- Advanced Platform-Level Interrupt Controller(APLIC)
  - 中断源和中断号
  - 中断域
  - Hart下标
  - 单个域的中断控制
  - 中断域的MMIO控制区域
    - 域配置
    - 源配置（sourcecfg[1] - sourcecfg[1023]）
    - M态MSI 地址配置（mmsiaddrcfg 和 mmsiaddrcfgh）
    - S态MSI地址配置（smsiaddrcfg 和 smsiaddrcfgh）
    - 设置中断的pending bits（setip[0] - setip[31]）
    - 通过编号设置 interrupt pending （setipnum）
    - 输入和清空interrupt-pending bits（in_clrip[0] - in_clrip[31]）
    - 通过编号来清空interrupt pending（clripnum）
    - 设置interrupt pending使能 bits （setie[0] - setie[31]）
    - 通过编号设置 interrupt pending使能bits（setienum）
    - 清空interrupt使能位（clrie[0] - clrie[31]）
    - 根据编号将interrupt 使能清空 （clrienum）
    - 根据编号以小端方式设置中断pending（setipnum_le）
    - 根据编号以大端方式设置中断pending（setipnum_be）
    - 生成MSI（genmsi）
    - 中断目标（target[1] - target[1023]）
      - 活跃源，直接转送模式
      - 活跃源，MSI传输模式
  - 中断pending bits的精确作用
  - APLIC的中断直接传输模式
    - IDC结构
    - 中断传输使能（idelivery）
    - Interrupt force （iforce）
    - 中断使能阈值（ithreshold）
    - 最高优先中断（topi）
    - 获取最高优先级中断（claimi）
    - 中断传递与处理
  - 通过MSI进行中断传递
    - MSI的目标地址和数据
    - 对于中断源的输入为电平敏感型的考虑
    - 同步hart和APLIC

（**Translated by Porterlu，any recommendations are welcomed.**）

Advanced Platform-Level Interrupt Controller(APLIC)

在RISC-V系统中，一个Platform-Level Interrupt Controller（PLIC)可以处理通过直连线传进来的外部中断（不是MSIs的情况）。当RISC-V核没有实现IMSICs，那么这些核就不支持MSIs，所有外部中断都传送到PLIC或者APLIC。但是即使核上有IMSICs并且大多数中断通过MSIs进行通信，但是有一些设备更多地是通过专门的直连线进行中断。一些不需要总线事务的设备，实现MSIs的成本会特别高，所以使用直连线是更好的选择。直连线的中断被所有的平台支持，而MSIs不是这样，所以几乎所有的商业设备选择直连线而不是MSIs，除非实现了一个像PCIe这样的标准要求MSIs。

所以这章介绍了Advanced PLIC(APLIC)，它并不向后兼容PLIC。完全符合标准的AIA需要APLIC作为直连线和MSI信号的转化器。然而可以建立一个可行的系统来替代老的PLIC，使用老的直连线中断，而不是MSIs也是可行的。

在一个没有IMSICs的机器上，每一个核都会直接从PLIC或者APLIC接受自己的外部中断。外部中断信号也通过直连线接入到对应的核上，这个核上的每一个特权级都可能收到中断信号。在一个没有IMSICs的系统典型地只会有一个PLIC或者APLIC，做为RISC-V核的外部中断控制器。

如果RISC-V核实现IMSICs做为自己的外部中断控制器，接受MSIs形式的外部中断。在这种情况下，APLIC就会将一个直连线信号转化为MSIs信号。当核有IMSICs，可以支持MSIs，一个系统可以包含多个APLIC，每个APLIC将部分外设的直连线中断转化为MSIs信号。

中断源和中断号

一个特定APLIC支持固定数量的中断源，就是进入的直连线中断信号。更多情况，每一个直连线都来来自一个设备或者设备控制器。如果APLIC中断源都一个具体的中断号，范围从1到N，N就是APLIC的中断源总量。0号并不是APLIC的合法中断号，APLIC支持的最大中断数量为1023。当APLIC将一个中断传送到一个核的特定特权级（不是通过MSIs），如果APLIC是这个核这个特权级的外部中断控制器，在APLIC的中断号是这个核外部中断的次中断号（相对于外部中断这个固定的中断号）。从其他方面讲，当APLIC通过MSIs传递中断，软件对于每一个MSIs配置一个中断号，那么原来的APLIC上的中断号只可以从进入APLIC进行区分，出了APLIC就不区分了。

中断域

一个APLIC支持一个或者多个中断域，每一个都连接一部分RISC-V核的一个特权级（M态或者S态）。在一个中断域的核可以将中断传送特定的特权级。每一个域中都可以用自己的MMIO区域来控制一个完整APLIC，即使事实上所有的域接口会访问一个同一个的中断控制器。

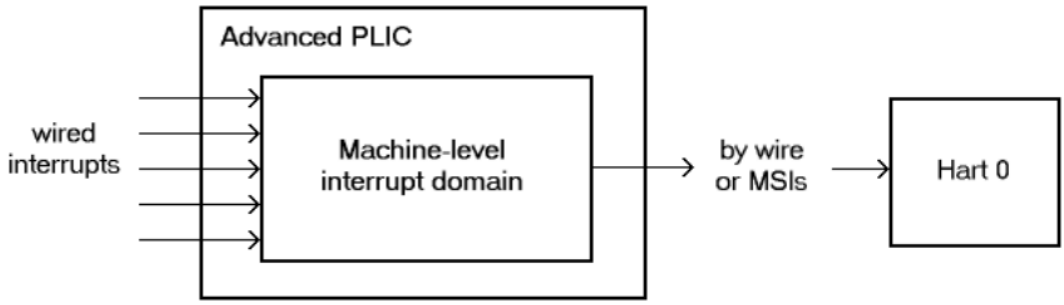

Figure 4.1: Example of a RISC-V system that has a single hart implementing only M-mode, with a single machine-level interrupt domain for that hart.

图4.1中展示了在一个不支持S态的hart上的最小系统，只有一个M态的中断接到hart上。在图4.2中，展示了基本的多核系统的配置，有多个hart并且每个hart都实现了S态。在这种情况下，APLIC会提供一个独立的S态中断域。它允许运行在hart上的S态操作系统直接控制它接受到的中断，而不需要转发到M态进行处理。

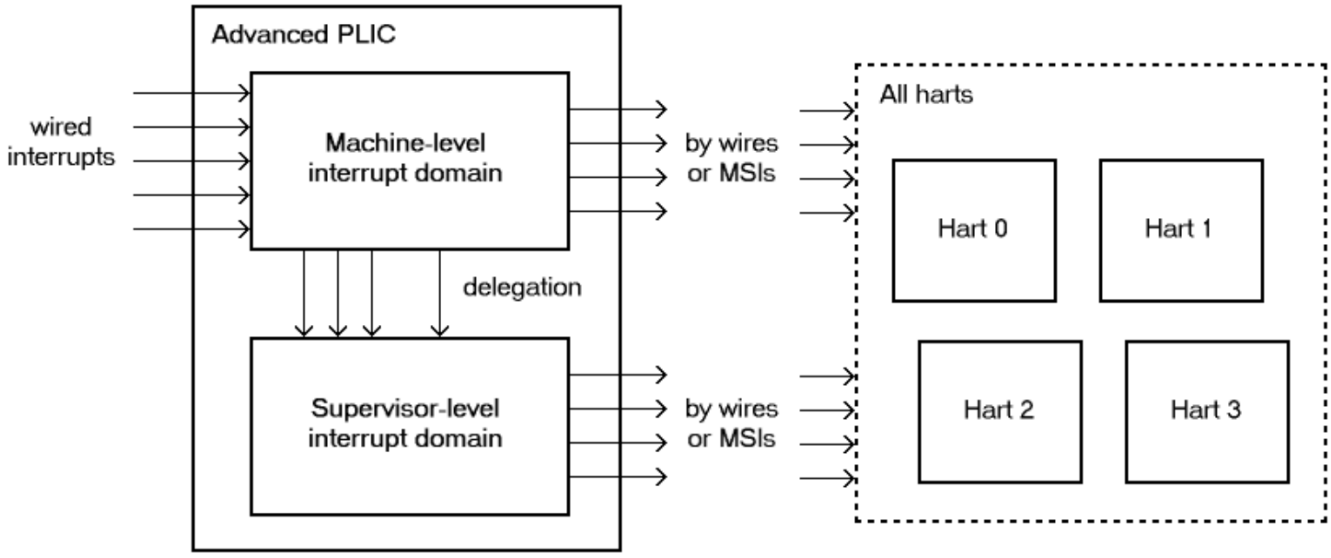

Figure 4.2: An example system with four harts that implement M-mode and S-mode, with two APLIC interrupt domains, one each for machine and supervisor levels.

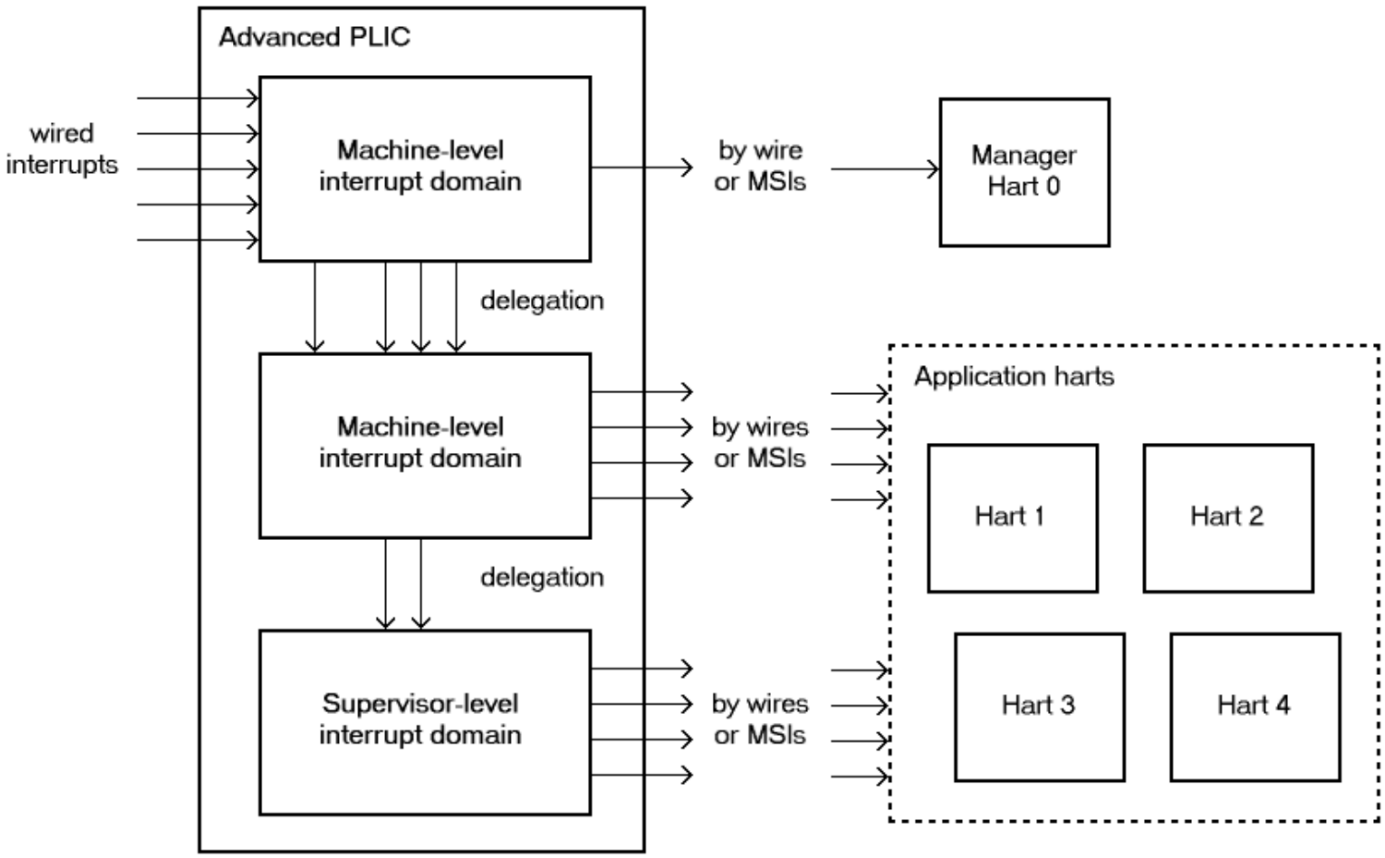

Figure 4.3: A RISC-V system that extends the example of Figure 4.2 with a fifth M-mode-only "manager" hart, with a separate machine-level interrupt domain above the other domains.

APLIC的中断域被设置为树状结构，根域总是M态的。进入的直连中断总是先进入根域。每一个域都可以选择将自己的中断委托给子域。在给定的APLIC，中断号在不同的中断域是不变的，所以一个中断i在不同的中断域中是同一个中断源。对于根域之下的中断域，没有被传播到自己的子域或者发出的中断应该是没有实现的。

图4.3中存在两个M态的中断域和一个S态的中断域，和PMP搭配到一起可以允许M态软件将专门划归一部分中断给harto，即使在其他核的M态也不能接收到。

为了在一个中断域中hart应该可以直接控制来自这个域中的中断，在域所对应的特权级中必须要使用软件对所有的核进行控制。具体地说，一个操作系统应该控制在一个S态中断域中的所有的hart。在图4.2和4.3的例子中，一个中断域中的核并不能被安全地分配给多个操作系统。假设域的层次就是图中所示，如果需要将hart分给操作系统，M态的软件就因该阻止OS直接访问S态中断域而是使用SBI的服务去访问APLIC。

一个APLIC的中断域应该满足下面的规则：

- 根域应该在M态
- S态中断域的父态应该是包含了相同hart的M态中断域，它应该包含了更多的hart。
- 对于每一个中断域，域中来的一个中断会以相同的方式而不是不同的方式通知所有的核。

当一个RISC-V hart的外部中断控制器是APLIC而且不存在IMSIC时，该APLIC在每个特权级别上只能在该APLIC的一个中断域内。从另一方面讲，一个hart如果将IMSIC作为外部中断控制器，每个特权级可以在不同的APLIC的中断域中，并且接受来自不同APLIC的MSIs。一个平台可以给软件在任何给定的APLIC的中断域进行选择。

Hart下标

在一个给定的中断域中，每一个hart都有一个独一无二的标号（从 0 到 $2^{14}-1=16383$）。这个域中给这个hart的标号可以和RISC-V 特权架构中每个hart的hart id没有任何关系。两个不同的中断域中可以对于同一组的hart使用完全不同的标号。然而，如果一个APLIC域可以发送MSI，那么所有的APLIC的M态域都要对于每一个hart使用相同的标号。

单个域的中断控制

APLIC的每一个中断域在机器的地址空间都有自己的MMIO接口，允许通过PMP和页表翻译的方式进行访问控制。所有中断域的接口都是相同的结构。在大多数方面，每个域在软件看来都像是一个根域，在层次结构中看不见上面的域。

一个中断域中，APLIC中对于每个中断源有如下的配置：

- 源信息。这个决定这个源是否在这个域中是活跃的，则来的信号可以配置为电平触发或者是边缘触发的。对于一个域中的不活跃的源，会将中断传递给子域。
- 中断pending和enable bit。对于一个非活跃的源，这两个bits只会读出0。否则，pending bit会记录一个中断是否到来并且还没被处理，同时enable bit会决定这个中断是否会被传入到核中还是继续pending。
- 目标选择。对于一个活跃的源，目标选择决定接收到中断的hart，还有作为MSI时的中断号和优先级。

对于一个中断域直接传递中断到核中而不是通过MSIs的情况，会有专门的组件传递中断。

中断域的MMIO控制区域

对于APLIC支持的每一个中断域，会有专门的MMIO空余区域来管理中断。这个控制区域时多个4KB组成的区域，且每个区域都是4KB对齐的。最小的控制区域大小为16KB。每一个控制区域中有一系列32比特的寄存器。

offset	size	register name	
0x0000	4 bytes	domaincfg	
0x0004	4 bytes	sourcecfg[1]	
0x0008	4 bytes	sourcecfg[2]	
...		...	
0x0FFC	4 bytes	sourcecfg[1023]	
0x1BC0	4 bytes	mmsiaddrcfg	(machine-level interrupt domains only)
0x1BC4	4 bytes	mmsiaddrcfgh	”
0x1BC8	4 bytes	smsiaddrcfg	”
0x1BCC	4 bytes	smsiaddrcfgh	”
0x1C00	4 bytes	setip[0]	
0x1C04	4 bytes	setip[1]	
...		...	
0x1C7C	4 bytes	setip[31]	
0x1CDC	4 bytes	setipnum	
0x1D00	4 bytes	in_clrip[0]	
0x1D04	4 bytes	in_clrip[1]	
...		...	
0x1D7C	4 bytes	in_clrip[31]	
0x1DDC	4 bytes	clripnum	
0x1E00	4 bytes	setie[0]	
0x1E04	4 bytes	setie[1]	
...		...	
0x1E7C	4 bytes	setie[31]	
0x1EDC	4 bytes	setienum	
0x1F00	4 bytes	clrie[0]	
0x1F04	4 bytes	clrie[1]	
...		...	
0x1F7C	4 bytes	clrie[31]	
0x1FDC	4 bytes	clrienum	
0x2000	4 bytes	setipnum_le	
0x2004	4 bytes	setipnum_be	
0x3000	4 bytes	genmsi	
0x3004	4 bytes	target[1]	
0x3008	4 bytes	target[2]	
...		...	
0x3FFC	4 bytes	target[1023]	

Table 4.1: The registers of the first 16 KiB of an interrupt domain's memory-mapped control region.

从0x4000的偏移开始，一个控制区域后面可以选择接上一个Interrupt Delivery Control结构数组，每一个IDC都用于一个中断域中的hart index。IDC用于配置将中断直接传入到hart中而不是使用MSIs的情况。一个中断域中如果使用MSIs传递中断信号而不是直接传递那么就不需要IDC结构。第一个IDC结构是0号hart的IDC，第二个是1号的以此类推。每一个IDC结构都是32bytes的，有如下的寄存器。IDC结构是连续打包的，每一个结构都是32bytes连续的，所以每个结构之间相差0x20。

offset	size	register name
0x00	4 bytes	idelivery
0x04	4 bytes	iforce
0x08	4 bytes	ithreshold
0x18	4 bytes	topi
0x1C	4 bytes	claimi

IDC结构数组包含了潜在hart下标，它并不包含了实际在域中。例如，第一个IDC结构属于hart index 0，但是0可能并不是域中一个有效的hart下标。对于每一个IDC结构并不需要对应到一个域中的有效下标。当没有对应的hart时，对应的寄存器就是只读的0。只有自然32比特对齐的读写才能对于中断域的控制区域进操作。对于只读bytes的写将被忽略。对于第一个16KB的寄存器将在下面一节进行说明。

域配置

domaincfg寄存器的格式如下：

bits 31:24　只读的0x80

bit 8　　IE

bit 7　　只读的0

bit 2　　DM(WARL)

bit 0　　BE(WARL)

IE的意思时Interrupt Enable在一个中断域的全局的使能，只有在IE使能时，enable并且pending的中断才能传送到hart中。

Field DM(Delivery Mode)是WARL并且可以决定如何将中断域中的中断传送到harts。两个可能的值是：

0 = direct delivery mode

1 = MSI delivery mode

在direct delivery mode，中断会被设定优先级并且直接通过APLIC传送到hart中。在MSI模式中，中断会通过APLIC转化为MSIs传送到harts中，之后通过IMSIC具体处理。一个给定APLIC实现可以支持任意一个或者两个都支持。

如果中断域中的harts有IMSI，除非IMSICs的相关中断文件支持eidelivery寄存器为0x40000000，将DM设置为0（直接传输模式），那么就和将IE设置为0的结果一致。

BE(Big-Endian) 是一个WARL寄存器，可以决定中断域中MMIO寄存器的字节序。如果BE=0, 则比特序是小端，BE=1则是大端。对于只支持小端的RISC-V系统，BE是一个只读的0，对于只支持大端的系统，BE是一个只读的1。对于同时支持大端和小端的系统，BE是可写的。

BE域可以影响访问domaincfg本身的字节序。

源配置（sourcecfg[1] - sourcecfg[1023])

对于每一个可能的中断源i，寄存器sourcecfg[i]可以控制中断域中的中断源i。当一个源i没有被实现时，或者在这个域中没有被实现时，sourcecfg[i]是一个只读的0。如果源i没有被委托到这个中断域，之后被改变为委托到这个中断域中，sourcecfg[i]会保持0，直到被写入一些新的值。

sourcecfg[i]的Bit 10是一个1比特的域叫做D(Delegate)。如果D = 1，那么这个中断源将被委托给子域，如果D = 0， 这个中断将不被委托给子域。将下来的Bit的意义取决于D。

如果这个中断源i被委托给子域，那么sourcecfg[i]的格式如下：

bit 10　　D = 1

bit 9:0　Child Index(WLRL)

其他寄存器bit将保留并且是只读的0。

Child Index是一个WLRL的域，其中的编号将决定源i被委托到哪个子域。对于一个有C个子域的中断域，这个域中的值必须是0 到 C-1。每个中断域将这个固定中断源i委托到子域。

如果一个域没有子域，那么这个中断域中的任何一个sourcecfg的bit D都不能设置为1。对于这样的一个叶子域，尝试对于这样个寄存器的bit 10写入值，这个寄存器都会被设置为0。

如果源i没有被委托给子域，那么sourcecfg[i] 将是如下的格式：

bit 10　　D = 0

bits 2:0　SM(WARL)

其他寄存器比特将都是保留的且是只读的0

SM(Source Mode)域是WARL并且控制是否中断源是活跃的。如果是活跃的，那么从直连线传入的信号将被翻译为中断。SM允许的值如下所示。SM总是支持0。实现上非常自由，每一个中断源可以独立实现它支持哪些值。

Value	Name	Description
0	Inactive	Inactive in this domain (and not delegated)
1	Detached	Active, detached from the source wire
2–3	—	*Reserved*
4	Edge1	Active, edge-sensitive; interrupt asserted on rising edge
5	Edge0	Active, edge-sensitive; interrupt asserted on falling edge
6	Level1	Active, level-sensitive; interrupt asserted when high
7	Level0	Active, level-sensitive; interrupt asserted when low

Table 4.2: Encoding of the SM (Source Mode) field of a `sourcecfg` register when bit D = 0

如果这个源被分配给子域，或者并没有进行委托并且SM是非活跃的，那么这个中断源就是非活跃的。在这个中断源是非活跃时，这个域中对应的中断pending bit和 中断enable bit就是只读的0，target[i] 也会是一个只读的0。如果源i从非活跃转变为活跃的时，中断源的pending和enable bit都会保持0，除非有特殊原因，target[i]将变为一个未加规定的非标准值。

当源被设置为Detached，那么这时的输入将被忽略。然而，但是中断pending bit仍然可以通过写入setip或者setipnum寄存器进行设置。

一个边缘敏感的源可以配置为上升沿触发也可以配置为下降沿触发，当配置为下降沿缘触发时，我们将这个源称为反向的；一个电平敏感的源可以配置高电平触发也可以配置低电平触发，当被配置低电平触发时，我们将这个源称为反向的。

对于一个中断源：

$$Input\ value = (\text{直连线传入信号}) XOR (\text{源是否是反向的})$$

对于一个非活跃的源，输入值（Input value）总是0。

任何对于sourcecfg寄存器的写，如果在新的模式下输入值判断为1，会造成中断pending寄存器变为1。一个对于sourcecfg的写本身在源处于非活跃时，不会造成中断pending。

M态MSI 地址配置（mmsiaddrcfg 和 mmsiaddrcfgh）

对于M态的中断域，寄存器mmsiaddrcfg和mmsiaddrcfgh可以提供参数来决定一些地址是否输出为MSIs。

如果APLIC没有中断域支持MSI传输模式（domaincfg.DM是只读的0的情况），任何中断域都不会实现这两个寄存器，当一个中断域没有实现mmsiaddrcf和mmsiaddrcfgh时，这些bytes会变成只读的0。

默认来说，mmsiaddrcfg寄存器和mmsiaddrcfgh寄存器是只能在根中断域进行写。对于其他M态中断域，它们是只读的。

其中mmsiaddrcfg的格式如下：

bits 31:0 Low Base PPN(WARL)

并且 mmsiaddrcfgh的格式如下：

bits 31 L

bits 28:24 HHXS(WARL)

bits 22:20 LHXS(WARL)

bits 18:16 HHXW(WARL)

bits 15:12 LHXW(WARL)

bits 11:0 High Base PPN(WARL)

所有其他的mmsiaddrcfgh bits都是保留的并且是只读的0。

高部分的PPN和低部分的PPN结合起来变成一个44bits的PPN（Physic Page Number）与HHXS（High Hart Index Shift）、LHXS（Low Hart Index Shift）、HHXW（High Hart Index Width）和 LHXW（Low Hart Index Width）一起可以用来决定MSIs的目标地址。

当mmsiaddrcfg和mmsiaddrcfgh是可写的，所有的部分除了L都是WARL。一个具体的实现可以选择自己所支持的值。典型地，一些bits是可写的，但是一些bits是一些只读的常量。在极端情况下，所有的部分都是常量，已经在实现中被固定好了。

如果mmsiaddrcfgh的L bit被设置为1，那么mmsiaddrcfg和mmsiaddrcfgh都将被锁住，对其的写入将被忽略，这些寄存器将变为只读的。当L = 1，除了L外的所有bits可以选择设置读出时全部变为0。在这种情况下，如果在根中断域中除了L有部分值不为0，它的值会在APLIC中保持不变，但是不再是通过读mmsiaddrcfg和mmsiaddrcfgh可见的了。

设置mmsiaddrcfgh.L为1也会锁住smsiaddrcfg和smsiaddrcfgh（如果这些寄存器实现的话）。

对于根中断域，L bit会在系统初始化时就设置为0或者1，任何一种在具体的实现中都是合理的。如果初始化时将L bit设置为1，其他部分将被APLIC硬件设置为常数，或者APLIC有Spec以外的方法来决定输出的MSI写的值。对于后一种，除了L的部分可以时读出时全0，那么这时对mmsiaddrcfg和mmsiaddrcfgh进行读出就只有分别是全0和0x80000000的情况了。如果mmsiaddrcfg和mmsiaddrcfg中不是这两个值，那么MSI就会根据可见值传入到对应的M态。

如果一个M态中断域不是根中断域，并且对应的寄存器已经实现了，并且bit L是1，那么从mmsiaddrcfgh和mmsiaddrcfg中读出来的值要么和根中断域保持一致，要么为全0。

假设软件有能力任意决定哪个地址传送哪个MSI那么就可以绕过PMP的检查，即使这个地址是只能由M态软件访问。如果APLIC支持MSI传送，那么建议将APLIC硬件连接到所有IMSICs的物理地址。然而，并不是所有实现都会遵循这一建议。

建议大多数系统将IMSICs的物理地址安排在一个根据hart下标排列的简单线性空间中。mmsiaddrcfg和mmsiaddrcfgh寄存器会运行可信的M态软件和早期的启动软件对于IMSICs的物理地址空间进行排布，之后将配置锁上。

APLICs被硬件写死IMSIC地址，那么这些寄存器可以实现为只读的0和0x80000000。或者如果IMSICs地址可以被软件配置但是处理mmsiaddrcfg和mmsiaddrcfgh过于复杂，这些寄存器也可以简单实现为只读的0和0x80000000，有专门的机制来配置IMSIC地址。

如果一个APLIC支持除了系统重启之外的重启方式，它需要定义如何影响根中断域中的mmsiaddrcfg和mmsiaddrcfgh寄存器。然而，它不应该让特权级不够的软件进行局部重启来解锁寄存器，将L bit从1变为0。出于这个理由，只有系统的重启才可以影响这两个寄存器，其他形式的重启无法做到这一点。

S态MSI地址配置（smsiaddrcfg 和 smsiaddrcfgh）

对于M态中断域，smiaddrcfg和smsiaddrcfgh可以提供参数来让S态中断域决定MSI所在的地址。如果一个中断域实现了smsiaddrcfgh和smiaddrcfg寄存器。如果一个域实现了mmsiaddrcfg和mmsiaddrcfgh寄存器，并且APLIC至少有一个S态中断域，那么这个域就应该实现smiaddrcfg和smiaddrcfgh寄存器。

就像mmsiaddrcfg和mmsiaddrcfgh寄存器一样，默认来说，只有在根中断域中才能对smsiaddrcfg和smsiaddrcfgh寄存器进行写。对于其他实现了M态中断域，这些寄存器只能是只读的。

当我们实现了这两个寄存器，smiaddrcfg有如下的格式：

bits 31:0　Low Base PPN(WARL)

和 smsiaddrcfgh有如下的格式：

bits 22:20　LHXS(WARL)

bits 11:0　High Base PPN(WARL)

其他为说明的bits都是保留的并且是只读的0.

smsiaddrcfgh中的高位PPN和smsiaddrcfg中的低位PPN会组成一个44位的PPN（Physic Page Number）。这个值将和LHXS(Low Hart Index Shift)一起决定MSIs的目标地址。

当smsiaddrcfg和smsiaddrcfgh寄存器是可写的，那么所有的部分都是WARL的。一个具体的实现可以选择它所支持的值。

如果这个中断域中的bit L被设置为1，那么smsiaddrcfg和smsiaddrcfgh就被锁住了。当mmsiaddrcfgh.L = 1, 那么mmsiaddrcfg和mmsiaddrcfgh寄存器读出来的值就分别是0和0x80000000，因为其中的值已经被隐藏了，同理smsiaddrcfg和smsiaddrcfgh也会被隐藏即读出的值是全0。

对根中断域，如果mmsiaddrcfgh.L = 1，那么地址配置就是隐藏的，之后不管对mmsiaddrcfgh.L做什么操作，smsiaddrcfg和smsiaddrcfgh的值将保持APLIC一开始设置的值，这些值就不再被用户可见。但是在具体实现中，可以提供非标准的方法来进行一些修改。

任何时间mmsiaddrcfg和mmsiaddrcfgh不是0和0x80000000时，传到S态的MSI信号可以从mmsiaddrcfgh、smsiaddrcfg和smisaddrcfgh的可见值中获取。

对于不是根域的M态中断域，如果有S态的地址配置寄存器且不是只读的0，那么它们只能是对于根域的拷贝。

设置中断的pending bits（setip[0] - setip[31]）

读写寄存器setip[k] 会修改中断源 32k -> 32k + 31 。在这个范围内的中断源i，对应寄存器中的bit i。对于setip寄存器的读会返回相关中断源的pending bits。如果对应bit的中断源不是活跃的，那么这个bit恒为0。对于setip寄存器的写，如果定义的bit对应了一个实现了的中断，将这个bit置1。

通过编号设置 interrupt pending （setipnum）

如果i是一个中断域中的活跃中断源，写入一个32bit的值i到setipnum寄存器中会使得域中中断源i的pending bit被置1。如果对应的bit的中断源没有实现，这个写被忽略，对这个bit的读出将返回0。

输入和清空interrupt-pending bits（in_clrip[0] - in_clrip[31]）

对于in_clrip[k]寄存器的读将会返回纠正过的输入(经过Source Mode修正果的输入，如将一个上升沿翻译为1)，对于这个寄存器的写将会修改将对应的pending bit进行修改，如果写入对应的bit为1，则将对应的bit清空。

通过编号来清空interrupt pending（clripnum）

如果i是中断域中的一个中断，对clripnum寄存器中写入一个32-bit的值i会造成定这个pending bit的清空。同理如果这个中断源没有被实现将会对写进行无视，对读返回0。

设置interrupt pending使能 bits （setie[0] - setie[31]）

读写setie[k]寄存器会潜在地修改中断源32k 到 32k + 31的中断使能。

一个对于setie寄存器的读，将会返回相关中断源的使能位。和上面一样如果对应的中断没有实现将会返回0。对于这个寄存器的写，会将对应的bit置1。

通过编号设置 interrupt pending使能bits（setienum）

如果中断域中的中断源i是活跃的，写入一个32bits的i到setienum寄存器中，将会置对应的使能位为1。如果对应中断源没有实现将会无视对应的写，读出时为0。

清空interrupt使能位（clrie[0] - clrie[31]）

对于clrie[k]寄存器进行写入会修改32k 到 32k + 31 的使能bit。

对于这个寄存器的写，会造成对应的interrupt使能被清空，对于这个寄存器的读总是返回0。

根据编号将interrupt 使能清空 （clrienum）

如果i是中断域中一个活跃中断源，写入一个32bits的值i将会对应的中断enable清空。如果对应的中断源没有实现，对其的写将被无视，同时读会返回0。

根据编号以小端方式设置中断pending（setipnum_le）

这个寄存器将会无视domaincfg中的配置，以小端的方式对中断pending进行配置，但是如果系统不支持小端，这个寄存器不需要被实现。

根据编号以大端方式设置中断pending（setipnum_be)

这个寄存器将会不是domaincfg中的配置，以大端的方式对中断pending进行配置，但是如果系统不只支持大端，这个寄存器不需要被实现。

生成MSI（genmsi）

当中断域已经被配置为MSI传输模式（domaincfg.DM = 1），genmsi寄存器可以被用于生成一个即时的MSI信号从APLIC传送到hart。主要的目的就是帮助建立hart对APLIC寄存器进行写和MSI从APLIC到hart两者之间的顺序。

寄存器的格式如下：

bits 31：18 Hart Index（WLRL）

bits 12 Busy（Read Only）

bits 10：0 EIID（WARL）

其它未说明的bits都是保留的并且为读出时为0。

Busy位一般来说时0，但是写一次genmsi会导致Busy被置1，表示一个即时MSI正在pending。Hart Index表示目标Hart，EIID(External Interrupt Identity)表示MSI的值。Field Hart Index和EEID与在target寄存器中的格式一致并且行为一致，将在下一节中介绍。对于一个M态的中断域，一个即时的MSI传送到hart的M态，对于一个S态的中断域，它会将一个即时MSI传送到目标核的S态。

一个pending的即时MSI应该以最小的延迟被APLIC处理。一旦它离开了APLIC那么就可以马上接受下一个对于genmsi的写。在MSI对于hart的IMSIC可见之前，所有的从之前从APLIC传送到同一个hart的MSI必须对于该hart可见。

如果Busy是true，所有对于genmsi的写将被忽略。

即时MSIs不会受domaincfg寄存器的IE bit的影响，即时IE= 0，即时MSI仍让可以发送。

如果中断域被配置为直接传送模式，寄存器genmsi就是只读的0

中断目标（target[1] - target[1023])

如果在中断域中如果中断源i是没有实现的，那么target[i]是只读的0。如果源i是实现了的，target[i]决定了决定了这个中断源所传送的目标hart。具体对于target[i]的翻译取决于domaincfg的DM部分的传送模式配置。

如果domaincfg的DM配置被修改了，那么目标寄存器target，所有实现了的中断源的目标寄存器将被设置为一个非标准化的值。

活跃源，直接转送模式

对于一个活跃中断源，如果中断域被配置为直接传送模式，那么寄存器target[i]的格式如下：

bits 31：18 Hart Index（WLRL）

bits 7: 0 IPRIO(WARL)

所有没被提及的bits都是被保留并且读出时为0

Hart Index决定了从这个源来的中断会传送到哪一个hart。

IPRIO（Interrupt Priority）决定了中断源所对应的特权级。这个部分是一个WARL 的无符号证书，长度为IPRIOLEN bits，其中IPRIOLEN是一个给定APLIC上的常数，可以是1到8内的任何一个值。IPRIO可以是从1到$2^{IPRIOLEN}-1$ 的任何一个值，但是不能是0。如果一个写将target寄存器的IPRIO设置对应的值，除非写入的值0则默认写入1。

越小的数字表示越高的特权级。当中断源有相同的中断等级，则identity number小的优先级更高。

活跃源，MSI传输模式

对于活跃的实现的中断i，如果中断域被配置为了MSI传输模式，那么target寄存器的格式如下：

bits 31：18 Hart Index（WLRL）

bits 17：12 Guest Index（WLRL）

bits 10：0 EIID（WARL）

bit 11 是保留的，读出时为0。

其中Hart Index会说明中断从这个源来了之后会传向哪个核。

如果中断域时在S态，那么之后域中的hart实现了Hypervisor扩展，那么Guest Index就会时WLRL的部分可以存有一个从0到GEILEN的证书。否则，Gudex Index将是一个只读的0。对于S态的中断域，一个非0的Guest Index就是MSI传入时的中断文件号。当Guest Index是0，S态中断域中的MSIs将传送到目标hart的S态。对于一个M态中断域，Guest Index是一个只读的0，MSIs总是传到目标核的M态。

同时，target[i]寄存器的Hart Index 和 Guest Index决定了中断源i传导MSIs所使用的地址。剩下的EIID（External Interrupt Identity）指定MSIs的数据值，最终变为外部中断的次中断号。

如果中断域中的hart有IMSIC中断文件实现了N个不同的中断号，那么EIID就是一个k bits无符号整数，且$log_2^N \leq k \leq 11$。对于target寄存器的写将EIID实现的k个bits设置为32bits中对应的k bits部分。

中断pending bits的精确作用

如果尝试通过写中断域中的寄存器设置或者清除中断源的pending bit，是否成功是不能确定的，这还取决于对应的源模式，中断域的中断传递模式，还有中断源的输入值。在给定源模式下，下面列出了所有设置和清除pending bit的情况：

1. 如果源模式是Detached：
    - 只有在写setip或者setipnum寄存器时才会将pending bits置1
    - 如果中断已经被取得了或者已经作为MSI转发了再或者写了一个值到in_clrip或者clripnum寄存器中，则相应的bit会被清空。
2. 如果源模式是Edge1或者Edge0：
    - 一个输入从低到高的上升沿将会对pending bit置1，或者直接像setip或setipnum寄存器中写入一个值。
    - pending bit会在中断被获取或者MSI已经做了转发是被清空，再或者对in_clrip或clripnum寄存器写入了一个值。
3. 如果源模式是Level1 或者 Level0并且中断域被配置为了直接传输模式：
    - 外面的输入值为高，则将pending bit置1。但是pending bit 不能通过对setip或者setipnum进行写来置位。
    - pending bit在输入值变为低电平时就会清空，但是在APLIC中获取这个中断或者写in_clrip或clripnum寄存器不能对pending bit进行清空。
4. 如果源模式是Level1或者Level0，并且处于MSI传输模式
    - pending bit 在有一个上升沿到来时会被置1。当输入是高电平时，对于setip或者setipnum寄存器进行写仍然会对pending bit进行置1，但是低电平时不能。
    - 如果输入为低电平时对应的pending bit将被清空，如果中断被作为MSI进行转发，再或者对in_clrip或者clripnum寄存器进行写都会进行清空操作。

当中断域的模式是直接传输模式，源模式是低电平敏感的，那么pending bit总是输入值的拷贝。即使再MSI传输模式下，在处于低电平敏感模式时，当输入为低电平时，pending bit一定不会被置位。

除了上面的规则，对于sourcecfg寄存器的一个写会将源的中断pending bit置为1。

APLIC的中断直接传输模式

当一个中断域的传输模式为直接传输模式，APLIC会将中断通过专门的信号传输到hart里，往往就是直连线的方式。在这种情况下，中断域中的MMIO控制区域的尾部会有一个IDC结构数组，每一个IDC都会对应一个hart index。第一个IDC结构是专门给0号，第二个给1号，以此类推。

IDC结构

每一个IDC结构都是32 bytes的（32byte自然对齐）并且格式定义如下：

offset size register name

0x00 4bytes idelivery

0x04 4bytes iforce

0x08 4bytes ithreshold

0x18 4bytes topi

0x1c 4bytes claimi

如果在中断域中存在对应的hart，那么这些寄存器应该实现为只读的0。

一个专门的APLIC可能构建用于支持最大数量的harts而不知道这个系统会使用哪些hart。在这种情况下，对于没有使用的hart index，APLIC可以使用IDC结构且不是只读的0，只需要不将其连接到任何hart上。

中断传输使能（idelivery）

idelivery是一个WARL寄存器，它可以控制是否中断是否会被传输到对应的hart上，这样hart中 的mip CSR寄存器就会显示对应的pending interrupt。现在idelivery只定义了两个值：

0 = interrupt delivery is disabled

1 = interrupt delivery is enabled

如果一个IDC结构对应一个不存在的hart，则将idelivery设置为1，这样就不会传递中断。

Interrupt force （iforce）

iforce 是一个WARL寄存器，可以用于测试。其中的值只允许是0或者1。将iforce设置为1，当domaincfg.IE = 1 并且 interrupt delivery是使能的，这时就会强制一个中断被触发。当topi寄存器中式0时，会伪造一个外部中断。

一旦对于claimi寄存器的读取返回的中断实体号为0（表示这是一个假的中断），iforce会被自动清0。

中断使能阈值（ithreshold）

ithreshold 是一个WLRL寄存器决定可以发送一个特定核的最小中断优先级。ithreshold寄存器实现正好了IPRIOLEN个bits，因此可以处理的优先级号为从0到$2^{IPRIOLEN} - 1$。

如果ithreshold寄存器中是一个非0值P，中断优先级为P或者大于P不能将中断信号发送到hart中，无视它们配置中的interrupt使能bit。当ithreshold是0，所有的使能中断源都可以向hart发送中断。

最高优先中断（topi）

topi是一个只读的寄存器，它的值表示当前最高特权级的pending并且使能的中断，同时中断的优先级需要超过ithreshold所要求的优先级。

如果没有这个hart上没有中断pending并且使能那么对于topi寄存器的读将返回0，或者没有中断优先级低于阈值时也会返回0。除此之外，读取topi的返回值如下：

bits 25: 16 Interrupt identity(source number)

bits 7:0 Interrupt priority

所有的其它bits都是0

这里的中断实体就是目标hart上的次中断号。

对于topi的写会被忽略。

获取最高优先级中断（claimi）

claimi中有和topi一样的值。当其中的值不为0时，对claimi进行读取会清空pending bit并报告中断。对于claimi寄存器的读取还会将iforce寄存器归0。

对于claimi寄存器的写将被无视。

中断传递与处理

当APLIC的中断域被配置为直接传递模式，在下面任何一种情况，APLIC将发出一个外部中断信号而不是MSI：

(a) hart 没有IMSIC

(b) IMSIC中断文件的eidelivery寄存器设置在0x40000000的位置。

对于M态中断域，从APLIC来的中断信号体现为一个MEIP到CSR的mip寄存器中。

对于S态中断域，从APLIC来的中断信号体现为一个SEIP到CSR的sip寄存器中。

每一个中断信号可以被任意地延迟。

在APLIC中，会从domaincfg的IE部分和中断域对应IDC结构中提取中断信号。如果domaincfg.IE = 0或者idelivery = 0, 则中断信号不会被触发。当domaincfg.IE = 1并idelivery被使能，中断信号就会在iforce或者topi不等于0的时候触发。

由于hart和APLIC通讯存在延迟，这就可能发生一个外部中断已经被获取了，但是hart上并没有一个pending并且enable的中断，而hart已经发起了一次对iclaim寄存器的读。在这种情况下，中断实体就会是0，造成一次APLIC的假中断。可移植的软件必须准备好应对假中断。处于测试目的，可以通过设置IDC中的iforce为1来触发。

一个APLIC的外部中断的trap handler大概如下所示：

```
1. 存储处理器上下文
2. i = 读取IDC数据结构中的claimi寄存器
3. i = i >> 16
4. 调用外部中断i的处理函数
5. 恢复处理器上下文
6. 从Trap返回
```

为了处理假中断，外部中断0将不会做任何操作。

通过MSI进行中断传递

在MSI传输模式下，中断域中通过MSI传来一个中断信号到hart。

只有当源的pending和使能bit是1并且domaincfg的IE为1，这个源的中断才会以MSI的形式发出。如果MSI被送出，这个中断源的pending bit就会被清空。

MSI的目标地址和数据

为了通过MSI传递中断，APLIC需要直到每个hart的MSI目标地址。对于一个给定系统，这些地址固定的并写死到APLIC中。然而，一些APLIC可能需要通过软件进行一些MSI目标地址的配置。在这种情况下，根中断域中的寄存器mmsiaddrcfg、mmsiaddrcfgh、smsiaddrcfg、smsiaddrcfgh可以用于配置MSI的地址。当然可以自己定制标准外的配置方法。如果MSI的目标地址可以通过软件配置，这些必只能由合适的特权软件进行配置，或者，在系统启动早期进行配置。

对于一个M态中断域，如果MSI的目标地址可以通过mmsiaddrcfg和mmsiaddrcfgh进行配置，之后中断源i来的MSI可以通过如下方式计算地址：

$$
\begin{aligned}
&g = (\text{Hart Index}\texttt{>>}\text{LHXW})\ \texttt{\&}\ (2^{\text{HHXW}}-1)\\
&h = \text{Hart Index}\ \texttt{\&}\ (2^{\text{LHXW}}-1)\\
&\text{MSI address} = \big(\text{Base PPN}\ \texttt{|}\ (g\texttt{<<}(\text{HHXS}+12))\ \texttt{|}\ (h\texttt{<<}\text{LHXS})\big)\texttt{<<}12
\end{aligned}
$$

假设IMSIC中断文件的排布和AIA第三章第6节说明的排布一致，这里的g就代表了hart group，h代表了group中目标hart的编号。HHXS（High Hart Index Shift）、LHXS（Low Hart Index Shift）、HHXW（High Hart Index Width）和 LHXW（Low Hart Index Width）。

$$
\begin{aligned}
&g = (\text{machine-level hart index}\texttt{>>}\text{LHXW})\ \texttt{\&}\ (2^{\text{HHXW}}-1)\\
&h = \text{machine-level hart index}\ \texttt{\&}\ (2^{\text{LHXW}}-1)\\
&\text{MSI address} = \big(\text{Base PPN}\ \texttt{|}\ (g\texttt{<<}(\text{HHXS}+12))\ \texttt{|}\ (h\texttt{<<}\text{LHXS})\ \texttt{|}\ \text{Guest Index}\big)\texttt{<<}12
\end{aligned}
$$

对于S态的中断域，如果MSI的目标地址已经被根中断域中的smsiaddrcfg等寄存器决定了，之后对于中断源i的target[i]需要先转为M态中断域中相同hart的hart下标。之后MSI使用M态的hart下标进行计算，公式如上，LHXW等还是来自smsiaddrcfg和smsiaddrcfgh寄存器。

MSI传送的数据是target[i]中的EIID 0扩展至32位。一个MSI的32位写总是小端序的，这里无视domaincfg中BE配置。

对于中断源的输入为电平敏感型的考虑

一旦电平敏感型中断被作为MSI发出，APLIC就会将中断源的pending bit清空，之后无视这个源，直到这个输入被取消。MSI被送出之后将pending bit给清空显然足够阻止从APLIC重复发出同一个中断的MSI信号。然而，在一次中断服务后，APLIC的中断直连线可能还保持了触发状态，经过中断的pending bit已经被清除了，只要软件没有干预就会一致保持这个状态。如果中断服务例程结束了，之后从这个源来的中断就可能得不到响应。

为了防止中断的丢失，对于电平敏感型的中断需要中断服务例程结束之间做如下的操作：

第一个选项就是测试APLIC的中断直连线是否仍然保持了触发状态，可以通读取in_clrip寄存器来得知。如果进入的中断被持续的触发，那么中断服务的代码可能会不断地发现和处理一个中断。一旦发现这个直连线没有被触发，这个中断服务例程就可以结束了，之后再触发一个新的中断就会导致pending bit被置位生成一个新的MSI。

第二个选项就是中断服务程序在结束之前在中断域的setipnum寄存器中写入源的实体号。如果中断源仍处于触发状态，这会导致中断pending bit被设置为1；反之则不会。

同步hart和APLIC

当一个APLIC向一个hart发送MSI信号是，在MSI发出到IMSIC观察到这个MSI会有一个不确定的延迟。从结果上，在通过写APLIC的寄存器修改APLIC的配置后，在写之前会看到APLIC上不断发出MSI信号。

如果有事需要知道之后这个MSIs不会再到来。例如，如果这个hart被关机了，所有指向这个hart的中断需要重新导向其他的hart，这可能涉及到重新配置APLIC。即使再APLIC重新配置后，直到没有MSI传向这个hart，这个hart才可以安全关机。

genmsi寄存器的存在会允许软件来决定是否之前所有的MSIs已经到达了一个hart。为了使用genmsi达到这一目的，软件可以在IMSIC的中断文件专门设置一个外部中断号用于APLIC的同步。假设这里存在多个harts，一个APLIC的genmsi寄存器应该被通过标准的互斥锁进行保护。接下来就是同步APLIC和一个hart的流程：

1. 在hart的IMSIC上，清空次中断号为i的pending bit，其中i专门用于APLIC的同步
2. 获取APLIC genmsi 寄存器的共享锁
3. 写如genmsi寄存器来向hart产生一个MSI
4. 不断读取genmsi直到Busy bit变成了false
5. 释放genmsi的锁
6. 不断读取hart的IMSIC次中断号i的pending bit直到发现它已经被置位。

其中4和6的循环往往非常块，只需要一到两次的尝试。当这个过程被完成，之前所有APLIC来的所有MSIs一定都已经到达hart的IMSIC。