Sapphire 高性能 RISC-V SoC 数据手册

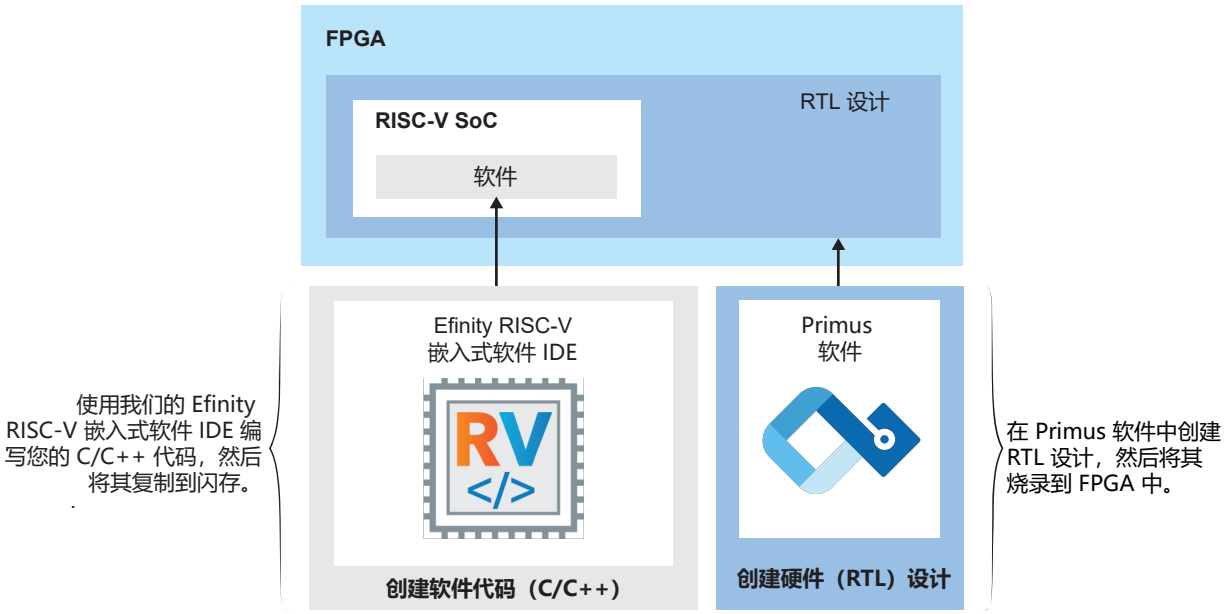
DS-SAPPHIREHPB-v1.0
2024-10
www.elitestek.com

目录

简介

易灵思 提供高性能硬核 Sapphire RISC-V SoC，其系统时钟最高频率达 1 GHz。Sapphire 高性能 RISC-V SoC 提供支持各种外围器件的接口，例如内存控制器、直接内存访问通道、自定义指令和 I/O 器件。 您可以通过在 IP Manager 中配置 SoC 来选择要使用的接口。

图 1: Sapphire RISC-V SoC 设计流程



功能

- 4 个 VexRiscv 处理器，具有 6 条流水线阶段（获取、注入、解码、执行、内存和写回）、中断和异常处理，具有机器模式和监督模式。
- 最高 1 GHz 系统时钟频率
- 16 KB 带有 SPI 闪存 boot loader 的片上 RAM
- LPDDR4x 内存控制器
  - 支持 3.7 GB 内存模块
  - 1 个全双工 512 位 AXI4 接口，用于与外部内存通信
  - 用户可配置的外部内存总线频率
- 1 个用于用户逻辑的 AXI 主机通道，数据宽度为 128 位
- 1 个用于用户逻辑的 AXI 从机通道
- 每个内核包括：
  - 4 路 16 KB 数据和指令缓存
  - 浮点单元 (FPU)
  - Linux 内存管理单元 (MMU)
  - 具有 1,024 个 ID 的自定义指令接口，用于执行各种功能
- 支持 RISC-V 扩展，例如整数、乘法、原子、压缩、单精度浮点数和双精度浮点数。
- 具有 8 个硬件断点的 JTAG 调试模块
- 外设：
  - 2 个用户计时器
  - 24 个用户中断

FPGA 支持

Sapphire 高性能 RISC-V SoC 使用 TJ135 和 TJ375 FPGA 的硬核 RISC-V 模块，并且仅支持这些 FPGA。

性能基准

可以使用 Dhrystone 和 Coremark 基准程序对 CPU 性能进行基准测试，处理器间差别比较更容易。

开发板： 钛金系列

Primus 2023.2 版本

表 1: 性能基准

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

(1) 待定特性

功能描述

硬核 RISC-V SoC 模块 (HRB) 由 4 个 Vexriscv 内核组成。每个内核由一个内存管理单元 (MMU) 组成，MMU 负责处理与处理器相关的所有内存和缓存操作。内核以 1.0 GHz 频率运行并控制以下 RISC-V 扩展：

- 32 位整数 (I)
- 乘法 (M)
- 原子 (A)
- 压缩 (C)
- 单精度和双精度浮点 (FD) 指令扩展

您可以使用自定义指令接口执行自定义操作，并使用具有 8 个硬件断点且符合 RISC-V 调试规范的 JTAG 调试模块进行调试。

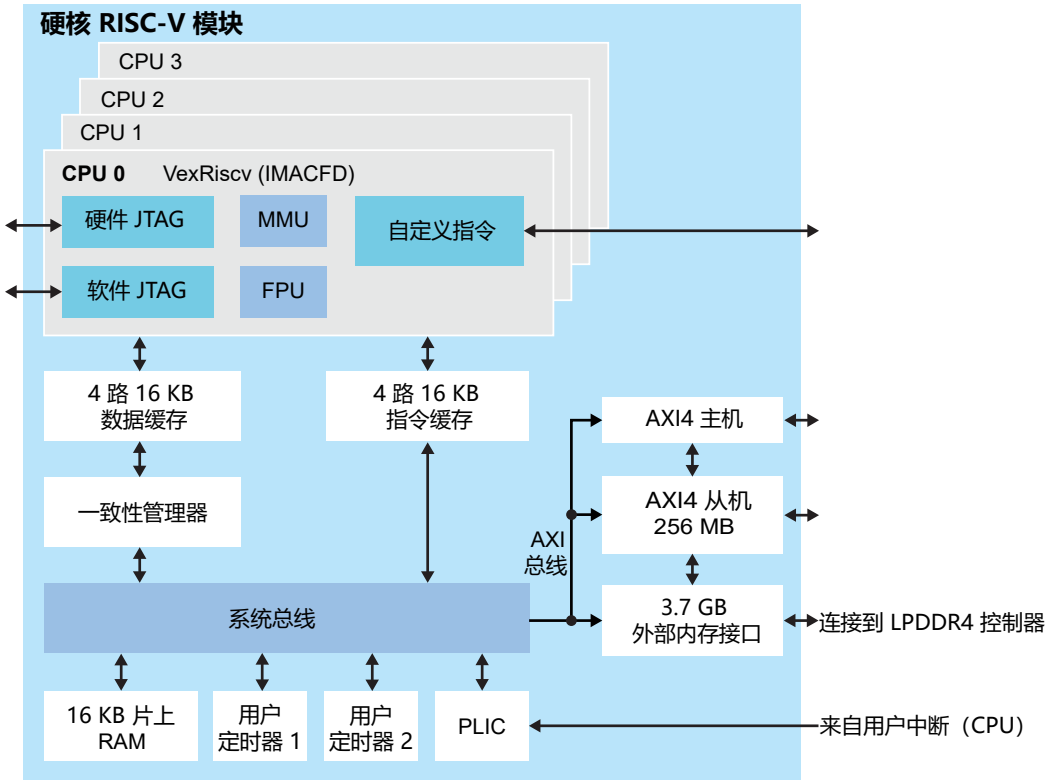
HRB 提供 4 路 16 KB 数据和指令缓存，以适应与主内存的数据和指令交换。

此外，HRB 具有 16 KB 片上 RAM，可用于多种用途。默认情况下，此片上 RAM 保存 bootloader，bootloader 从 SPI 闪存中检索 124 KB 数据，并在 HRB 启动期间传输到主内存。HRB 还提供一个中断控制器，可用作 CLINT 计时器、24 个用户中断和 2 个用户计时器的辅助控制器。您可以使用 Efinity RISC-V IDE 软件配置中断设备间的优先级。

为了与 LPDDR4 内存通信，HRB 提供了一个 512 位数据宽度的 AXI4 全双工接口和一个 128 位数据宽度的 AXI4 主机，您可连接到直接内存访问 (DMA)。但是，AXI 主机只能访问 LPDDR4 内存。HRB 还提供了一个 32 位、256 MB AXI4 从机接口，用于与来自 FPGA 内核元架构的逻辑进行通信。

您可以使用 Primus® 软件中的 IP Manager 自定义 Sapphire RISC-V SoC。

图 2: Sapphire 硬核 RISC-V 框图



地址映射

参数名称和地址映射定义见 **/embedded_sw/efx_hard_soc/bsp/efinix/EfxSapphireSoc/include/soc.h**。

表 2: 默认地址映射、中断 ID 和缓存通道

AXI 用户从机通道位于无缓存区域 (I/O)，以兼容 AXI-Lite。

设备	参数	大小	中断 ID	区域
片外内存	SYSTEM_DDR_BMB	3.7 GB	–	缓存
AXI 用户从机	SYSTEM_AXI_A_BMB	256 MB	–	I/O
用户计时器 0	SYSTEM_USER_TIMER_0_CTRL	4 KB	49	I/O
用户计时器 1	SYSTEM_USER_TIMER_1_CTRL	4 KB	50	I/O
CLINT 计时器	SYSTEM_CLINT_CTRL	4 KB	–	I/O
PLIC	SYSTEM_PLIC_CTRL	4 MB	–	I/O
片上 BRAM	SYSTEM_RAM_A_BMB	16 KB	–	缓存
外部中断	–	–	[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

访问 I/O 区域中的地址时，使用关键字 **volatile** 对指针进行类型转换。编译器会将其识别为内存映射的 I/O 寄存器，不会优化读/写访问。转换示例见以下命令：

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

对于缓存区域，突发长度相当于 AXI 突发长度 8。对于 I/O 区域，突发长度相当于 AXI 突发长度 1。AXI 用户从机通过断开未使用的输出并将常数 1 驱动到输入端口与 AXI-Lite 兼容。

注意： RISC-V GCC 编译器不支持从 0x0000_0000 开始的用户地址空间。

时钟

表 3: 时钟端口

端口	方向	描述
io_systemClock	输入	为 SoC 提供高达 1 GHz 的时钟。
io_peripheralClock	输入	为 APB3 外设和 AXI4 从机提供高达 250 MHz 的时钟。
io_memoryClock	输入	为外部内存总线提供高达 250 MHz 的时钟。
io_ddrMaster_0_clk	输入	为 AXI 主机总线提供高达 250 MHz 的时钟。
io_cfuClk	输入	为自定义指令总线提供高达 250 MHz 的时钟。

中断

表 4: 中断端口

端口	方向	描述
userInterruptA userInterruptB userInterruptC userInterruptD userInterruptE userInterruptF userInterruptG userInterruptH userInterruptI userInterruptJ userInterruptK userInterruptL userInterruptM userInterruptN userInterruptO userInterruptP userInterruptQ userInterruptR userInterruptS userInterruptT userInterruptU userInterruptV userInterruptW userInterruptX	输入	提供外部中断。
axiAInterrupt	输入	用户 AXI 从机通道中断。

复位

Sapphire 高性能 RISC-V SoC 具有主复位信号 `io_asyncReset`，可触发系统复位。您的 RTL 设计应保持 `io_asyncReset` 至少 3 个 `io_systemClk` 高电平状态周期，以完全复位整个 SoC 系统。当您断言 `io_asyncReset` 时，SoC 会断言：

- `io_systemReset`，这会复位 RISC-V 处理器和片上内存。
- `io_peripheralReset`，这会复位 APB3 外设和 AXI4 从机。
- `io_memoryReset`，这会复位 LPDDR4 内存控制器的 AXI 接口。
- `io_ddrMasters_0_reset`，这会响应 AXI 主机通道 0 的复位并与 `io_ddrMasters_0_clk` 同步。
- `io_cfuReset`，这会响应自定义指令的复位，并与 `io_cfuClk` 同步。

SoC 同时断言 `io_memoryReset`、 `io_ddrMaster_0_reset`、`io_peripheralReset`、 `io_cfuReset` 和 `io_systemReset` 信号，以允许 AXI 主机在复位完成后访问 AXI 交叉开关。

一旦 `io_systemReset` 输出低电平，执行用户二进制代码。

图 3: 复位时序图

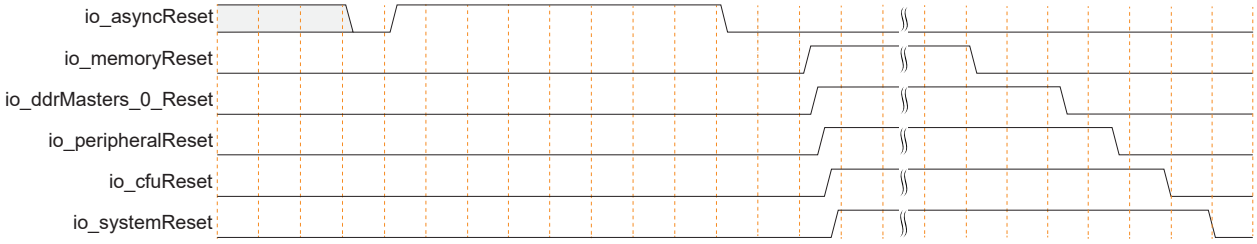


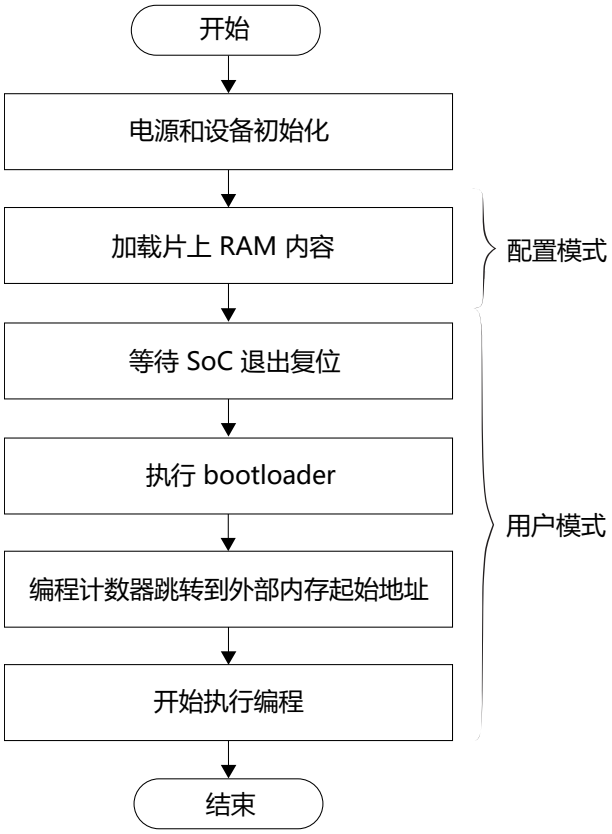
表 5: 复位端口

端口	方向	说明
io_asyncReset	输入	整个系统的高电平有效异步复位。
io_systemReset	输出	系统时钟 (io_systemClk) 的同步高电平有效复位。
io_peripheralReset	输出	外设时钟 (io_peripheralClock) 的同步高电平有效复位。
io_memoryReset	输出	来自 RISC-V SoC 的外部内存复位源。
io_ddrMasters_0_reset	输出	响应 AXI 主机的复位。
io_cfuReset	输出	自定义指令时钟 (io_cfuClock) 的同步高电平有效复位。

启动流程

Sapphire 高性能 RISC-V SoC 的启动流程见以下流程图。

图 4: 正常启动流程 - 流程图



在设备通电并经过初始化序列后，配置模块开始将 RAM 内容（默认情况下为 bootloader）卸载到 Sapphire SoC 的片上 RAM。其他接口模块（如专用于 SoC 的 PLL、软逻辑模块和 GPIO 配置）应在此阶段进行编程。此外，SPI 控制器需要包含在 bootloader 的软逻辑模块中。

一旦 `io_asyncReset` 从软逻辑模块中释放， Sapphire SoC 就会运行其复位序列。此外，CPU 开始执行 bootloader，允许 SPI 控制器从 SPI 闪存启动 RISC-V 固件二进制文件的检索过程。检索到的数据被重定向并存储在 LPDDR4 内存中。此循环过程持续到 bootloader 完成其任务。然后，CPU 跳转到 LPDDR4 内存起始地址以开始程序执行。

门控 SoC 复位

`io_asyncReset` 信号高电平有效，可使 Sapphire 高性能 RISC-V SoC 所有时钟域中的所有逻辑恢复到其已知的初始状态。有关复位时序图，请参阅 第8页的**图 3: 复位时序图**。`io_asyncReset` 信号在配置模式下必须处于高电平有效状态，并且在 LPDDR4 控制器校准完成后取消置位。 易灵思® 建议按下图所示门控复位。

图 5: 门控 SoC 复位

PLL 锁定信号必须来自专用 `io_systemClk` PLL。用户 I/O 是指通常连接到外部 GPIO 开关的用户自控信号。但是，您可以使用软逻辑从另一个源设计此信号。用于生成 `io_asyncReset` 的所有相关信号都必须具有高电平有效属性。

AXI 接口

Sapphire 高性能 RISC-V SoC 具有 512 位全双工 AXI4 接口，用于与外部内存通信。

此外，Sapphire 高性能 RISC-V SoC 还具有全双工 AXI4 接口，用于连接到用户逻辑。

- 有一个 AXI4 从机接口，与 AXI-Lite 兼容（axlen 始终为 0）。
- 有一个可选的全双工 AXI4 主机接口，128 位数据宽度。

了解更多： 有关 AXI 通道描述和握手信息，请参阅 AMBA AXI 和 ACE 协议规范。

连接到外部内存的 AXI 接口

表 6: 用于读写的 AXI 从机全双工地址通道

端口	方向	描述
io_ddrA_aw_valid	输出	外部内存写地址有效。
io_ddrA_aw_ready	输入	外部内存写地址就绪。
io_ddrA_aw_payload_addr[31:0]	输出	外部内存写地址。
io_ddrA_aw_payload_id[7:0]	输出	外部内存写地址 ID。
io_ddrA_aw_payload_region[3:0]	输出	外部内存写区域标识符。
io_ddrA_aw_payload_len[7:0]	输出	外部内存写突发长度。
io_ddrA_aw_payload_size[2:0]	输出	外部内存写突发大小。
io_ddrA_aw_payload_burst[1:0]	输出	外部内存写突发类型，仅 INCR。
io_ddrA_aw_payload_lock	输出	外部内存写锁定类型。
io_ddrA_aw_payload_cache[3:0]	输出	外部内存写内存类型。
io_ddrA_aw_payload_qos[3:0]	输出	外部内存写服务质量。
io_ddrA_aw_payload_prot[2:0]	输出	外部内存写保护类型。
io_ddrA_aw_payload_allStrb	输出	外部内存写全部选通。
io_ddrA_ar_valid	输出	外部内存读地址有效。
io_ddrA_ar_ready	输入	外部内存读地址就绪。
io_ddrA_ar_payload_addr[31:0]	输出	外部内存读地址。
io_ddrA_ar_payload_id[7:0]	输出	外部内存读地址 ID。
io_ddrA_ar_payload_region[3:0]	输出	外部内存读区域标识符。
io_ddrA_ar_payload_len[7:0]	输出	外部内存突发长度。
io_ddrA_ar_payload_size[2:0]	输出	外部内存读突发大小。
io_ddrA_ar_payload_burst[1:0]	输出	外部内存读突发类型，仅 INCR。
io_ddrA_ar_payload_lock	输出	外部内存读锁定类型。
io_ddrA_ar_payload_cache[3:0]	输出	外部内存读内存类型。
io_ddrA_ar_payload_qos[3:0]	输出	外部内存读服务质量。
io_ddrA_ar_payload_prot[2:0]	输出	外部内存读保护类型。

表 7: AXI 从机写数据通道

端口	方向	描述
io_ddrA_w_valid	输出	外部内存写有效。
io_ddrA_w_ready	输入	外部内存写就绪。
io_ddrA_w_payload_data[n:0]	输出	外部内存写数据。 长度固定为 512 位。
io_ddrA_w_payload_strb[m:0]	输出	外部内存写选通。 m 是 io_ddrA_w_payload_data[n:0] 的宽度除以 8。 长度固定为 64 位。
io_ddrA_w_payload_last	输出	外部内存写最后。

表 8: AXI 从机写响应通道

端口	方向	描述
io_ddrA_b_valid	输入	外部内存写响应有效。
io_ddrA_b_ready	输出	外部内存响应就绪。
io_ddrA_b_payload_id[7:0]	输入	外部内存响应 ID。
io_ddrA_b_payload_resp[1:0]	输入	外部内存写响应。

表 9: AXI 从机读数据通道

端口	方向	描述
io_ddrA_r_valid	输入	外部内存读有效。
io_ddrA_r_ready	输出	外部内存读就绪。
io_ddrA_r_payload_data[n:0]	输入	外部内存读数据。 长度固定为 512 位。
io_ddrA_r_payload_id[7:0]	输入	外部内存读 ID。
io_ddrA_r_payload_resp[1:0]	输入	外部内存读响应。
io_ddrA_r_payload_last	输入	外部内存读最后。

连接到用户逻辑的 AXI 接口

表 10: 用户从机写地址通道

端口	方向	描述
axiA_awvalid	输出	用户写地址有效。
axiA_awready	输入	用户写地址就绪。
axiA_awaddr[31:0]	输出	用户写地址。
axiA_awid[7:0]	输出	用户写地址 ID。
axiA_awregion[3:0]	输出	用户区域标识符。
axiA_awlen[7:0][(2)]	输出	用户突发长度。
axiA_awsize[2:0]	输出	用户突发大小。
axiA_awburst[1:0]	输出	用户突发类型，仅 INCR。
axiA_awlock	输出	用户锁定类型。
axiA_awcache[3:0]	输出	用户内存类型。
axiA_awqos[3:0]	输出	用户服务质量。
axiA_awprot[2:0]	输出	用户保护类型。

[(2)] axiA_awlen 始终输出 0，即突发长度为 1。此设置使 axiA 通道与 AXI-Lite 兼容。

表 11: 用户从机写数据通道

端口	方向	描述
axiA_wvalid	输出	用户写有效。
axiA_wready	输入	用户写就绪。
axiA_wdata[31:0]	输出	用户写数据。
axiA_wstrb[3:0]	输出	用户写选通。
axiA_wlast	输出	用户写最后。

表 12: 用户从机写响应通道

端口	方向	描述
axiA_bvalid	输入	用户写响应有效。
axiA_bready	输出	用户响应就绪。
axiA_bresp[1:0]	输入	用户写响应。

表 13: 用户从机读地址通道

端口	方向	描述
axiA_arvalid	输出	用户读地址有效。
axiA_arready	输入	用户读地址就绪。
axiA_araddr[31:0]	输出	用户读地址。
axiA_arregion[3:0]	输出	用户区域标识符。
axiA_arlen[7:0][(3)]	输出	用户突发长度。
axiA_arsize[2:0]	输出	用户突发大小。
axiA_arburst[1:0]	输出	用户突发类型，仅 INCR。
axiA_arlock	输出	用户锁定类型。
axiA_arcache[3:0]	输出	用户内存类型。
axiA_arqos[3:0]	输出	用户服务质量。
axiA_arprot[2:0]	输出	用户保护类型。

表 14: 用户从机读数据通道

端口	方向	描述
axiA_rvalid	输入	用户读有效。
axiA_rready	输出	用户读就绪。
axiA_rdata[31:0]	输入	用户读数据。
axiA_rresp[1:0]	输入	用户读响应。
axiA_rlast	输入	用户读最后。

表 15: 用户主机时钟和复位

端口	方向	描述
io_ddrMasters_0_clk	输入	AXI 主机时钟。
io_ddrMasters_0_reset	输出	AXI 主机高电平有效复位。

(3) axiA_arlen 始终输出 0，即突发长度为 1。此设置使 axiA 通道与 AXI-Lite 兼容。

AXI 主机接口

表 16: 用户主机写地址通道

端口	方向	描述
io_ddrMasters_0_aw_valid	输入	用户写地址有效。
io_ddrMasters_0_aw_ready	输出	用户写地址就绪。
io_ddrMasters_0_aw_payload_addr[31:0]	输入	用户写地址。
io_ddrMasters_0_aw_payload_id[7:0]	输入	用户写地址 ID。
io_ddrMasters_0_aw_payload_region[3:0]	输入	用户区域标识符。
io_ddrMasters_0_aw_payload_len[7:0]	输入	用户突发长度。
io_ddrMasters_0_aw_payload_size[2:0]	输入	用户突发大小。
io_ddrMasters_n_aw_payload_burst[1:0]	输入	用户突发类型，仅 INCR。
io_ddrMasters_0_aw_payload_lock	输入	用户锁定类型。
io_ddrMasters_0_aw_payload_cache[3:0]	输入	用户内存类型。
io_ddrMasters_0_aw_payload_qos[3:0]	输入	用户服务质量。
io_ddrMasters_0_aw_payload_prot[2:0]	输入	用户保护类型。
io_ddrMasters_0_aw_payload_allStrb[2:0]	输入	用户所有选通脉冲类型。

表 17: 用户主机写数据通道

端口	方向	描述
io_ddrMasters_0_w_valid	输入	用户写有效。
io_ddrMasters_0_w_ready	输出	用户写就绪。
io_ddrMasters_0_w_payload_data[m:0]	输入	用户写数据。 长度固定为 128 位。
io_ddrMasters_0_w_payload_strb[15:0]	输入	用户写选通脉冲。
io_ddrMasters_0_w_payload_last	输入	用户写最后。

表 18: 用户主机写响应通道

端口	方向	描述
io_ddrMasters_0_b_valid	输出	用户写响应有效。
io_ddrMasters_0_b_ready	输入	用户响应就绪。
io_ddrMasters_0_b_payload_id[7:0]	输出	用户响应 ID。
io_ddrMasters_0_b_payload_resp[1:0]	输出	用户写响应。

表 19: 用户主机读地址通道

端口	方向	描述
io_ddrMasters_0_ar_valid	输入	用户读地址有效。
io_ddrMasters_0_ar_ready	输出	用户读地址就绪。
io_ddrMasters_0_ar_payload_addr[31:0]	输入	用户读地址。
io_ddrMasters_0_ar_payload_id[7:0]	输入	用户读地址 ID。
io_ddrMasters_0_ar_payload_region[3:0]	输入	用户区域标识符。
io_ddrMasters_0_ar_payload_len[7:0]	输入	用户突发长度。
io_ddrMasters_0_ar_payload_size[2:0]	输入	用户突发大小。
io_ddrMasters_0_ar_payload_burst[1:0]	输入	用户突发类型，仅 INCR。
io_ddrMasters_0_ar_payload_lock	输入	用户锁定类型。
io_ddrMasters_0_ar_payload_cache[3:0]	输入	用户内存类型。
io_ddrMasters_0_ar_payload_qos[3:0]	输入	用户服务质量。
io_ddrMasters_0_ar_payload_prot[2:0]	输入	用户保护类型。

表 20: 用户主机读数据通道

端口	方向	描述
io_ddrMasters_0_r_valid	输出	用户读有效。
io_ddrMasters_0_r_ready	输入	外部内存读就绪。
io_ddrMasters_0_r_payload_data[m:0]	输出	外部内存读数据。 长度固定为128位。
io_ddrMasters_0_r_payload_id[7:0]	输出	外部内存读 ID。
io_ddrMasters_0_r_payload_resp[1:0]	输出	外部内存读响应。
io_ddrMasters_0_r_payload_last	输出	外部内存读最后。

JTAG 接口

Sapphire 高性能 RISC-V SoC 使用 JTAG 用户 TAP 接口模块与 OpenOCD 调试软件进行通信。

表 21: JTAG 端口

端口	方向	描述
jtagCtrl_enable	输入	表示用户指令对接口有效。
jtagCtrl_capture	输入	TAP 控制器处于捕获状态。
jtagCtrl_shift	输入	TAP 控制器处于移位状态。
jtagCtrl_update	输入	TAP 控制器处于更新状态。
jtagCtrl_reset	输入	TAP 控制器处于复位状态。
jtagCtrl_tdi	输入	用于调试的 JTAG TDI。
jtagCtrl_tdo	输出	用于调试的 JTAG TDO。
jtagCtrl_tck	输入	用于调试的 JTAG TCK。

JTAG 接口 - GPIO

Sapphire 高性能 RISC-V SoC 通过 GPIO 使用 JTAG 与 OpenOCD 调试软件通信。

表 22: JTAG 端口

端口	方向	描述
io_jtag_tdi	输入	用于调试的 JTAG TDI。
io_jtag_tdo	输出	用于调试的 JTAG TDO。
io_jtag_tck	输入	用于调试的 JTAG TCK。
io_jtag_tms	输入	用于调试的 JTAG TMS。

自定义指令接口

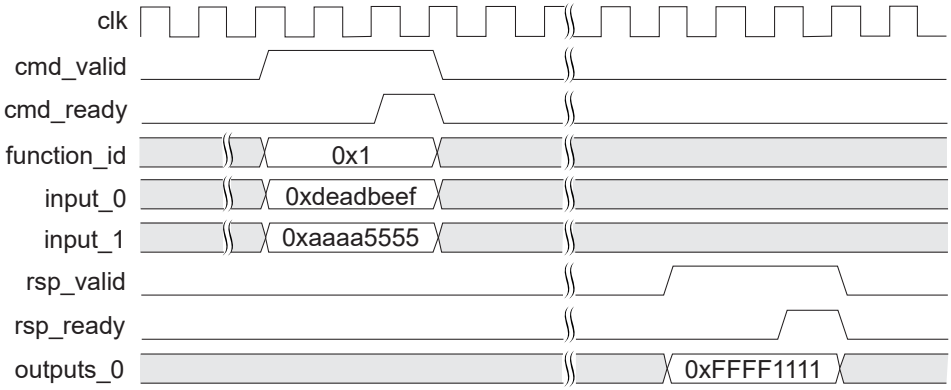
Sapphire 高性能 RISC-V SoC 支持自定义指令接口，您可以通过自定义硬件逻辑加速软件功能。 自定义指令支持R型指令，提供两个寄存器（`rs1` 和 `rs2`）处理自定义指令处理逻辑和最多 1,024 个 ID 来执行不同的功能。

表 23: 自定义指令端口

其中 n 是内核编号（0、1、2 或 3）。

端口	方向	说明
cpun_customInstruction_cmd_valid	输出	表示寄存器 rs1 和 rs2 存在并准备好进行处理。
cpun_customInstruction_cmd_ready	输入	表示自定义处理逻辑已准备好处理来自 CPU 的寄存器rs1和rs2。
cpun_customInstruction_function_id[9:0]	输出	自定义指令的功能 ID。
cpun_customInstruction_inputs_0[31:0]	输出	自定义指令使用的寄存器 rs1。
cpun_customInstruction_inputs_1[31:0]	输出	自定义指令使用的寄存器 rs2。
cpun_customInstruction_rsp_valid	输入	表示自定义指令结果可用。
cpun_customInstruction_rsp_ready	输出	表示 CPU 已准备好接受自定义指令结果。
cpun_customInstruction_outputs_0[31:0]	输入	自定义指令的结果。

图 6: 自定义指令波形



PLIC 外设接口

使用 SYSTEM_PLIC_CTRL 参数引用接口 PLIC 接口。

表 24: RISC-V PLIC 操作参数

定义	描述
中断优先级寄存器	每个中断源的中断优先级。
中断挂起位寄存器	每个中断源的中断挂起状态。
中断使能寄存器	使能每个上下文的中断源。
优先级阈值寄存器	每个上下文的中断优先级阈值。
中断声明寄存器	用于获取每个上下文中断源 ID 的寄存器。
中断完成寄存器	用于向关联网关发送中断完成消息的寄存器。

soc.h 文件包含许多 PLIC 参数，用于指定各种外设的中断 ID。

表 25: PLIC 中断 ID 参数

其中 n 是外设编号， m 是中断 ID。

参数	参考
SYSTEM_PLIC_SYSTEM_AXI_A_INTERRUPT	第7页的**中断**
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	第7页的**中断**
SYSTEM_PLIC_SYSTEM_USER_TIMER_n_INTERRUPTS_m	**定时器限制寄存器：0x0000_00044**

用户定时器

您可以根据系统时钟或外设时钟（如果已使能）设置预分频器寄存器，调整间隔周期以生成定时器滴答脉冲。

表 26: 用户定时器寄存器映射

地址偏移	寄存器名称	特权	宽度
0x0000_0000	预分频器	读/写	32
0x0000_0040	定时器配置	读/写	32
0x0000_0044	定时器限制	读/写	32
0x0000_0048	定时器值	读	32

预分频器寄存器：0x0000_0000

31	16	15	0
保留		预分频器值	

位	字段	描述	特权
0-15	预分频器值	时钟分频器比率。例如： 16'd0：除以 1 16'd1：除以 2 ... 16'd65534：除以 65535 16'd65535：除以 65536	读/写
16-31	保留	保留。	-

定时器配置寄存器：0x0000_0040

31	17	16	15	2	1	0
保留		自重启	保留		使用预分频器	不使用预分频器

位	字段	描述	特权
0	不使用预分频器	写入 1'b1 以运行定时器，不使用预分频器。	读/写
1	使用预分频器	写入 1'b1 以运行定时器，使用预分频器。	读/写
2-15	保留	保留。	N/A
16	自重启	写入 1'b1 以在达到定时器限制时使能自重启。	读/写
17-31	保留	保留。	N/A

定时器限制寄存器：0x0000_0044

31	0
限制值	

位	字段	描述	特权
0-31	限制值	定时器生成触发脉冲的值。使能预分频器时最终值为： (限制值 + 1) * (预分频器值 + 1)	读/写

定时器值寄存器：0x0000_0048

31	0
值	

位	字段	描述	特权
0-31	值	递增计数器的值。	读

Clint

内核本地中断 (clint) 由 `io_systemClk` 或 `io_peripheralClk` (如果使能) 驱动的 64 位实时计数器组成。clint 计数器值单调递增。Clint 还负责通过软件中断处理控制和状态。

表 27: Clint 寄存器映射

地址偏移	寄存器名称	特权	宽度
0x0000_0000	PIP	读/写	32
0x0000_4000	MTIMECMP (LO)	写	32
0x0000_4004	MTIMECMP (HI)	写	32
0x0000_BFF8	MTIME (LO)	读	32
0x0000_BFFC	MTIME (HI)	读	32

PIP 寄存器：0x0000_0000

31	2	0
保留		软件中断

位	字段	描述	特权
0	软件中断	机器模式软件中断。	读/写
2-31	保留	保留。	-

MTIMECMP 寄存器 (LO)：0x0000_4000

31	0
CMP 值	

位	字段	描述	特权
0-31	CMP 值	定时器中断触发值（低 32 位）。	写

MTIMECMP 寄存器 (HI): 0x0000_4004

31	0
CMP 值	

位	字段	描述	特权
0-31	CMP 值	定时器中断触发值 (高 32 位)。	写

MTIME 寄存器 (LO)：0x0000_BFF8

31	0
定时器值	

位	字段	描述	特权
0-31	定时器值	增量计数器的值（低 32 位）。	读

MTIME 寄存器 (HI)：0x0000_BFFC

31	0
定时器值	

位	字段	描述	特权
0-31	定时器值	增量计数器的值（高 32 位）。	读

控制和状态寄存器

机器级 CSR 实现见下表。

表 28: 机器信息寄存器

地址	寄存器名称	特权	描述	宽度
0xF14	mhartid	读	硬件线程 ID。	32

表 29: 机器陷阱寄存器

地址	寄存器名称	特权	描述	宽度
0x300	mstatus	读/写	机器状态寄存器。	13
0x304	mie	读/写	机器中断使能寄存器。	12
0x305	mtvec	读/写	机器陷阱处理程序基址。	32

表 30: 机器陷阱处理寄存器

地址	寄存器名称	特权	描述	宽度
0x340	mscratch	读/写	机器陷阱处理程序的临时寄存器。	32
0x341	mpec	读/写	机器异常程序计数器。	32
0x342	mcause	读	机器陷阱原因。	32
0x343	mtval	读	机器地址或指令错误。	32
0x344	mip	读/写	机器中断待处理。	12

机器级 CSR

机器模式状态寄存器 (mstatus)：0x300

`mstatus` 寄存器是一个 13 位读/写寄存器格式。 `mstatus` 寄存器跟踪并控制 HART 的当前操作状态。 `mstatus` 寄存器的受限视图分别显示为 S 级和 U 级 ISA 中的 `sstatus` 和 `ustatus` 寄存器。

31	30 20	19	18	17	16 15	14 13	12 11	10 8	7	6 4	3	2	1	0
SD	保留	MXR	SUM	MPRV	保留	FS	MPP	保留	MPIE	保留	MIE	保留	SIE	保留

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SIE	机器全局中断使能寄存器。	N/A	N/A	读/写
2	保留	保留。	N/A	N/A	N/A
3	MIE	机器中断使能寄存器。	读/写	读/写	读/写
4-6	保留	保留。	N/A	N/A	N/A
7	MPIE	机器先前中断使能。	读/写	读/写	读/写
8-10	保留	保留。	N/A	N/A	N/A
11-12	MPP	机器先前特权模式。	读/写	读/写	读/写
13-14	FS	浮点单元的状态。 2'b00：关闭 2'b01：初始 2'b10：干净 2'b11：脏	N/A	读	N/A
15-16	保留	保留。	N/A	N/A	N/A
17	MPRV	修改加载和存储可执行文件的特权级别。 1'b1：加载和存储内存地址经过转换和保护 1'b0：正常模式	N/A	N/A	读/写
18	SUM	修改 S 模式加载和存储访问虚拟内存的特权。 1'b1：允许访问 1'b0：S-模式 对 U-模式可访问的页面进行内存访问将触发故障	N/A	N/A	读/写
19	MXR	修改加载访问虚拟内存的特权。 1'b1：从标记为可读或可执行的页面加载将成功 1'b0：只有从标记为可读的页面加载才会成功	N/A	N/A	读/写
20-30	保留	保留。	N/A	N/A	N/A
31	SD	表示存在具有脏状态的 FS 字段，需要将扩展的用户上下文保存到内存中。	N/A	读	N/A

机器模式中断使能寄存器 (mie)：0x304

`mie` 寄存器是一个包含中断使能位的 12 位读/写寄存器。

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

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SSIE	监督模式软件中断。	N/A	N/A	读/写
2	保留	保留。	N/A	N/A	N/A
3	MSIE	机器软件中断使能。	读/写	读/写	读/写
4	保留	保留。	N/A	N/A	N/A
5	STIE	监督模式定时器中断使能。	N/A	N/A	N/A
6	保留	保留。	N/A	N/A	N/A
7	MTIE	机器定时器中断使能。	读/写	读/写	读/写
8	保留	保留。	N/A	N/A	N/A
9	SEIE	监督模式外部中断使能。	N/A	N/A	读/写
10	保留	保留。	N/A	N/A	N/A
11	MEIE	机器外部中断使能。	读/写	读/写	读/写

机器模式陷阱向量基址寄存器 (mtvec)：0x305

`mtvec` 寄存器是一个 32 位读/写寄存器，用于保存陷阱向量配置，由向量基址 (base) 和向量模式 (mode) 组成。

31	2	1	0
base		mode	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-1	mode	向量模式。 0：直接。所有异常将 pc 设置为 BASE 1：向量。异步中断将 pc 设置为 BASE + 4xcause ≥ 2：保留	读/写	读/写	读/写
2-31	base	向量基址。	读/写	读/写	读/写

机器模式暂存寄存器 (mscratch)：0x340

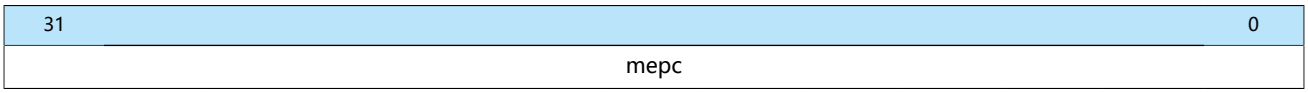
`mscratch` 寄存器是专用于机器模式的 32 位读/写寄存器。通常，它用于保存指向机器模式 hart 本地上下文空间的指针，并在进入 M 模式陷阱处理程序时与用户寄存器交换。

31	0
mscratch	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	mscratch	机器模式软件可以使用的临时暂存空间。	读/写	读/写	读/写

机器模式异常程序计数器 (mepc)：0x341

`mepc` 是一个 32 位读/写寄存器。 `mepc` (`mepc[0]`) 的低位始终为零。在仅支持 IALIGN=32 的实现中，两个低位 (`mepc[1:0]`) 始终为零。



位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	mepc	机器异常程序计数器。	读/写	读/写	读/写

机器模式原因寄存器 (mcause)：0x342

`mcause` 寄存器是一个 32 位读写寄存器。当陷阱被捕获并进入 M 模式时， `mcause` 会被写入指示陷阱原因事件的代码。在没有异常发生时， `mcause` 永远不会由实现写入，但可能由软件显式写入。

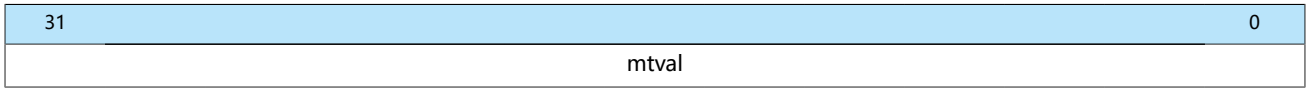
位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-30	异常代码	参见 第26页的**表 31: 陷阱后的机器原因寄存器 (mcause) 值**.	读	读	读
31	中断	mcause 中断位。	读	读	读

表 31: 陷阱后的机器原因寄存器 (mcause) 值

中断	异常代码	描述
1	0	保留。
1	1	监督软件中断。
1	2	保留。
1	3	机器软件中断。
1	4	用户定时器中断。
1	5	监督定时器中断。
1	6	保留。
1	7	机器定时器中断。
1	8	用户外部中断。
1	9	监督外部中断。
1	10	保留。
1	11	机器外部中断。
1	≥12	保留。
0	0	指令地址未对齐。
0	1	指令访问错误。
0	2	非法指令。
0	3	断点。
0	4	加载地址未对齐。
0	5	加载访问错误。
0	6	存储/AMO 地址未对齐。
0	7	存储/AMO 访问故障。
0	8	保留。
0	9	保留。
0	10	保留。
0	11	从 M 模式调用环境。
0	12	指令页面错误。
0	13	加载页面错误。
0	14	保留。
0	15	存储/AMO 页面错误。
0	≥16	保留。

机器模式陷阱值寄存器 (mtval)：0x343

`mtval` 寄存器是一个 32 位寄存器。当陷阱被捕获并进入 M 模式时， `mtval` 被设置为零或写入异常特定信息，以帮助软件处理陷阱。其他情况下， `mtval` 永远不会由实现写入，但可能由软件显式写入。硬件平台将指定哪些异常必须在 `mtval` 中写入具体的信息以及哪些可以无条件将其设置为零。



位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	mtval	机器陷阱值寄存器位。	读/写	读/写	读/写

机器模式中断待处理寄存器 (mip)：0x344

`mip` 寄存器是一个 12 位读/写寄存器，包含有关待处理中断的信息。

11	10	9	8	7	6	5	4	3	2	1	0
MEIP	保留	SEIP	保留	MTIP	保留	STIP	保留	MSIP	保留	SSIP	保留

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SSIP	监督软件中断待处理。	N/A	N/A	读/写
2	保留	保留。	N/A	N/A	N/A
3	MSIP	机器软件中断待处理。	读/写	读/写	读/写
4	保留	保留。	N/A	N/A	N/A
5	STIP	监督定时器中断待处理。	N/A	N/A	读/写
6	保留	保留。	N/A	N/A	N/A
7	MTIP	机器定时器中断待处理。	读	读	读
8	保留	保留。	N/A	N/A	N/A
9	SEIP	监督外部中断待处理。	N/A	N/A	读/写
10	保留	保留。	N/A	N/A	N/A
11	MEIP	机器外部中断待处理。	读	读	读

Hart ID 寄存器 (mhartid)：0xF14

`mhartid` CSR 是一个 32 位只读寄存器，包含运行代码的硬件线程整数 ID。此寄存器在任何实现中都必须可读。在多处理器系统中，Hart ID 可能不一定是连续的，但至少一个 Hart 的 hart ID 必须为零。Hart ID 必须是唯一的。

31	0
Hart ID	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	Hart ID	硬件线程 ID。	读	读	读

定时器相关 CSR

CSR	名称	描述	单核/多核	配备 FPU	配备 MMU
0xC00	周期	执行 **RDCYCLE** 指令的周期计数器。	N/A	N/A	读
0xC01	时间	执行 **RDTIME** 指令的定时器。	N/A	N/A	读
0xC02	timeh	执行 **RDINSTRET** 指令的指令执行计数器。	N/A	N/A	读

浮点相关 CSR

CSR	名称	描述	单核/多核	配备 FPU	配备 MMU
0x001	fflags	浮点累积异常。	N/A	读/写	N/A
0x002	frm	浮点动态舍入模式。	N/A	读/写	N/A
0x003	fcsr	浮点控制和状态寄存器 (frm + fflags)。	N/A	读/写	N/A

监督级 CSR

监督模式下运行的操作系统应仅查看应该对其可见的 CSR 状态。监督模式下运行的操作系统可访问的 CSR 中不存在有关更高特权级别（机器级或其他）存在（或不存在）的信息。许多监督 CSR 是等效机器模式 CSR 的子集。

监督模式状态寄存器 (sstatus)：0x100

sstatus 寄存器是 mstatus 寄存器的一个子集。

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

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SIE	监督模式全局中断使能寄存器。	N/A	N/A	读/写
2-4	保留	保留。	N/A	N/A	N/A
5	SPIE	监督模式先前中断使能寄存器。	N/A	N/A	读/写
6-7	保留	保留。	N/A	N/A	N/A
8	SPP	监督模式先前特权模式。	N/A	N/A	读/写
9-12	保留	保留。	N/A	N/A	N/A
13-14	FS	浮点单元的状态。 2'b00：关闭 2'b01：初始 2'b10：干净 2'b11：脏	N/A	读/写	N/A
15-16	保留	保留	N/A	N/A	N/A
17	MPRV	修改加载和存储可执行文件的特权级别。 1'b1：加载和存储内存地址被转换和保护 1'b0：正常模式	N/A	N/A	读/写
18	SUM	修改 S 模式加载和存储访问虚拟内存的特权。 1'b1：允许访问 1'b0：S 模式内存访问 U 模式可访问的页面将导致故障	N/A	N/A	读/写
19	MXR	修改加载访问虚拟内存的特权。 1'b1：从标记为可读或可执行的页面加载都会成功 1'b0：只有从标记为可读的页面加载才会成功	N/A	N/A	读/写
20-30	保留	保留。	N/A	N/A	N/A
31	SD	指示存在脏状态的 FS 字段，需要将扩展的用户上下文保存到内存中。	N/A	读/写	N/A

监督模式中断使能寄存器 (sie)：0x104

`sie` 寄存器是一个包含中断使能位的 12 位读/写寄存器。

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

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SSIE	监督模式软件中断。	N/A	N/A	读/写
2-4	保留	保留。	N/A	N/A	N/A
5	STIE	监督模式定时器中断使能。	N/A	N/A	读/写
6-8	保留	保留。	N/A	N/A	N/A
9	SEIE	监督模式外部中断使能。	N/A	N/A	读/写
10-31	保留	保留。	N/A	N/A	N/A

监督模式陷阱向量基址寄存器 (stvec)：0x305

`stvec` 寄存器是一个 32 位读/写寄存器，用于保存陷阱向量配置，由向量基址 (base) 和向量模式 (mode) 组成。

31	2	1	0
base		mode	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-1	mode	向量模式。 0：直接。所有异常将 pc 设置为 BASE 1：向量。异步中断将 pc 设置为 BASE + 4xcause ≥ 2：保留	N/A	N/A	读/写
2-31	base	向量基址。	N/A	N/A	读/写

监督模式暂存寄存器 (sscratch)：0x140

`sscratch` 寄存器是一个 32 位读/写寄存器，专供机器模式使用。通常， `sscratch` 用于在 hart 执行用户代码时保存指向 hart 本地监督模式上下文的指针。在陷阱处理程序开始时，`sscratch` 与用户寄存器交换以提供初始工作寄存器。

31	0
sscratch	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	sscratch	可供监督模式软件使用的临时暂存空间。	N/A	N/A	读/写

监督模式异常程序计数器 (sepc)：0x141

`sepc` 是一个 32 位读/写寄存器。 `sepc` (`sepc[0]`) 的低位始终为零。在仅支持 IALIGN=32 的实现中，两个低位 (`sepc[1:0]`) 始终为零。

31	0
sepc	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	sepc	监督模式异常程序计数器。	N/A	N/A	读/写

监督模式原因寄存器 (scause)：0x142

scause 寄存器是一个 32 位读写寄存器。当陷阱被捕获并进入 S 模式时， scause 会被写入指示陷阱原因事件的代码。在没有异常发生时， scause 永远不会由实现写入，但可能由软件显式写入。

31	30	0
中断	异常代码	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-30	异常代码	参见 第30页的**表 32: 陷阱后的机器原因寄存器 (scause) 值**。	N/A	N/A	读
31	中断	scause 中断位。	N/A	N/A	读

表 32: 陷阱后的机器原因寄存器 (scause) 值

中断	异常代码	描述
1	0	保留。
1	1	监督模式软件中断。
1	2-4	保留。
1	5	监督模式定时器中断。
1	6-8	保留。
1	9	监督模式外部中断。
1	10-15	保留。
0	0	指令地址未对齐。
0	1	指令访问错误。
0	2	非法指令。
0	3	断点。
0	4	加载地址未对齐。
0	5	加载访问故障。
0	6	存储/AMO 地址未对齐。
0	7	存储/AMO 访问故障。
0	8	从 U 模式调用环境。
0	9	从 S 模式调用环境。
0	10	保留。
0	11	从 M 模式调用环境。
0	12	指令页面错误。
0	13	加载页面错误。
0	14	保留。
0	15	存储/AMO 页面错误。
0	≥16	保留。

监督模式陷阱值寄存器 (stval)：0x143

`stval` 寄存器是一个 32 位寄存器。当陷阱被捕获并进入 S 模式时， `stval` 被设置为零或写入异常特定信息，以帮助软件处理陷阱。其他情况下， `stval` 永远不会由实现写入，但可能由软件显式写入。硬件平台将指定哪些异常必须在 `stval` 中写入具体的信息以及哪些异常可以无条件将其设置为零。

31	0
stval	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-31	stval	监督模式陷阱值寄存器位。	N/A	N/A	读/写

监督模式中断待处理寄存器 (sip)：0x144

`sip` 寄存器是一个 12 位读/写寄存器，其中包含有关待处理中断的信息。

11	10	9	8	7	6	5	4	3	2	1	0
保留		SEIP	保留			STIP	保留			SSIP	保留

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0	保留	保留。	N/A	N/A	N/A
1	SSIP	监督模式软件中断待处理。	N/A	N/A	读/写
2-4	保留	保留。	N/A	N/A	N/A
5	STIP	监督模式定时器中断待处理。	N/A	N/A	读/写
6-8	保留	保留。	N/A	N/A	N/A
9	SEIP	监督模式外部中断待处理。	N/A	N/A	读/写
10-11	保留	保留。	N/A	N/A	N/A

监督模式地址转换保护寄存器 (satp)：0x180

`satp` 寄存器是一个 32 位寄存器，用于控制监督模式地址转换和保护。此寄存器保存根页表的物理页号 (PPN)。例如，其监督模式物理地址除以 4KB；地址空间标识符 (ASID)，用于在每个地址空间基础上促进地址转换保护；以及选择当前地址转换方案的 `MODE` 字段。

31	30	22	21	0
MODE	ASID		PPN	

位	字段	描述	单核/多核	配备 FPU	配备 MMU
0-21	PPN	物理页码。	N/A	N/A	读/写
22-30	ASID	地址空间标识符。	N/A	N/A	读/写
31	MODE	1'b1：基于页面的 32 位虚拟处理。 1'b0：无转换或保护。	N/A	N/A	读/写

修订记录

表 33: 修订记录

日期	版本	说明
2024 年 10 月	1.0	首次发布。