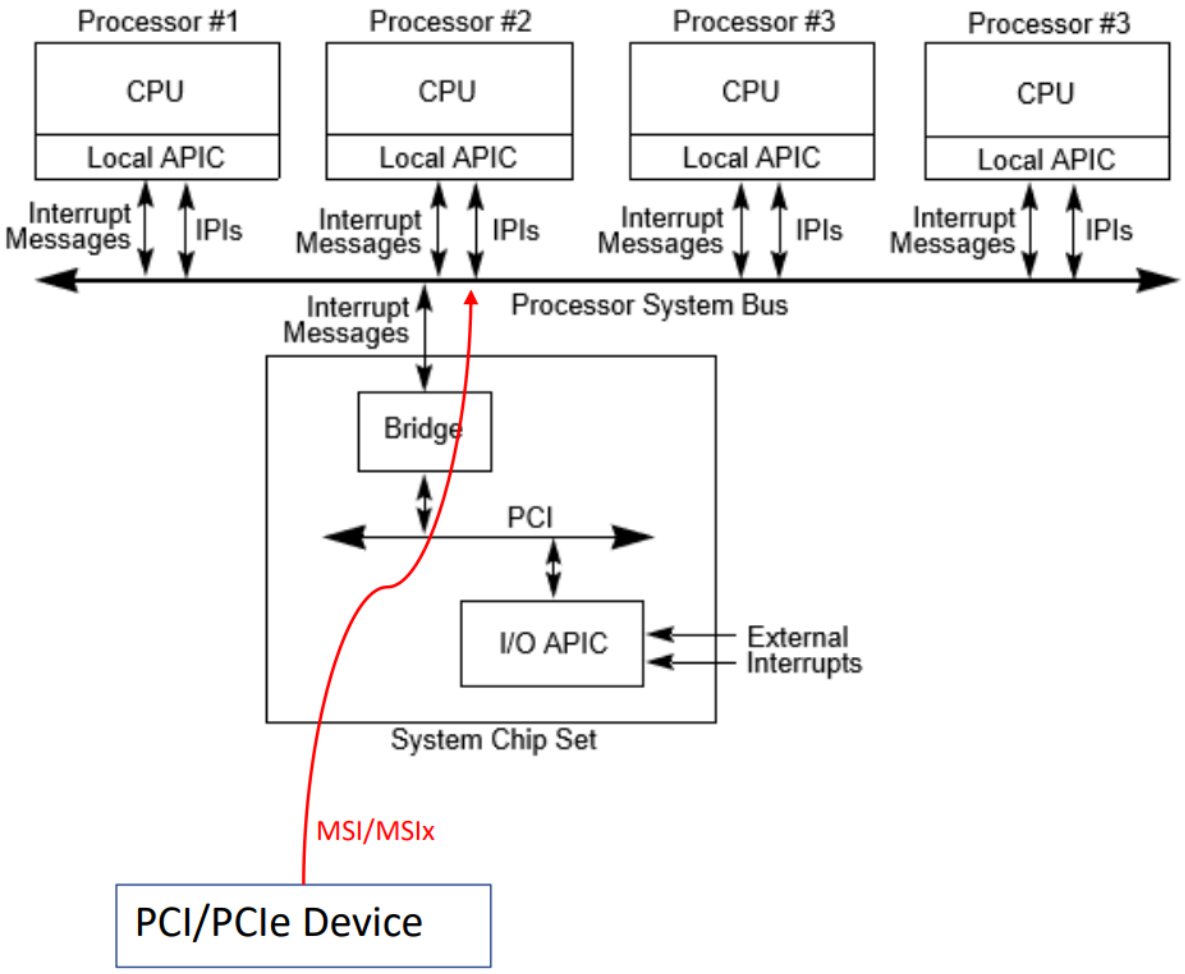
深入理解 MSI/MSI-X 中断和中断虚拟化

本文将以 Intel 平台为背景，首先介绍 MSI/MSI-X 的背景知识，然后 介绍 MSI/MSI-X 的相关虚拟化技术。

1. MSI/MSI-X 基础知识

MSI/MSI-X 的功能是设备的中断路由。如下图所示，PCI/PCIe 设备通过 MSI/MSI-X，无需经过 I/O APIC，可以直接将中断发送到目的 LAPIC 中。



MSI capability 首先在 PCI 2.2 中引入，后来在 PCI 3.0 中得到增强，允许单独屏蔽每个中断。 PCI 3.0 还引入了 MSI-X capability， 与 MSI 相比，它可以让每个设备支持更多的中断，并允许独立配置中断。

1.1 MSI

MSI Capability Structure 的格式如下：

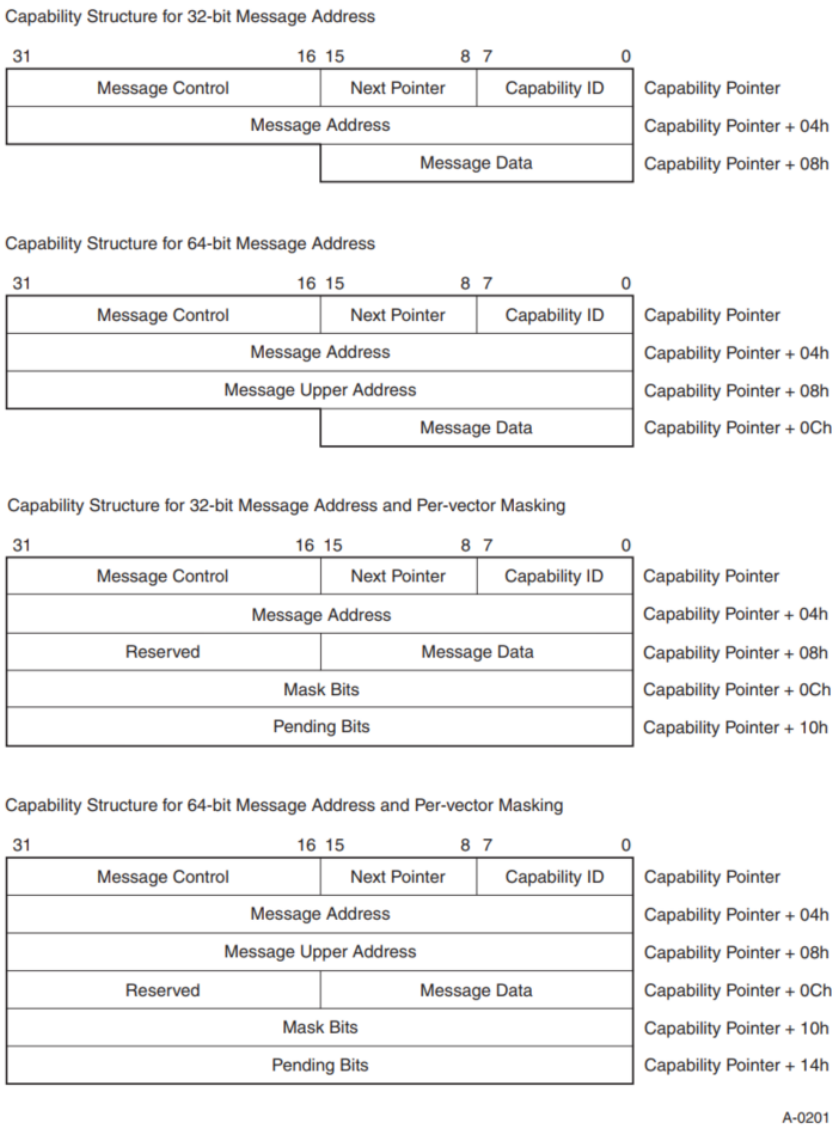


Figure 6-9: MSI Capability Structures

根据 Message Control 的取值不同，MSI Capability Structure 可以有四种格式，其中两种使用 32 位地址，两种使用 64 位地址。Message Address 即 MSI Transaction 写入的地址，Message Data 即 MSI Transaction 写入的数据。

Message Address 和 Message Data 的格式由各个架构自己定义，对于 x86 架构，Message Address 就是位于 0xFEE00000 - 0xFEEFFFFF 的一块地址区域（和 LAPIC 的地址范围重合），Message Address/Data 的格式定义于 Intel SDM Vol.3 的第 10 章。

1.1.1 Message Address

Message Address 寄存器（低 32 位）的格式如下图所示：

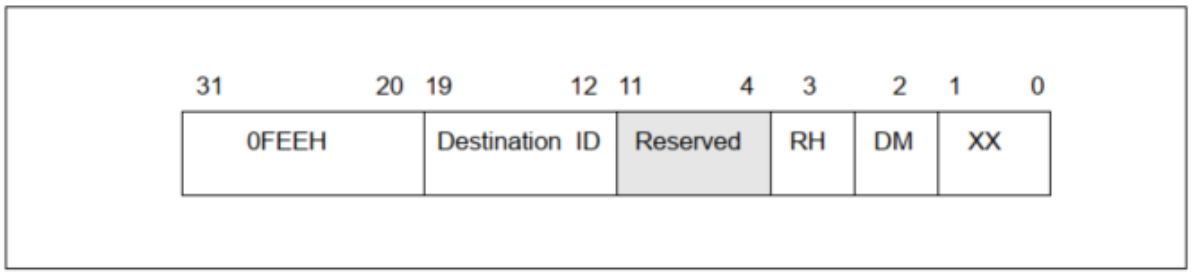


Figure 10-24. Layout of the MSI Message Address Register

为了方便阐述，我们只介绍 Destination ID 这一 field。

- 第 12-19 位为 Destination ID，决定了 MSI 发送的目标 CPU(s)
- 第 20-31 位必须为 0xFEE

1.1.2 Message Data

Message Data 寄存器的格式如下图所示：

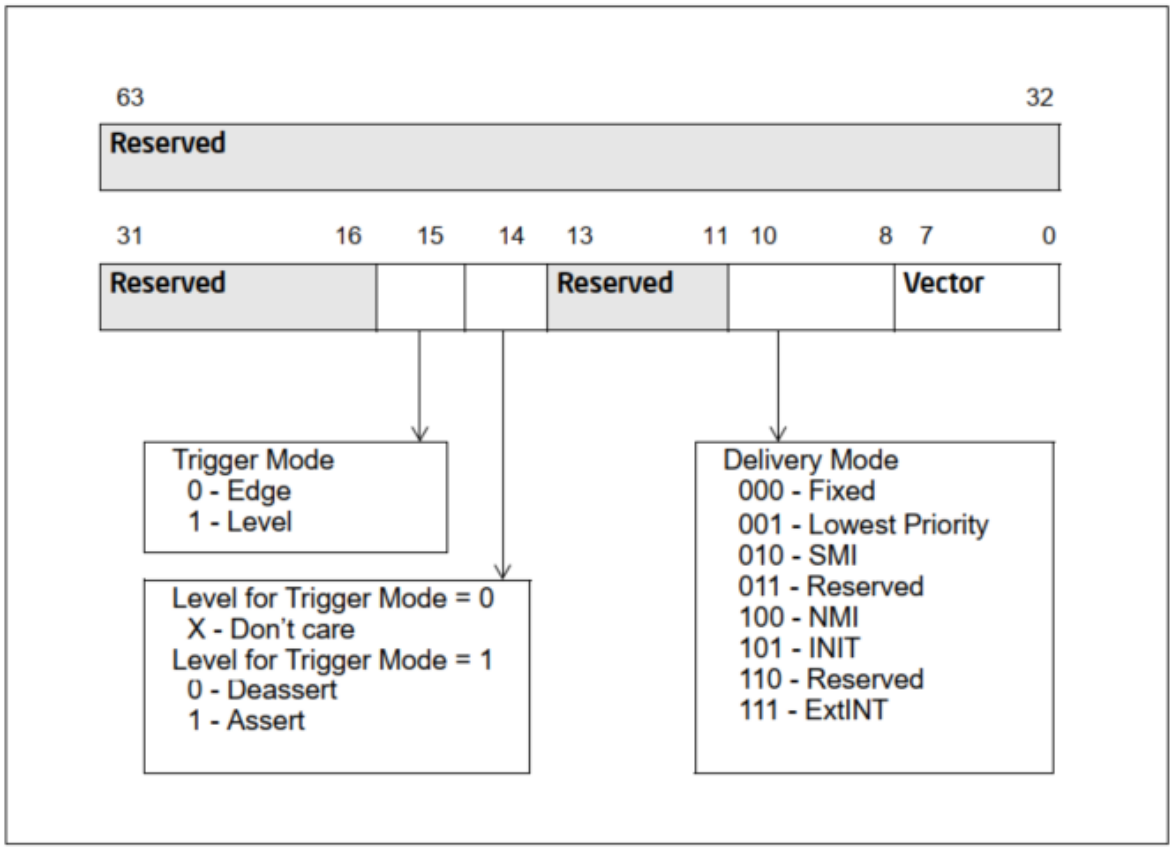


Figure 10-25. Layout of the MSI Message Data Register

为了方便阐述，我们只介绍 Vector 这一 field。

- 第 0-7 位为 Vector，即目标 CPU 收到的中断向量号。

1.1.3 Message Control

- 第 0 位为 MSI Enable，开机时默认为 0，表示禁用 MSI
- 第 1-3 位为 Multiple Message Capable，表示设备支持多少 Vector，设备遇到不同的事件，可以使用不同的 Vector 来产生中断
  - 其取值为 X，则支持 2^X 个 Vector，X 最大为 5，即允许使用 32 个不同的 Vector
  - 设备使用的 Vector 号受到 Message Data 的限制，它只能改变 Message Data 的末 X 位来变更 Vector，从而产生 2^X 个不同的 Vector
- 第 4-6 位为 Multiple Message Enable，表示软件允许设备使用多少个 Vector，取 X 则只允许 Device 改动 Message Data 的末 X 位，即使用 2^X 个 Vector
- 第 7 位为 64 Bit Address Capable，决定了 Message Address 是否为 64 位
- 第 8 位为 Per-vector Masking Capable，决定了是否有 Mask Bits 和 Pending Bits 这两个域

1.1.4 Mask Bits and Pending Bits

上面已经提到 MSI 允许设备使用多个 Vector，这里 Mask Bits 和 Pending Bits 的每个 Bit 就对应于一个 Vector（若无该 Vector 就取 0）：

- Mask Bits 由软件设置，其某个位取 1，表示禁止发送对应的 Vector
- Pending Bits 由设备设置，当某个 Vector 被 Mask，而设备又产生了一个中断从而生成该 Vector 的 MSI Message 时，Pending Bits 中该 Vector 对应的位就会置为 1，当 Mask 取消后，设备就会立即发送该 Vector 对应的 MSI Message，并将该位清零

1.2 MSI-X

MSI-X Capability Structure 的格式如下：

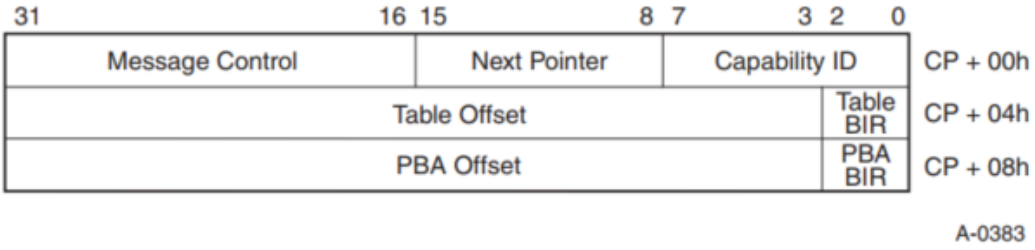


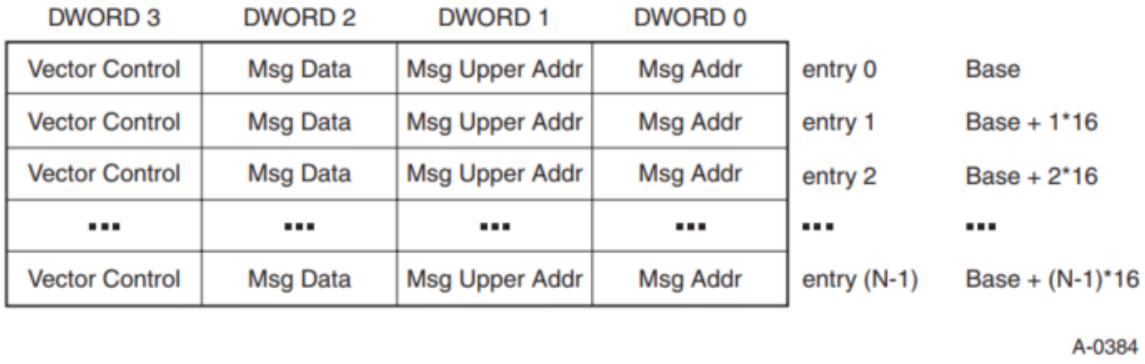
Figure 6-10: MSI-X Capability Structure

可以看到它实际上将真正的配置移到了 MSI-X Table 和 PBA（Pending Bit Array）中去了，这两个结构都位于 MMIO 而不是 PCI Configuration Space。我们通过最低 3 位（BIR, BAR Indicator Register）指定一个 BAR Region（必须是 MMIO Region 不能是 I/O Region），其余位则用于指定在该 Region 中的偏移量（Offset 是 32 位而不是 29 位的，末 3 位取 0），从而可以索引到 MSI-X Table 和 PBA。

Message Control 的格式如下：

- 第 0-10 位为 Table Size，表示 MSI-X Table 的大小，若 MSI-X Table 有 N 项，则其取值为 N-1
- 第 14 位为 Function Mask，取 1 表示屏蔽该 Function 的中断，取 0 则根据 Per-vector Mask 只屏蔽部分 Vector
- 第 15 位为 MSI-X Enable，开机时默认为 0，表示禁用 MSI-X

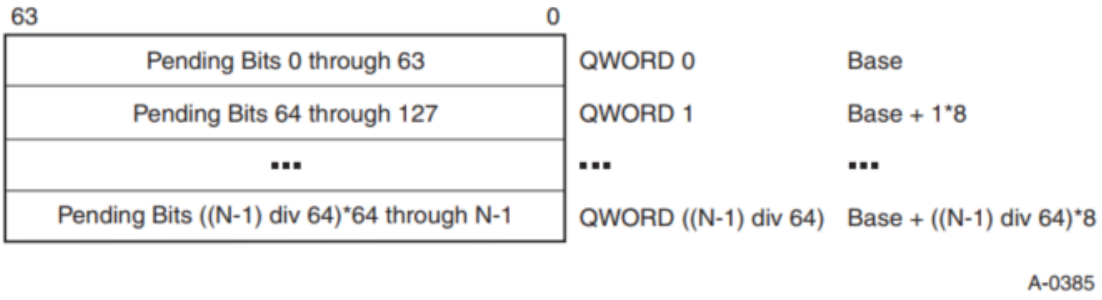
MSI-X Table 和 PBA 的格式如下：



DWORD 3	DWORD 2	DWORD 1	DWORD 0		
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 0	Base
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 1	Base + 1*16
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 2	Base + 2*16
...	...	...	...	...	...
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry (N-1)	Base + (N-1)*16

A-0384

Figure 6-11: MSI-X Table Structure



63 ... 0		
Pending Bits 0 through 63	QWORD 0	Base
Pending Bits 64 through 127	QWORD 1	Base + 1*8
...	...	...
Pending Bits ((N-1) div 64)*64 through N-1	QWORD ((N-1) div 64)	Base + ((N-1) div 64)*8

A-0385

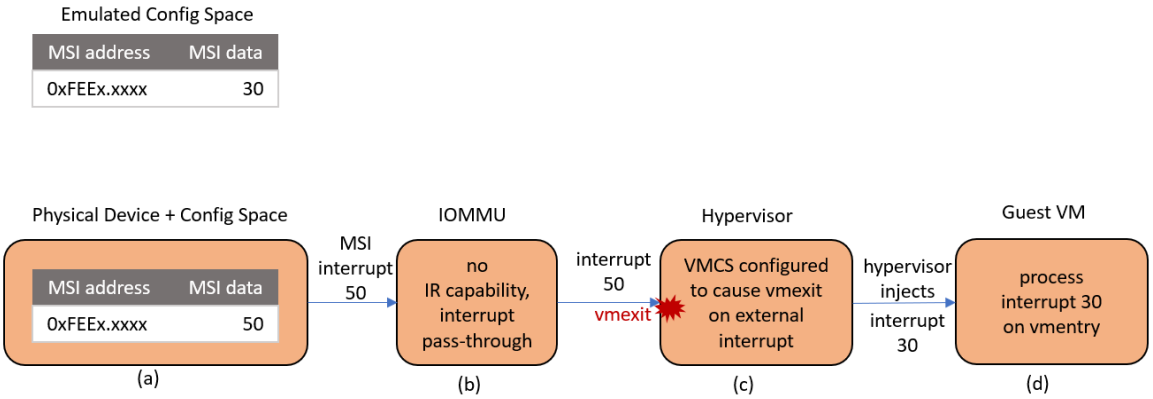
Figure 6-12: MSI-X PBA Structure

MSI-X Table 每个条目大小为 16 字节，最多 2048 个条目，也就是说设备最多可以为 2048 个不同事件分配不同的 Vector。设其有 N 个条目，则 PBA 就是一个有 N 个 Bit 的 bitmap，每 64 位为一组，最后一组不足 64 位的补零。MSI-X Table 条目的 Vector Control 项仅第 0 位有效，其余均为保留位，它的含义是屏蔽该条目，禁止发送对应的 MSI Message，屏蔽后积压的中断会体现在 PBA 中的对应 Bit 上。

2. MSI/MSI-X 虚拟化

本节将介绍设备 Pass-thru 时，MSI/MSI-X 的虚拟化情况。MSI 的配置信息位于 PCI Configuration Space，MSI-X 的配置信息位于 MMIO BAR 中，当虚拟机访问 MSI 或者 MSI-X 的寄存器时，都需要发生 VM Exit。为了阐述的方便，本节只介绍 MSI 的虚拟化情况。

2.1 Without interrupt remapping

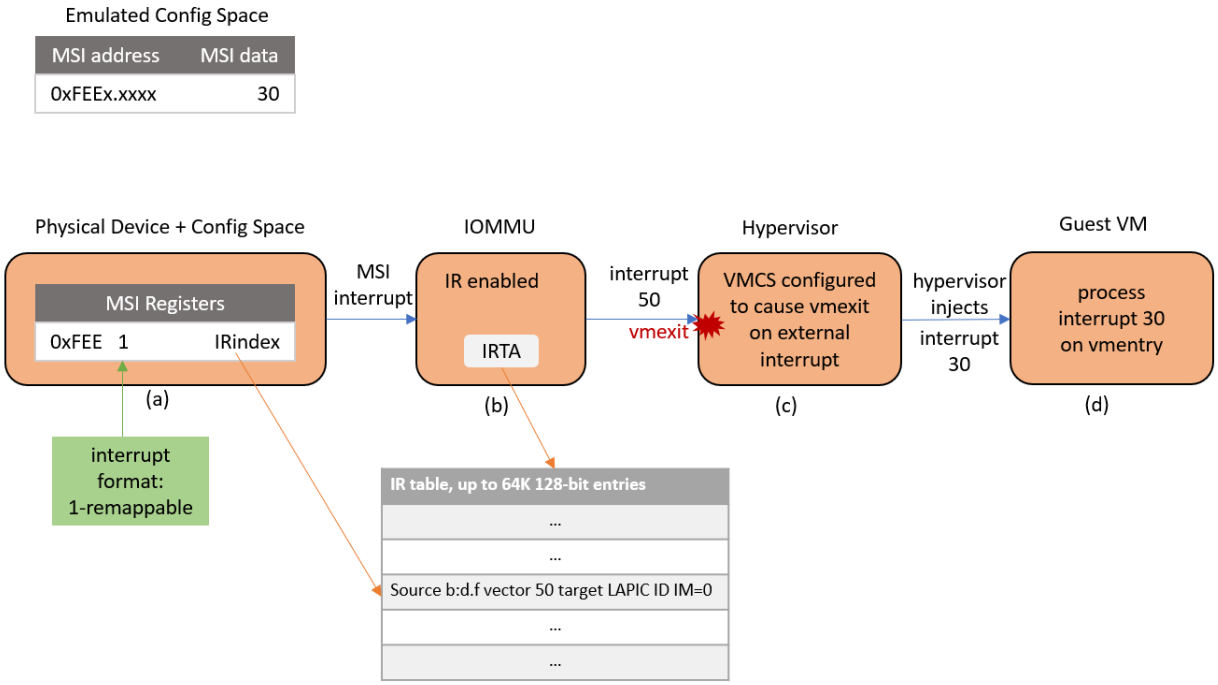


上图中的过程如下：

1. 当 guest 配置 MSI address 和 MSI data 时，hypervisor(hv)会 trap，解析相关 field，得到 virtual interrupt 的 vector 为 30，而对应的 physical interrupt 的 vector 为 50。这样 hv 会记录 physical interrupt 到 virtual interrupt 的映射(vector 50 -> vector 30)。
2. 设置 VMCS(configured to cause vmexit on external interrupt)。当物理 CPU 收到 vector 为 50 的中断时，会发生 VM-Exit。
3. hv 处理 VM-Exit，根据 physical interrupt 到 virtual interrupt 的映射(vector 50 -> vector 30)关系，为 guest 注入 vector 为 30 的 interrupt。
4. VM-Entry 到 guest VM，process vector 为 30 的 interrupt。

这样的实现无法避免 BDF spoofing 攻击。BDF spoofing 是一种针对 VT-d 的硬件攻击，其中恶意设备生成带有欺骗 BDF 地址的 PCIe 数据包。为了提高安全性，我们将引入 VT-d 的 interrupt remapping 技术。

2.2 Interrupt remapping



上图中的过程如下：

1. 当 guest 配置 MSI address 和 MSI data 时，hypervisor(hv)会 trap，解析相关 field，得到 virtual interrupt 的 vector 为 30，而对应的 physical interrupt 的 vector 为 50。这样 hv 会记录 physical interrupt 到 virtual interrupt 的映射(vector 50 -> vector 30)。
2. 分配一个 IRTE 并且按照 IRTE 的格式要求填好 IRTE 的每个属性。如：B:D.F, Vector(50), 运行目标 vCPU 的物理 CPU 的 LAPIC ID。
3. 按照 Remapping format 的格式对 MSI 进行编程。如：将 Interrupt Format 置 1，设置中断的 interrupt_index。
4. 设置 VMCS(configured to cause vmexit on external interrupt)。当物理 CPU 收到 vector 为 50 的中断时，会发生 VM-Exit。
5. hv 处理 VM-Exit，根据 physical interrupt 到 virtual interrupt 的映射(vector 50 -> vector 30)关系，为 guest 注入 vector 为 30 的 interrupt。
6. VM-Entry 到 guest VM，process vector 为 30 的 interrupt。

Interrupt remapping 可以避免 BDF spoofing 攻击。因为 IRTE 中设置了 bdf 号，恶意的硬件就算生成数据包，IOMMU 硬件会发现恶意硬件的 bdf 号与 IRTE 中设置了 bdf 号不一致，从而会拒绝产生中断。

物理 MSI/MSI-X 的配置细节如下：

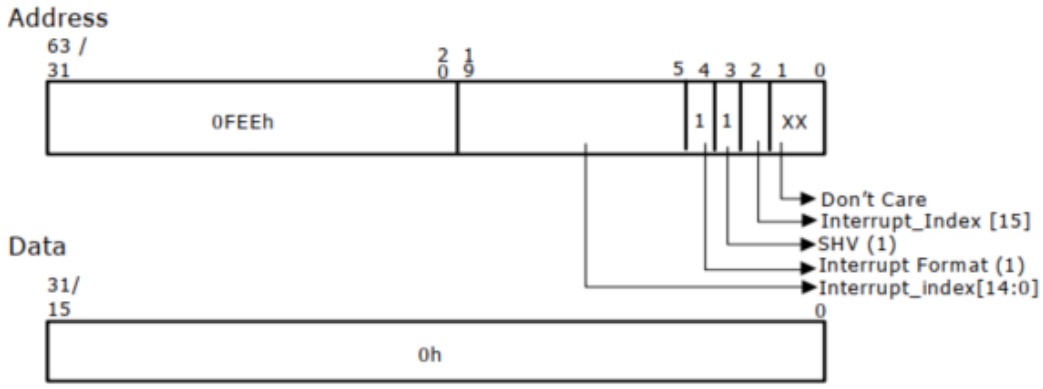


Figure 5-18. MSI-X Programming

这里只解析几个关键的 field。

- 第 2 位和第 5-19 位构成了 Interrupt_Index 以用来索引 IRTE
- 第 4 位必须为 1，以表示这个中断是 Remappable 格式

Remapped Interrupts 的 IRTE 目格式如下:

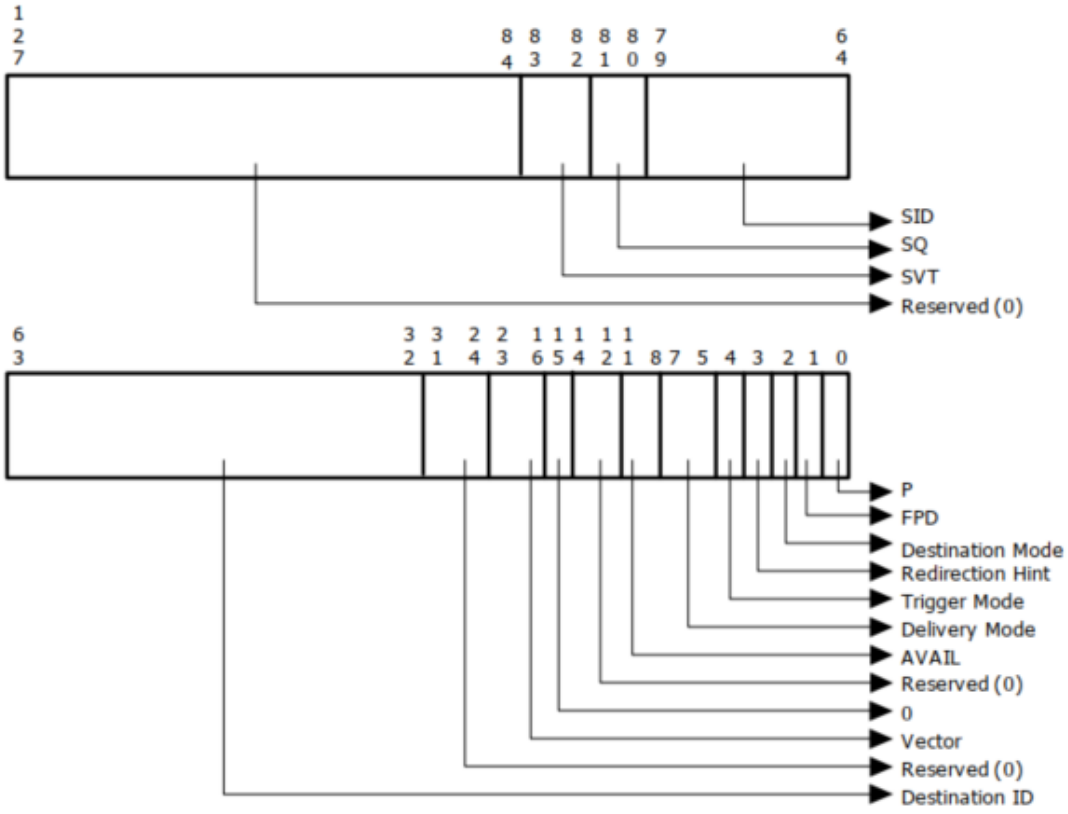


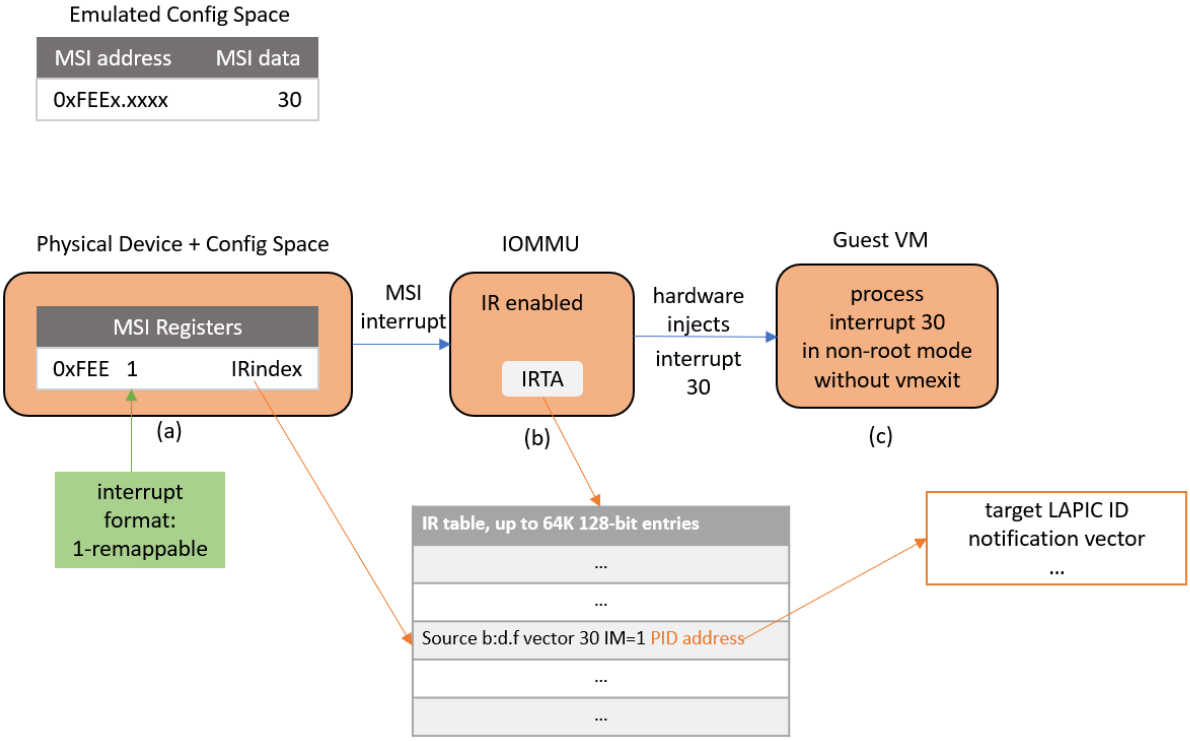
Figure 9-41. Interrupt Remap Table Entry Format for Remapped Interrupts

这里只解析几个关键的 field。

- 第 0 位 present 位，表示改 IRTE 是否有效
- 第 15 位为 IM（IRTE Mode）位，此时必须为 0，表示是 interrupt remapping 而不是 interrupt posting。interrupt posting 的内容会在下一小节中详细介绍
- 第 16 到 23 位表示往目标 LAPIC 发送的物理 vector
- 第 32 到 63 位表示发送的目标 CPU(s)
- 第 64 到 79 位表示设备的 bdf 号

Interrupt remapping 的步骤 5（hv 处理 VM-Exit，根据 physical interrupt 到 virtual interrupt 的映射(vector 50 -> vector 30)关系，为 guest 注入 vector 为 30 的 interrupt）需要将运行在 non-root 下 vCPU trap 下来从而注入 virtual interrupt。由于 VM Exit 引起的 trap 会影响性能，posted interrupt 应运而生。在 posted interrupt 机制中，即使 vCPU 运行在 non-root 下，硬件会直接注入 virtual interrupt，无需产生 VM Exit。

2.3 Interrupt posting



上图中的过程如下：

1. 当 guest 配置 MSI address 和 MSI data 时，hypervisor(hv)会 trap，解析相关 field，得到 virtual interrupt 的 vector 为 30。
2. 分配一个 IRTE 并且按照 IRTE 的格式要求填好 IRTE 的每个属性。如：B:D.F, virtual vector(30),PID 的 address。PID 中会设置好目标物理 LAPIC 等信息。
3. 按照 Remapping format 的格式对 MSI 进行编程。如：将 Interrupt Format 置 1，设置中断的 interrupt_index。
4. 硬件会往运行在 non-root mode 下的 vCPU 直接注入 vector 为 30 的 interrupt。vCPU 会 process vector 为 30 的 interrupt。

Posted Interrupts 的 IRTE 目格式如下:

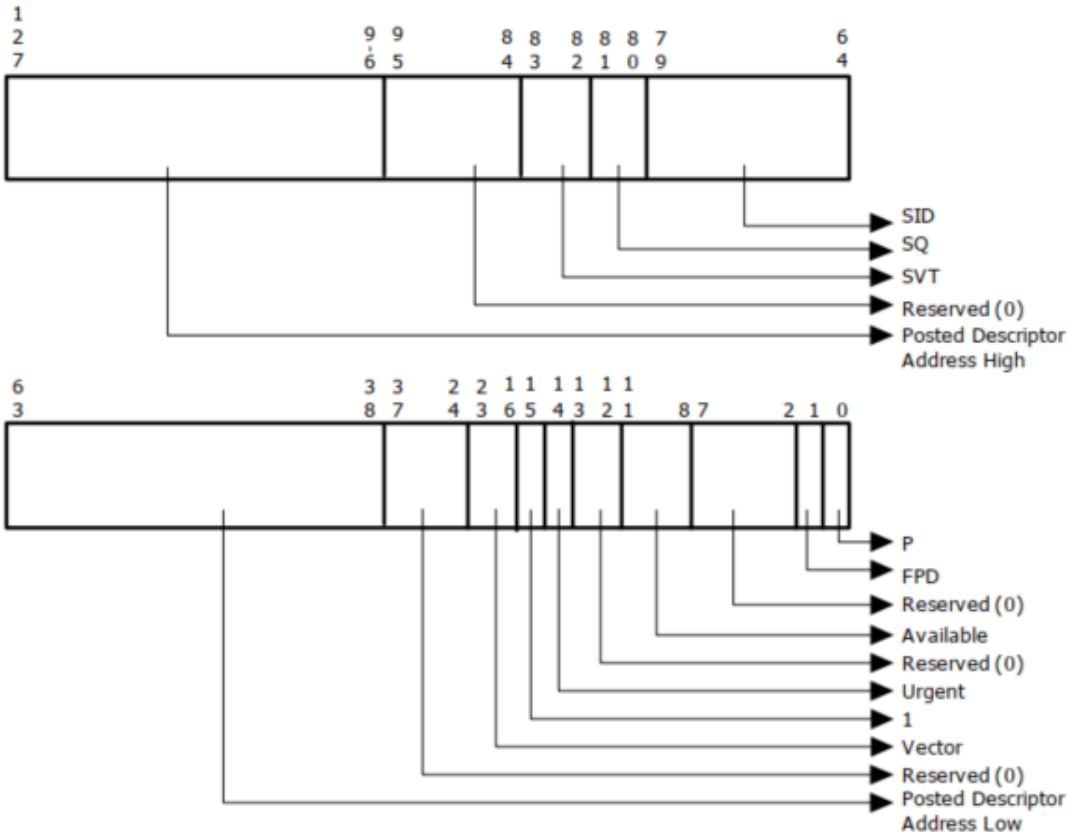


Figure 9-42. Interrupt Remap Table Entry Format for Posted Interrupts

这里只解析几个关键的 field

- 第 0 位 present 位，表示改 IRTE 是否有效
- 第 15 位为 IM（IRTE Mode）位，此时必须为 1，表示是 interrupt posting 而不是 interrupt remapping
- 第 16 到 23 位表示要往 PID(Posted Interrupt Descriptor)中设置的 virtual vector
- 第 38 到 63 位和第 96 到 127 位构成了 PID 的物理地址
- 第 64 到 79 位表示设备的 bdf 号

PID 的格式如下:

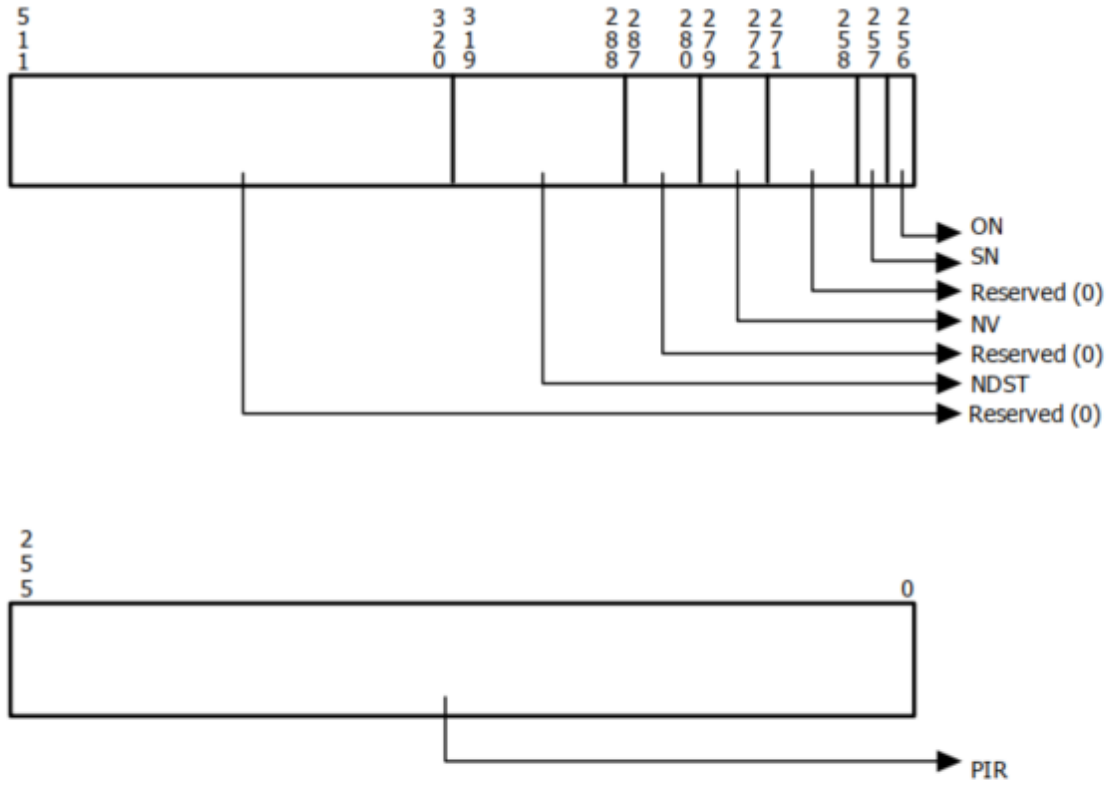


Figure 9-43. Posted Interrupt Descriptor Format

这里只解析几个关键的 field

- 第 0 位到 255 位表示 PIR
- 第 272 到 279 位表示 Notification Vector

Posted Interrupt 的细节知识不在本文讨论的范畴之内，感兴趣的读者请参考 SDM Vol3 29.6 POSTED-INTERRUPT PROCESSING 一节。

参考资料：

1. Intel SDM
2. PCI LOCAL BUS SPECIFICATION, REV. 3.0
3. Intel® Virtualization Technology for Directed I/O
4. https://tcbbd.moe/ref-and-spec/pci/

Signed off by liujunming

5. https://invisiblethingslab.com/resources/2011/Software%20Attacks%20on%20Intel%20VT-d.pdf

Signed off by liujunming