AIA

第一章：介绍

这篇文档描述了RISC-V上的Advanced Interrupt Architecture，包含了如下的部分：

1. 对于RISC-V指令集手册第二章中的RISC-V标准特权架构进行了一个扩展；

2. RISC-V系统中的两个标准中断控制器，即Advanced Platform-Level Interrupt Controller(APLIC)和Incoming Message-Signaled Interrupt(IMSIC);

3. 在考虑中断的情况下，其他组件有什么要求。

目标

RISC-V Advanced Interrupt Architecture有如下的目标：

1. 在RISC-V特权架构中构建中断处理的功能，但是尽可能不要修改已有的功能。
2. 使得RISC-V系统可以直接和Message-Signaled Interrupts(MSIs)一起工作（满足PCIe等设备的需求）。
3. 对于种类繁多的中断类型，定义了一种Platform-Level Interrupt Control（the Advance PLIC），对各个特权级提供了接口，并且能将这些中断转化为MSIs。
4. 扩展RISC-V核的Local Interrupts。
5. 允许让软件配置一个核上的中断相对特权（包括标准时钟中断和软件中断等等），而不是只能让中断控制器控制外部中断的特权级。
6. 当核实现了hypervisor特权架构，需要对于中断虚拟化提供充分的支持。
7. IOMMU可以重定向MSIs，最大化一个虚拟机中的客户操作系统对于设备的直接控制，减少hypervisor的参与。
8. 不要让硬件的中断支持成为增多虚拟机数量的阻碍。
9. 实现如上的全部要求，并且在速度、效率和灵活性之间做好平衡。

Advanced Interrupt Architecture的最初版本，主要思考的是大型、高性能RISC-V系统。现在并没有考虑那些为了在实时系统上为了减少中断响应时间，但是高性能RISC-V系统无关的特性：

1. 给每一个中断源，在每一个核上，设置一个单独的陷入地址；
2. 中断陷入时自动存储寄存器，退出时自动恢复；
3. 自动嵌套处理中断（根据中断优先级）。

这些特性用于实时系统上的优化，可以由之后的扩展来完成，或者作为未来版本的Interrupt Architecure文档的一部分。

局限性

在当前版本的RISC-V Advanced Interrupt Architecture最多只能在SMP系统支持16384个核。如果这些核都是64位核并实现了虚拟化拓展，且所有的Advanced Interrupt Architecture特性都被充分实现了，那么对于每一个物理核最多只能支持63个活跃的虚拟核，当然可以存在成千个不活跃的虚拟核被换出，这些虚拟都可以在直接控制一个或者多个物理设备。

表格1.1中说明了在Advanced Interrupt Architecture中，物理核数量、虚拟核数量、中断实体数量的支持。

	Maximum	Requirements
Physical harts	16,384	
Active virtual harts having direct control of a device, per physical hart	31 for RV32, 63 for RV64	RISC-V hypervisor extension; IMSICs with guest interrupt files; and an IOMMU
Idle (swapped-out) virtual harts having direct control of a device, per physical hart	potentially thousands	An IOMMU with support for memory-resident interrupt files
Wired interrupts at a single APLIC	1023	
Distinct identities usable for MSIs at each hart (physical or virtual)	2047	IMSICs

Table 1.1: Absolute limits on the numbers of harts and interrupt identities in a system. Individual implementations are likely to have smaller limits.

主要部件概览

RISC-V系统的signaling interrupts的整体架构取决于它是为了MSIs还是传统的形形色色的中断实现的。如果是一个完整支持MSIs的系统，那么每一个核都一个Incoming MSI Controller（IMSIC）作为一个私有的中断控制器，用于处理外部中断。相反，如果在一个使用传统的形形色色中断的系统中，核上没有IMSICs。在大型系统中，特别是那些带有PCI设备的系统，最好能通过核的IMSICs来完全支持MSIs。

外部中断(without IMSICs)

当一个RISC-V核没有IMSIC的时候，外部中断信号直接通过直联线来接入到核。这种情况下，Advanced Platform-Level Interrupt Controller（APLIC）作为一个传统的中断处理中心，可以为每一个核进行路由和对特权级进行排序。如图1.1中所示，中断可以被有选择地路由到Machine Level或者Supervisor Level。APLIC将在第4章中介绍。

没有IMSICs， 现在的AIA就不能直接将外部中断传到虚拟机中，即使RISC-V核上已经实现了hypervisor特权架构。取而代之的是，一个中断必须送到对应的hypervisor，之哦再注入一个虚拟中断到虚拟机中。

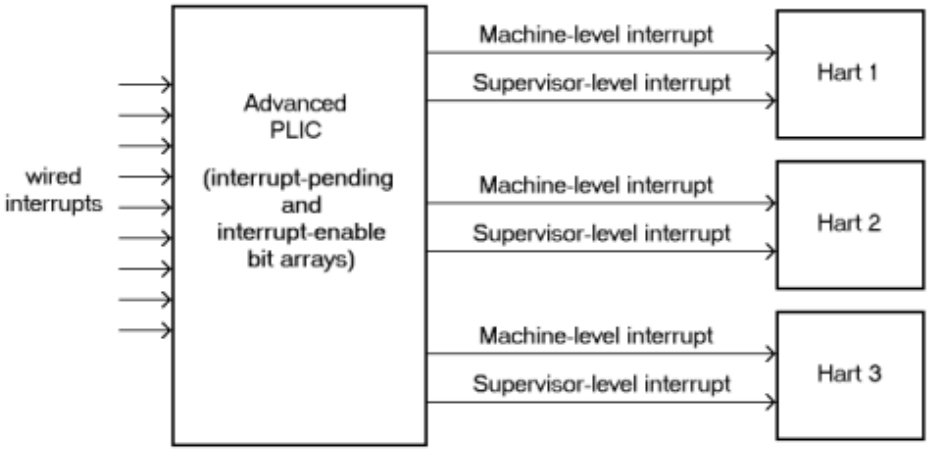

Figure 1.1: Traditional delivery of wired interrupts to harts without support for MSIs.

外部中断(with IMSICs)

为了接受MSIs，每一个RISC-V核上都有一个IMSIC，如下图1.2展示。从根本上讲，一个MSI就是写入一个值到一个特定的地址，然后硬件会把它认为是一个中断。出于这个目的，每一个IMSIC都会被放到一个或者多个确切的地址中。当写入的内容符合预期的格式，那么IMSIC就会将中断信号送到对应的核。

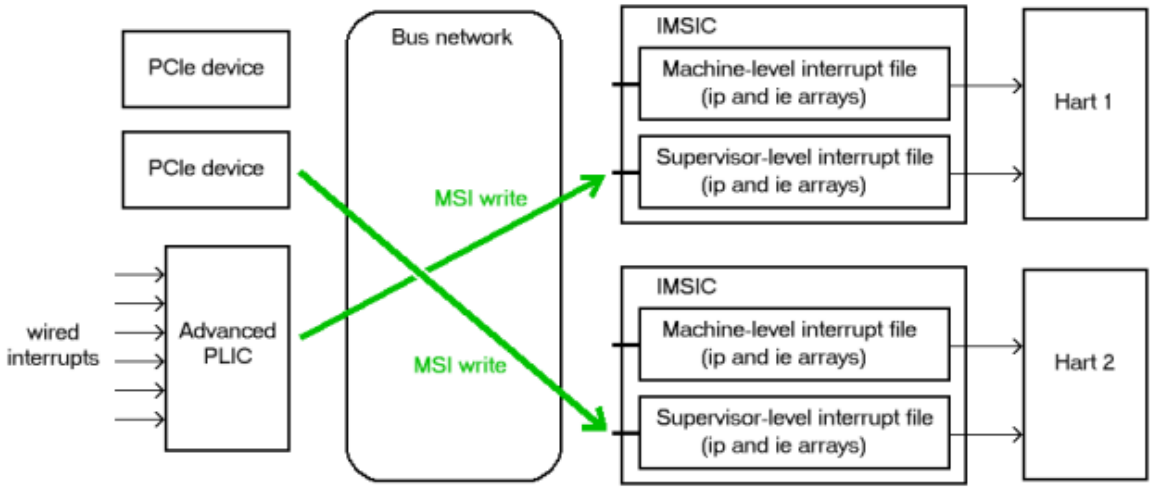

Figure 1.2: Interrupt delivery by MSIs when harts have IMSICs for receiving them.

因为所有的IMSICs都有一个独一无二的地址，所以每一个IMSIC会接受来自设备或者其他核的写。IMSICs对于machine level和supervisor level有独立的地址用于MSIs的传输，这样就可以对各个特权级下的中断进行配置，也更好地支持虚拟化。某一个特权级下的MSIs可以通过一个中断文件进行配置，这个文件中包含了一个Pending数组和一个Enable数组，后者可以表示当前核是否会接受某个中断。

第三章中会详细介绍IMSIC单元，RISC-V AIA中对于MSIs格式的介绍将在第3.2节中介绍。

当一个RISC-V系统的核上有IMSICs时，系统一般还会有一个APLIC。当时它的功能会和在上一节讲的功能不同，这里APLIC会将中断信号转化为一个MSI的写发送给核上的IMSIC单元。根据软件在APLIC上的配置，每一个MSI都会传送给一个特定的核。

如果RISC-V核实现了hypervisor特权扩展，IMSICs可能会添加guest中断文件，用于向虚拟机传输中断信号。除了在第三章中介绍了IMSIC，第六章会介绍虚拟机中断的相关内容。如果一个系统包含了一个IOMMU用于设备访问时对地址进行翻译，那么从这些设备来的MSIs需要特殊的处理，这部分内容将在第八章中介绍。

其它中断

除了IO设备来的外部中断，RISC-V特权架构说明一些其他中断类型，特权架构下的时钟中断和软件中断。对于软件中断，将在第七章中介绍，标题为核间中断。

AIA添加了相当多的在一个核上的对于Local Interrupt的支持，因此一个核可以中断自己来响应一些异步的事件。就像RISC-V标准时钟中断和软件中断一样，他们不用通过APLIC和IMSIC。

一个核上的中断实体

RISC-V特权架构下，给每一个中断原因分配了一个特定的主实体号码，这个号码会自动写入到mcause寄存器或者scause寄存器。0-15在特权级架构下是被标准化的，之后的中断号都是平台定制的。AIA要求16-23号和32-47号也被标准化，34-31和其他号码可以让平台定制。

Major identity	Minor identity	
0	–	*Reserved by Privileged Architecture*
1	–	Supervisor software interrupt
2	–	Virtual supervisor software interrupt
3	–	Machine software interrupt
4	–	*Reserved by Privileged Architecture*
5	–	Supervisor timer interrupt
6	–	Virtual supervisor timer interrupt
7	–	Machine timer interrupt
8	–	*Reserved by Privileged Architecture*
9	Determined by	Supervisor external interrupt
10	external interrupt	Virtual supervisor external interrupt
11	controller	Machine external interrupt
12	–	Supervisor guest external interrupt
13	–	Counter overflow interrupt
14–15	–	*Reserved by Privileged Architecture*
16–23	–	*Reserved for standard local interrupts*
24–31	–	*Designated for custom use*
32–34	–	*Reserved for standard local interrupts*
35	–	Low-priority RAS event interrupt
36–42	–	*Reserved for standard local interrupts*
43	–	High-priority RAS event interrupt
44–47	–	*Reserved for standard local interrupts*
≥ 48	–	*Designated for custom use*

Table 1.2: Major and minor identities for all interrupt causes at a hart. Major identities 0–15 are the purview of the RISC-V Privileged Architecture.

大多数IO设备的中断都是通过外部中断控制器传到核上的，可以是IMSIC或者APLIC。如上图1.2中所描述的一样，所有的在一个特权级下的外部中断共享一个主实体号，11是machine level，9是supervisor level，10是hypervisor level。之后通过外部中断控制器提供的从实体号可以进一步区分。

除了外部中断的其他其他中断原因，可能有它们自己的从主体号。但是这份文档只讨论外部中断的从主体号。

AIA定义的Local Interrupts会在第五章中进行介绍，标题为"Interrupts for Machine and Supervisor Level"。

选择一个核来接收中断

每一个signaled Interrupt都会被传送一个核上的一个特定特权级。RISC-V AIA上没有提供标准的硬件机制来核间广播或者组播中断。

对于Local Interrupts，一个软件向低特权级注入虚拟中断，这个中断就完全是一个本地事件，对于其它的核是不可见的。RISC-V特权架构下，时间中断也是独一无二地绑定到一个核上的。对于其它中断，这个中断被另外一个核接收到，那么这个中断信号必须被软件配置送到一个指定的核上。

为了发送一个核间中断到多个核，发送核需要执行一个循环，给每一个核都发送一次核间中断。对于传送一个特定核的核间中断，可以看第七章。

很少有情况，一个外部IO中断需要发送到多个核上，这个中断可以发送到一个特定的核上，再通过核间中断来通知其它核。

Smaia 和 Ssaia

AIA定了两个RISC-V指令集扩展，一个用于Machine level的执行环境，一个用于Supervisor level的执行环境。对于Machine level的执行环境，Smaia扩展包含了它所有添加的CSR寄存和对于中断响应行为的修改。对于Supervisor level的执行环境，Ssaia扩展和Smaia保持一致，除了它排除了Machine level的CSR寄存器和对于Supervisor level不直接可见的一些行为。

Smaia和Ssaia扩展包含了影响一个核上ISA行为的AIA特性。即使接下来讨论的内容是AIA的内容，但是但是它们可能不是Smaia或者Ssaia的内容，因为它们不属于ISA如APLICs、IOMMUs和任何和写IMSICs无关的核间中断内容。

在接下来的章节中，AIA添加的CSR寄存器和修改的行为，受XLEN、Supervisor或者hypervisor是否实现，核上是否实现了IMSIC影响。但是这些组合不会单独提供一个名字。

像编译器和汇编器这类开发工具不用关心是否有IMSIC存在，只需要在存在Smaia或者Ssaia存在时允许访问IMSIC CSR寄存器。没有IMSIC，会触发一个异常，但是对于开发工具来说这不是一个问题。

第二章：核上添加的CSR寄存器

在RISC-V核上的任何一个特权级都可以中断陷入，AIA 添加了CSR来进行控制和处理。

Machine-level CSRs

图2.1 中列出了添加到machine level的CSRs。每一个寄存器都是MXLEN比特的。

对于RV32, CSRs寄存器的高32位罗列在最后5个扩展寄存器，来支持一个完整64位。寄存器的低半部分（如mideleg）包含了中断号0-31，每一个bit都代表了一个中断号。同样地，高半部分覆盖了中断号32-63， 每一个bit都代表了一个中断中号。对于RV64, 一个64比特的CSR寄存器（如mideleg）包含了相同的中断号0-63。AIA需要在RV32中存在高32位的寄存器，不过它们可能仅仅是只读的全零。

CSRs miselect 和 mireg提供了一个窗口来访问在下表之外的一些寄存器。miselect决定了通过mireg可以访问哪些寄存器。miselect是一个WARL寄存器，它需要根据实现的特性支持一个范围内的值。如果IMSIC没有被实现，miselect的值只能从0到0x3f。如果IMSIC实现了，那么miselect可以实现8比特的值，可以从0到0xff。miselect从0到0xff的值目前的分配如下所示：

0x00 - 0x2f　保留

0x30 - 0x3f　主中断

0x40 - 0x6f　保留

0x70 - 0xff 外部中断（with IMSIC）

Number	Privilege	Name	Description
Machine-Level Window to Indirectly Accessed Registers			
0x350	MRW	miselect	Machine indirect register select
0x351	MRW	mireg	Machine indirect register alias
Machine-Level Interrupts			
0x35C	MRW	mtopei	Machine top external interrupt (only with an IMSIC)
0xFB0	MRO	mtopi	Machine top interrupt
Virtual Interrupts for Supervisor Level			
0x308	MRW	mvien	Machine virtual interrupt enables
0x309	MRW	mvip	Machine virtual interrupt-pending bits
Machine-Level High-Half CSRs for Interrupts 32–63 (RV32 only)			
0x313	MRW	midelegh	Extension of mideleg (only with S-mode)
0x314	MRW	mieh	Extension of mie
0x318	MRW	mvienh	Extension of mvien (only with S-mode)
0x319	MRW	mviph	Extension of mvip (only with S-mode)
0x354	MRW	miph	Extension of mip

Table 2.1: Machine-level CSRs added by the Advanced Interrupt Architecture.

miselect还可以支持0x00-0xff之外的值，但是在0xff没有再设置标准寄存器。

miselect最高位用于一些定制，可以通过mireg来访问一些定制的寄存器。如果MXLEN被修改了，那么miselect的最高位移动到了一个新的位置，并保留之前的值。但是实现的时候并不需要支持任何定制的值。

当miselect是一个reserved范围内的值时，试图访问mireg将触发一个非法指令异常。

通常情况下，只用实现了IMSIC，miselect中的0x70-0xff才能被使用；否则，当miselect在这个范围，试图访问mireg寄存器将触发一个非法指令异常。外部中断区域的内容将第三章介绍。

CSR mtopei寄存器，只有IMSIC实现时才存在，它也会在第三章介绍。

CSR mtopi寄存器，其中记录Machine level目前pending并且使能的最高等级的中断。

当实现了S模式时，CSRs mvien和mvip可以在supervisor level实现中断过滤和虚拟中断，这个将在第五章的第三节介绍。

Supervisor-level CSRs

图2.2中列出在S模式添加的CSR寄存器，每个寄存器都是SXLEN比特长的，它们的功能和Machine level是对应的。从siselect/sireg中访问的寄存器空间是和machine level分开的，用于supervisor level的中断而不是machine level的中断。siselect从0到0xff的分布如下所示：

0x00-0x2f reserved

0x30-0x3f 主中断

0x40-0x6f reserved

0x70-0xff 外部中断（with IMSIC）

为了最大的兼容性，建议siselect最少实现9比特，不管IMSIC是否存在，可以从0到0x1ff。

Number	Privilege	Name	Description
Supervisor-Level Window to Indirectly Accessed Registers			
0x150	SRW	siselect	Supervisor indirect register select
0x151	SRW	sireg	Supervisor indirect register alias
Supervisor-Level Interrupts			
0x15C	SRW	stopei	Supervisor top external interrupt (only with an IMSIC)
0xDB0	SRO	stopi	Supervisor top interrupt
Supervisor-Level High-Half CSRs for Interrupts 32–63 (RV32 only)			
0x114	SRW	sieh	Extension of sie
0x154	SRW	siph	Extension of sip

Table 2.2: Supervisor-level CSRs added by the Advanced Interrupt Architecture.

就像miselect，siselect的最高位是定制功能设定的，如果SXLEN改变了，那么最高位要移动一个新的位置，并且保持原来的值。实现时并不要求支持siselect任何定制值。

当siselect的值在一个reserved的范围内，或者在当没有IMSIC的时候在0x70-0xff内，试图访问sireg寄存器会导致一个非法指令异常。

CSR stopei将在第三章中讲述。

stopi寄存器中存储着supervisor level当前pending并使能的最高等级中断。

Hypervisor and VS CSRs

当一个实现了hypervisor特权级拓展，那么将会添加图2.3中的hypervisor and VS CSR寄存器。这些VS CSR寄存器都是VSXLEN比特宽的，hypervisor CSR寄存器都是HSXLEN比特宽的。

Number	Privilege	Name	Description
Virtual Interrupts and Interrupt Priorities for VS Level			
0x608	HRW	hvien	Hypervisor virtual interrupt enables
0x609	HRW	hvictl	Hypervisor virtual interrupt control
0x646	HRW	hviprio1	Hypervisor VS-level interrupt priorities
0x647	HRW	hviprio2	Hypervisor VS-level interrupt priorities
VS-Level Window to Indirectly Accessed Registers			
0x250	HRW	vsiselect	Virtual supervisor indirect register select
0x251	HRW	vsireg	Virtual supervisor indirect register alias
VS-Level Interrupts			
0x25C	HRW	vstopei	Virtual supervisor top external interrupt (only with an IMSIC)
0xEB0	HRO	vstopi	Virtual supervisor top interrupt
Hypervisor and VS-Level High-Half CSRs (RV32 only)			
0x613	HRW	hidelegh	Extension of hideleg
0x618	HRW	hvienh	Extension of hvien
0x655	HRW	hviph	Extension of hvip
0x656	HRW	hviprio1h	Extension of hviprio1
0x657	HRW	hviprio2h	Extension of hviprio2
0x214	HRW	vsieh	Extension of vsie
0x254	HRW	vsiph	Extension of vsip

Table 2.3: Hypervisor and VS CSRs added by the Advanced Interrupt Architecture.

图中的VS CSR寄存器和supervisor CSR是对应的，当运行在虚拟机中时，这些寄存器用于替代那些supervisor的CSR（在VS或者VU模式）。

vsiselect需要至少支持9个bit范围从0到0x1ff，不管IMSIC是否实现了。就像siselect一样，vsiselelct的最高位用于定制的目的，如果VSXLEN修改了，那么最高位需要移动新的位置，并且保持原来的值。

vsiselect的空间相较于Machine level和Supervisor level有更多的限制。

0x000 - 0x06f　inaccessible

0x070 - 0x0ff　外部中断（with IMSIC），或者inaccessible

0x100 - 0x1ff inaccessible

当vsiselect当中的是inaccessible的，包括大于0x1ff并且最高位没有置位。从M模式或者HS模式访问vsireg寄存器将触发一次非法指令异常，试图从VS模式去访问sireg会触发一次虚拟指令异常。

外部中断部分的寄存器，只有当hstatus中VGEIN域是一个已经实现的guest 外部中断时，才能访问。如果VGEIN中不是一个已经实现的guest外部中断号，那么vsiselect的所有非自定义值都指向不可访问寄存器。

相似地，当hstatus的VGEIN不是已经实现的guest外部中断号，那么试图从M模式或者HS模式访问CSR vstopei将会触发一次非法指令异常，在VS模式试图访问stoepi将触发一次虚拟指令异常。

高位CSR寄存器除了hidelegh，AIA还添加了hvien、hvictl、hviprio1、hviprio2和hypervisor CSR寄存器的高位用于加强hvip注入中断到VS。这些寄存器的使用将在第六章中介绍。

虚拟指令异常

遵从hypervisor拓展的基本规则，从VS模式直接访问hypervisor或者VS寄存器，从VU模式访问任何supervisor level的CSR（包括hypervisor和VS的CSR寄存器）都会触发一次虚拟指令异常，而不是非法指令异常。细节可以查看RISC-V特权架构。

state-enable CSR的访问控制

如果实现了Smstateen扩展和AIA，state-enable寄存器mstateen和mstateen0h中的3个比特可以控制M模式以下的特权级对于AIA添加状态的访问。

bit 60: CSRs siselect, sireg, vsiselect和vsireg

bit 59: AIA添加的状态但是不受比特60和比特58控制

bit 58: 所有IMSIC状态，包括CSRs stopei和vstopei

如果在mstateen0/mstateeen0h中存在一个bit为0，从一个低于M模式的特权级来的访问相应的状态将会触发一次非法指令陷入。和以往一样，这些state-enable CSRs不影响M模式下的状态访问。更多的相关说明可以在Smstateen拓展的文档中找到。

Bit59 控制了对于AIA CSRs siph、sieh、stopi、hidelegh、hvien/hvienh、hviph、hvictl、hviprio1/hviprio1h、hviprio2/hviprio2h、vsiph、vsieh和vstopi，还有supervisor-level中断相关的siselect和sireg。

Bit58 只在实现了IMSIC时实现，如果虚拟化拓展也实现了，这个比特并不影响hypervisor CSRs hgeip和hgeie，或者hstatus中的VGEIN的行为和可访问性。特别地，从IMSIC来的guest外部中断，即使在bit58是0的时候，仍然对于HS可见。

如果实现了hypervisor拓展，那么在hypervisor CSR寄存器hstateen0/hstateen0h中也会相应的3个bits，但是只关心VS和VU的状态访问。

bit 60 CSRs siselect 和 sireg（实际上是vsiselect和vsireg）

bit 59 CSRs siph、sieh和stopi（实际上是vsiph、vsieh和vstopi）

bit 58 IMSIC guest中断文件的所有状态，包裹CSR stopei（实际上是vstopei）

如果在hstateen0/hstateen0h上面其中一个比特为0，但是mstateen0/mstateen0h相应的比特为1，那么在VS或者VU状态访问相应的状态会导致一次虚拟指令异常。（但是对于高部分CSRs siph和sieh，只有在RV32时使用，当XLEN>32时，试图访问siph和sieh会导致非法指令异常，而不是虚拟指令异常。）

当且在实现了IMSIC时，在hstateen0和hstateen0h才会实现Bit58。即使实现了IMSIC，如果IMSIC对于guest外部中断没有实现guest中断文件，这个bit也只能是一个只读的0。当这个bit是0时，和hstatus.VGEIN一样，虚拟机中不能访问核的IMSIC。

Sstateen扩展定义在supervisor-level视角的Smstateen，Sstateen因此可以和hstateen0 / hstateen0h 中定义到相应的bit进行协作。

第三章：IMSIC

IMSIC是一个可选的RISC-V硬件，一个IMSIC和一个核紧紧联系在一起。IMSIC接受和记录发送给一个核的所有MSIs，并且通知核哪些已经pending并使能的中断等待被处理。

一个IMSIC可以在地址空间有一个或者多个MMIO寄存器，用于接受MSIs。除了这些MMIO寄存器，软件可以通过添加到核上的几个CSR寄存器，和IMSICs进行通信。

中断文件和中断实体

在RISC-V系统中，MSIs不仅仅可以导向一个特定的核，还可以导向一个核上一个特定的特权级，如Machine或者Supervisor level。当一个核实现了hypervisor特权级，一个IMSIC可以有选择地将MSIs发送给一个特定的在VS特权级的虚拟核。

这些MSIs可能指向的核上的每一个特权级和每一个虚拟核，核中的IMSIC都包含一个独立的中断文件。假设这个核实现了supervisor模式，那么这个核的IMSIC包含了至少两个中断文件，一个用于Machine level和一个用于Supervisor level。当一个核实现了hypervisor扩展时，它的IMSIC可以添加一个中断文件到虚拟核，这个中断文件称之为guest中断文件。IMSIC上虚拟核的guest中断文件数量是GEILEN，即支持的guest外部中断的数量，已经在RISC-V特权架构hypervisor扩展部分介绍。

每一个中断文件包含两个大小相同的bits数组，一个数组用于记录已经到了倒是没被服务的MSIs（pending），另一个数组用于说明，现在哪些中断将被接受（enable）。两个数组中的每一个bit都会对应一个都对应一个中断实体号，通过中断实体号，就可以区分不同的MSIs。因为一个IMSIC是一个核的外部中断控制器，所以一个中断文件的中断实体号，就是一个核上的外部中断的从身份。

一个中断文件支持的中断实体数量小于64的倍数，最小是63，最大是2047。

当一个中断文件支持N个不同的中断实体，那么1到N就是合法的中断实体号。在这一范围的实体号就是中断文件实现的，超出这个范围就是没有实现的。0永远不是一个合法的实体号。

IMSIC硬件并不假设一个中断文件中的中断实体号和另外一个中断文件中的存在任何联系。软件通常在不同的中断文件中将同一中断号分配给不同的MSI源，不同中断文件间不需要协作。因此在一个系统中可以单独区分的MSI源总数量，很可能是当个中断文件中中断实体的数量乘以总的中断文件数量。

一个系统中的中断文件大小并不要求保持一致。对于一个给定的核，guest外部中断的中断文件必须保持大小一致，但是machine level和supervisor level的中断文件大小可以不一致。相似地，不同核的中断文件大小可以不一样。

一个平台可以一个方法让软件可以配置IMSIC中的中断文件数量，和中断文件的大小，例如在machine level将中断文件设置得小一点，在supervisor level将中断文件设置得大一点。这些配置已经在本份文档的讨论范围之外了。但是最好只有在Machine level有能力修改IMSIC中的中断文件大小和数量。

MSI编码

已经确立的标准中（如PCI和PCI Express）说明一个MSI是一个自然对齐的32bits的写，软件可以在设备控制器配置地址和写的值。根据设备控制器的版本不同，这个地址可以限制在低4GB，写的值可以限制在16bits宽，同时高的16bits总是为0。

当RISC-V核有IMSICs，一个设备上来的MSI可以直接送到软件选择的一个核上，之后进行中断处理。

一个MSI会通过核上IMSIC相应的中断文件送到一个特定的特权级，或者一个特定的虚拟核。MSI写的地址是一个word-size寄存器的物理地址，这个寄存器和一个中断文件相连。MSI写的数据就是中断文件已经pending的中断的实体号（最终成为这个外部中断的从身份号）。

通过配置设备的一个MSI的地址和数据，这个系统就可以由软件完全控制：

1. 哪个核可以接受特定设备的中断
2. 目标特权级或者虚拟核
3. 在目标中断文件中代表MSI的实体号码

第1点和第2点，由MSI地址指向哪个中断文件决定；第3点由MSI的内容决定。

当实现了hypervisor拓展，并且设备直接由guest操作系统操控，从设备来的一个MSI地址是一个guest物理地址，因为它们都是guest操作系统配置的。这些地址还需要经过IOMMU的翻译，这些都是hypervisor配置的，可以将这些MSI重定向到正确的guest中断文件。更多细节，可以看第八章。

中断等级

在单个中断文件中，中断等级是直接由中断实体号决定的，低的实体号有更高的特权级。

Reset 和 显示状态

一旦IMSIC reset了，所有中断文件的状态就会变成有效和一致，但是在其他方面不做保证。正如machine level和supervisor level的eidelivery寄存器。

如果IMSIC包含了一个supervisor level中断文件，对应核上的软件使能了之前被disabled的S模式，所有supervisor level的中断文件都会变得有效和一致，但是其他方面不做保证。相似地，如果一个IMSIC包含guest中断文件，并且这个核上的软件使能了之前被disabled的hypervisor拓展，IMSIC guest中断文件会变得有效并且一致，但是其他方面不做保证。

中断文件的内存区域

每一个中断文件都用一个或者两个MMIO的32bits的寄存器用于接受MSI写。这些MMIO寄存器都是位于自然对齐的4KB区域，这一页的物理地址专门用于中断文件。

中断文件的内存布局如下所示：

offset size register_name

0x000 4 bytes seteipnum_le

0x004 4bytes seteipnum_be

中断文件中的其他bytes是保留的，并且必须被实现为只读的0。

中断文件的内存区域只支持32bits对齐的读和写。向只读区域写，操作将被忽略。对于其他形式的访问（其他大小、非对齐、AMO），IMSIC在实现上最好报告一次访问异常或者总线错误，否则需要忽略这些访问。

如果i是一个已经实现的中断实体号，将值i以小端的形式写入到seteipnum_le(Set External Interrupt-Pending bit by Number, Little-Endian)会使得中断i的pending bit被置1。如果这个写的中断实体号没有被实现，那么这次写将被忽略。

对于实现了大端的系统，如果i是一个已经实现的中断实体号，以大端模式将值i写入到seteipnum_be(Set External Interrupt-Pending bit by Number，Big-Endian)会中断i的pending bit被置1.如果对应中断实体号的大端模式没有被实现，那么这次写将被忽略。系统只实现了小端模式会选择忽略对于seteipnum_be的写。

在大多数系统中，seteipnum_le是中断文件的写口。但是对于大端的系统，seteipnum_be是默认的写口。

对于seteipnum_le和seteipnum_be的读会在任何情况下返回0。

当写操作没被忽略时，对于一个中断文件的写，会最终会反应到中断文件上，但是这个不一定是立即发生。对于一个中断文件，对于一个内存区域的多次写，即使被任意地延迟，但是在全局内存序上是一致的，这个已经在RISC-V Unprivileged ISA中定义了。

多个中断文件的内存区域排布

IMSIC中每一个中断文件都会有自己的内存区域，该内存区域都是一个机器上的4KB大小空间。在实践中，所有IMSICs的所有中断文件的页必须分布在一块内存区域，同时所有supervisor-level和guest中断文件必须一起放在另外一块内存区域。

如果机器规定核需要被分成小组，每一个小组都要有自己的一部分地址空间，那么最好每一个小组的machine level中断都各自放到一起，相似地，将每一个小组supervisor level和guest中断文件放到一起。

为了定位每一个中断文件在地址空间中的位置，假设每一个核都有一个独立的核号（可能和RISC-V Privileged Architecture中分配的Hart IDs无关）。为了方便地址映射，所有machine-level的中断文件的页都需要根据核号进行布局，表达式为$A + h \times 2^C$, 其中A和C都是常数，对于最大的核号h_{max}, 我们设$k = \lceil log_2(h_{max} + 1) \rceil$，这就是我们需要的用于表示核号的比特数。同时基地址A需要是2^{k+C}对齐，所以$A + h \times 2^C = A|(h \times 2^C)$。

C的最小值必须是12， 所以2^C最小值是4096KB的页。如果C 大于 12，那么machine level中断文件地址的其实地址就不能只是4KB对齐，而要是2^C对齐的。C大于12，那些没被中断文件占用的页，需要被填充为只读的0，同时对于这些区域的写需要被忽略。

对于所有supervisor level中断文件的内存页，应该被安排到 $B + h \times 2^D$, 及地址B应该和2^{k+D}对齐，k还是上面讲的一致，表示需要的核号位数。

如果一个IMSIC实现了guest中断文件，IMSIC的supervisor level中断文件和guest中断文件的内存页应该是连续的，最低地址从supervisor level中断文件开始，之后guest中断文件，排列如下：

$$S, G_1, G_2, G_3, \ldots$$

其中S是supervisor level中断文件，G_i 表示i号guest中断文件的页。结果上，D最小要是$\lceil log_2(maximum\ GEILEN + 1) \rceil + 12$, GEILEN表示了IMSIC实现了的guest中断页表的数量。

当一个系统将核分组，每一个都独立拥有地址空间的一部分。Machine level中断文件的页基址为$g \times 2^E + A + h \times 2^C$ ，同时supervisor level中断文件的页基址$g \times 2^E + B + h \times 2^D$, g是组号，h是组中的核号，E是另外一个整数，$E \geq k + max(C, D)$，通常情况下会比这个下限大很多。如果最大的组号是$g_{max}$, 让$j = \lceil log_2(g_{max} + 1) \rceil$。

通过CSR访问的外部中断寄存器

软件访问一个核上的IMSIC主要通过第二章中介绍的寄存器。在每一个特权级都有一组CSRs可以接受中断。Machine level CSRs和Machine level中断文件进行交互，但是，如果实现了supervisor level，那么supervisor level CSR和IMSIC的supervisor-level中断文件进行交互。当一个IMSIC有guest中断文件，那么VS CSRs可以和一个guest中断文件进行交互，这个中断文件将由CSR hstatus中的VGEIN决定。

对于Machine level，相关的CSR寄存器是miselect、mireg和mtopei。当实现了supervisor-level，相关的寄存器是siselect, sireg, stopei。当实现了hypervisor拓展，相关VS CSR是vsiselect、vsireg和vstopei。

正如第二章中介绍的那样，miselect和mireg寄存器提供了访问非直接访问添加在machine-level CSR的方式。相似地，supervisor-level有siselect和sireg，和VS-level vsiselect和vsireg。在各种情况下，*iselect CSR寄存器（miselect, siselect, vsiselect) 在范围0x70 到 0xff中选择相应IMSIC中断文件中的一个寄存器，要么是Machine level中断文件、Supervisor level中断文件或者guest中断文件。

每个特权级的中断文件表现一致。对于一个给定的特权级，在0x70-0xff范围内的*iselect CSR会选择相应中断文件的对应寄存器：

0x70　eidelivery

0x72　eithreshold

0x80　eip0

0x81　eip1

0xBF　eip63

0xC0　eie0

0xC1　eie1

…　…

0xFF　eie63

寄存器号0x71和0x73-0x7F是保留的。当一个*iselect CSR有一个如上保留值，从对应的*ireg将读到0，同时写*ireg会被忽略。

eip0寄存器到eip63寄存器包含了所有实现的中断实体的pending bits，因此被称之eip数组；寄存器eie0到eie63包含了中断实体的所有enable bits，因此被称为eie 数组。

不直接访问的中断文件寄存器和CSR寄存器mtopei, stopei, vstopei将在下面两节更加具体地介绍。

不直接访问的中断文件寄存器

这一节会介绍可以通过*iselect寄存器和*ireg寄存器访问的中断文件寄存器。对于每一个中断文件，对于每一个不可直接访问的寄存器都是* ireg 相同宽度的，是MXLEN、SXLEN或者VSXLEN。

外部中断传递使能寄存器（eidelivery）

eidelivery是一个WARL寄存器，它控制了是否可以从IMSIC的中断文件的信号传送到对应的核，之后触发一个mip或者hgeip CSR寄存器中的中断bit pending。eidelivery可以有选择地支持从PLIC或者APLIC传输中断信号到对应核。现在eidelivery定义了三个可能的值：

0 = 中断传递禁止

1 = 中断文件的中断传递使能

0x40000000 = 从PLIC或者APLIC的中断传递使能

如果eidelivery支持0x40000000, 那么系统中的PLIC或者APLIC，可以在作为当前特权级当前中断文件的一个可选的外部中断控制器。当eidelivery是0x40000000，中断文件的功能和eidelivery的值是0的情况是一致的，并且PLIC或者APLIC可以代替中断文件来在特定核上的特定特权级提供pending的外部中断。

guest中断文件目前并不支持eidelivery的0x40000000。如果eidelivery支持0x40000000，那么Reset可以将eidelivery的值初始化为0x40000000。否则eidelivery初始化后将将会是一个不确定的0或者1。

eithreshold

eip0-eip63

eie0-eie63

Top external interrupt CSRs（mtopei，stopei，vstopei）

这个在*topei CSR寄存器中的中断实体是这个核上的外部中断的次中断号。

这里从*topei CSR中读取出来的冗余值是为了兼容Advanced PLIC，它会返回一个中断实体号还有它的中断等级，但是这两个号码在PLIC中是独立的。

一个对于*topei CSR寄存器的写预示着报告这个中断时间，同时清楚相应的bit。这个写的值会被忽略，但是当前这个寄存器的可读值决定那个中断将会被清楚。特别地，当一个*topei CSR寄存器被写时，如果一个寄存器中是一个特定中断号，那么这个中断相应的pending bit将被清楚。如果这个寄存器中的值为0，那么将不产生任何效果。

如果同时对这个CSR寄存器进行读写，例如一条CSR指令（CSRRW、CSRRS或者CSRRC），那么这个被读出来的值意味这个pending的bit将被清除。

Interrupt delivery and handling

一个IMSIC中断文件提供外部中断信号给给定的核，每一个中断文件一个中断信号。从Machine-level中断文件来的中断信号体现为CSR mip中的MEIP，Supervisor level中断文件来的中断信号体现为MIP和SIP中的SEIP。guest中断文件来的中断信号体现为hypervisor CSR寄存器hgeip的active bits。

当中断delivery被一个中断文件的eidelivery寄存器给disabled了，那么从一个中断文件来会被held de-asserted, 当一个从一个中断文件来的interrupt delivery被使能了，当且仅当中断文件有pending并且enable的中断并且超出了中断阈值，那么这个信号才会asserted。

一个外部中断的trap handler通过IMSIC可以被如下的值写：

save processor registers

i = read CSR mtopei 或者 stopei, 并且write来claim中断

i = i >> 16

调用中断i的interrupt handler

恢复processor registers

从trap返回

第二步中的对于mtopei或者stopei的读写可以通过一条CSRRW指令实现

```
csrrw rd, mtopei/stopei, x0
```

where rd is the destination regster for value i.