto Incr or Clear Counter action.
	target	16	31:16		
	counter	16	15:0		
c_dbg_cla_counter1_cfg				0x002108	Configure CLA counter1. One of the 4 counters used for counting cycles after a event match, and trigger a action on match.
	rsvd	31	63:33		
	reset_on_target	1	32:32		When Counter = target, reset to 0. With this bit set, counter will provide a periodic tick w/ frequency of (target +1) if the action is set to Auto Incr. The periodic tick will continue till Stop Auto Incr or Clear Counter action.

Name	Fields			Address	Description
	target	16	31:16		
	counter	16	15:0		
c_dbg_cla_counter2_cfg				0x002110	Configure CLA counter2. One of the 4 counters used for counting cycles after a event match, and trigger a action on match.
	rsvd	31	63:33		
	reset_on_target	1	32:32		When Counter = target, reset to 0. With this bit set, counter will provide a periodic tick w/ frequency of (target +1) if the action is set to Auto Incr. The periodic tick will continue till Stop Auto Incr or Clear Counter action.
	target	16	31:16		
	counter	16	15:0		
c_dbg_cla_counter3_cfg				0x002118	Configure CLA counter3. One of the 4 counters used for counting cycles after a event match, and trigger a action on match.
	rsvd	31	63:33		
	reset_on_target	1	32:32		When Counter = target, reset to 0. With this bit set, counter will provide a periodic tick w/ frequency of (target +1) if the action is set to Auto Incr. The periodic tick will continue till Stop Auto Incr or Clear Counter action.
	target	16	31:16		
	counter	16	15:0		
c_dbg_node0_eap0				0x002120	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node0_eap1				0x002128	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		

Name	Fields			Address	Description
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node1_eap0				0x002130	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node1_eap1				0x002138	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.

Name	Fields			Address	Description
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node2_eap0				0x002140	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node2_eap1				0x002148	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action

Name	Fields			Address	Description
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node3_eap0				0x002150	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_node3_eap1				0x002158	CLA Action(s) are tied to events(s) using "event-action pairing" registers. One of he 8 EAPs (Two EAPs per Node).
	rsvd	26	63:38		
	custom_action1_enable	1	37:37		custom_action1 is valid only if custom_action1_enable is set
	custom_action0_enable	1	36:36		custom_action0 is valid only if custom_action0_enable is set
	custom_action1	4	35:32		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	custom_action0	4	31:28		Select the bit position of custom action bus to be set when EAP trigger is met. External blocks can define and implement the specifc action for a given bit.
	event_type1	6	27:22		Select a trigger event
	event_type0	6	21:16		Select a trigger event
	logical_op	2	15:14		Relation to be satisfied among events to activate the actions
	action1	6	13:8		Select an Action
	action0	6	7:2		Select an Action
	dest_node	2	1:0		Select Destination Node
c_dbg_signal_mask0				0x002160	Used to define debug bus match event. Match event (0x2) is triggered when debug_bus & mask = match.
	value	64	63:0		Mask to be applied to debug bus before match

Name	Fields			Address	Description
c_dbg_signal_match0				0x002168	Used to define debug bus match event. Match event (0x2) is triggered when debug_bus & mask = match.
	value	64	63:0		Value used for debug bus match.
c_dbg_signal_mask1				0x002170	Used to define debug bus match event. Match event (0x4) is triggered when debug_bus & mask = match.
	value	64	63:0		Mask to be applied to debug bus before match
c_dbg_signal_match1				0x002178	Used to define debug bus match event. Match event (0x4) is triggered when debug_bus & mask = match.
	value	64	63:0		Value used for debug bus match.
c_dbg_signal_edge_detect_cfg				0x002180	Register to configure debug bus for edge triggers.
	pos_edge_signal1	1	13:13		Defines which edge to detect for singal from signal1_select. 1: Pos edge, 0: Neg Edge
	signal1_select	6	12:7		Define which signal to select for edge detection.
	pos_edge_signal0	1	6:6		Defines which edge to detect for singal from signal0_select. 1: Pos edge, 0: Neg Edge
	signal0_select	6	5:0		Define which signal to select for edge detection.
c_dbg_eap_status				0x002188	Register to indicate if EAP Pair was activated. Use the corresponding w2c register to reset the status. There are 2 bits for each EAP. The bit corresponds to action-0 and action-1 of the EAP. The register is used by SW to know if an action was taken (ex: NMI ISR needs to know which of the EAP triggered the NMI)
	node3_eap1_w2c	1	39:39		
	node3_eap0_w2c	1	38:38		
	node2_eap1_w2c	1	37:37		
	node2_eap0_w2c	1	36:36		
	node1_eap1_w2c	1	35:35		
	node1_eap0_w2c	1	34:34		
	node0_eap1_w2c	1	33:33		
	node0_eap0_w2c	1	32:32		
	rsvd_31_8	24	31:8		
	node3_eap1	1	7:7		
	node3_eap0	1	6:6		
	node2_eap1	1	5:5		
	node2_eap0	1	4:4		
	node1_eap1	1	3:3		
	node1_eap0	1	2:2		
	node0_eap1	1	1:1		
	node0_eap0	1	0:0		
c_dbg_cla_ctrl_status				0x002190	Ctrl/Status register to Enable EAP after EAP programming is complete, and read the current node.

Name	Fields			Address	Description
	cla_chain_loop_delay	7	13:7		Creates a window in which we don't forward the incoming Xtrigger and Clock_Halt signal
	enable_cla	1	6:6		Set this bit to 1 to enable CLA. If this bit is set, CLA is clock gated (save pwr)
	enable_eap	1	5:5		Set Enable EAP to 1 after EAP programming is complete
	current_node	2	1:0		Read the current node.
c_dbg_mux_sel				0x002198	
	Muxselseg7	6	63:58		
	Muxselseg6	6	57:52		
	Muxselseg5	6	51:46		
	Muxselseg4	6	45:40		
	Muxselseg3	6	39:34		Mux Select Bits for Lane 3. Split debug bus into 16 bit segments. Seg0 = debug bus [15:0], Seg1 = debug bus[31:16],.... If all bits are 0, Lane3= Seg3, if bit[0] = 1, Lane3 = Seg4, if bit[1] =1, Lane3 = Seg5,...
	Muxselseg2	6	33:28		Mux Select Bits for Lane 2. Split debug bus into 16 bit segments. Seg0 = debug bus [15:0], Seg1 = debug bus[31:16],.... If all bits are 0, Lane2= Seg2, if bit[0] = 1, Lane2 = Seg4, if bit[1] =1, Lane2 = Seg5,...
	Muxselseg1	6	27:22		Mux Select Bits for Lane 1. Split debug bus into 16 bit segments. Seg0 = debug bus [15:0], Seg1 = debug bus[31:16],.... If all bits are 0, Lane1= Seg1, if bit[0] = 1, Lane1 = Seg4, if bit[1] =1, Lane1 = Seg5,...
	Muxselseg0	6	21:16		Mux Select Bits for Lane 0. Split debug bus into 16 bit segments. Seg0 = debug bus [15:0], Seg1 = debug bus[31:16],.... If all bits are 0, Lane0 = Seg0, if bit[0] = 1, Lane0 = Seg4, if bit[1] =1, Lane0 = Seg5,...
	rsvd_15_8	8	15:8		
	DbmId	6	7:2		Unique DBM ID of the DBM instance
	DbmMode	2	1:0		Mode selection, 0: DBM off, 1: Normal debug mode, 2: DBM ID output mode, 3: Toggle Mode
c_dbg_debug_trace_cfg				0x0021A0	
	frame_closure_mode	1	21:21		Closure Mode: 1: Close frame with a packet no larger than max packet size before any overflow (due to packets crossing frame boundary). 0: Close frame when packets cross boundary. Push back packet crossing frame boundary to packet generator.

Name	Fields			Address	Description
	frame_mode_enable	1	20:20		Enable Frame Mode (Frame Mode: Always pack "frame length" number of bytes for trace transmission). If flush to memory is stalled (due to silicon bug), we can have incomplete packets in memory. Frame Mode will help SW decoder to avoid reading incomplete trasnmitted packets. In Frame Mode, SW will get a memory pointer that is aligned to Frame Length. HW will ensure there are no incomplete packets incluced in the region pointed by memory pointer.
	frame_lenght_in_bytes	4	15:12		Use trDstImpl[trDstVendorFrameLength] for Frame Length programming. Register field ZBB-ed.
	trace_frame_fill_byte	8	11:4		Use "Filler Packet" to align a stream of trace packets within a frame boundary.
	trace_source_id	4	3:0		Trace Source ID (FIXME: Base Address, Tool Issue)
c_dbg_ntrace_frame_cfg				0x0021A8	
	frame_closure_mode	1	21:21		Closure Mode: 1: Close frame with a packet no larger than max packet size before any overflow (due to packets crossing frame boundary). 0: Close frame when packets cross boundary. Push back packet crossing frame boundary to packet generator.
	frame_mode_enable	1	20:20		Enable Frame Mode (Frame Mode: Always pack "frame length" number of bytes for trace transmission). If flush to memory is stalled (due to silicon bug), we can have incomplete packets in memory. Frame Mode will help SW decoder to avoid reading incomplete trasnmitted packets. In Frame Mode, SW will get a memory pointer that is aligned to Frame Length. HW will ensure there are no incomplete packets incluced in the region pointed by memory pointer.
	frame_lenght_in_bytes	4	15:12		Use trDstTe[trDstVendorFrameLength] for Frame Length programming. Register field ZBB-ed.
	trace_frame_fill_byte	8	11:4		Use "Filler Packet" to align a stream of trace packets within a frame boundary.
	trace_source_id	4	3:0		Trace Source ID (FIXME: Base Address, Tool Issue)
c_dbg_transition_mask				0x0021B0	3 registers to configure "transition event". This register is to select the signals of interest using a mask.
	value	64	63:0		
c_dbg_transition_from_value				0x0021B8	3 registers to configure "transition event". The event is triggered on transition from Value A to Value B. This register specifies Value A.

Name	Fields			Address	Description
	value	64	63:0		
c_dbg_transition_to_value				0x0021C0	3 registers to configure "transition event". The event is triggered on transition from Value A to Value B. This register specifies Value B.
	value	64	63:0		
c_dbg_ones_count_mask				0x0021C8	2 registers to configure "ones count" event. This register is to select the signals of interest using a mask.
	value	64	63:0		
c_dbg_ones_count_value				0x0021D0	2 registers to configure "ones count" event. Event triggered when the sum of the signals match the value specified in this register.
	value	64	63:0		
c_dbg_any_change				0x0021D8	Event triggered when a subset of debug signals change. The mask is used to select the subset of the debug signals.
	mask	64	63:0		
c_dbg_signal_snapshot_node0_eap0				0x0021E0	debug Bus Value when Node0 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node0_eap1				0x0021E8	debug Bus Value when Node0 Eap1 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node1_eap0				0x0021F0	debug Bus Value when Node1 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node1_eap1				0x0021F8	debug Bus Value when Node1 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node2_eap0				0x002200	debug Bus Value when Node0 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node2_eap1				0x002208	debug Bus Value when Node0 Eap1 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node3_eap0				0x002210	debug Bus Value when Node1 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_signal_snapshot_node3_eap1				0x002218	debug Bus Value when Node1 Eap0 Trigger is met.
	value	64	63:0		
c_dbg_cla_time_match				0x00221C	This value is compared against the internal time register value of the core, and is used to generate a time match event for CLA. Need to be non zero for the time match event signal to trigger. The event will only trigger an action if the EAP programming is done.
	time_match_val	64	63:0		
clock_ratios				0x003000	
	cluster_clk_2ref_clk	64	63:0		
dfs_transition_req				0x003008	

Name	Fields			Address	Description
	requested_freq	10	9:0		Requested core clock frequency. Frequency will step in multiples of 4MHz. The 10-bit register field can support frequency range from 0 to 4 Ghz (4Mz *2^10).
low_power_req				0x003010	
	exit_latency_limit	8	11:4		Placeholder field for exit latency limit. Not used for now.
	requested_c_state	4	3:0		Requested C-State. Requested C-State equals vaue of this field. If the value is 4, the requested C-state is C4. For Ascalon, the supported C-state is C2, C3 andC4.
power_management_capability				0x003018	
	supported_core_low_power_states	5	4:0		Supported C-States. Bit[0]= C2 Supported, Bit[1]= C4 Supported, Bit [2] = C6 Supported
crCsrDataPort				0x004000	
	Data	64	63:0		
crCsrCommandPort				0x004008	Usage: CSR Read: Check if Command Port[Busy] = 0,Write Command Port, Poll for CommandPort [Busy] to be 0, Read Data Port. (CommandPort[Enable] will be when busy transitions from 1 to 0) CSR Write: Check if Command Port[Busy] = 0,Write Command Port, Poll for CommandPort [Busy] to be 0. (CommandPort[Enable] will be when busy transitions from 1 to 0)
	Busy	1	63:63		See above
	Enable	1	62:62		See above
	Wr	1	61:61		0: read, 1; write
	WrInstType	2	60:59		Specify CSR type: RW=0,RS=1,RC=2 (Default is RW)
	Unit	4	15:12		Specify which core unit the CSR resides in, multihot encoding: [MC (1000), MS (0100), FE (0010), LS (0001)]
	Address	12	11:0		Address Offset within the unit
c_ResetVector				0x005300	Reset Vector
	lock	1	63:63		Lock bit, when set, write to ResetVector field is ignored.
	ResetVector	63	62:0		Reset Vector
c_pversion				0x005000	Patch version
	rsvd_63_8	56	63:8		
	pversion	8	7:0		Patch version
c_pcontrol				0x005040	Patch control
	patch3_lock	1	63:63		Patch3 lock, when set, disable update of corresponding preg and pcontrol section
	patch3_dispatch_stall	1	62:62		Patch3 Dispatch stall
	rsvd_61_59	3	61:59		
	patch3_other_conservative mode	1	58:58		Patch3 other conservative mode
	patch3_vec_conservative mode	1	57:57		Patch3 vector conservative mode only

Name	Fields			Address	Description
	patch3_prv_multi_hot_enc	8	56:49		Patch3 prv+v multi-hot encoding
	patch3_enable	1	48:48		Patch3 Enable
	patch2_lock	1	47:47		Patch2 lock, when set, disable update of corresponding preg and pcontrol section
	patch2_dispatch_stall	1	46:46		Patch2 Dispatch stall
	rsvd_45_43	3	45:43		
	patch2_other_conservative mode	1	42:42		Patch2 other conservative mode
	patch2_vec_conservative mode	1	41:41		Patch2 vector conservative mode only
	patch2_prv_multi_hot_enc	8	40:33		Patch2 prv+v multi-hot encoding
	patch2_enable	1	32:32		Patch2 Enable
	patch1_lock	1	31:31		Patch1 lock, when set, disable update of corresponding preg and pcontrol section
	patch1_dispatch_stall	1	30:30		Patch1 Dispatch stall
	rsvd_29_27	3	29:27		
	patch1_other_conservative mode	1	26:26		Patch1 other conservative mode
	patch1_vec_conservative mode	1	25:25		Patch1 vector conservative mode only
	patch1_prv_multi_hot_enc	8	24:17		Patch1 prv+v multi-hot encoding
	patch1_enable	1	16:16		Patch0 Enable
	patch0_lock	1	15:15		Patch0 lock, when set, disable update of corresponding preg and pcontrol section
	patch0_dispatch_stall	1	14:14		Patch0 Dispatch stall
	rsvd_13_11	3	13:11		
	patch0_other_conservative mode	1	10:10		Patch0 other conservative mode
	patch0_vec_conservative mode	1	9:9		Patch0 vector conservative mode only
	patch0_prv_multi_hot_enc	8	8:1		Patch0 prv+v multi-hot encoding
	patch0_enable	1	0:0		Patch0 Enable
c_pcontrolu				0x005078	Ucode Patch control
	lock	1	63:63		Ucode Patch lock, when set, disable update of pcontrolu
	rsvd_62_12	51	62:12		Reserved.
	vsret_en	1	11:11		VSRET patch Enable
	sret_en	1	10:10		SRET patch Enable
	mnret_en	1	9:9		MNRET patch Enable
	mret_en	1	8:8		MRET patch Enable
	rsvd_7_6	2	7:6		Reserved.
	ebreak_p_en	1	5:5		Ebreak_p patch Enable
	ebreak_d_en	1	4:4		Ebreak_d patch Enable
	ecall_vs_en	1	3:3		Ecall_vs patch Enable
	ecall_s_en	1	2:2		Ecall_s patch Enable
	ecall_mn_en	1	1:1		Ecall_mn patch Enable
	ecall_m_en	1	0:0		Ecall_m patch Enable
c_preg0				0x005080	
	patch_mask	32	63:32		Patch mask

Name	Fields			Address	Description
	patch_opcode	32	31:0		Patch instruction opcode
c_preg1				0x005088	
	patch_mask	32	63:32		Patch mask
	patch_opcode	32	31:0		Patch instruction opcode
c_preg2				0x005090	
	patch_mask	32	63:32		Patch mask
	patch_opcode	32	31:0		Patch instruction opcode
c_preg3				0x005098	
	patch_mask	32	63:32		Patch mask
	patch_opcode	32	31:0		Patch instruction opcode
CrSatp				0x005100	
	Data	64	63:0		Core i satp value
CrMatp				0x005108	
	Data	64	63:0		Core i matp value
CrHgatp				0x005110	
	Data	64	63:0		Core i hgatp value
CrVsatp				0x005118	
	Data	64	63:0		Core i vsatp value
CrMModeKey				0x005200	Core i generated mode key for M Mode
	Data	64	63:0		
CrHSModeKey				0x005208	Core i generated mode key for HS Mode
	Data	64	63:0		
CrUModeKey				0x005210	Core i generated mode key for U Mode
	Data	64	63:0		
CrVSModeKey				0x005218	Core i generated mode key for VS Mode
	Data	64	63:0		
CrVUModeKey				0x005220	Core i generated mode key for VU Mode
	Data	64	63:0		
trFunnelControl				0x081000	
	trFunnelActive	1	0:0		
	trFunnelEnable	1	1:1		
	trFunnelEmpty	1	3:3		
trFunnelImpl				0x081004	
	trFunnelVerMajor	4	3:0		
	trFunnelVerMinor	4	7:4		
	trFunnelCompType	4	11:8		
trFunnelDisInput				0x081008	
	trFunnelDisInput	16	15:0		Bits 7:0 are reserved for N-trace srcs and 15:8 for Dst sources
trRamControl				0x080000	
	trRamActive	1	0:0		
	trRamEnable	1	1:1		
	trRamEmpty	1	3:3		
	trRamMode	1	4:4		
	trRamStopOnWrap	1	8:8		
trRamImpl				0x080004	

Name	Fields			Address	Description
	trRamVerMajor	4	3:0		
	trRamVerMinor	4	7:4		
	trRamCompType	4	11:8		
	trRamHasSRAM	1	12:12		
	trRamHasSMEM	1	13:13		
	trRamVendorFrameLength	4	27:24		
trRamStartLow				0x080010	
	trRamStartLow	30	31:2		
	rsvd_1_0	2	1:0		
trRamStartHigh				0x080014	
	trRamStartHigh	32	31:0		
trRamLimitLow				0x080018	
	trRamLimitLow	30	31:2		
	rsvd_1_0	2	1:0		
trRamLimitHigh				0x08001C	
	trRamLimitHigh	32	31:0		
trRamWPLow				0x080020	
	trRamWrap	1	0:0		
	trRamWPLow	30	31:2		
trRamWPHigh				0x080024	
	trRamWPHigh	32	31:0		
trRamRPLow				0x080028	
	trRamRPLow	30	31:2		
	rsvd_1_0	2	1:0		
trRamRPHigh				0x08002C	
	trRamRPHigh	32	31:0		
trRamData				0x080040	
	trRamData	32	31:0		
trCustomRamSMEMLimitLow				0x080E00	
	trCustomRamSMEMLimitLow	30	31:2		
	rsvd_1_0	2	1:0		
trDstRamControl				0x082000	
	trDstRamActive	1	0:0		
	trDstRamEnable	1	1:1		
	trDstRamEmpty	1	3:3		
	trDstRamMode	1	4:4		
	trDstRamStopOnWrap	1	8:8		
trDstRamImpl				0x082004	
	trDstRamVerMajor	4	3:0		
	trDstRamVerMinor	4	7:4		
	trDstRamCompType	4	11:8		
	trDstRamHasSRAM	1	12:12		
	trDstRamHasSMEM	1	13:13		
	trDstRamVendorFrameLength	4	27:24		
trDstRamStartLow				0x082010	
	trDstRamStartLow	30	31:2		
	rsvd_1_0	2	1:0		
trDstRamStartHigh				0x082014	
	trDstRamStartHigh	32	31:0		

Name	Fields			Address	Description
trDstRamLimitLow				0x082018	
	trDstRamLimitLow	30	31:2		
	rsvd_1_0	2	1:0		
trDstRamLimitHigh				0x08201C	
	trDstRamLimitHigh	32	31:0		
trDstRamWPLow				0x082020	
	trDstRamWrap	1	0:0		
	trDstRamWPLow	30	31:2		
trDstRamWPHigh				0x082024	
	trDstRamWPHigh	32	31:0		
trDstRamRPLow				0x082028	
	trDstRamRPLow	30	31:2		
	rsvd_1_0	2	1:0		
trDstRamRPHigh				0x08202C	
	trDstRamRPHigh	32	31:0		
trDstRamData				0x082040	
	trDstRamData	32	31:0		
mtime				0x180000	Machine-level time counter
	mtime	64	63:0		
waketime				0x180008	Minimum wake time across all cores
	wake_time	64	63:0		
wakecore				0x180010	Core to interrupt when wake_time expires
	rsvd_63_3	61	63:3		
	core_id	3	2:0		
timesync				0x180018	Used for CPL to force-sync time on all Cores
	rsvd_63_8	56	63:8		
	core7	1	7:7		
	core6	1	6:6		
	core5	1	5:5		
	core4	1	4:4		
	core3	1	3:3		
	core2	1	2:2		
	core1	1	1:1		
	core0	1	0:0		
mtimecmp0				0x188000	Machine-level time compare register for Core 0
	mtimecmp	64	63:0		
mtimecmp1				0x188008	Machine-level time compare register for Core 1
	mtimecmp	64	63:0		
mtimecmp2				0x188010	Machine-level time compare register for Core 2
	mtimecmp	64	63:0		
mtimecmp3				0x188018	Machine-level time compare register for Core 3
	mtimecmp	64	63:0		
mtimecmp4				0x188020	Machine-level time compare register for Core 4
	mtimecmp	64	63:0		

Name	Fields			Address	Description
mtimecmp5				0x188028	Machine-level time compare register for Core 5
	mtimecmp	64	63:0		
mtimecmp6				0x188030	Machine-level time compare register for Core 6
	mtimecmp	64	63:0		
mtimecmp7				0x188038	Machine-level time compare register for Core 7
	mtimecmp	64	63:0		
mtimecmp8				0x188040	Machine-level time compare register for Cluster CPL
	mtimecmp	64	63:0		
CrGenModeKey				0x188050	Side channel security : Indication for generating new mode keys
	rsvd_63_61	3	63:61		
	GenVUKey7	1	60:60		
	GenVSKey7	1	59:59		
	GenUKey7	1	58:58		
	GenHSKey7	1	57:57		
	GenMKey7	1	56:56		
	rsvd_55_53	3	55:53		
	GenVUKey6	1	52:52		
	GenVSKey6	1	51:51		
	GenUKey6	1	50:50		
	GenHSKey6	1	49:49		
	GenMKey6	1	48:48		
	rsvd_47_45	3	47:45		
	GenVUKey5	1	44:44		
	GenVSKey5	1	43:43		
	GenUKey5	1	42:42		
	GenHSKey5	1	41:41		
	GenMKey5	1	40:40		
	rsvd_39_37	3	39:37		
	GenVUKey4	1	36:36		
	GenVSKey4	1	35:35		
	GenUKey4	1	34:34		
	GenHSKey4	1	33:33		
	GenMKey4	1	32:32		
	rsvd_31_29	3	31:29		
	GenVUKey3	1	28:28		
	GenVSKey3	1	27:27		
	GenUKey3	1	26:26		
	GenHSKey3	1	25:25		
	GenMKey3	1	24:24		
	rsvd_23_21	3	23:21		
	GenVUKey2	1	20:20		
	GenVSKey2	1	19:19		
	GenUKey2	1	18:18		
	GenHSKey2	1	17:17		
	GenMKey2	1	16:16		

Name	Fields			Address	Description
	rsvd_15_13	3	15:13		
	GenVUKey1	1	12:12		
	GenVSKey1	1	11:11		
	GenUKey1	1	10:10		
	GenHSKey1	1	9:9		
	GenMKey1	1	8:8		
	rsvd_7_5	3	7:5		
	GenVUKey0	1	4:4		
	GenVSKey0	1	3:3		
	GenUKey0	1	2:2		
	GenHSKey0	1	1:1		
	GenMKey0	1	0:0		
sidech_spec_pred_ctrl				0x188058	Side channel security : Configuration for RAND Gen
	rsvd_63_21	43	63:21		
	period	20	20:1		Sets the period between rand refreshes. Value indicates number of millions reference clock cycles
	rekey	1	0:0		If set, allows rand refreshes, otherwise, only way to set is to use KeyGenRandLo and KeyGenRandHi
KeyGenRandLo				0x188060	Side channel security : Lower bits of 128-bit rand key. Only rand_hi writes flush values.
	rand_lo	64	63:0		
KeyGenRandHi				0x188068	Side channel security : Higher bits of 128-bit rand key. When written to, flushes both rand_lo and rand_hi to 128-bit rand key.
	rand_hi	64	63:0		
cluster_fuse				0x00FFF8	
cluster_fuse				0x18FFF8	
cluster_fuse				0x1BFFF8	
cluster_fuse				0x19FFF8	
	rsvd_63_48	16	63:48		Reserved for future use
	core7_vid	3	47:45		Physical core7 vid
	core7_enable	1	44:44		Physical core7 availability
	core6_vid	3	43:41		Physical Core6 vid
	core6_enable	1	40:40		Physical Core6 availability
	core5_vid	3	39:37		Physical Core5 vid
	core5_enable	1	36:36		Physical Core5 availability
	core4_vid	3	35:33		Physical Core4 vid
	core4_enable	1	32:32		Physical Core4 availability
	core3_vid	3	31:29		Physical Core3 vid
	core3_enable	1	28:28		Physical Core3 availability
	core2_vid	3	27:25		Physical Core2 vid
	core2_enable	1	24:24		Physical Core2 availability
	core1_vid	3	23:21		Physical Core1 vid
	core1_enable	1	20:20		Physical Core1 availability
	core0_vid	3	19:17		Physical Core0 vid
	core0_enable	1	16:16		Physical Core0 availability
	lock	1	15:15		Lock bit

Name	Fields			Address	Description
	rsvd_14_12	3	14:12		Reserved for future use
	export_restriction_enabled	1	11:11		Use reduced FP throughput to comply with export restriction
	debug_enable	2	10:9		Debug enable. 00 - No Debug 01 - Default (JTAG based debug) 10 - AXI based debug 11 - Support both enabled at same time. Same cycle collision causes error packet
	trace_enable	1	8:8		CLA and Trace enable
	sc_harvest_strap	8	7:0		1 bit / 4 ways (total 8 bit /32 ways possible) 1: enable 0: disabled
dm_cluster_cfg				0x19FFF0	
	rsvd_63_11	53	63:11		Reserved for Future Use
	sc_slice_clk_gating	1	10		1: enable 0: disable
	sc_cla_enable	1	9		1: enable 0: disable
	sc_cacheable_sp	1	8		Support SC Scratch Pad as cacheable
	core_in_low_power_state	8	7:0		Core disabled for power saving
axi_sel_hash_ids				0x1bFFE8	
	rsvd_63_2	62	63:2		Reserved for Future Use
	sel_hash	2	1:0		Selects which hashing schema to use: 00 - Default (Folding), 01 Core ID + I/D, 10 Core ID + Strongly Ordered / Relaxed Ordered, 11 bit-3 is I/D xor Strongly Ordered/Relaxed Ordered
tr_cluster_fuse_cfg_low				0x08FFF8	
	core3_vid	3	31:29		Physical Core3 vid
	core3_enable	1	28:28		Physical Core3 availability
	core2_vid	3	27:25		Physical Core2 vid
	core2_enable	1	24:24		Physical Core2 availability
	core1_vid	3	23:21		Physical Core1 vid
	core1_enable	1	20:20		Physical Core1 availability
	core0_vid	3	19:17		Physical Core0 vid
	core0_enable	1	16:16		Physical Core0 availability
	lock	1	15:15		Lock bit
	rsvd_14_11	4	14:11		Reserved for future use
	debug_enable	2	10:9		Debug enable. 00 - No Debug 01 - Default (JTAG based debug) 10 - AXI based debug 11 - Support both enabled at same time. Same cycle collision causes error packet
	trace_enable	1	8:8		CLA and Trace enable
	sc_harvest_strap	8	7:0		1 bit / 4 ways (total 8 bit /32 ways possible) 1: enable 0: disabled
tr_cluster_fuse_cfg_hi				0x08FFFC	
	rsvd_31_16	16	31:16		Reserved for future use
	core7_vid	3	15:13		Physical core7 vid
	core7_enable	1	12:12		Physical core7 availability
	core6_vid	3	11:9		Physical Core6 vid
	core6_enable	1	8:8		Physical Core6 availability
	core5_vid	3	7:5		Physical Core5 vid
	core5_enable	1	4:4		Physical Core5 availability

Name	Fields			Address	Description
	core4_vid	3	3:1		Physical Core4 vid
	core4_enable	1	0:0		Physical Core4 availability
cpl_error_header_vendor				0x1b0000	Vendor and implementation ID
	WPRI_reserved_63_48	16	63:48		Reserved for future RISCV RERI standard use
	imp_id	16	47:32		Provides a unique identity, defined by the vendor, to idectify the component and revisions of the component
	vendor_id	32	31:0		Provides the JEDEC manufacturer ID of the provider of the component hosting the error bank
cpl_error_header_bank_info				0x1b0008	Error bank information
	version	8	63:56		Version of the arch register layout specification implemented by the error bank
	WPRI_reserved_55_24	32	55:24		Reserved for future RISCV RERI standard use
	layout	2	23:22		Indicates the layout of the registers in the error bank and the error records. The layout encoding 0 indicates the registers are arranged and have meaning as defined by the specification.
	n_err_recs	6	21:16		Indicates the number of error records implemented by the error bank
	inst_id	16	15:0		Identifies a uniqiue instance of an error bank within a package or die, ideally in the whole system
cpl_error_header_valid_summary				0x1b0010	Summary of valid error records
	valid_bitmap	63	63:1		Summary of the valid bits from status registers in the error bank
	sv	1	0:0		Indicates that the valid_bitmap provides a summary of the valid bits from the status registers
cpl_error_header_reserved_0				0x1b0018	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_header_reserved_1				0x1b0020	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_header_reserved_2				0x1b0028	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_header_reserved_3				0x1b0030	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_header_custom_0				0x1b0038	Reserved for custom use
	ususued	64	63:0		Reserved for custom use
cpl_error_record_0_control				0x1b0040	Control register of error record 0
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
cpl_error_record_0_status				0x1b0048	Status register of error record 0
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
cpl_error_record_0_addr				0x1b0050	Address register of error record 0
	address	64	63:0		Address associated with the detected error
cpl_error_record_0_info				0x1b0058	Information register of error record 0
	info	64	63:0		Reserved for custom error information
cpl_error_record_0_suppl_info				0x1b0060	Supplemental information register of error record 0
	suppl_info	64	63:0		Reserved for custom supplemental error information
cpl_error_record_0_timestamp				0x1b0068	Timestamp register of error record 0
	timestamp	64	63:0		Timestamp of the error record
cpl_error_record_0_reserved_7				0x1b0070	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_record_0_reserved_8				0x1b0078	Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_record_1_control				0x1b0080	Control register of error record 1
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
cpl_error_record_1_status				0x1b0088	Status register of error record 1
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow

Name	Fields			Address	Description
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
cpl_error_record_1_addr				0x1b0090	Address register of error record 1
	address	64	63:0		Address associated with the detected error
cpl_error_record_1_info				0x1b0098	Information register of error record 1
	info	64	63:0		Reserved for custom error information
cpl_error_record_1_suppl_info				0x1b00A0	Supplemental information register of error record 1
	suppl_info	64	63:0		Reserved for custom supplemental error information
cpl_error_record_1_timestamp				0x1b00A8	Timestamp register of error record 1
	timestamp	64	63:0		Timestamp of the error record
cpl_error_record_1_reserved_7				0x1b00B0	Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_record_1_reserved_8				0x1b00B8	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_record_2_control				0x1b00C0	Control register of error record 2
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
cpl_error_record_2_status				0x1b00C8	Status register of error record 2
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
cpl_error_record_2_addr				0x1b00D0	Address register of error record 3
	address	64	63:0		Address associated with the detected error
cpl_error_record_2_info				0x1b00D8	Information register of error record 3
	info	64	63:0		Reserved for custom error information
cpl_error_record_2_suppl_info				0x1b00E0	Supplemental information register of error record 3
	suppl_info	64	63:0		Reserved for custom supplemental error information
cpl_error_record_2_timestamp				0x1b00E8	Timestamp register of error record 3

Name	Fields			Address	Description
	timestamp	64	63:0		Timestamp of the error record
cpl_error_record_2_reserved_7				0x1b00F0	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
cpl_error_record_2_reserved_8				0x1b00F8	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
sw_chicken_bit_register				0x1b1000	Chicken Bits
	reserved_63_3	61	63:3		Reserved
	cpl_atom_disable	3	2:0		Bit 0 : Disable Triggering of 1st WDT Error Atom , Bit 1 : Disable Triggring of 2nd WDT Error Event Atom , Bit 2 : Disable triggering of Halt Error Atom
c_fe_error_header_vendor				0x008000	Vendor and implementation ID
	WPRI_reserved_63_48	16	63:48		Reserved for future RISCV RERI standard use
	imp_id	16	47:32		Provides a unique identity, defined by the vendor, to idectify the component and revisions of the component
	vendor_id	32	31:0		Provides the JEDEC manufacturer ID of the provider of the component hosting the error bank
c_fe_error_header_bank_info				0x008008	Error bank information
	version	8	63:56		Version of the arch register layout specification implemented by the error bank
	WPRI_reserved_55_24	32	55:24		Reserved for future RISCV RERI standard use
	layout	2	23:22		Indicates the layout of the registers in the error bank and the error records. The layout encoding 0 indicates the registers are arranged and have meaning as defined by the specification.
	n_err_recs	6	21:16		Indicates the number of error records implemented by the error bank
	inst_id	16	15:0		Identifies a uniqiue instance of an error bank within a package or die, ideally in the whole system
c_fe_error_header_valid_summary				0x008010	Summary of valid error records
	valid_bitmap	63	63:1		Summary of the valid bits from status registers in the error bank
	sv	1	0:0		Indicates that the valid_bitmap provides a summary of the valid bits from the status registers
c_fe_error_header_reserved_0				0x008018	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_header_reserved_1				0x008020	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_header_reserved_2				0x008028	Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_header_reserved_3				0x008030	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_header_custom_0				0x008038	Reserved for custom use
	ususued	64	63:0		Reserved for custom use
c_fe_error_record_0_control				0x008040	Control register of error record 0
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
c_fe_error_record_0_status				0x008048	Status register of error record 0
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec

Name	Fields			Address	Description
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
c_fe_error_record_0_addr				0x008050	Address register of error record 0
	address	64	63:0		Address associated with the detected error
c_fe_error_record_0_info				0x008058	Information register of error record 0

Name	Fields			Address	Description
	info	64	63:0		Reserved for custom error information
c_fe_error_record_0_suppl_info				0x008060	Supplemental information register of error record 0
	suppl_info	64	63:0		Reserved for custom supplemental error information
c_fe_error_record_0_timestamp				0x008068	Timestamp register of error record 0
	timestamp	64	63:0		Timestamp of the error record
c_fe_error_record_0_reserved_7				0x008070	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_record_0_reserved_8				0x008078	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_record_1_control				0x008080	Control register of error record 1
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
c_fe_error_record_1_status				0x008088	Status register of error record 1
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
c_fe_error_record_1_addr				0x008090	Address register of error record 1

Name	Fields			Address	Description
	address	64	63:0		Address associated with the detected error
c_fe_error_record_1_info				0x008098	Information register of error record 1
	info	64	63:0		Reserved for custom error information
c_fe_error_record_1_suppl_info				0x0080A0	Supplemental information register of error record 1
	suppl_info	64	63:0		Reserved for custom supplemental error information
c_fe_error_record_1_timestamp				0x0080A8	Timestamp register of error record 1
	timestamp	64	63:0		Timestamp of the error record
c_fe_error_record_1_reserved_7				0x0080B0	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_fe_error_record_1_reserved_8				0x0080B8	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_header_vendor				0x008100	Vendor and implementation ID
	WPRI_reserved_63_48	16	63:48		Reserved for future RISCV RERI standard use
	imp_id	16	47:32		Provides a unique identity, defined by the vendor, to idectify the component and revisions of the component
	vendor_id	32	31:0		Provides the JEDEC manufacturer ID of the provider of the component hosting the error bank
c_ls_error_header_bank_info				0x008108	Error bank information
	version	8	63:56		Version of the arch register layout specification implemented by the error bank
	WPRI_reserved_55_24	32	55:24		Reserved for future RISCV RERI standard use
	layout	2	23:22		Indicates the layout of the registers in the error bank and the error records. The layout encoding 0 indicates the registers are arranged and have meaning as defined by the specification.
	n_err_recs	6	21:16		Indicates the number of error records implemented by the error bank
	inst_id	16	15:0		Identifies a uniqiue instance of an error bank within a package or die, ideally in the whole system
c_ls_error_header_valid_summary				0x008110	Summary of valid error records
	valid_bitmap	63	63:1		Summary of the valid bits from status registers in the error bank
	sv	1	0:0		Indicates that the valid_bitmap provides a summary of the valid bits from the status registers
c_ls_error_header_reserved_0				0x008118	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
c_ls_error_header_reserved_1				0x008120	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_header_reserved_2				0x008128	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_header_reserved_3				0x008130	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_header_custom_0				0x008138	Reserved for custom use
	ususued	64	63:0		Reserved for custom use
c_ls_error_record_0_control				0x008140	Control register of error record 0
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
c_ls_error_record_0_status				0x008148	Status register of error record 0
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use

Name	Fields			Address	Description
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected
	v	1	0:0		If 1, error record holds a valid error
c_ls_error_record_0_addr				0x008150	Address register of error record 0

Name	Fields			Address	Description
	address	64	63:0		Address associated with the detected error
c_ls_error_record_0_info				0x008158	Information register of error record 0
	info	64	63:0		Reserved for custom error information
c_ls_error_record_0_suppl_info				0x008160	Supplemental information register of error record 0
	suppl_info	64	63:0		Reserved for custom supplemental error information
c_ls_error_record_0_timestamp				0x008168	Timestamp register of error record 0
	timestamp	64	63:0		Timestamp of the error record
c_ls_error_record_0_reserved_7				0x008170	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_record_0_reserved_8				0x008178	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_record_1_control				0x008180	Control register of error record 1
	custom_unused	4	63:60		Reserved
	WPRI_reserved_55_48	10	59:50		Reserved for future RISCV RERI standard use
	srdp	1	49:49		Set read in progress. When 1, causes read in progress (rdip) to be set
	sinv	1	48:48		Status register invalidate. When written to 1, causes the valid (v) field of the status register to be cleared.
	eid	16	47:32		Error injection delay
	WPRI_reserved_31_8	24	31:8		Reserved for future RISCV RERI standard use
	uecs	2	7:6		Used to enable signaling of uncorrected critical errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ueds	2	5:4		Used to enable signaling of uncorrected deferred errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	ces	2	3:2		Used to enable signaling of corrected errors. If 0, disabled. If 1, signal using low-priority RAS signal. If 2, signal using high-priority RAS signal. If 3, signal using platform specific RAS signal.
	cece	1	1:1		If 1, harware will count corrected errors in the cec field of the status register. If 0, cec does not count and retains its value.
	else	1	0:0		If 1, the hardware unit logs and signals errors in the error record. If 0, the hardware unit will not log RAS errors
c_ls_error_record_1_status				0x008188	Status register of error record 1

Name	Fields			Address	Description
	cec	16	63:48		Corrected error counter. Incremented on each corrected error if cece is set to 1.
	WPRI_reserved_47_32	16	47:32		Reserved for future RISCV RERI standard use
	ec	8	31:24		Error code, provides a description of the detected error. Encodings defined in section 2.7 of RISC-V RERI spec
	rdip	1	23:23		Read in progress. Set to 1 when a new error is recorded in an invalid status register. Cleared to 0 when a valid status is overwritten. When control.sinv is set, status.v is cleared to 0 only if rdip is 1.
	WPRI_reserved_22_22	2	22:22		Reserved for future RISCV RERI standard use
	ceco	1	21:21		Corrected error counter overflow
	scrub	1	20:20		Set to 1 to indicate that the storage location has been updated with the corrected value. Only valid for corrected errors.
	WPRI_reserved_19_18	2	19:18		Reserved for future RISCV RERI standard use
	tsv	1	17:17		Timestamp valid, set to 1 if the error record holds a valid timestamp in the timestamp register
	siv	1	16:16		Supplemental information valid, set to 1 if error reports supplemental information in the suppl_info register
	at	4	15:12		Address type, indicates the type of address reported in the addr register. 0=None, 1=Supervisor Physical Address, 2=Guest Physical Address, 3=Virtual Address, 4-15=Component Specific
	iv	1	11:11		Information valid, set to 1 if error reports additional information in the info register
	tt	3	10:8		Transaction type. 0=Unspecified or NA, 1=UNSPECIFIED, 2-3=Reserved for furture standard use, 4=Explicit read, 5=Explicit write, 6=Implicit read, 7=Implicit write
	c	1	7:7		Containable bit. If 1, indicates that the error has not propogated beyond the boundaries of the hardware unit
	mo	1	6:6		Set to 1 if more errors of the same class and severity occur
	pri	2	5:4		Priority of the error. Indicates relative priority among errors of the same class.
	uec	1	3:3		Set to 1 if the error could not be corrected or deferred and needs critical handling by RAS handler
	ued	1	2:2		Set to 1 if the error could not be corrected but was deferred
	ce	1	1:1		Set to 1 if if the detected error was corrected

Name	Fields			Address	Description
	v	1	0:0		If 1, error record holds a valid error
c_ls_error_record_1_addr				0x008190	Address register of error record 1
	address	64	63:0		Address associated with the detected error
c_ls_error_record_1_info				0x008198	Information register of error record 1
	info	64	63:0		Reserved for custom error information
c_ls_error_record_1_suppl_info				0x0081A0	Supplemental information register of error record 1
	suppl_info	64	63:0		Reserved for custom supplemental error information
c_ls_error_record_1_timestamp				0x0081A8	Timestamp register of error record 1
	timestamp	64	63:0		Timestamp of the error record
c_ls_error_record_1_reserved_7				0x0081B0	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
c_ls_error_record_1_reserved_8				0x0081B8	Reserved for future RISCV RERI standard use
	reserved_63_0	64	63:0		Reserved for future RISCV RERI standard use
dm_scratchpad				0x19ffe8	Scratchpad register for DV
	data	64	63:0		
cr_scratchpad				0x002400	Scratchpad register for DV
	data	64	63:0		
cr4b_scratchpad_lo				0x002010	Scratchpad register for DV
	data	32	31:0		
cr4b_scratchpad_hi				0x002014	Scratchpad register for DV
	data	32	31:0		
sw_scratchpad				0x1bfff0	Scratchpad register for DV
	data	64	63:0		
ac_scratchpad				0x18fff0	Scratchpad register for DV
	data	64	63:0		
tr_scratchpad_lo				0x08fff0	Scratchpad register for DV
	data	32	31:0		
tr_scratchpad_hi				0x08fff4	Scratchpad register for DV
	data	32	31:0		
cr_physical_id				0x002FF8	Core Physical ID (external pins)
	rsvd_63_3	61	63:3		
	pid	3	2:0		

Table 41. Ascalon Cluster MMR Detail