RISC-V 平台级中断控制器规范

RISC-V Task Group

Version 1.0.0_rc5, 10/2022: 本文档已处于冻结状态，极不可能发生改动。见 http://riscv.org/spec-state 获得更多细节

这是一份完全免费的文档，发布在

译者：沉默之海 xiaowuzxc
如有谬误，请联系我，邮箱 xiaowuzxc@outlook.com

中英文对照/释义

中文翻译、含义	英文原文
平台级中断控制器	Platform-Level Interrupt Controller
操作参数	Operation parameter
机器模式	M(machine)-mode
监督模式	S(supervisor)-mode
用户模式	U(user)-mode
偏移量	Offset
特权架构	Privileged architecture
中断通知	Interrupt Notification
中断目标	Interrupt target
中断信号	Interrupt signalled
中断请求	Interrupt request
中断声明	Interrupt claim
声明响应	Claim response
中断完成	Interrupt completion
闸口	Gateway
有效的	Active
中断未决(等待/挂起)	Interrupt pending
外部中断未决	External interrupt pending
消息信号中断	Message-signalled interrupt

目录

约首

本文档已处于冻结状态。

极不可能发生改动。改动的阈值很高，只有在公共审查周期中发现一些非常关键的问题才会进行改动。任何展望或更改都将作为后续新扩展的主题。

修改日志

version 1.0.0_rc6

- 2023-1-5

从规格书中移除 H/U 模式

version 1.0.0_rc5

- 2022-10-29

明确了访问内存映射寄存器时的“寄存器宽度”。
合并寄存器宽度章节和内存映射章节。

version 1.0.0_rc4

- 2022-8-27

将规格书更新为冻结状态

version 1.0.0_rc3

- 2022-6-11

修订版权和许可信息
添加了 Andrew Waterman 和 Krste Asanovic 作为原始设计和编写规范的贡献者。

version 1.0.0_rc2

- 2022-5-30

根据 RISC-V Regulations 的附录 A“知识产权政策”第 3.3 节修改版权声明。
修正 PLIC 中断使能位的内存映射偏移量。

version 1.0.0_rc1

- 2022-4-16

公开前的审查版本

版权和许可信息

贡献者

RISC-V PLIC 规格书的贡献者按字母顺序排列：

Abner Chang <abner.chang@hpe.com>
Andrew Waterman <andrew@sifive.com>
Bin Meng <bmeng.cn@gmail.com>
Drew Barbier <drew@sifive.com>
Jeff Scheel <jeff@riscv.org>
Jessica Clarke <jrtc27@jrtc27.com>
Krste Asanovic <krste@sifive.com>
Palmer Dabbelt <palmer@dabbelt.com>
Robert Balas <balasr@iis.ee.ethz.ch>
Yan <phantom@zju.edu.cn>

第 1 章 简介

该规格书根据 RISC-V 平台级中断控制器(PLIC)规范(已从 RISC-V Privileged Spec v1.11-draft 移除)，叙述了通用 PLIC 架构在 RISC-V 系统的 context 中的操作参数。

PLIC 将各种设备发出的中断源复用到 hart contexts 的外部中断线上，在硬件上支持中断优先级。PLIC 支持最多 1023 个中断源(0 是保留的)和 15872 个 contexts，但实际的中断源和 context 数量取决于 PLIC 的实现方式。但是，实现 PLIC 必须遵守 PLIC 工作流程对每个寄存器的偏移量的定义(译者注：对寄存器的功能和偏移量定义需要符合规范)。遵循以下内容实现的 PLIC 可称为兼容 PLIC 标准。

1.1 中断目标和 hart contexts

当一个 hart context 在一个 hart 上被赋予一个特权模式，中断目标通常是 hart context(也可能是其他的中断目标，例如 DMA)。例如，在一个 2 路超线程的 4 核系统中，有 8 个硬件线程，可能每个 hart 至少有两种特权模式：机器模式和监督模式。

并不是所有的 hart contexts 都需要成为中断目标。例如，如果处理器核不支持将外部中断委托给低特权的模式，那么低特权的 hart contexts 就不是中断目标。PLIC 产生的中断通知分别会出现在 M/S 模式的 mip/sip(译者注：机器/监督模式的中断未决寄存器)寄存器的 meip/seip 位上。

老版本规范说明 PLIC 支持用户模式中断，删除此文本的原因是特权架构没有定义用户模式中断。未来如果特权架构定义了用户模式中断，那么将会直接扩展本 PLIC 规范。

如果外部中断已经委托给低特权的模式，中断通知就只会出现在低特权的 xip 寄存器(译者注：x 指代 s、h 等不同的中断未决(等待)寄存器)。

每个处理器核必须定义一个策略，声明内核上的 hart contexts 如何接收同时到达的有效中断。对于 hart contexts 进行叠加的简单情况(每个 hart context 对应一个特权模式)，发往高特权 contexts 的中断可以抢占低特权 contexts 的中断服务程序。多线程的处理器核可以在同一时间的不同 hart contexts 上运行多个独立的中断服务程序。处理器核还可以提供仅用于中断处理的 hart contexts，以减少中断服务延迟，这些 contexts 可能会抢占同一个处理器核上其他 hart 的中断服务程序。

PLIC 独立处理每个中断目标，不会对多个中断目标使用的任何的中断优先级方案(译者注：大概意思是 PLIC 不负责中断目标优先级问题)。因此，PLIC 不提供中断抢占或嵌套的概念，这些功能必须由托管多个中断目标 contexts 的内核来处理。

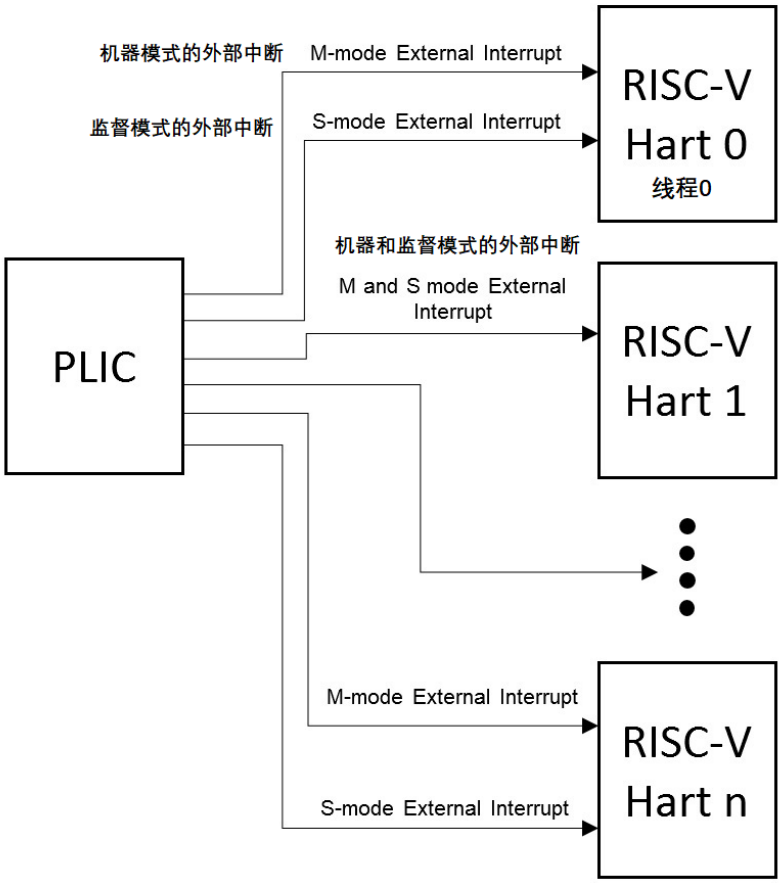

图 1 RISC-V PLIC 中断架构框图

1.2 中断闸口

中断闸口负责将全局“中断信号”转换为通用形式的“中断请求”，并让中断请求流向 PLIC 内核。无论如何，在 PLIC 内核中，每个中断源最多只能发起一个中断请求，中断请求显示在 IP 位。只有完成了中断处理，向同一中断源的闸口发送了“中断完成”通知，闸口才会将一个新的中断请求发给 PLIC 内核。(译者注：建议参考图 2)

如果全局的中断源是电平触发的，闸口会将电平的第一个断言(译者注：可以理解为 PLIC 第一次对电平起反应)转换为中断请求，但是此闸口不再发送新的中断请求，直到此闸口收到中断完成信号。闸口收到中断完成后，如果中断依然是电平触发且中断信号有效，那么闸口就向 PLIC 内核发送新的中断请求。一旦发到 PLIC 内核，闸口就无法收回中断请求。在 PLIC 内核接受请求之后、中断完成之前，如果一个电平触发的中断信号被取消了，则中断请求仍然存在于 PLIC 内核的 IP 位，并由处理机制提供服务，然后处理机制将去确认中断设备不再需要服务(译者注：原文难以理解)。

如果全局的中断源是边沿触发的，闸口将第一个有效信号沿转换为中断请求。根据设备和中断处理程序的设计，在发送中断请求之后、中断完成之前，闸口可能会忽略额外的有效信号沿或增加的中断未决计数器。无论是哪种情况，下一个中断请求都不会被转发到 PLIC 内核，直到闸口收到上一个中断已完成的消息。如果闸口有一个中断未决计数器，当 PLIC 内核接收中断请求后，计数器将递减。与专用的中断信号线不同，消息信号中断（MSIs）通过系统互连线发送一个消息包，该消息包描述了正处于断言状态的中断。对消息进行译码，通知相应的闸口处理 MSI，类似于边缘触发的中断。(译者注：一种是单独的中断信号线，一种是数据包，MSI 就像是有效信号沿)

1.3 中断通知

每个中断目标在 PLIC 内核中都有一个外部中断未决（EIP）位，指示相应的中断目标有一个未决的中断等待它去服务。EIP 可能会随着 PLIC 内核状态的变化而变化，这些变化是由中断源、中断目标或其他代理操作 PLIC 中的寄存器值引起的。EIP 的值作为中断通知传递给中断目标。如果中断目标是 RISC-V hart context，则根据 hart context 的特权级别，中断通知到达 meip/seip 位。(在简单的系统中，中断通知是连接到 hart 的信号线。在更复杂的平台中，中断通知可以作为消息通过系统互连进行路由。)

PLIC 硬件仅支持中断广播，因此所有使能的中断目标都会收到有效中断给出的中断通知。(广播提供了快速响应，因为最快的响应器会声明中断，但在高中断率的场景中，如果多个 hart 进入了只有一个 hart 可以成功声明的中断，则会造成浪费。软件可以将 PLIC IE 位调制为每个中断处理程序的一部分，以提供替代策略，例如中断亲和(译者注：不了解此概念)或轮询单播。)

根据平台架构和传输中断通知的方式，中断目标可能需要些时间才能收到中断通知。PLIC 保证最终会将 EIP 中的所有状态变化交付给所有中断目标，前提是 PLIC 内核中没有干预活动。(中断通知的值只保证保存在过去某个时间点有效的 EIP 值(译者注：理不顺，看例子吧)。例如，中断目标 2 可以在中断目标 1 的中断通知有效的时候，响应并声明中断。这样一来，当中断目标 1 试图声明中断的时候，会发现 PLIC 内核中不存在有效中断。)

1.4 中断标识符（IDs）

全局的中断源都会分配一个无符号整数的中断标识符，从 1 开始向上分配。

中断 ID 为 0 表示“无中断”。

当两个或多个中断源具有相同的优先级时，中断标识符也用于仲裁。较小的中断 ID 优先于较大的中断 ID，即 ID 越小仲裁优先级越高。

1.5 中断流程

图 2 显示了通过 PLIC 处理中断时层级之间的信息流。

- 全局的中断从它们的中断源发送到对应的中断闸口。
- 然后，中断闸口向 PLIC 内核发送一个中断请求，后者将这些请求锁存在内核的中断未决位(IP)。
- 如果中断目标使能(IE)了任意的未决中断，并且未决中断的最高优先级超过每个中断目标的优先级阈值，则 PLIC 内核将中断通知转发给一个或多个中断目标。
- 中断目标接收了外部中断，它会发送一个中断声明，从 PLIC 内核中检索符合该目标且具有最高优先级的中断源标识符(ID)。
- 然后，PLIC 内核清除关联的中断源未决位(Source’s IP)。
- 中断目标完成中断服务后，它向关联的中断闸口发送中断完成信号。
- 现在，中断闸口可以将同源的另一个中断请求发给 PLIC 内核。

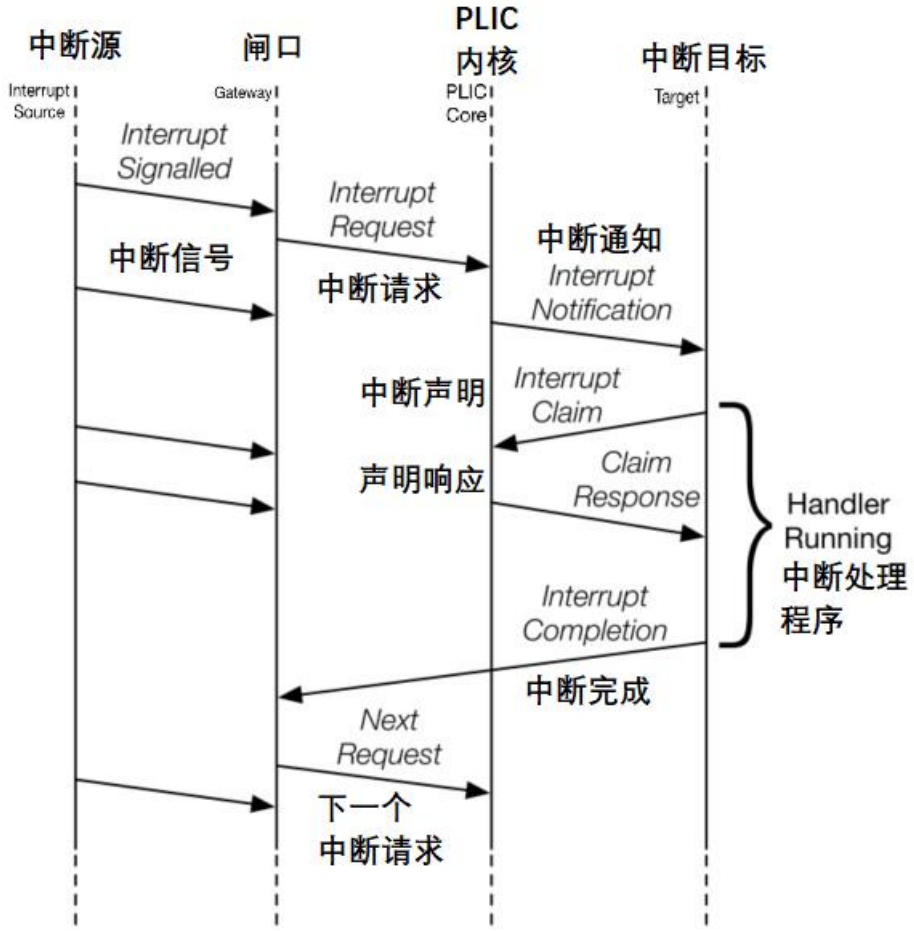
中断源
Interrupt Source
闸口
Gateway
PLIC 内核
PLIC Core
中断目标
Target
Interrupt Signalled
中断信号
Interrupt Request
中断请求
中断通知
Interrupt Notification
中断声明
Interrupt Claim
声明响应
Claim Response
Handler Running
中断处理程序
Interrupt Completion
中断完成
Next Request
下一个中断请求

图 2 PLIC 中断流程

第 2 章 RISC-V PLIC 操作参数

本规范定义了一般的 PLIC 操作参数寄存器组，它们是：

- 中断优先级寄存器：
  每个中断源的中断优先级。
- 中断未决位寄存器(IP)：
  每个中断源的中断未决状态。
- 中断使能寄存器(IE)：
  每个 context 的中断使能。
- 优先级阈值寄存器：
  每个 context 的中断优先级阈值。
- 中断请求寄存器：
  用于获取每个 context 的中断源 ID 的寄存器。
- 中断完成寄存器：
  发送中断完成信息到相关闸口的寄存器。

下图为 PLIC 操作参数框图

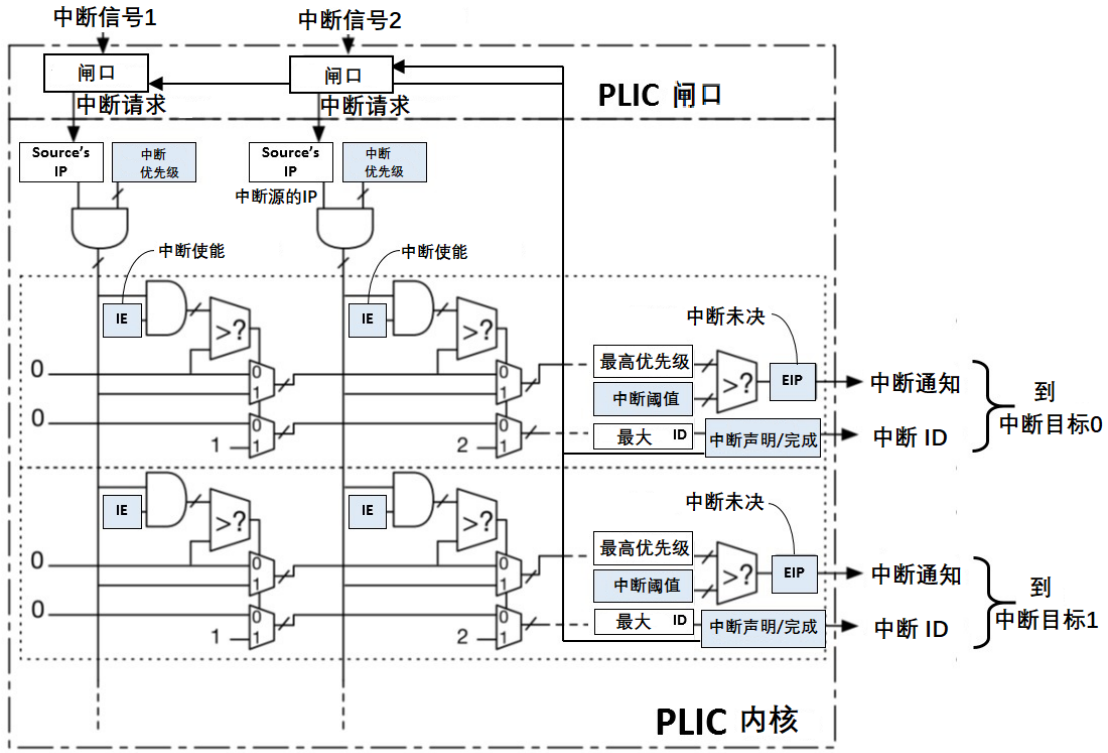

图 3 PLIC 操作参数框图

(译者注：蛋蛋色方框是可以内存映射访问的寄存器)

第 3 章 内存映射

PLIC 内存映射的基地址是由平台实现决定的。本章中指定的内存映射寄存器的宽度为 32 位。使用 lw 和 sw 指令原子地访问这些位。

PLIC 内存映射：

base + 0x000000: 保留 Reserved (中断源 0 不存在 interrupt source 0 does not exist)
base + 0x000004: 中断源 1 优先级 Interrupt source 1 priority
base + 0x000008: 中断源 2 优先级 Interrupt source 2 priority
...
base + 0x000FFC: 中断源 1023 优先级 Interrupt source 1023 priority
base + 0x001000: 中断未决位(IP)0-31Interrupt Pending bit 0-31
...
base + 0x00107C: 中断未决位 992-1023Interrupt Pending bit 992-1023
...
base + 0x002000: context0 的中断源 0-31 使能位(IE)Enable bits for sources 0-31 on context 0
base + 0x002004: context0 的中断源 32-63 使能位 Enable bits for sources 32-63 on context 0
...
base + 0x00207C: Enable bits for sources 992-1023 on context 0
base + 0x002080: Enable bits for sources 0-31 on context 1
base + 0x002084: Enable bits for sources 32-63 on context 1
...
base + 0x0020FC: Enable bits for sources 992-1023 on context 1
base + 0x002100: Enable bits for sources 0-31 on context 2
base + 0x002104: Enable bits for sources 32-63 on context 2
...
base + 0x00217C: Enable bits for sources 992-1023 on context 2
...
base + 0x1F1F80: Enable bits for sources 0-31 on context 15871
base + 0x1F1F84: Enable bits for sources 32-63 on context 15871
base + 0x1F1FFC: Enable bits for sources 992-1023 on context 15871
...
base + 0x1FFFFC: 保留 Reserved
base + 0x200000: 中断源 0 的优先级阈值 Priority threshold for context 0
base + 0x200004: 中断源 0 的声明/完成 Claim/complete for context 0
base + 0x200008: Reserved
...
base + 0x200FFC: Reserved
base + 0x201000: 中断源 1 的优先级阈值 Priority threshold for context 1
base + 0x201004: 中断源 1 的声明/完成 Claim/complete for context 1
...
base + 0x3FFF000: 中断源 15871 的优先级阈值 Priority threshold for context 15871
base + 0x3FFF004: 中断源 15871 的声明/完成 Claim/complete for context 15871
base + 0x3FFF008: Reserved
...

base + 0x3FFFFFC: Reserved

下面几章描述 PLIC 操作参数的控制寄存器组。

第 4 章 中断优先级

中断优先级是数值较小的无符号整数，最大级别由具体平台决定。优先级 0 是保留的，表示“永不中断”，数值越大、中断优先级越高。每个全局的中断源在内存映射寄存器中都有一个关联的中断优先级。不同的中断源不需要支持相同的优先级值的集合。有效的实现可以硬连接到所有输入优先级级别(译者注：不太能理解)。中断源优先级寄存器应该是 WARL 字段，以允许软件确定每个优先级规范中读写位的数量和位置(如果有的话)。为了简化优先级判别，每个优先级寄存器必须支持任意数值组合，即如果寄存器中有 2 个变量位，那么共 4 种数值组合必须都有效。(这段原文晦涩难懂)

如果 PLIC 支持中断优先级，则可以通过写入 32 位内存映射的优先级寄存器来为每个 PLIC 中断源分配优先级。保留优先级 0 表示“从不中断”，并有效地禁用中断。优先级 1 是最低的有效优先级，而最高优先级取决于 PLIC 的实现方案。具有相同优先级的中断冲突被中断 ID 解决；ID 最低的中断具有最高的有效优先级。

PLIC 内存映射区域内，中断源优先级的基地址固定为 0x000000。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
中断源优先级	中断源从优先级 0 到 1023	1024*4=4096(0x1000) 字节	这是一个包含 PLIC 中断源优先级的连续内存块。 此内存块共有 1024 个中断源优先级。中断源优先级 0 保留，表示它不存在。

PLIC 中断源优先级的内存映射：

0x000000: Reserved (interrupt source 0 does not exist)

0x000004: Interrupt source 1 priority

0x000008: Interrupt source 2 priority

...

0x000FFC: Interrupt source 1023 priority

第 5 章 中断未决位

在 PLIC 内核中，中断源未决位的当前状态可以从未决阵列中读取，未决阵列被分配为 32 位寄存器。中断 ID N 的未决位存储在第(N/32)字的(N mod 32)位。不存在的中断源 0 的 0 字 0 位的硬连线为零。

通过设置关联的使能位(IE)然后发出中断声明，可以清除 PLIC 内核中的中断未决位。

PLIC 内存映射区域内，中断未决位的基地址固定为 0x001000。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
中断未决位	中断源 0 至 N 的中断未决位	1024/8=128(0x80)字节	这是一个包含 PLIC 中断未决位的连续内存块。每个中断未决位占用此寄存器组的 1bit。

PLIC 中断挂起位内存映射：

0x001000: Interrupt Source #0 to #31 Pending Bits

...

0x00107C: Interrupt Source #992 to #1023 Pending Bits

第 3 章 中断使能

通过设置使能寄存器中的关联位，可以启用每个全局中断。使能寄存器被组织为 32 位寄存器的连续阵列，其封装方式与未决位相同。使能寄存器 0 的位 0 表示不存在的中断 ID 0，并硬连线为 0。PLIC 具有 15872 个 context 的中断使能位。

PLIC 如何为 context（Hart 和特权模式）分配中断不在 RISC-V PLIC 规范的范围内，但必须在供应商的 PLIC 规范中加以说明。

(若某些中断源只能路由到中断目标子集，大量潜在的 IE 位可能会硬连线为零。对于具有固定中断路由的嵌入式设备，可能会有更多位硬连线为 1。中断优先级、阈值和 hart 内部中断屏蔽提供了相当大的灵活性，即使始终启用全局中断源，也可以忽略外部中断。)

PLIC 内存映射区域内，中断使能位的基地址固定在 0x002000。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
中断使能位	源 0 到 1023 指向 15872 个 context 的中断使能位	(1024/8)*15872=2031616 (0x1f0000)字节	这是一个连续内存块，包含 15872 个 context 的 PLIC 中断使能位。每个中断使能位占用该寄存器块中的 1 位，总共占用 15872 个中断使能位

PLIC 中断使能位内存映射：

0x002000: Interrupt Source #0 to #31 Enable Bits on context 0

...

0x00207C: Interrupt Source #992 to #1023 Enable Bits on context 0

0x002080: Interrupt Source #0 to #31 Enable Bits on context 1

...

0x0020FC: Interrupt Source #992 to #1023 Enable Bits on context 1

0x002100: Interrupt Source #0 to #31 Enable Bits on context 2

...

0x00217C: Interrupt Source #992 to #1023 Enable Bits on context 2

0x002180: Interrupt Source #0 to #31 Enable Bits on context 3

...

0x0021FC: Interrupt Source #992 to #1023 Enable Bits on context 3

...

...

...

0x1F1F80: Interrupt Source #0 to #31 on context 15871

...

0x1F1FFC: Interrupt Source #992 to #1023 on context 15871

第 7 章 优先级阈值

PLIC 为每个 context 的中断优先级阈值提供了基于 context 的阈值寄存器。阈值寄存器是一个 WARL 字段。PLIC 会屏蔽所有优先级小于或等于阈值的 PLIC 中断。例如，阈值为 0，就允许所有优先级为非零的中断通过。

优先级阈值寄存器的基地址位于 4K 对齐位置，从偏移量 0x200000 开始。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
优先级阈值	15872 个 context 的优先级阈值	4096*15872=65011712 (0x3e00000)字节	这是设置每个 context 的优先级阈值寄存器

中断优先级阈值内存映射：

0x200000: Priority threshold for context 0

0x201000: Priority threshold for context 1

0x202000: Priority threshold for context 2

0x203000: Priority threshold for context 3

...

...

...

0x3FFF000: Priority threshold for context 15871

第 8 章 中断声明处理

中断目标接收到中断通知后的某个时间，它可能决定为中断提供服务。

一般情况下，通过非幂等的内存映射 I/O 控制寄存器读操作，中断目标向 PLIC 内核发送中断声明。PLIC 内核接收到中断声明后，将自动确认中断目标对应的最高优先级的未决中断 ID，然后清除相应的中断源 IP 位。最后，PLIC 内核将 ID 返回给中断目标。如果中断目标请求服务时没有未决中断，PLIC 内核返回的 ID 为 0。

最高优先级的未决中断被中断目标给声明了并且相应的 IP 位被清除后，其他较低优先级的未决中断可能对中断目标产生影响。因此，中断声明后 PLIC 的 EIP 位可能不会被清除。中断处理程序可以在退出之前检查本地的 meip/seip/ueip 位，从而更高效地处理其他中断，无需先恢复中断的上下文再进入另一个中断陷阱。(译者注：类似中断咬尾机制)

PLIC 可以通过读取声明/完成寄存器来实现中断声明，该寄存器返回未决的最高优先级中断 ID，如果没有未决中断，则返回 0。成功声明后，自动清除中断源上相应的未决位(IP)。PLIC 可以随时进行声明，声明操作不受优先级阈值寄存器配置的影响。

中断声明进程寄存器是基于 context，位于(4K 对齐+4)，从偏移量 0x200000 开始。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
中断声明寄存器	15872 个 context 的中断声明处理	4096*15872=65011712 (0x3e00000)字节	这是用于每个 context 获取中断 ID 的寄存器

PLIC 中断声明处理的内存映射：

0x200004: Interrupt Claim Process for context 0

0x201004: Interrupt Claim Process for context 1

0x202004: Interrupt Claim Process for context 2

0x203004: Interrupt Claim Process for context 3

...

...

...

0x3FFF004: Interrupt Claim Process for context 15871

第 9 章 中断完成

PLIC 通过写入声明/完成寄存器，发出中断处理程序已完成的信号，写入值来自于中断声明期间读到的中断 ID(译者注：声明时读的 ID 写回去，表示完成)。PLIC 不检查完成 ID 是不是和该目标最后一个声明的 ID 相同。如果完成 ID 不匹配当前为中断目标使能的中断源，则中断完成信号被静默地忽略。

处理程序完成中断服务后，必须向关联的闸口发送中断完成信息，通常作为对非幂等内存映射 I/O 控制寄存器的写入。闸口只有收到了中断完成，才会将额外的中断发给 PLIC 内核。

中断完成寄存器是基于 context 的，与中断请求处理寄存器处于相同的地址，该地址位于(4K 对齐+4)，从偏移量 0x200000 开始。

PLIC 寄存器组名称	功能	寄存器组大小(字节)	描述
中断完成寄存器	15872 个 context 的中断完成	4096*15872=65011712 (0x3e00000)字节	这是一个通过写操作来完成中断处理的寄存器

PLIC 中断完成的内存映射：

0x200004: Interrupt Completion for context 0

0x201004: Interrupt Completion for context 1

0x202004: Interrupt Completion for context 2

0x203004: Interrupt Completion for context 3

...

...

...

0x3FFF004: Interrupt Completion for context 15871