易 灵 思

Sapphire High-Performance RISC-V SoC Data Sheet

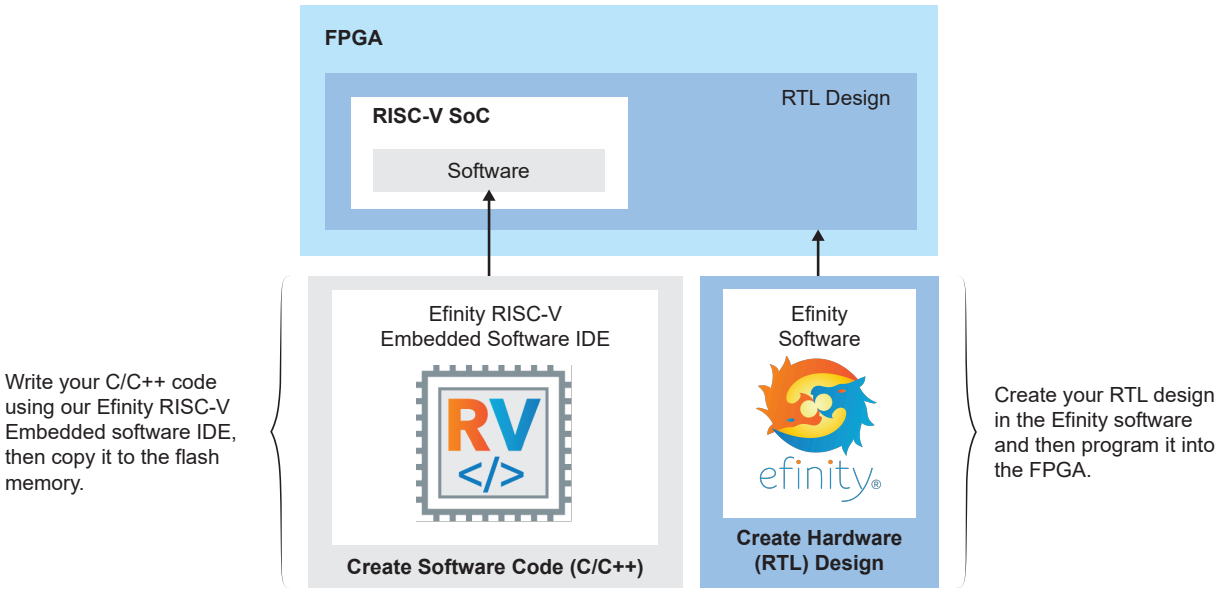
DS-SAPPHIREHPB-v1.0
January 2024
www.elitestek.com

Contents

Introduction

Elitestek provides the high-performance hardened Sapphire RISC-V SoC, that has a system clock up to 1 GHz. The Sapphire high-performance RISC-V SoC provides interfaces to support a variety of peripherals e.g., memory controller, direct memory access channel, custom instruction, and I/O devices. You can choose the interface you want to use by configuring the SoC in the Efinity® IP Manager.

Figure 1: Sapphire RISC-V SoC Design Flow



Features

- 4 VexRiscv processor(s) with 6 pipeline stages (fetch, injector, decode, execute, memory, and write back), interrupts, and exception handling with machine mode and supervisor mode.
- Up to 1 GHz system clock frequency
- 16 KB on-chip RAM with boot loader for SPI flash
- Memory controller for LPDDR4x
  - Supports memory module sizes of 3.7 GB
  - 1 full-duplex 512-bits AXI4 interface to communicate with the external memory
  - User-configurable external memory bus frequency
- 1 AXI master channel for user logic, data width of 128-bits
- 1 AXI slave channel to user logic
- Each core includes:
  - 4-way 16 KB data and instruction caches
  - Floating point unit (FPU)
  - Linux memory management unit (MMU)
  - Custom instruction interface with 1,024 IDs to perform various functions
- Supports RISC-V extensions such as integer, multiply, atomic, compressed, single, and double-digit floating point.
- JTAG debug module with 8 hardware breakpoints
- Peripherals:
  - 2 user timers
  - 24 user interrupts

FPGA Support

The Sapphire high-performance RISC-V SoC uses the hardened RISC-V block in Ti135, Ti200, and Ti375 FPGAs, and only support these FPGAs.

Performance Benchmark

The performance of the CPU can be benchmarked with Dhrystone and Coremark benchmark programs for easier comparison between processors.

Development board: 钛金系列
Efinity version 2023.2

Table 1: Performance Benchmark

GCC Option	Coremark (/MHz)	Dhrystone (/MHz)
(1)	(1)	(1)

(1) Pending characterization

Functional Description

The hardened RISC-V SoC block (HRB) consists of 4 Vexriscv cores. Each core consists of a memory management unit (MMU) that handles all memory and caching operations associated with the processors. The core runs at 1.0 GHz and controls the following RISC-V extensions:

- 32-bit integer (I)
- Multiply (M)
- Atomic (A)
- Compressed (C)
- Single and double precision floating-point (FD) instruction extensions

You can use custom instruction interfaces to execute self-defined operations and debug with a JTAG debug module that has 8 hardware breakpoints and complies with the RISC-V debug specifications.

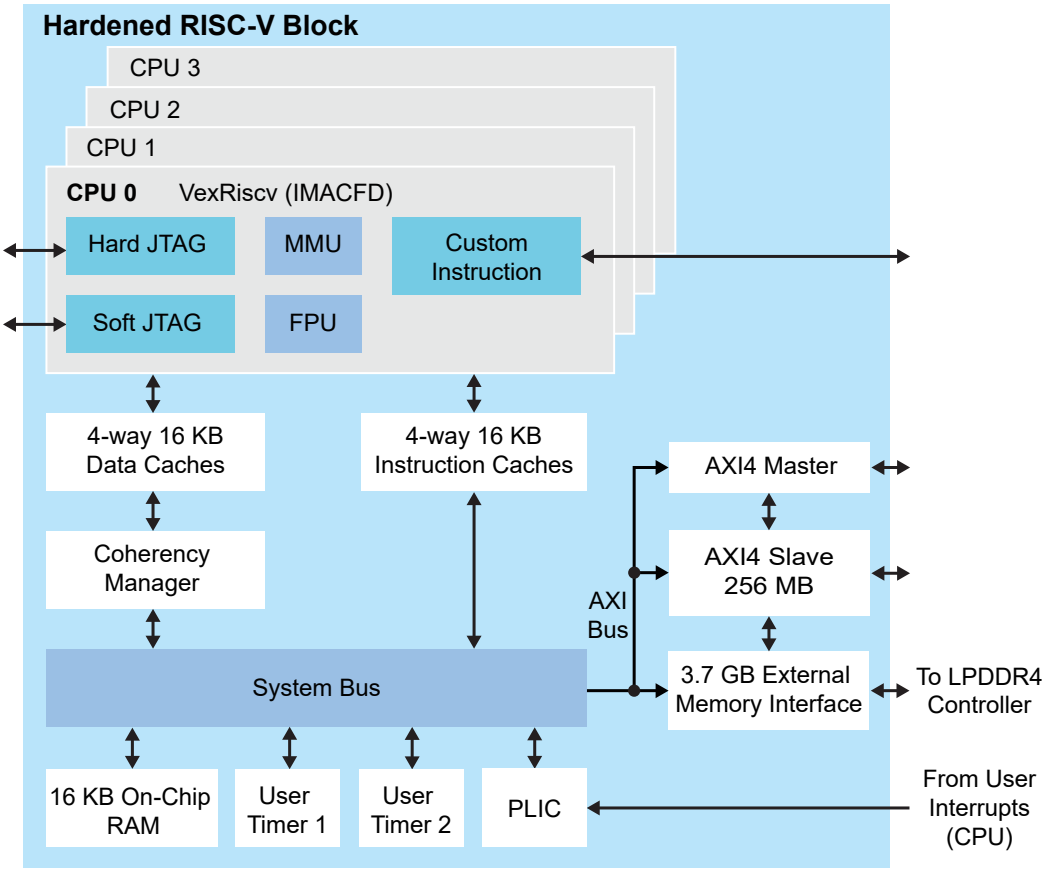
The HRB provides 4-way 16 KB data and instruction cache(s) to accommodate data and instruction exchanges with the main memory.

Additionally, the HRB has a 16 KB on-chip RAM for multi-purpose usage, which, by default, holds the bootloader that retrieves 124 KB data from the SPI flash and transfers it to the main memory during HRB power-up. An interrupt controller is also available that serves as an auxiliary controller for the CLINT timer, 24 user interrupts, and 2 user timers. You can configure the priority between the interrupt devices using the Efinity RISC-V IDE software.

To interface with the LPDDR4 memory, the HRB provides an AXI4 full-duplex interface with a 512-bit data width, and together with a 128-bit data width AXI4 master, allows you to connect to the direct memory access (DMA). However, the AXI master can only access the LPDDR4 memory. The HRB further provides another 32-bit, 256 MB AXI4 slave interface for communication with the logic from the FPGA core fabric.

You customize the Sapphire RISC-V SoC using the IP Manager in the Efinity® software.

Figure 2: Sapphire Hardened RISC-V Block Diagram



Address Map

The parameter names and address mappings are defined in **/embedded_sw/efx_hard_soc/bsp/efinix/EfxSapphireSoc/include/soc.h**.

Table 2: Default Address Map, Interrupt ID, and Cached Channels

The AXI user slave channel is in a cacheless region (I/O) for compatibility with AXI-Lite.

Device	Parameter	Size	Interrupt ID	Region
Off-chip memory	SYSTEM_DDR_BMB	3.7 GB	–	Cache
AXI user slave	SYSTEM_AXI_A_BMB	256 MB	–	I/O
User timer 0	SYSTEM_USER_TIMER_0_CTRL	4 KB	49	I/O
User timer 1	SYSTEM_USER_TIMER_1_CTRL	4 KB	50	I/O
CLINT Timer	SYSTEM_CLINT_CTRL	4 KB	–	I/O
PLIC	SYSTEM_PLIC_CTRL	4 MB	–	I/O
On-chip BRAM	SYSTEM_RAM_A_BMB	16 KB	–	Cache
External interrupt	–	–	[A]: 1 [B]: 2 [C]: 3 [D]: 4 [E]: 5 [F]: 6 [G]: 7 [H]: 8 [I]: 9 [J]: 10 [K]: 11 [L]: 12 [M]: 13 [N]: 14 [O]: 15 [P]: 16 [Q]: 17 [R]: 18 [S]: 19 [T]: 20 [U]: 21 [V]: 22 [W]: 23 [X]: 24	I/O

When accessing the addresses in the I/O region, type cast the pointer with the keyword **`volatile`**. The compiler recognizes this as a memory-mapped I/O register without optimizing the read/write access. The following command shows an example of the casting:

```
*((volatile u32*) address);
```

For the cached regions, the burst length is equivalent to an AXI burst length of 8. For the I/O region, the burst length is equivalent to an AXI burst length of 1. The AXI user slave is compatible with AXI-Lite by disconnecting unused outputs and driving a constant 1 to the input port.

Note: The RISC-V GCC compiler does not support user address spaces starting at 0x0000_0000.

Clocks

Table 3: Clock Ports

Port	Direction	Description
io_systemClock	Input	Provides up to I GHz clock for the SoC.
io_peripheralClock	Input	Provides up to 250 MHz clock for the APB3 peripherals and AXI4 slave.
io_memoryClock	Input	Provides up to 250 MHz clock for the external memory bus.
io_ddrMaster_0_clk	Input	Provides up to 250 MHz clock for the AXI master bus.
io_cfuClk	Input	Provides up to 250 MHz clock for the custom instruction bus.

Interrupts

Table 4: Interrupt Ports

Port	Direction	Description
userInterruptA userInterruptB userInterruptC userInterruptD userInterruptE userInterruptF userInterruptG userInterruptH userInterruptI userInterruptJ userInterruptK userInterruptL userInterruptM userInterruptN userInterruptO userInterruptP userInterruptQ userInterruptR userInterruptS userInterruptT userInterruptU userInterruptV userInterruptW userInterruptX	Input	Provides external interrupts.
axiAInterrupt	Input	User AXI slave channel interrupt.

Resets

The Sapphire high-performance RISC-V SoC has a master reset signal, `io_asyncReset` that triggers a system reset. Your RTL design should hold `io_asyncReset` for a minimum of 3 high state cycles of `io_systemClk` to reset the whole SoC system completely. When you assert `io_asyncReset`, the SoC asserts:

- `io_systemReset`, which resets the RISC-V processor and on-chip memory.
- `io_peripheralReset`, which resets the APB3 peripherals and AXI4 slave.
- `io_memoryReset`, which resets the LPDDR4 memory controller's AXI interface.
- `io_ddrMasters_0_reset`, which responds to the reset for AXI master channel `0` and is synchronized to `io_ddrMasters_0_clk`.
- `io_cfuReset`, which responds to the reset for custom instruction and is synchronized to `io_cfuClk`.

The SoC asserts the `io_memoryReset`, `io_ddrMaster_0_reset`, `io_peripheralReset`, `io_cfuReset`, and `io_systemReset` signals at the same time to allow the AXI masters access to the AXI cross bar once the reset completes.

Once `io_systemReset` goes low, the user binary code is executed.

Figure 3: Reset Timing Diagram

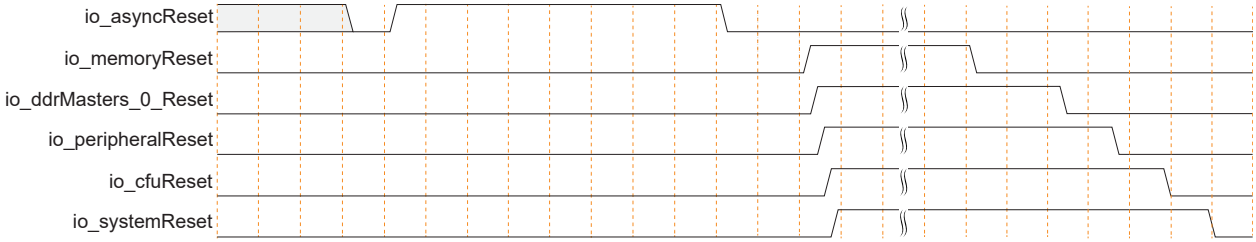


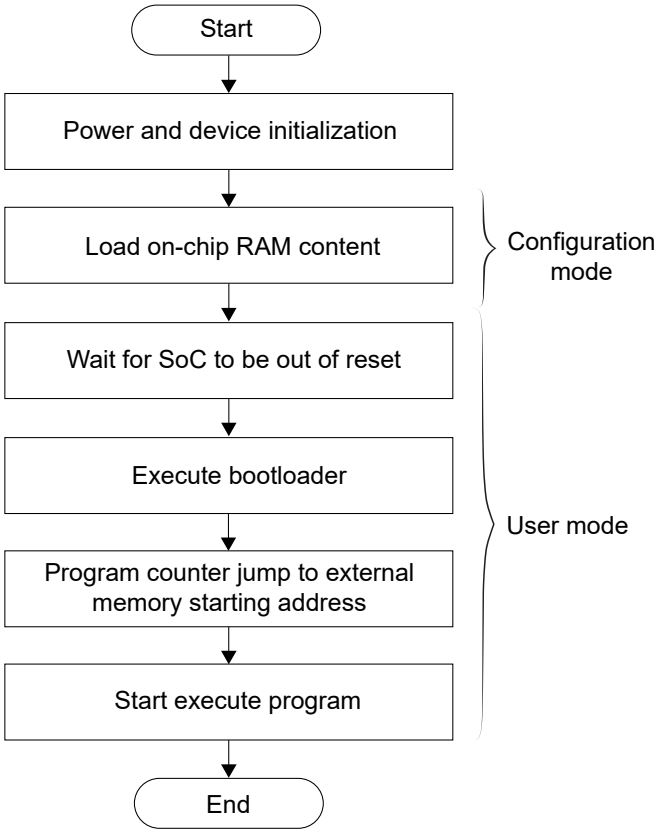
Table 5: Reset Ports

Port	Direction	Description
io_asyncReset	Input	Active-high asynchronous reset for the entire system.
io_systemReset	Output	Synchronous active-high reset for the system clock (io_systemClk).
io_peripheralReset	Output	Synchronous active-high reset for the peripheral clock (io_peripheralClock).
io_memoryReset	Output	External memory reset source from the RISC-V SoC.
io_ddrMasters_0_reset	Output	Responds to the reset for the AXI master.
io_cfuReset	Output	Synchronous active-high reset for the custom instruction clock (io_cfuClock).

Boot Flow

For the boot flow of the Sapphire high-performance RISC-V SoC, refer to the following flow chart.

Figure 4: Normal Boot Flow - Flow Chart



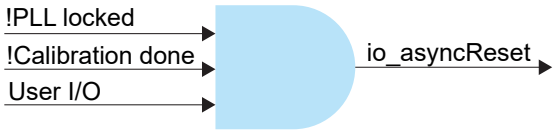
The configuration block starts offloading the RAM content, which is the bootloader by default, to the Sapphire SoC's on-chip RAM after the device is powered up and goes through initialization sequences. Other interface blocks such as PLLs dedicated to the SoC, soft logic block, and GPIO configuration should be programmed at this stage. Additionally, the SPI controller is required to be included in the soft logic block of the bootloader.

The Sapphire SoC runs its reset sequence once the `io_asyncReset` has been released from the soft logic block. Also, the CPU starts to execute the bootloader that allows the SPI controller to start the retrieval process of the RISC-V firmware binary from the SPI flash. The retrieved data is redirected and stored in the LPDDR4 memory. This looping process continues until the bootloader completes its task. Then, the CPU jumps to the LPDDR4 memory start address to start the program execution.

Gating SoC Reset

An active high `io_asyncReset` signal brings all the logic back to their known initial state across all clock domains of the Sapphire high-performance RISC-V SoC. For the reset timing diagram, refer to **Figure 3: Reset Timing Diagram** on page 10. The `io_asyncReset` signal must be in active high during configuration mode and de-assert once the LPDDR4 controller calibration is completed. Elitestek recommends gating the reset as shown in the following diagram.

Figure 5: Gating SoC Reset



The PLL locked signal must be sourced from the dedicated `io_systemClk` PLL. The user I/O refers to the user's self-controlled signal that is normally attached to an external GPIO switch. However, you can design this signal from another source using soft logic. All the associated signals used to generate the `io_asyncReset` must have active high attributes.

AXI Interface

The Sapphire high-performance RISC-V SoC has a 512-bit full duplex AXI4 interface to communicate to external memory.

Additionally it has a full duplex AXI4 interface to connect to user logic.

- There is one AXI4 slave interface, which is compatible with AXI-Lite (axlen is always 0).
- There are one optional full duplex AXI4 master interfaces with 128-bit data width.

Learn more: Refer to the AMBA AXI and ACE Protocol Specification for AXI channel descriptions and handshake information.

AXI Interface to External Memory

Table 6: AXI Slave Full-Duplex Address Channel for Read and Write

Port	Direction	Description
io_ddrA_aw_valid	Output	External memory write address valid.
io_ddrA_aw_ready	Input	External memory write address ready.
io_ddrA_aw_payload_addr[31:0]	Output	External memory write address.
io_ddrA_aw_payload_id[7:0]	Output	External memory write address ID.
io_ddrA_aw_payload_region[3:0]	Output	External memory write region identifier.
io_ddrA_aw_payload_len[7:0]	Output	External memory write burst length.
io_ddrA_aw_payload_size[2:0]	Output	External memory write burst size.
io_ddrA_aw_payload_burst[1:0]	Output	External memory write burst type, INCR only.
io_ddrA_aw_payload_lock	Output	External memory write lock type.
io_ddrA_aw_payload_cache[3:0]	Output	External memory write memory type.
io_ddrA_aw_payload_qos[3:0]	Output	External memory write quality of service.
io_ddrA_aw_payload_prot[2:0]	Output	External memory write protection type.
io_ddrA_aw_payload_allStrb	Output	External memory write all strobe.
io_ddrA_ar_valid	Output	External memory read address valid.
io_ddrA_ar_ready	Input	External memory read address ready.
io_ddrA_ar_payload_addr[31:0]	Output	External memory read address.
io_ddrA_ar_payload_id[7:0]	Output	External memory read address ID.
io_ddrA_ar_payload_region[3:0]	Output	External memory read region identifier.
io_ddrA_ar_payload_len[7:0]	Output	External memory burst length.
io_ddrA_ar_payload_size[2:0]	Output	External memory read burst size.
io_ddrA_ar_payload_burst[1:0]	Output	External memory read burst type, INCR only.
io_ddrA_ar_payload_lock	Output	External memory read lock type.
io_ddrA_ar_payload_cache[3:0]	Output	External memory read memory type.
io_ddrA_ar_payload_qos[3:0]	Output	External memory read quality of service.
io_ddrA_ar_payload_prot[2:0]	Output	External memory read protection type.

Table 7: AXI Slave Write Data Channel

Port	Direction	Description
io_ddrA_w_valid	Output	External memory write valid.
io_ddrA_w_ready	Input	External memory write ready.
io_ddrA_w_payload_data[*n*:0]	Output	External memory write data. The length is fixed at 512-bit.
io_ddrA_w_payload_strb[*m*:0]	Output	External memory write strobe. *m* is the width of io_ddrA_w_payload_data[*n*:0] divided by 8. The length is fixed at 64-bit.
io_ddrA_w_payload_last	Output	External memory write last.

Table 8: AXI Slave Write Respond Channel

Port	Direction	Description
io_ddrA_b_valid	Input	External memory write respond valid.
io_ddrA_b_ready	Output	External memory respond ready.
io_ddrA_b_payload_id[7:0]	Input	External memory respond ID.
io_ddrA_b_payload_resp[1:0]	Input	External memory write respond.

Table 9: AXI Slave Read Data Channel

Port	Direction	Description
io_ddrA_r_valid	Input	External memory read valid.
io_ddrA_r_ready	Output	External memory read ready.
io_ddrA_r_payload_data[*n*:0]	Input	External memory read data. The length is fixed at 512-bit.
io_ddrA_r_payload_id[7:0]	Input	External memory read ID.
io_ddrA_r_payload_resp[1:0]	Input	External memory read respond.
io_ddrA_r_payload_last	Input	External memory read last.

AXI Interface to User Logic

Table 10: User Slave Write Address Channel

Port	Direction	Description
axiA_awvalid	Output	User write address valid.
axiA_awready	Input	User write address ready.
axiA_awaddr[31:0]	Output	User write address.
axiA_awid[7:0]	Output	User write address ID.
axiA_awregion[3:0]	Output	User region identifier.
axiA_awlen[7:0][(2)]	Output	User burst length.
axiA_awsize[2:0]	Output	User burst size.
axiA_awburst[1:0]	Output	User burst type, INCR only.
axiA_awlock	Output	User lock type.
axiA_awcache[3:0]	Output	User memory type.
axiA_awqos[3:0]	Output	User quality of service.
axiA_awprot[2:0]	Output	User protection type.

Table 11: User Slave Write Data Channel

Port	Direction	Description
axiA_wvalid	Output	User write valid.
axiA_wready	Input	User write ready.
axiA_wdata[31:0]	Output	User write data.
axiA_wstrb[3:0]	Output	User write strobe.
axiA_wlast	Output	User write last.

Table 12: User Slave Write Respond Channel

Port	Direction	Description
axiA_bvalid	Input	User write respond valid.
axiA_bready	Output	User respond ready.
axiA_bresp[1:0]	Input	User write respond.

(2) axiA_awlen always outputs 0, that is, a burst length of 1. This setting makes the, axiA channel compatible with AXI-Lite.

Table 13: User Slave Read Address Channel

Port	Direction	Description
axiA_arvalid	Output	User read address valid.
axiA_arready	Input	User read address ready.
axiA_araddr[31:0]	Output	User read address.
axiA_arregion[3:0]	Output	User region identifier.
axiA_arlen[7:0][3]	Output	User burst length.
axiA_arsize[2:0]	Output	User burst size.
axiA_arburst[1:0]	Output	User burst type, INCR only.
axiA_arlock	Output	User lock type.
axiA_arcache[3:0]	Output	User memory type.
axiA_arqos[3:0]	Output	User quality of service.
axiA_arprot[2:0]	Output	User protection type.

Table 14: User Slave Read Data Channel

Port	Direction	Description
axiA_rvalid	Input	User read valid.
axiA_rready	Output	User read ready.
axiA_rdata[31:0]	Input	User read data.
axiA_rresp[1:0]	Input	User read respond.
axiA_rlast	Input	User read last.

Table 15: User Master Clock and Reset

Port	Direction	Description
io_ddrMasters_*0*_clk	Input	AXI master clock.
io_ddrMasters_*0*_reset	Output	AXI master active high reset.

[3] axiA_arlen always outputs 0, that is, a burst length of 1.This setting makes the, axiA channel compatible with AXI-Lite.

AXI Master Interface

Table 16: User Master Write Address Channel

Port	Direction	Description
io_ddrMasters_*0*_aw_valid	Input	User write address valid.
io_ddrMasters_*0*_aw_ready	Output	User write address ready.
io_ddrMasters_*0*_aw_payload_addr[31:0]	Input	User write address.
io_ddrMasters_*0*_aw_payload_id[7:0]	Input	User write address ID.
io_ddrMasters_*0*_aw_payload_region[3:0]	Input	User region identifier.
io_ddrMasters_*0*_aw_payload_len[7:0]	Input	User burst length.
io_ddrMasters_*0*_aw_payload_size[2:0]	Input	User burst size.
io_ddrMasters_*n*_aw_payload_burst[1:0]	Input	User burst type, INCR only.
io_ddrMasters_*0*_aw_payload_lock	Input	User lock type.
io_ddrMasters_*0*_aw_payload_cache[3:0]	Input	User memory type.
io_ddrMasters_*0*_aw_payload_qos[3:0]	Input	User quality of service.
io_ddrMasters_*0*_aw_payload_prot[2:0]	Input	User protection type.
io_ddrMasters_*0*_aw_payload_allStrb[2:0]	Input	User all strobe type.

Table 17: User Master Write Data Channel

Port	Direction	Description
io_ddrMasters_*0*_w_valid	Input	User write valid.
io_ddrMasters_*0*_w_ready	Output	User write ready.
io_ddrMasters_*0*_w_payload_data[*m*:0]	Input	User write data. The length is fixed at 128-bit.
io_ddrMasters_*0*_w_payload_strb[15:0]	Input	User write strobe.
io_ddrMasters_*0*_w_payload_last	Input	User write last.

Table 18: User Master Write Respond Channel

Port	Direction	Description
io_ddrMasters_*0*_b_valid	Output	User write respond valid.
io_ddrMasters_*0*_b_ready	Input	User respond ready.
io_ddrMasters_*0*_b_payload_id[7:0]	Output	User respond ID.
io_ddrMasters_*0*_b_payload_resp[1:0]	Output	User write respond.

Table 19: User Master Read Address Channel

Port	Direction	Description
io_ddrMasters_*0*_ar_valid	Input	User read address valid.
io_ddrMasters_*0*_ar_ready	Output	User read address ready.
io_ddrMasters_*0*_ar_payload_addr[31:0]	Input	User read address.
io_ddrMasters_*0*_ar_payload_id[7:0]	Input	User read address ID.
io_ddrMasters_*0*_ar_payload_region[3:0]	Input	User region identifier.
io_ddrMasters_*0*_ar_payload_len[7:0]	Input	User burst length.
io_ddrMasters_*0*_ar_payload_size[2:0]	Input	User burst size.
io_ddrMasters_*0*_ar_payload_burst[1:0]	Input	User burst type, INCR only.
io_ddrMasters_*0*_ar_payload_lock	Input	User lock type.
io_ddrMasters_*0*_ar_payload_cache[3:0]	Input	User memory type.
io_ddrMasters_*0*_ar_payload_qos[3:0]	Input	User quality of service.
io_ddrMasters_*0*_ar_payload_prot[2:0]	Input	User protection type.

Table 20: User Master Read Data Channel

Port	Direction	Description
io_ddrMasters_*0*_r_valid	Output	User read valid.
io_ddrMasters_*0*_r_ready	Input	External memory read ready.
io_ddrMasters_*0*_r_payload_data[*m*:0]	Output	External memory read data. The length is fixed at 128-bit.
io_ddrMasters_*0*_r_payload_id[7:0]	Output	External memory read ID.
io_ddrMasters_*0*_r_payload_resp[1:0]	Output	External memory read respond.
io_ddrMasters_*0*_r_payload_last	Output	External memory read last.

JTAG Interface

The Sapphire high-performance RISC-V SoC uses the JTAG User TAP interface block to communicate with the OpenOCD debugger.

Table 21: JTAG Ports

Port	Direction	Description
jtagCtrl_enable	Input	Indicates that the user instruction is active for the interface.
jtagCtrl_capture	Input	TAP controller is in the capture state.
jtagCtrl_shift	Input	TAP controller is in the shift state.
jtagCtrl_update	Input	TAP controller in the update state.
jtagCtrl_reset	Input	TAP controller is in the reset state.
jtagCtrl_tdi	Input	JTAG TDI for debugging.
jtagCtrl_tdo	Output	JTAG TDO for debugging.
jtagCtrl_tck	Input	JTAG TCK for debugging.

JTAG Interface - GPIO

The Sapphire high-performance RISC-V SoC uses the JTAG through GPIO to communicate with the OpenOCD debugger.

Table 22: JTAG Ports

Port	Direction	Description
io_jtag_tdi	Input	JTAG TDI for debugging.
io_jtag_tdo	Output	JTAG TDO for debugging.
io_jtag_tck	Input	JTAG TCK for debugging.
io_jtag_tms	Input	JTAG TMS for debugging.

Custom Instruction Interface

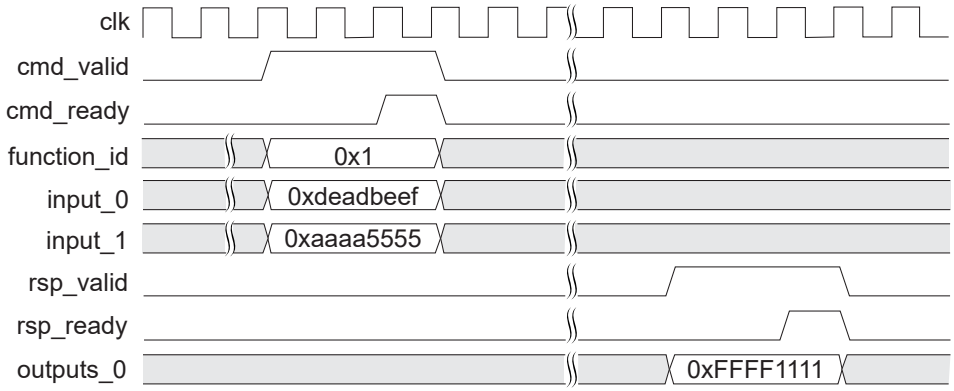
The Sapphire high-performance RISC-V SoC supports a custom instruction interface so you can accelerate software functions with custom hardware logic. The custom instruction supports R-type instructions, which provides two registers (`rs1` and `rs2`) to custom instruction processing logic and up to 1,024 IDs to perform different functions.

Table 23: Custom Instruction Ports

Where *n* is the core number (0, 1, 2, or 3).

Port	Direction	Description
cpu*n*_customInstruction_cmd_valid	Output	Indicates that registers rs1 and rs2 are present and ready for processing.
cpu*n*_customInstruction_cmd_ready	Input	Indicates that the custom processing logic is ready to process register rs1 and rs2 from the CPU.
cpu*n*_customInstruction_function_id[9:0]	Output	Function id for the custom instruction.
cpu*n*_customInstruction_inputs_0[31:0]	Output	Register rs1 for the custom instruction.
cpu*n*_customInstruction_inputs_1[31:0]	Output	Register rs2 for the custom instruction.
cpu*n*_customInstruction_rsp_valid	Input	Indicates that the custom instruction result is available.
cpu*n*_customInstruction_rsp_ready	Output	Indicates that the CPU is ready to accept the custom instruction result.
cpu*n*_customInstruction_outputs_0[31:0]	Input	Result of the custom instruction.

Figure 6: Custom Instruction Waveform



PLIC Peripheral Interface

Use the `SYSTEM_PLIC_CTRL` parameter to reference the interface PLIC interface.

Table 24: RISC-V PLIC Operation Parameters

Defines	Description
Interrupt priorities registers	The interrupt priority for each interrupt source.
Interrupt pending bits registers	The interrupt pending status of each interrupt source.
Interrupt enables registers	Enables the interrupt source of each context.
Priority thresholds registers	The interrupt priority threshold of each context.
Interrupt claim registers	The register to acquire interrupt source ID of each context.
Interrupt completion registers	The register to send interrupt completion message to the associated gateway.

The **soc.h** file contains a number of PLIC parameters to specify the interrupt ID for the various peripherals.

Table 25: PLIC Interrupt ID Parameters

Where *n* is the peripheral number and *m* is the interrupt ID.

Parameter	Refer to
SYSTEM_PLIC_SYSTEM_AXI_A_INTERRUPT	Interrupts on page 9
SYSTEM_PLIC_USER_INTERRUPT_A_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_B_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_C_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_D_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_E_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_F_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_G_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_H_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_I_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_J_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_K_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_L_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_M_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_N_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_O_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_P_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_Q_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_R_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_S_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_T_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_U_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_V_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_W_INTERRUPT SYSTEM_PLIC_USER_INTERRUPT_X_INTERRUPT	Interrupts on page 9
SYSTEM_PLIC_SYSTEM_USER_TIMER_*n*_INTERRUPTS_*m*	Timer Limit Register: 0x0000_00044

User Timer

You can adjust the interval period to generate a timer tick pulse by setting the prescaler register, based on the system clock or peripheral clock (if enabled).

Table 26: User Timer Register Map

Address Offset	Register Name	Privilege	Width
0x0000_0000	Prescaler	Read/Write	32
0x0000_0040	Timer configuration	Read/Write	32
0x0000_0044	Timer limit	Read/Write	32
0x0000_0048	Timer value	Read	32

Prescaler Register: 0x0000_0000

31 … 16	15 … 0
Reserved	Prescaler value

Bits	Field	Description	Privilege
0-15	Prescaler value	The clock divider ratio. Example: 16'd0: divide by 1 16'd1: divide by 2 ... 16'd65534: divide by 65535 16'd65535: divide by 65536	Read/Write
16-31	Reserved	Reserved.	-

Timer Configuration Register: 0x0000_0040

31 … 17	16	15 … 2	1	0
Reserved	Self-restart	Reserved	With prescaler	Without prescaler

Bits	Field	Description	Privilege
0	Without prescaler	Write 1'b1 to run timer without prescaler.	Read/Write
1	With prescaler	Write 1'b1 to run timer with prescaler.	Read/Write
2-15	Reserved	Reserved.	N/A
16	Self-restart	Write 1'b1 to enable self-restart when reach timer limit.	Read/Write
17-31	Reserved	Reserved.	N/A

Timer Limit Register: 0x0000_0044

31	0
Limit value	

Bits	Field	Description	Privilege
0-31	Limit value	Value for the timer to generate a trigger pulse. The final value with the prescaler enabled is: (limit value + 1) * (prescaler value + 1)	Read/Write

Timer Value Register: 0x0000_0048

31	0
Value	

Bits	Field	Description	Privilege
0-31	Value	Value of the increment counter.	Read

Clint

The core local interrupt (clint) consists of a 64-bit realtime counter, which is driven by `io_systemClk` or `io_peripheralClk` (if enabled). The clint counter value increases monotonically. Clint is also responsible for handling the control and status via software interrupt.

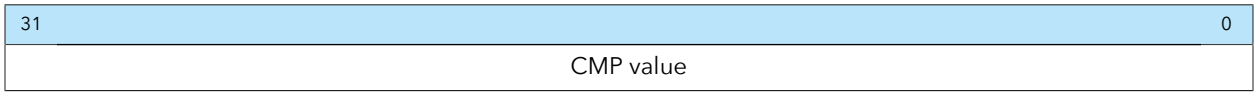
Table 27: Clint Register Map

Address Offset	Register Name	Privilege	Width
0x0000_0000	PIP	Read/Write	32
0x0000_4000	MTIMECMP (LO)	Write	32
0x0000_4004	MTIMECMP (HI)	Write	32
0x0000_BFF8	MTIME (LO)	Read	32
0x0000_BFFC	MTIME (HI)	Read	32

PIP Register: 0x0000_0000

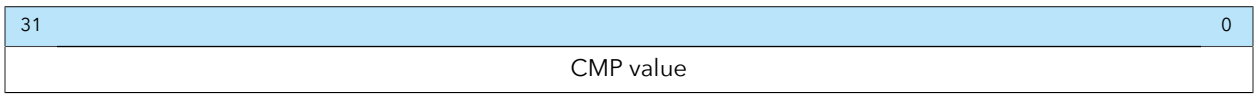
Bits	Field	Description	Privilege
0	Software Interrupt	Machine mode software interrupt.	Read/Write
2-31	Reserved	Reserved.	-

MTIMECMP Register (LO): 0x0000_4000



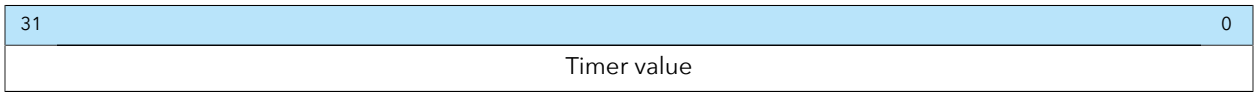
Bits	Field	Description	Privilege
0-31	CMP value	Timer interrupt trigger value (low 32 bits).	Write

MTIMECMP Register (HI): 0x0000_4004



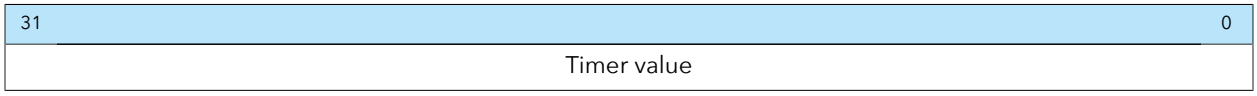
Bits	Field	Description	Privilege
0-31	CMP value	Timer interrupt trigger value (high 32 bits).	Write

MTIME Register (LO): 0x0000_BFF8



Bits	Field	Description	Privilege
0-31	Timer value	Value of increment counter (low 32 bits).	Read

MTIME Register (HI): 0x0000_BFFC



Bits	Field	Description	Privilege
0-31	Timer value	Value of increment counter (high 32 bits).	Read

Control and Status Registers

The following tables show the machine-level CSR implementation.

Table 28: Machine Information Register

Address	Register Name	Privilege	Description	Width
0xF14	mhartid	Read	Hardware thread ID.	32

Table 29: Machine Trap Registers

Address	Register Name	Privilege	Description	Width
0x300	mstatus	Read/Write	Machine status register.	13
0x304	mie	Read/Write	Machine interrupt enable register.	12
0x305	mtvec	Read/Write	Machine trap handler base address.	32

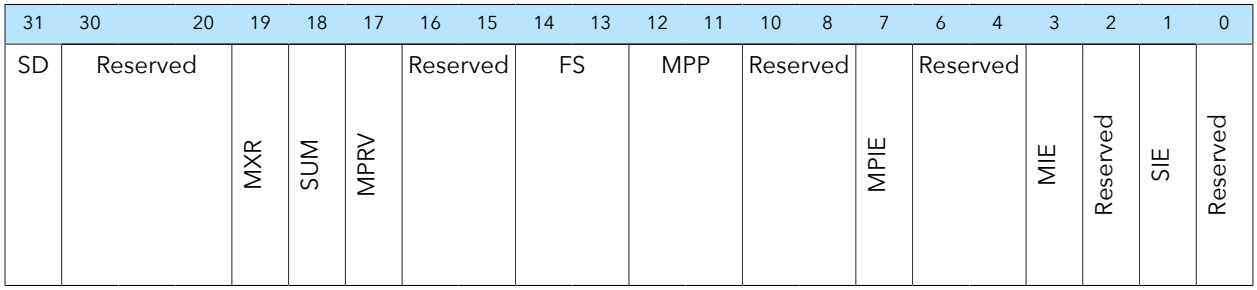
Table 30: Machine Trap Handling Registers

Address	Register Name	Privilege	Description	Width
0x340	mscratch	Read/Write	Scratch register for machine trap handlers.	32
0x341	mpec	Read/Write	Machine exception program counter.	32
0x342	mcause	Read	Machine trap cause.	32
0x343	mtval	Read	Machine bad address or instruction.	32
0x344	mip	Read/Write	Machine interrupt pending.	12

Machine-Level CSR

Machine Status Register (mstatus): 0x300

The `mstatus` register is a 13-bits read/write register formatted. The `mstatus` register keeps track of and controls the hart's current operating state. Restricted views of the `mstatus` register appear as the `sstatus` and `ustatus` registers in the S-level and U-level ISAs, respectively.



Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SIE	Machine global interrupt enable register.	N/A	N/A	Read/Write
2	Reserved	Reserved.	N/A	N/A	N/A
3	MIE	Machine interrupt enable register.	Read/Write	Read/Write	Read/Write
4-6	Reserved	Reserved.	N/A	N/A	N/A
7	MPIE	Machine previous interrupt enable.	Read/Write	Read/Write	Read/Write
8-10	Reserved	Reserved.	N/A	N/A	N/A
11-12	MPP	Machine previous privilege mode.	Read/Write	Read/Write	Read/Write
13-14	FS	Status of the floating-point unit. 2'b00: Off 2'b01: Initial 2'b10: Clean 2'b11: Dirty	N/A	Read	N/A
15-16	Reserved	Reserved.	N/A	N/A	N/A
17	MPRV	Modifies the privilege level that loads and stores the executables. 1'b1: Load and store memory address are translated and protected 1'b0: Normal mode	N/A	N/A	Read/Write

Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
18	SUM	Modifies the privilege with which S-mode loads and stores access virtual memory. 1'b1: Access permitted 1'b0: S-mode memory accesses to pages that are accessible by U-mode will fault	N/A	N/A	Read/Write
19	MXR	Modifies the privilege that loads access virtual memory. 1'b1: Loads from pages marked with either readable or executable will succeed 1'b0: Only loads from pages marked readable will succeed	N/A	N/A	Read/Write
20-30	Reserved	Reserved.	N/A	N/A	N/A
31	SD	Indicates the presence of FS field with dirty state that requires saving extended user context to memory.	N/A	Read	N/A

Machine Interrupt Enable Register (mie): 0x304

The `mie` register is a 12-bit read/write register containing interrupt enable bits.

11	10	9	8	7	6	5	4	3	2	1	0
MEIE	Reserved	SEIE	Reserved	MTIE	Reserved	STIE	Reserved	MSIE	Reserved	SSIE	Reserved

Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SSIE	Supervisor mode software interrupt.	N/A	N/A	Read/Write
2	Reserved	Reserved.	N/A	N/A	N/A
3	MSIE	Machine software interrupt enable.	Read/Write	Read/Write	Read/Write
4	Reserved	Reserved.	N/A	N/A	N/A
5	STIE	Supervisor mode timer interrupt enable.	N/A	N/A	N/A
6	Reserved	Reserved.	N/A	N/A	N/A
7	MTIE	Machine timer interrupt enable.	Read/Write	Read/Write	Read/Write
8	Reserved	Reserved.	N/A	N/A	N/A
9	SEIE	Supervisor mode external interrupt enable.	N/A	N/A	Read/Write
10	Reserved	Reserved.	N/A	N/A	N/A
11	MEIE	Machine external interrupt enable.	Read/Write	Read/Write	Read/Write

Machine Trap-Vector Base-Address Register (mtvec): 0x305

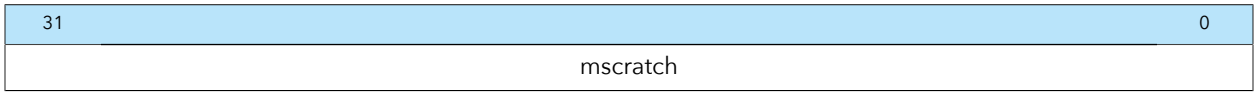
The `mtvec` register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (base) and a vector mode (mode).

31	2	1	0
base		mode	

Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-1	mode	Vector mode. 0: Direct. All exceptions set pc to BASE 1: Vectored. Asynchronous interrupts set pc to BASE + 4xcause ≥ 2: Reserved	Read/Write	Read/Write	Read/Write
2-31	base	Vector base address.	Read/Write	Read/Write	Read/Write

Machine Scratch Register (mscratch): 0x340

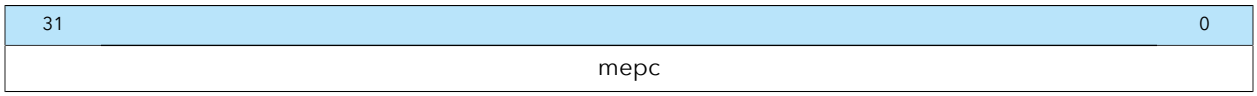
The `mscratch` register is a 32-bit read/write register dedicated for use by machine mode. Typically, it is used to hold a pointer to a machine mode hart-local context space and swapped with a user register upon entry to an M-mode trap handler.



Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-31	mscratch	A temporary scratch space that can be used by machine mode software.	Read/Write	Read/Write	Read/Write

Machine Exception Program Counter (mepc): 0x341

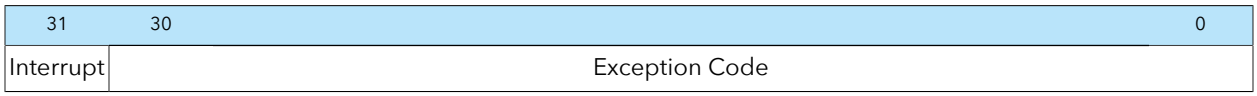
`mepc` is a 32-bit read/write register. The low bit of `mepc` (`mepc[0]`) is always zero. On implementations that support only IALIGN=32, the two low bits (`mepc[1:0]`) are always zero.



Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-31	mepc	Machine exception program counter.	Read/Write	Read/Write	Read/Write

Machine Cause Register (mcause): 0x342

The `mcause` register is a 32-bit read-write register. When a trap is taken into M-mode, `mcause` is written with a code indicating the event that caused the trap. Otherwise, `mcause` is never written by the implementation, though it may be explicitly written by software.



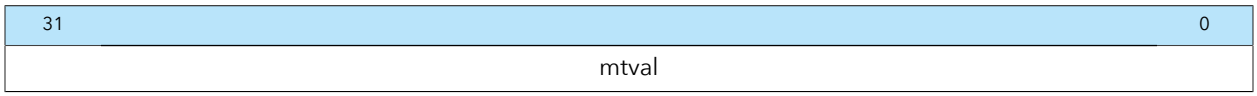
Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0-30	Exception code	See **Table 31: Machine Cause Register (mcause) Values after Trap** on page 31.	Read	Read	Read
31	Interrupt	mcause interrupt bit.	Read	Read	Read

Table 31: Machine Cause Register (mcause) Values after Trap

Interrupt	Exception Code	Description
1	0	Reserved.
1	1	Supervisor software interrupt.
1	2	Reserved.
1	3	Machine software interrupt.
1	4	User timer interrupt.
1	5	Supervisor timer interrupt.
1	6	Reserved.
1	7	Machine timer interrupt.
1	8	User external interrupt.
1	9	Supervisor external interrupt.
1	10	Reserved.
1	11	Machine external interrupt.
1	≥12	Reserved.
0	0	Instruction address misaligned.
0	1	Instruction access fault.
0	2	Illegal instruction.
0	3	Breakpoint.
0	4	Load address misaligned.
0	5	Load access fault.
0	6	Store/AMO address misaligned.
0	7	Store/AMO access fault.
0	8	Reserved.
0	9	Reserved.
0	10	Reserved.
0	11	Environment call from M-mode.
0	12	Instruction page fault.
0	13	Load page fault.
0	14	Reserved.
0	15	Store/AMO page fault.
0	≥16	Reserved.

Machine Trap Value Register (mtval): 0x343

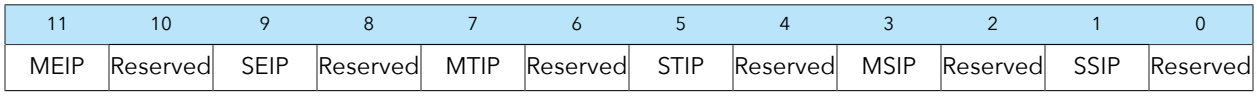
The `mtval` register is a 32-bit register. When a trap is taken into M-mode, `mtval` is either set to zero or written with exception-specific information to assist software in handling the trap. Otherwise, `mtval` is never written by the implementation, though it may be explicitly written by software. The hardware platform will specify which exceptions must set `mtval` informatively and which may unconditionally set it to zero.



Bits	Field	Description	Single/Multi Core	w/FPU	w/MMU
0-31	mtval	Machine trap value register bit.	Read/Write	Read/Write	Read/Write

Machine Interrupt Pending Register (mip): 0x344

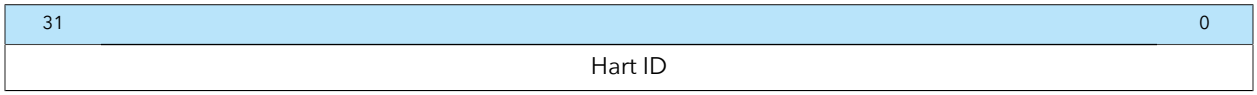
The `mip` register is a 12-bit read/write register containing information on pending interrupts.



Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SSIP	Supervisor software interrupt pending.	N/A	N/A	Read/Write
2	Reserved	Reserved.	N/A	N/A	N/A
3	MSIP	Machine software interrupt pending.	Read/Write	Read/Write	Read/Write
4	Reserved	Reserved.	N/A	N/A	N/A
5	STIP	Supervisor timer interrupt pending.	N/A	N/A	R/W
6	Reserved	Reserved.	N/A	N/A	N/A
7	MTIP	Machine timer interrupt pending.	Read	Read	Read
8	Reserved	Reserved.	N/A	N/A	N/A
9	SEIP	Supervisor external interrupt pending.	N/A	N/A	Read/Write
10	Reserved	Reserved.	N/A	N/A	N/A
11	MEIP	Machine external interrupt pending.	Read	Read	Read

Hart ID Register (mhartid): 0xF14

The `mhartid` CSR is a 32-bit read-only register containing the integer ID of the hardware thread running the code. This register must be readable in any implementation. Hart IDs might not necessarily be numbered continuously in a multiprocessor system, but at least one hart must have a hart ID of zero. Hart IDs must be unique.



Bits	Field	Description	Single / Multi-Core	w/FPU	w/MMU
0-31	Hart ID	Hardware thread ID.	Read	Read	Read

Timer Related CSR

CSR	Name	Description	Single / Multi-Core	w/FPU	w/MMU
0xC00	cycle	Cycle counter for **RDCYCLE** instruction.	N/A	N/A	Read
0xC01	time	Timer for **RDTIME** instruction.	N/A	N/A	Read
0xC02	timeh	Instructions-retired counter for **RDINSTRET** instruction.	N/A	N/A	Read

Floating-Point Related CSR

CSR	Name	Description	Single / Multi-Core	w/FPU	w/MMU
0x001	fflags	Floating-point accrued exceptions.	N/A	Read/Write	N/A
0x002	frm	Floating-point dynamic rounding mode.	N/A	Read/Write	N/A
0x003	fcsr	Floating-point control and status register (frm + fflags).	N/A	Read/Write	N/A

Supervisor-Level CSR

The supervisor should only view CSR state that should be visible to a supervisor-level operating system. There is no information about the existence (or non-existence) of higher privilege levels (machine level or other) visible in the CSRs accessible by the supervisor. Many supervisor CSRs are a subset of the equivalent machine mode CSR.

Supervisor Status Register (sstatus): 0x100

the sstatus register is a subset of the mstatus register.

31	30 20	19	18	17	16 15	14 13	12 9	8	7 6	5	4 2	1	0
SD	Reserved	MXR	SUM	MPRV	Reserved	FS	Reserved	SPP	Reserved	SPIE	Reserved	SIE	Reserved

Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SIE	Supervisor global interrupt enable register.	N/A	N/A	Read/Write
2-4	Reserved	Reserved.	N/A	N/A	N/A
5	SPIE	Supervisor previous interrupt enable register.	N/A	N/A	Read/Write
6-7	Reserved	Reserved.	N/A	N/A	N/A
8	SPP	Supervisor previous privilege mode.	N/A	N/A	Read/Write
9-12	Reserved	Reserved.	N/A	N/A	N/A
13-14	FS	Status of the floating-point unit. 2'b00: Off 2'b01: Initial 2'b10: Clean 2'b11: Dirty	N/A	Read/Write	N/A
15-16	Reserved	Reserved	N/A	N/A	N/A
17	MPRV	Modifies the privilege level at which loads and stores the executables. 1'b1: Load and store memory address are translated and protected 1'b0: Normal mode	N/A	N/A	Read/Write

Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
18	SUM	Modifies the privilege with which S-mode loads and stores access virtual memory. 1'b1: Access permitted 1'b0: S-mode memory accesses to pages that are accessible by U-mode will fault	N/A	N/A	Read/Write
19	MXR	Modifies the privilege that loads access virtual memory. 1'b1: loads from pages marked with either readable or executable will succeed 1'b0: only loads from page marked readable will succeed	N/A	N/A	Read/Write
20-30	Reserved	Reserved.	N/A	N/A	N/A
31	SD	Indicates the presence of FS field with dirty state that requires saving extended user context to memory.	N/A	Read/Write	N/A

Supervisor Interrupt Enable Register (sie): 0x104

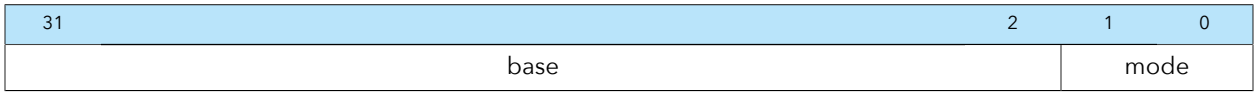
The `sie` register is a 12-bit read/write register containing interrupt enable bits.

31	10	9	8	7	6	5	4	2	1	0
Reserved		SEIE	Reserved			STIE	Reserved		SSIE	Reserved

Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SSIE	Supervisor mode software interrupt.	N/A	N/A	Read/Write
2-4	Reserved	Reserved.	N/A	N/A	N/A
5	STIE	Supervisor mode timer interrupt enable.	N/A	N/A	Read/Write
6-8	Reserved	Reserved.	N/A	N/A	N/A
9	SEIE	Supervisor mode external interrupt enable.	N/A	N/A	Read/Write
10-31	Reserved	Reserved.	N/A	N/A	N/A

Supervisor Trap-Vector Base-Address Register (stvec): 0x305

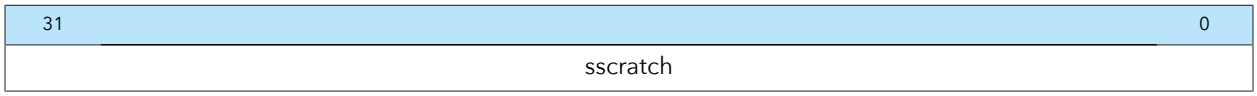
The `stvec` register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (base) and a vector mode (mode).



Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-1	mode	Vector mode. 0: Direct. All exceptions set pc to BASE 1: Vectored. Asynchronous interrupts set pc to BASE + 4xcause ≥ 2: Reserved	N/A	N/A	Read/Write
2-31	base	Vector base address.	N/A	N/A	Read/Write

Supervisor Scratch Register (sscratch): 0x140

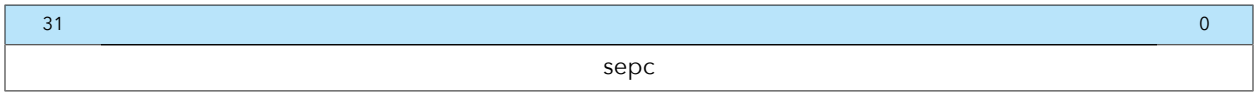
The `sscratch` register is a 32-bit read/write register dedicated for use by machine mode. Typically, `sscratch` is used to hold a pointer to the hart-local supervisor context while the hart is executing user code. At the beginning of a trap handler, `sscratch` is swapped with a user register to provide an initial working register.



Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-31	sscratch	A temporary scratch space that can be used by supervisor mode software.	N/A	N/A	Read/Write

Supervisor Exception Program Counter (sepc): 0x141

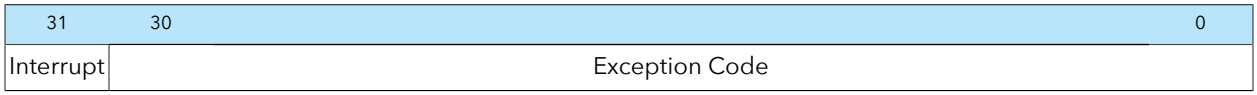
`sepc` is a 32-bit read/write register. The low bit of `sepc` (`sepc[0]`) is always zero. On implementations that support only IALIGN=32, the two low bits (`sepc[1:0]`) are always zero.



Bits	Field	Description	Single/ Multi-Core	w/FPU	w/MMU
0-31	sepc	Supervisor exception program counter.	N/A	N/A	Read/Write

Supervisor Cause Register (scause): 0x142

The `scause` register is a 32-bit read-write register. When a trap is taken into S-mode, `scause` is written with a code indicating the event that caused the trap. Otherwise, `scause` is never written by the implementation, though it may be explicitly written by software.



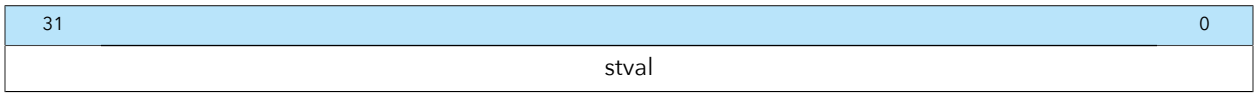
Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0-30	Exception code	See Table 32: Machine Cause Register (scause) Values after Trap on page 37.	N/A	N/A	Read
31	Interrupt	scause interrupt bit.	N/A	N/A	Read

Table 32: Machine Cause Register (scause) Values after Trap

Interrupt	Exception Code	Description
1	0	Reserved.
1	1	Supervisor software interrupt.
1	2-4	Reserved.
1	5	Supervisor timer interrupt.
1	6-8	Reserved.
1	9	Supervisor external interrupt.
1	10-15	Reserved.
0	0	Instruction address misaligned.
0	1	Instruction access fault.
0	2	Illegal instruction.
0	3	Breakpoint.
0	4	Load address misaligned.
0	5	Load access fault.
0	6	Store/AMO address misaligned.
0	7	Store/AMO access fault.
0	8	Environment call from U-mode.
0	9	Environment call from S-mode.
0	10	Reserved.
0	11	Environment call from M-mode.
0	12	Instruction page fault.
0	13	Load page fault.
0	14	Reserved.
0	15	Store/AMO page fault.
0	≥16	Reserved.

Supervisor Trap Value Register (stval): 0x143

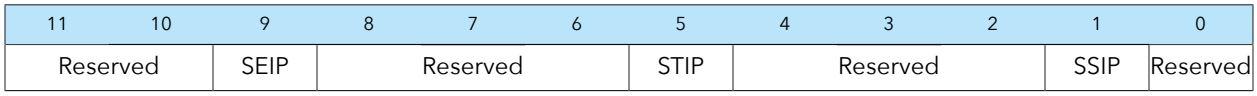
The `stval` register is a 32-bit register. When a trap is taken into S-mode, `stval` is either set to zero or written with exception-specific information to assist software in handling the trap. Otherwise, `stval` is never written by the implementation, though it may be explicitly written by software. The hardware platform will specify which exceptions must set `stval` informatively and which may unconditionally set it to zero.



Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0-31	stval	Supervisor trap value register bit.	N/A	N/A	Read/Write

Supervisor Interrupt Pending Register (sip): 0x144

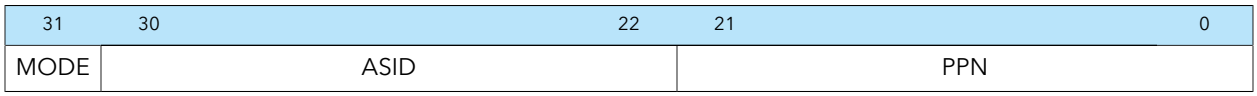
The `sip` register is a 12-bit read/write register containing information on pending interrupts.



Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0	Reserved	Reserved.	N/A	N/A	N/A
1	SSIP	Supervisor software interrupt pending.	N/A	N/A	Read/Write
2-4	Reserved	Reserved.	N/A	N/A	N/A
5	STIP	Supervisor timer interrupt pending.	N/A	N/A	Read/Write
6-8	Reserved	Reserved.	N/A	N/A	N/A
9	SEIP	Supervisor external interrupt pending.	N/A	N/A	Read/Write
10-11	Reserved	Reserved.	N/A	N/A	N/A

Supervisor Address Translation Protection Register (satp): 0x180

The `satp` register is a 32-bit register, which controls supervisor mode address translation and protection. This register holds the physical page number (PPN) of the root page table. For example, its supervisor physical address divided by 4KB; an address space identifier (ASID) that facilitates address translation fences on a per-address-space basis; and the `MODE` field that elects the current address translation scheme.



Bits	Field	Description	Single/Multi-Core	w/FPU	w/MMU
0-21	PPN	Physical page number.	N/A	N/A	Read/Write
22-30	ASID	Address space identifier.	N/A	N/A	Read/Write
31	MODE	1'b1: Page-based 32-bits virtual processing. 1'b0: No translation or protection.	N/A	N/A	Read/Write

Revision History

Table 33: Revision History

Date	Version	Description
January 2024	1.0	Initial release.