	Lane No modulo 8 = 0	Lane No modulo 8 = 1	Lane No modulo 8 = 2	Lane No modulo 8 = 3	Lane No modulo 8 = 4	Lane No modulo 8 = 5	Lane No modulo 8 = 6	Lane No modulo 8 = 7
Symbol 0	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 1	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 2	55h	00h	FFh	FFh	55h	FFh	FFh	FFh
Symbol 3	55h	00h	FFh	FFh	55h	FFh	F0h	F0h
Symbol 4	55h	00h	FFh	C0h	55h	FFh	00h	00h
Symbol 5	55h	00h	C0h	00h	55h	E0h	00h	00h
Symbol 6	55h	00h	00h	00h	55h	00h	00h	00h
Symbol 7	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}
Symbol 8	00h	1Eh	2Dh	3Ch	4Bh	5Ah	69h	78h
Symbol 9	00h	55h	00h	00h	00h	55h	00h	F0h
Symbol 10	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 11	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 12	00h	55h	0Fh	0Fh	00h	55h	07h	00h
Symbol 13	00h	55h	FFh	FFh	00h	55h	FFh	00h
Symbol 14	00h	55h	FFh	FFh	7Fh	55h	FFh	00h
Symbol 15	00h	55h	FFh	FFh	FFh	55h	FFh	00h

Key: P: Indicates the 4-bit encoding of the Transmitter preset being used.
~P: Indicates the bit-wise inverse of P.

3. Sync Header 为`01b 的 block，其 128bit 未加扰的负载构成如下：

	Lane No modulo 8 = 0	Lane No modulo 8 = 1	Lane No modulo 8 = 2	Lane No modulo 8 = 3	Lane No modulo 8 = 4	Lane No modulo 8 = 5	Lane No modulo 8 = 6	Lane No modulo 8 = 7
Symbol 0	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 1	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 2	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 3	F0h	F0h	55h	F0h	F0h	F0h	55h	F0h
Symbol 4	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 5	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 6	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 7	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}
Symbol 8	00h	1Eh	2Dh	3Ch	4Bh	5Ah	69h	78h
Symbol 9	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 10	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 11	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 12	FFh	0Fh	0Fh	55h	0Fh	0Fh	0Fh	55h
Symbol 13	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 14	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 15	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h

Key: P: Indicates the 4-bit encoding of the Transmitter preset being used.
~P: Indicates the bit-wise inverse of P.

4. 一个 EIEOS block。

5. 32 个 Data Block，每个 Data Block 的负载为 16 个 IDL 符号（00h）的加扰。

公众号：芯片业者拍卖双

IMPLEMENTATION NOTE

First Two Blocks of the Compliance Pattern

关于 Compliance Pattern 的头两个 Block：第一个 Block 是一个非常低频的码型，用于测量 preset 设置。第二个 Block 用于说明 Lane number 和 preset 编码，以确保 Compliance Pattern 是 DC 平衡的。

每个 Data Block 中的有效载荷是该通道中加扰器的输出（即输入数据为 0b）。Sync Header 是不加扰的。当发送 EIEOS 时，加扰器被初始化。用于 LFSR 种子值的 Lane number 是取决于如何进入 Polling.Compliance 状态的。如果通过配置 Link Control 2 寄存器中的 Enter Compliance 而进入，则 Lane number 就是是分配给通道的编号，并且要在每个通道上使用的接收器通道极性就是在最近一次 LinkUp 为 1b 时使用的。如果 Lane 不是之前已配置的链路的一部分，并且对于所有其他进入 Polling.Compliance 的方法，则 Lane number 是端口分配的默认号。这些默认数字必须唯一。例如，必须为 x16 链路的每个通道分配一个唯一的 Lane number（介于 0 到 15 之间）。Compliance Pattern 的 Data Block 不构成数据流，因此在有序集块到数据块的过渡期间无需传输 SDS 有序集或 EDS Token，反之亦然。

IMPLEMENTATION NOTE

Ordered Sets in Compliance and Modified Compliance Patterns in 128b/130b Encoding

各种有序集（例如 EIEOS 和 SKP OS）遵循与当前操作数据速率相对应的有序集定义。例如，在 32.0 GT/s 数据速率下，EIEOS 为 32.0 GT/s EIEOS；在 16.0 GT/s 数据速率下，EIEOS 为 16.0 GT/s EIEOS；而在 8.0 GT/s 数据速率下，EIEOS 为 8.0 GT/s EIEOS。如 Section 4.2.7 定义，SKP 有序集是标准 SKP 有序集。

4.2.11 Modified Compliance Pattern in 128b/130b Encoding

128b/130b 编码下的 Modified Compliance Pattern 由以下的 65792 个 Block 组成：

1. 一个 EIEOS Block。
2. 256 个 Data Block，其中每个对其中的 16 个 IDL Symbol 加扰。
3. 255 组下列序列组成：
   (1) 一个 SKP Order Set。
   (2) 256 个 Data Block，其中每个对其中的 16 个 IDL Symbol 加扰。

每个 Data Block 中的有效载荷是该通道中加扰器的输出（即输入数据为 0b）。Sync Header 是不加扰的。当发送 EIEOS 时，加扰器被初始化。用于 LFSR 种子值的 Lane number 是取决于如何进入 Polling.Compliance 状态的。如果通过配置 Link Control 2 寄存器中的 Enter Compliance 而进入，则 Lane number 就是是分配给通道的编号，并且要在每个通道上使用的接收器通道极性就是在最近一次 LinkUp 为 1b 时使用的。如果 Lane 不是之前已配置的链路的一部分，并且对于所有其他进入 Polling.Compliance 的方法，则 Lane number 是端口分配的默认号。这些默认数字必须唯一。例如，必须为 x16 链路的每个通道分配一个唯一的 Lane number（介于 0 到 15 之间）。Modified Compliance Pattern 的 Data Block 不构成数据流，因此在有序集块到数据块的过渡期间无需传输 SDS 有序集或 EDS Token，反之亦然。

4.2.12 Jitter Measurement Pattern in 128b/130b

抖动测量码型由以下重复块构成：

- 01b 的 Sync Header 后跟 16 个 55h 的 Symbol 的 128 bit 未加扰有效载荷。

这会生成交替的 1 和 0 码型，用于测量发送器的抖动。

公众号：芯片业者相关权

4.2.13 Lane Margining at Receiver

本节中定义的接收器上的 Lane Margining 对于支持 16.0 GT/s 或更高数据速率的所有端口都是强制性实现的，这包括 Pseudo Ports (Retimer)。 当链路处于 L0 状态时，接收器上的 Lane Margining 使系统软件能够获取给定接收器的裕量信息。裕量信息包括从当前接收器位置的任一方向的电压和时间。对于在接收器上实现 Lane Margining 的所有端口，接收器处关于时序的 Lane Margining 是必须的，而在 16.0 GT/s 和 32.0 GT/s 和更高的数据速率时需要在接收器处对电压进行 Lane Margining 支持。

当接收到一个 Margin Command、链路以 16.0 GT/s 或更高的数据速率运行且链路处于 L0 状态时，接收器的 Lane Margining 开始。当接收到 Go to Normal Settings command，链路改变速率或链路退出 L0 或 Recovery 状态时，接收器上的 Lane Margining 结束。当超过某些错误阈值时，接收器的 Lane Margining 可选择实现为结束。对于独立采样器，当链路处于 Recovery 状态时，允许暂停接收器的 Lane Margining。

以 2.5 GT/s、5.0 GT/s 或 8.0 GT/s 运行的 PCIe 链路不支持接收器上的 Lane Margining。

软件使用每个端口（下游或上游）中的每个通道 Margining Lane Control Register 和 Margining Lane Status Register 来发送 Margin Command 并获取与端口相关联的相应接收器的裕量状态信息。对于 Retimer，获取有关接收器的能力和状态的信息的命令以及为接收器设置裕度的命令在下游方向的 Control SKP Ordered Set 中传送。目标 Retimer 接收器的状态和错误报告在上行方向的 Control SKP Ordered Set 中传送。软件通过写入 Downstream Port 中 Margining Lane Control Register 中的对应字段来控制 Retimer 接收器中的裕量。Downstream Port 还更新由链路中的 Retimer 通过 Control SKP Ordered Set 传送到其 Margining Lane Status Register 中的状态信息。

4.2.13.1 Receiver Number, Margin Type, Usage Model, and Margin Payload Fields

Downstream Port 中 Margining Lane Control Register 的四个命令字段的内容总是反映在 Downstream Control SKP Ordered Sets 中的相同字段中。在 Downstream Port 接收到的 Upstream Control SKP Ordered Set 的内容总是反映在 Downstream Port 的 Margining Lane Status Register 的相应状态字段中。下表给出了这些字段在 Control SKP Ordered Set 中的比特位信息。

Table 4-25 Margin Command Related Fields in the Control SKP Ordered Set

Symbol	Description	
	Usage Model = 0b	Usage Model ≠0b
4*N + 2	Bit 7: Margin Parity (see Table 4-23) Bit 6: ***Usage Model*** = 0b: Lane Margining at Receiver Bits [5:3]: ***Margin Type*** Bits [2:0]: ***Receiver Number***	Bit 7: Margin Parity (see Table 4-23) Bit 6: Usage Model = 1b: Reserved Encoding Bits [5:0]: Reserved
4*N + 3	Bits [7:0]: ***Margin Payload***	Bits [7:0]: Reserved

Usage Model：编码为 0b 表示 Usage Model 是接收器的 Lane Margining。此字段中的 1b 编码保留供将来使用。

如果 Usage Model 字段是 1b，则 Symbol 4N+2 的 Bit[5:0]和 Symbol 4N+3 的 Bit[7:0]为 Reserved。

在评估接收到的用于 Margin Command 的 Control SKP Ordered Set 时，如果 Usage Model 字段是 1b，所有理解不了 Usage Model = 1b 相关用法的接收器都需要忽略 Control SKP Ordered Set 中的 Symbol 4N+2 的 Bit [5:0] 和 Symbol 4N 的 Bit [7:0] +3。

IMPLEMENTATION NOTE

Potential future usage of Control SKP Ordered Set

Control SKP Ordered Set 中的 15 bit 信息的预期用途（如 Table 4-25 中所定义）是接收器的 Lane Margining。然而，单个比特位（Symbol 4N+2 的 Bit 7）保留用于接收器上 Lane Margining 之外的任何未来使用。如果将来定义了这样的用法，则该比特位将设置为 1b，其余 14 比特可以根据新的 Usage Model 的需要进行定义。或者，Symbol 4N 可以使用与 78h 不同的编码用于任何未来的使用，允许为该 Usage Model 定义 Symbol 4N+1、4N+2 和 4N+3 中的所有比特位。

公众号·芯片业者相承双

Receiver Number：接收器在 Figure 4-35 中标识。以下 Receiver Number 编码用于 Downstream Port，主要用于针对该 Downstream Port 或该 Downstream Port 下方的 Retimer 的 Margin Command：

000b Broadcast (Downstream Port Receiver and all Retimer Pseudo Port Receivers)

001b Rx(A) (Downstream Port Receiver)

010b Rx(B) (Retimer X or Z Upstream Pseudo Port Receiver)

011b Rx(C) (Retimer X or Z Downstream Pseudo Port Receiver)

100b Rx(D) (Retimer Y Upstream Pseudo Port Receiver)

101b Rx(E) (Retimer Y Downstream Pseudo Port Receiver)

110b Reserved

111b Reserved

以下 Receiver Number 编码用于 Upstream Port，用于针对该 Upstream Port 的 Margin Command：

000b Broadcast (Upstream Port Receiver)

001b Reserved

010b Reserved

011b Reserved

100b Reserved

101b Reserved

110b Rx (F) (Upstream Port Receiver)

111b Reserved

公众号·芯片业者相承双

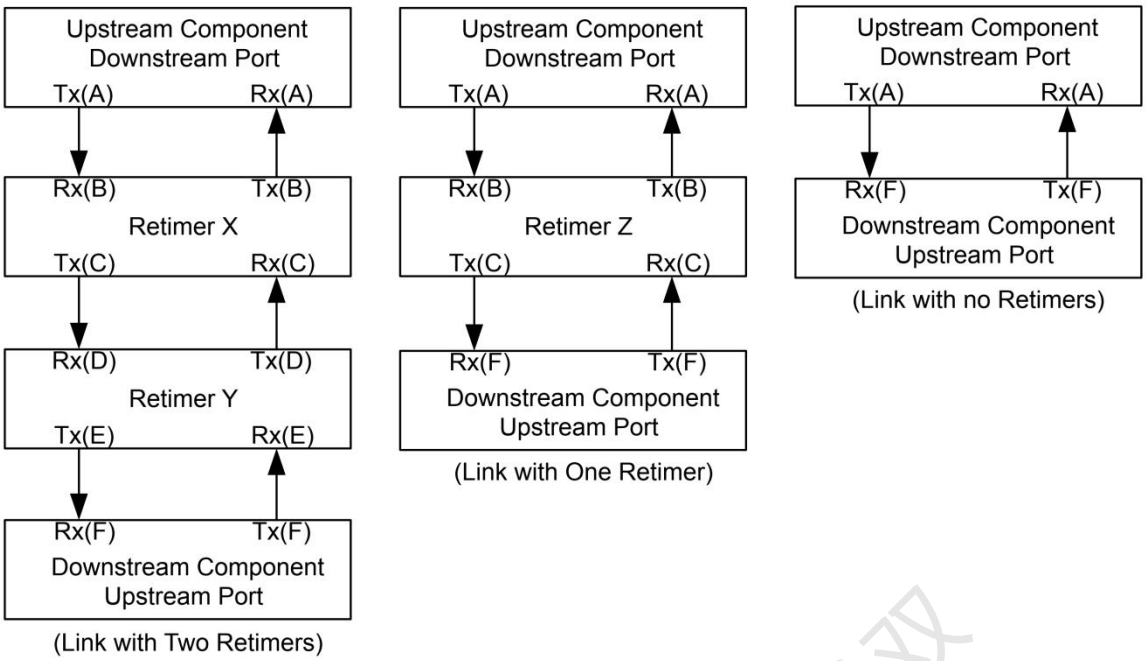

Figure 4-35 Receiver Number Assignment

Margin Type 和 Margin Payload：与 Margin Type 编码相关联的有效 Receiver Number 和 Margin Type 与特定的 Margin Payload 字段定义了用于 margining 的各种命令（称为 Margin Command）。Table 4-26 定义了有效的 Margin Command 的编码以及相应的响应，这些响应用于 Control SKP Ordered Set 以及 Margining Lane Control Register 和 Margining Lane Status Register。始终广播的 Margining Lane Status Register 将使用 Receiver Number 的广播编码，即使只有一个接收器是目标（例如，在没有 Retimer 的链路中，UP 或 DP）。响应 Margin Command 的 Receiver Number 字段（除了没有命令外）反映响应的接收器的数量，即使对于广播的 Margin Command 也是如此。Margin Command 会向下，而响应则在 Control SKP Ordered Set 中上游。响应反映目标接收器响应的 Margin Type。响应的 Receiver Number 字段对应于正在响应的目标接收器。此处使用的各种参数（如 MSampleCount）在 Section 8.4.4 中定义。下面描述的所有未使用的编码都保留，不得视为有效的 Margin Command。

公众号：芯片业者相承双

Table 4-26 Margin Commands and Corresponding Responses

Command				Response	
Margin Command	Margin Type [2:0]	Valid Receiver Number(s) [2:0]	Margin Payload [7:0]	Margin Type [2:0]	Margin Payload [7:0]
No Command	111b	000b	9Ch (No Command is also an independent command in Upstream direction. The expected Response is No Command with the Receiver Number = 000b.)		
Access Retimer register (Optional)	001b	010b, 100b	Register offset in bytes: 00h - 87h, A0h - FFh	001b	Register value, if supported. Target Receiver on Retimer returns 00h if it does not support accessing its registers.
Report Margin Control Capabilities	001b	001b through 110b	88h	001b	Margin Payload[7:5] = Reserved; Margin Payload[4:0] = {$M_{IndErrorSampler}$, $M_{SampleReportingMethod}$, $M_{IndLeftRightTiming}$, $M_{IndUpDownVoltage}$, $M_{VoltageSupported}$}
Report $M_{NumVoltageSteps}$	001b	001b through 110b	89h	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{NumVoltageSteps}$
Report $M_{NumTimingSteps}$	001b	001b through 110b	8Ah	001b	Margin Payload [7:6] = Reserved Margin Payload [5:0] = $M_{NumTimingSteps}$
Report $M_{MaxTimingOffset}$	001b	001b through 110b	8Bh	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{MaxTimingOffset}$
Report $M_{MaxVoltageOffset}$	001b	001b through 110b	8Ch	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{MaxVoltageOffset}$
Report $M_{SamplingRateVoltage}$	001b	001b through 110b	8Dh	001b	Margin Payload [7:6] = Reserved Margin Payload[5:0] = {$M_{SamplingRateVoltage}$ [5:0]}
Report $M_{SamplingRateTiming}$	001b	001b through 110b	8Eh	001b	Margin Payload [7:6] = Reserved Margin Payload[5:0] = {$M_{SamplingRateTiming}$ [5:0]}

公众号·芯片业者相聚及

Command				Response	
Margin Command	Margin Type [2:0]	Valid Receiver Number(s) [2:0]	Margin Payload [7:0]	Margin Type [2:0]	Margin Payload [7:0]
Report $M_{SampleCount}$	001b	001b through 110b	8Fh	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{SampleCount}$
Report $M_{MaxLanes}$	001b	001b through 110b	90h	001b	Margin Payload [7:5] = Reserved Margin Payload[4:0] = $M_{MaxLanes}$
Report Reserved	001b	001b through 110b	91-9Fh	001b	Margin Payload[7:0] = Reserved
Set Error Count Limit	010b	001b through 110b	Margin Payload [7:6] = 11b Margin Payload[5:0] = ***Error Count Limit***	010b	Margin Payload [7:6] = 11b Margin Payload[5:0] = Error Count Limit registered by the target Receiver
Go to Normal Settings	010b	000b through 110b	0Fh	010b	0Fh
Clear Error Log	010b	000b through 110b	55h	010b	55h
Step Margin to timing offset to right/left of default	011b	001b through 110b	See Section 4.2.13.1.2	011b	Margin Payload[7:6] = Step Margin Execution Status (see Section 4.2.13.1.1) Margin Payload[5:0] = $M_{ErrorCount}$
Step Margin to voltage offset to up/ down of default	100b	001b through 110b	See Section 4.2.13.1.2	100b	Margin Payload[7:6] = Step Margin Execution Status (see Section 4.2.13.1.1) Margin Payload[5:0] = $M_{ErrorCount}$
Vendor Defined	101b	001b through 110b	Vendor Defined	101b	Vendor Defined

Note ：

1. 术语 Step Margin 命令用于指代时间偏移到默认右侧/左侧的 Step Margin 或电压偏移到默认命令的向上/向下的 Step Margin。

公众号·芯片业者相关双

4.2.13.1.1 Step Margin Execution Status

Table 4-26 中使用的 Step Margin Execution Status 是一个 2 比特字段，定义如下：

00b NAK。表示发出了不受支持的 Lane Margining command。例如，时序余量超过±0.2 UI。MErrorCount 为 0。

01b Margining in progress。接收器正在执行 Step Margin 命令。MErrorCount 反映了 Section 8.4.4 中定义的检测到的错误数。

10b 正在设置 Margining 。这表示接收器已准备就绪，但尚未开始执行 Step Margin 命令。MErrorCount 为 0。

11b Too many errors - 接收器自动返回其默认设置。MErrorCount 反映了 Section 8.4.4 中定义的检测到的错误数。请注意，MErrorCount 可能大于错误计数限制。

4.2.13.1.2 Margin Payload for Step Margin Commands

对于 Step Margin to timing offset to right/left of default 命令，Margin Payload 字段定义如下：

- Margin Payload [7] 为 Reserved。
- 如果目标接收器的 $M_{IndLeftRightTiming}$ 设置为 1：
  - Margin Payload [6] 表示 Margin Command 是 right 还是 left。0b 表示将接收器移动到正常设置的右侧，而 1b 表示将接收器移动到正常设置的左侧。
  - Margin Payload [5:0] 表示正常设置左侧或右侧的步数。
- 如果目标接收器的 $M_{IndLeftRightTiming}$ 设置为 0：
  - Margin Payload [6] 为 Reserved。
  - Margin Payload [5:0] 表示超出正常设置的步数。

对于 Step Margin to voltage offset to up/down of default 命令，Margin Payload 字段定义如下：

- 如果目标接收器的 $M_{IndUpDownVoltage}$ 设置为 1：
  - Margin Payload [7] 表示 Margin Command 是 up 还是 down。0b 表示将接收器移动到正常设置的上方，而 1b 表示将接收器移动到正常设置的下方。
  - Margin Payload [6:0] 表示正常设置 up 或 down 的步数。
- 如果目标接收器的 $M_{IndUpDownVoltage}$ 设置为 0：
  - Margin Payload [7] 为 Reserved。
  - Margin Payload [6:0] 表示超出正常设置的步数。

4.2.13.2 Margin Command and Response Flow

每个接收器都按照 Section 8.4.4 中的定义通告其能力。接收器必须报告与在指定位置发生的数据样本一致的错误数量以执行 margin 过程。为简单起见，Margin Command 和需求是根据移动数据采样器位置来描述的，尽管实际的 margining 方法可能是特

公众号·芯片业者相承双

定于实现的。例如，时序余量可以在实际数据采样器或独立/错误采样器上实现。此外，可以通过向数据采样位置注入适量的应力/抖动或通过实际移动数据/错误采样位置来实现时序裕度。当使用独立数据/错误采样器时，即使链路可能没有遇到任何错误，也必须在 MErrorCount 中报告独立的数据/错误采样器遇到的错误。为了 margin 一个接收器，软件将目标接收器移动到与其默认采样位置的电压/时序偏移。

以下规则是必须要遵守的：

- 每个 Retimer 的 Upstream Pseudo Port Receiver 和 Downstream Port Receiver 必须计算 Margin CRC 和 Margin Parity 位，并与接收的 Margin CRC 和 Margin Parity 位进行比较。任何不匹配都会导致忽略 Symbol 4N+2 和 4N+3 的内容。Downstream Port Receiver 必须在 Lane Error Status Register 中报告 Margin CRC 和 Margin Parity 错误（参见 Section 7.7.3.3 ）。
- 允许 Upstream Port Receiver 忽略 Margin CRC、Margin Parity 位和 Control SKP Ordered Set 的 Symbol 4N+2 和 4N+3 中的所有位。如果它检查 Margin CRC 和 Margin Parity，则必须在 Lane Error Status Register 中报告任何不匹配。
- Downstream Port 必须在每个通道中发送 Control SKP Ordered Set，其中 Margin Type、Receiver Number、Usage Model 和 Margin Payload 字段反映了 Margining Lane Control Register 中的相应控制字段。在配置写入完成后超过 10 µs 发送的任何 Control SKP Ordered Set 必须反映由该配置写入写入的 Margining Lane Control Register 的值。
  - 无论 Margining Lane Control Register 中的值如何，此要求都适用。
  - 无论链路中的 Retimer 数量如何，此要求均适用。
- 对于 Upstream Pseudo Port 接收的 Control SKP Ordered Set，如果以下所有条件都为真，则 Retimer Receiver 是一个有效 Margin Command 的目标：
  - Margin Type 不是 No Command。
  - Receiver Number 是分配给接收器的编号，或者 Margin Type 是 Clear Error Log 或 Go to Normal Settings，并且 Receiver Number 是“Broadcast”。
  - Usage Model 字段为 0b。
  - Margin Type、Receiver Number 和 Margin Payload 字段与 Table 4-25 和 Table 4-26 中的定义一致。
  - Margin CRC 检查和 Margin Parity 检查通过。
- 对于 Upstream 和 Downstream Port，如果其 Margining Lane Control Register 满足以下所有条件，则接收器是一个有效 Margin Command 的目标：
  - Margin Type 不是 No Command。
  - Receiver Number 是分配给接收器的编号，或者 Margin Type 是 Clear Error Log 或 Go to Normal Settings，并且 Receiver Number 是“Broadcast”。
  - Usage Model 字段为 0b。
  - Margin Type、Receiver Number 和 Margin Payload 字段与 Table 4-26 中的定义一致。
- Upstream Port 必须发送带 No Command 的 Control SKP Ordered Set。
- 如果链路仍处于 L0 状态并以 16.0 GT/s 或更高的数据速率运行，则目标接收器必须在收到有效 Margin Command 后 1ms 内应用并响应 Margin Command。
  - Retimer 中的目标接收器必须在接收 Margin Command 后 1 ms 内在 Upstream Direction 的 Control SKP Ordered Set 中发送响应。

公众号：芯片业者相承双

- Upstream Port 中的目标接收器必须在收到 Margin Command 后 1 ms 内更新 Lane Margin Command and Status register 的 Status 字段。
- 如果命令未广播或不存在 Retimer，则 Downstream Port 中的目标接收器必须在接收 Margin Command 后 1 ms 内更新 Lane Margin Command and Status register 的 Status 字段。

- 对于由 Retimer 广播和接收的除 No Command 之外的有效 Margin Type：
  - 如果命令已应用，位置 X 的 Retimer（见 Figure 4-35）转发 Upstream Control SKP Ordered Set 中未修改的响应，否则发送 No Command。
  - 响应的 Receiver Number 字段必须设置为 Retimer 的 Pseudo Port 之一的编码。
  - Retimer 必须在两个 Pseudo Port 都完成 Margin Command 后才响应。
- Retimer 必须覆盖 Symbol 4N+1 的 Bit [4:0]、Symbol 4N+2 的 Bit [7, 5:0] 和 Symbol 4N+3 中的 Bit [7:0]，因为如果它是 Margin Command 的目标接收器并且正在执行该命令，则 Retimer 向上游方向转发 Control SKP Ordered Set。
- 收到 Control SKP Ordered Set 后，如果 Downstream Port 通过了 Margin CRC 和 Margin Parity 检查以及以下条件之一，则 Downstream Port 必须在 1 µs 内在 Margining Lane Status Register 反映接收到的 Control SKP Ordered Set 中的相应字段：
  - 在 Margining Lane Control Register 中：Receiver Number 为 010b 到 101b。
  - 在 Margining Lane Control Register：Receiver Number 为 000b，Margin Command 为 Clear Error Log、No Command 或 Go to Normal Settings，并且链路中有 Retimer。
  - 如果 Margining Lane Control Register Usage Model 字段为 1b，则为可选。
  - 可选地，如果 Margining Lane Control Register Receiver Number 字段是 110b 或 111b 则 Margining Lane Status Register 字段会更新，而不管接收到的 Control SKP Ordered Set 中的 Usage Model 的值是多少。
- 组件必须为 Section 8.4.4 中 Table 8-11 中定义的每个参数在其所有接收器上通告相同的值。除了在 Section 8.4.4 的 Table 8-11 中定义的 $M_{SampleCount}$ 和 $M_{ErrorCount}$ 之外，当 LinkUp = 1b 时，组件不得更改任何参数值。
- 接收到有效 Step Margin 命令的目标接收器必须继续应用该偏移量，直到发生以下任何一种情况：
  - 收到有效的 Go to Normal Settings 命令。
  - 它接收具有不同 Margin Type 或 Margin Payload 字段的后续有效 Step Margin 命令。
  - $M_{IndErrorSampler}$ 为 0b 且 $M_{ErrorCount}$ 超出错误计数限制。
  - 可选地，$M_{INdErrorSampler}$ 为 1b 并且 $M_{ErrorCount}$ 超过错误计数限制。
- 如果 Step Margin 命令因 $M_{ErrorCount}$ 超过错误计数限制而终止，则目标接收器必须自动返回其默认样本位置并在 Margin Payload 字段中指明这一点（Step Margin Execution Status = 00b）。注意：如果 $M_{IndErrorSampler}$ 为 1b，则因此终止是可选的。
- 如果 $M_{IndErrorSampler}$ 为 0b，则在以下情况下会检测到一个错误：
  - 目标接收器是从 L0 中进入 Recovery 状态或检测到数据奇偶校验不匹配的端口。
  - 目标接收器是一个 Pseudo Port，它进入转发训练集或在转发非训练集时检测到数据奇偶校验不匹配。
- 如果 $M_{IndErrorSampler}$ 为 1b，则在以下情况下会检测到一个错误：
  - 目标接收器是一个在 L0 中检测到比特错误的端口。
  - 目标接收器是一个当 Retimer 转发非训练集时检测到比特错误的 Pseudo Port。

公众号：芯片业者相承双

- 如果 $M_{IndErrorSampler}$ 为 0b 并且 (1) 目标接收器是进入 Recovery 的端口或 (2) 目标接收器是进入转发训练集的 Pseudo Port，则：
  - 目标接收器必须回到默认样本位置。
  - 如果目标接收器是仍在执行 margining 的端口，则必须在进入 L0 后的 128 µs 内恢复到 margin 位置。
  - 如果目标 Receiver 是仍在执行 margining 的 Pseudo Port，它必须在转发非训练集的 128 µs 内恢复到 margin 位置。
- 目标接收器需要在接收 Clear Error Log 命令时清除其累积错误计数，同时它继续留出余量（如果它是仍在进行的 Step Margin 命令的目标接收器），如果它正在这样做。
- 对于 Set Error Count Limit 命令的目标接收器，新值用于所有未来的 Step Margin 命令，直到接收到新的 Set Error Count Limit 命令。
- 如果接收器自进入 L0 后未收到 Set Error Count Limit，则默认值为 4。
- 如果在 Step Margin 命令有效时接收到 Set Error Count Limit 命令，则行为未定义。
- 一旦目标接收器报告 Step Margin Execution Status 为 11b (NAK) 或 00b（“错误太多”），只要 Step Margin 有效，它就必须继续报告相同的状态。
- 目标接收器在收到新的有效 Step Margin 命令后的 100 ms 内不得报告 Step Margin Execution Status 为 01b（“正在设置 margin”）。
- 报告 Step Margin Execution Status 不是 01b 的目标接收器随后不能报告 01b，除非它接收到一个新的有效 Step Margin 命令。
- Margin Payload 中的保留位必须遵循以下规则：
  - Downstream 或 Upstream Port 必须为保留位传输 0。
  - Retimer 必须转发未修改的保留位。
  - 所有接收器必须忽略保留位。
- Margin Command、Receiver Number 或 Margin Payload 字段的保留编码必须遵循以下规则：
  - Retimer 必须转发未修改的保留编码。
  - 所有接收器必须将保留编码视为不是 Margin Command 的目标。
- Vendor Defined Margin Command 或 response，不是由 Retimer 定义的，它们将被忽略并正常转发。
- 如果 Retimer 不支持寄存器访问，则该 Retimer 上的目标接收器必须在 Access Retimer Register 命令的响应有效载荷上返回 00h。如果 Retimer 支持 Access Retimer register 命令，则必须遵守以下事项：
  - 它必须分别在地址 80h 和 84h 处为 DWORD 返回一个非零值。
  - 不得放置与 Margin Payload 的地址 88h 到 9Fh 相对应的任何寄存器。

4.2.13.3 Receiver Margin Testing Requirements

软件必须确保在接收器执行 Lane Margining 之前满足以下条件：

- 当前链路数据速率必须为 16.0 GT/s 或更高。
- 当前链路宽度必须包括要测试的 Lane。

公众号·芯片业者相承双

- 必须将 Upstream Port 的 Function 编程为 D-state，以防止端口进入 L1 链路状态。有关更多信息，请参阅 Section 5.2 。
- 必须在 Downstream Port 和 Upstream Port 中将 Link Control Register 的 ASPM Control 字段设置为 00b（Disabled）。
- 必须保存 Link Control 2 Register 的 Hardware Autonomous Speed Disable 字段和 Link Control Register 的 Hardware Autonomous Width Disable 字段的状态，以便稍后在此过程中恢复。
- 如果可写，则必须在 Downstream Port 和 Upstream Port 中把 Link Control 2 Register 的 Hardware Autonomous Speed Disable 字段设置为 1。（如果硬连线到 0b，则不会实现自主变速机制，因此本质上是禁用的。）
- 如果可写，则必须在 Downstream Port 和 Upstream Port 中把 Link Control Register 的 Hardware Autonomous Width Disable 字段设置为 1。（如果硬连线到 0b，则不会实现自主宽度更改机制，因此本质上是禁用的。）

在调整 margining 时，软件必须确保以下几点：

- 所有 Margin Command 必须将 Margining Lane Control Register 中的 Usage Model 字段设置为 0b。 在检查 outstanding 的 Margin Command 的状态时，软件必须检查 Margining Lane Status Register 状态部分的 Usage Model 字段是否设置为 0b。
- 软件必须读取接收器提供的能力，并在其提供的农光里的限制范围内对其进行裕度处理。发出的命令和确定余量的过程必须与 Section 4.2.13 和 Section 8.4.4 中提供的定义一致。例如，如果端口不支持电压测试，则软件不得启动电压测试。此外，如果一个端口支持同时测试 2 个 Lane，那么软件只能同时测试 1 个或 2 个 Lane，并且不能超过 2 个 Lane。
- 对于 $M_{IndErrorSampler}$ 为 1b 的接收器，此类接收器的任何组合都允许并行设置裕量。
- 对于 $M_{IndErrorSampler}$ 为 0b 的接收器，一次最多允许一个这样的接收器进行裕度处理。然而，裕量可以同时在多个通道上执行，只要它在设备支持的最大通道数内。
- 软件必须确保它在 Margining Lane Control Register 中提供的 Margin Command 是有效的，如 Section 4.2.13.1 中所定义。例如，Margin Type 必须具有定义的编码，并且 Receiver Number 和 Margin Payload 与其一致。
- 通过以原子方式写入 Margining Lane Control Register 发出命令后，软件必须检查该命令是否完成。这是通过自动读取 Margining Lane Status Register 并检查 Status 字段是否与发出的命令的预期响应匹配来完成的（参见 Table 4-25）。如果在发出新的 Margin Command 后 10 ms 过去了，并且读取的值与预期的响应不匹配，则允许软件假设接收器不会响应，并声明目标接收器的裕度调整失败。 对于 No Command 以外的广播命令，响应中的 Receiver Number 必须对应于 Retimer Y 或 Retimer Z 中的 Pseudo Port 之一，如 Figure 4-35 所述。
- 对 Margining Lane Status Register 的任何两次读取应间隔至少 10 µs，以确保它们从不同的 Control SKP Ordered Set 读取结果。
- 软件必须广播 No Command 并等待它完成，然后才能在 Margining Lane Control Register 中发布新的 Margin Type 或 Receiver Number 或 Margin Payload。
- 在给定方向（电压/时序和上/下/左/右）的裕度调整结束时，软件必须在确保目标接收器已确认每个命令之后，在下行和上行端口中连续广播 Go to Normal Settings、No Command、Clear Error Log 和 No Command。
- 如果数据速率在裕量调整期间发生了变化，则裕量结果（如果有）将不准确，软件必须退出裕量调整程序。软件必须将 Margining Lane Control Register 设置为 No Command 以避免在 Data Rate 稍后更改为 16.0 GT/s 或更高时启动裕度。
- 允许软件在进行裕度调整时定期发出 Clear Error Log 命令，以在很长一段时间内收集错误信息。
- 软件不得尝试同时调整目标接收器的时序和电压。如果接收器接收到的命令会将电压和时序裕量位置同时放置在远离默认采样位置的位置，则结果是不确定的。
- 接收器在测试时在切换到下一个裕度步长位置之前，软件应允许运行至少 108 比特的裕度（除非超出误差限制）。
- 在测量裕量时间或接收器采样的位数时，软件必须考虑“正在设置余量”状态。

公众号：芯片业者相承攻

- 如果目标接收器在发出 Step Margin 命令之一后 200 ms 报告“正在设置余量”，则允许软件假定接收器不会响应并声明目标接收器余量设置失败。
- 如果接收器在 Margin Payload 状态字段中报告“NAK”并且相应的 Step Margin 命令有效且在允许范围内（如 Section 4.2.13 和 Section 8.4.4 所定义），则允许软件声明目标接收器余量调整失败。
- 当余量测试程序完成时，Hardware Autonomous Speed Disable 和 Hardware Autonomous Width Disable 的状态必须恢复为之前保存的值。

公众号·芯片业者相承双

IMPLEMENTATION NOTE

Example Software Flow for Lane Margining at Receiver

为了获得不变参数，可以遵循以下步骤。一旦获得，只要 LinkUp=1b 继续为真，就可以在多组裕度测试中使用相同的参数。对于链路中的每个组件，请执行以下步骤。软件可以对不同链路上的不同组件并行执行这些步骤。

Step A1：发送 Report Margin Control Capabilities（Margining Lane Control Register 中的 Margin Type = 001b，Margin Payload = 88h，Receiver Number = 目标设备）。

Step A2：读取 Margining Lane Status Register。

a. 如果 Margin Type = 001b 且 Receiver Number = 目标接收方则转到 Step A3。

b. 否则：如果自命令发出后 10 ms 已过期，则声明 Receiver 余量调整失败并退出；否则等待 >10 µs 并转到 Step A2。

Step A3：存储提供的 Margin Payload 状态字段的信息，以便在 margining 过程中使用。

Step A4：广播 No Command（Margining Lane Control Register 中的 Margin Type = 111b、Receiver Number = 000b 和 Margin Payload = 9Ch）并等待这些命令在 Margining Lane Status Register 中反映出来。如果 10 ms 到期而没有获得命令完成握手，则声明接收器的裕度调整失败并退出。

Step A5:重复 Step A1 到 Step A4 以执行 Report MNumVoltageSteps、Report MNumTimingSteps、Report MMaxTimingOffset、Report MMaxVoltageOffset、Report MSamplingRateVoltage 和 Report MSamplingRateTiming 这些命令。需要注意的是，对于不同的 Margin Type，可以跨不同 Lane 并行执行此步骤。

跨越链路的每个通道上的 Margining 可以是一系列单独的命令。在启动该命令序列之前，软件应读取允许同时运行裕度调整的最大通道数。这些步骤与上面的 Step A1 到 Step A4 类似，使用 Report MMaxLanes 命令。之后，该软件可以同时 margin up 链路的这么多通道。在每个链路上，在确保链路以 16.0 GT/s 或更高的数据速率以全宽度工作并且硬件自主宽度和速度变化以及 ASPM 功率状态已被禁用之后，每个接收器根据其能力被限制在这里描述的约束下。

如果软件想要设置一个不同于默认值 4 或任何最后编程的错误计数限制值，它会在进入下面的 Step C1 之前执行以下步骤。

Step B1：发送 Set Error Count Limit（Margining Lane Control Register 中的 Margin Type = 010b， = 88h，目标 Receiver Number 和 Margin Payload = {11b, Error Count Limit}）。

Step B2：读取 Margining Lane Status Register。

a. 如果 Margin Type = 010b，Receiver Number = 目标接收器，Margin Payload = Margin Payload control 字段的值（Bit [14:7]），则转到 Step B4。

b. 否则：如果自命令发出后 10 ms，占到 Step B3；否则，等待 > 10ms 时间转到 Step B2 。

Step B3：裕度调整失败。调用系统检查以查明链路的宽度/速度是否因可靠性原因而降低。

Step B4：广播 No Command 并等待那些在状态字段中反映出来。如果 10 ms 到期而没有获得命令完成握手，则声明接收器的裕度调整失败并退出。

以下步骤是给定接收器的一个 margin point 示例流程，该接收器执行 Step Margin 以从右侧 15 个步长开始对默认的右侧/左侧进行时序偏移：

Step C1：在 Margining Lane Control Register 写入 Margin Type = 011b、目标 Receiver Number 和 Margin Payload = {0000b, 1111b}。

Step C2：读取 Margining Lane Status Register。

a. 如果 Margin Type = 011b 且 Receiver Number = 目标接收器，则转到 Step C3。

b. 否则，如果自命令发出后 10 ms 已过期，则声明接收器的裕度调整失败并转到 Step C7。

c. 等待 >10 µs 并转到 Step C2。

Step C3：在 Margining Lane Status Register 中：

a. 如果 Margin Payload [7:6] = 11b：

① 如果我们超过了 0.2 UI，那就是具有了所需的 margin；

② 否则此时报告裕度调整失败并转到 Step C7；

b. 否则，如果 Margin Payload [7:6] = 00b：

① 报告裕度调整失败并转到 Step C7；

c. 否则，如果 Margin Payload [7:6] = 01b：

① 如果进入 Step C3 已经过去了 200ms，则报告 Receiver 裕量测试失败并退出；

② 否则等待 1 ms，读取 Margining Lane Status Register 并转到 Step C3。

d. 否则转到 Step C4。

Step C4：等待所需的时间进行裕量调整，同时周期性地对 Margining Lane Status Register 进行采样，以获取 Margin Payload 字段中报告的错误数量（Bit [5:0] - MErrorCount）。

对于更长时间的运行，即使保持在预期的 BER 目标范围内，如果该时间长度将导致错误计数超过 Set Error Count Limit，则发出 No Command 和 Clear Error Log 命令（使用类似于 Step B1 到 Step B4 的过程，以及相应的预期状态字段）。

如果总错误计数保持在预期错误计数内并且状态字段中的 Margin Payload [7:6] 保持 10b 直到结束，则接收器在时序余量过程具有所需的 Margin；否则，时序余量过程将失败，转到 Step C7。

Step C5：广播 No Command 并等待那些在状态字段中反映出来。如果 10 ms 到期而没有获得命令完成握手，则声明接收器的裕度调整失败并退出。

Step C6：转到 Step C1，如果我们想针对更高的 margin 进行测试，则通过 Margin Payload Control 字段（Bit [5:0]）增加计时步数；否则转到 Step C8，注意接收器通过的 margin 值。

Step C7：Margin 失败；Receiver 在 Step C6 中通过的前一个 margin step 是 Receiver 的 margin。

Step C8：广播 No Command、Clear Error Log、No Command、Go to Normal Settings 命令序列（使用类似于 Step B1 到 Step B4 的过程以及相应的预期状态字段）。

4.3 Retimers

Redriver 和 Retimer 都是中继器。Redriver 通过信号均衡化和预加强等技术，重新加强再发送出去。Retimer 则是通过内部时钟重构信号，再恢复后发送出去。

Retimer 就类似于一个 PHY 芯片，其信号在经过 Retimer 的时候，通过内部的时钟重构信号，使其信号传输能量增加，然后再继续传输。Retimer 是内部具有 CDR(数据时钟恢复)的 IC，实现数据的恢复之后然后再此按照串行通道把信号发送出去。可以减轻信号的抖动。

在云计算、数据中心领域，服务器侧 CPU 离 PCIe 加速卡设备越来越远，Retimer 就越来越有用。

公众号：芯片业者相承双

本节定义了具有物理层协议感知能力的 Retimer 的要求,以及与 Retimer 每侧具有任何兼容通道的任何组件对进行互操作的要求。物理层协议感知 Retimer 的一个重要功能是在每个方向上执行均衡过程的 Phase 2/3。上游和下游端口之间最多允许有两个 Retimer。

两个 Retimer 的限制是基于多种考虑的，最显著的是修改了 SKP 有序集的限制，以及在均衡过程的 Phase 2/3 中花费的时间的限制。为确保互操作性，平台设计人员必须确保所有 PCI Express 链路（包括涉及外形规格的电缆和涉及有源电缆的电缆）均遵守两个 Retimer 限制。外形规格可以定义其他必须遵守的 Retimer 规则。 评估不基于本节中的 Retimer 定义的任何扩展设备的互操作性不在本规范的范围之内。

扩展设备的许多架构都是可能的，例如，模拟中继器、协议无关的 Retimer 等。本规范描述了物理层协议无关的 Retimer。如果对特定通道、扩展设备和终端设备对进行了适当的分析，则可以在封闭系统中使用其他类型的扩展设备，但执行此分析的特定方法不在本规范的范围内。

Retimer 具有两个伪端口（Pseudo Port），一个面向上游，另一个面向下游。每个伪端口的发送器必须从 100 MHz 参考时钟中获取其时钟。参考时钟必须满足 Section 8.6 的要求。Retimer 支持 Section 8.6 中 Electrical Sub-block 中定义的一种或多种参考时钟架构。

在大多数操作中，Retimer 仅转发接收到的有序集，DLLP，TLP，逻辑空闲和电气空闲。Retimer 对数据链路层和事务层是完全透明的。系统软件不得在存在 Retimer 的任何链路上启用 L0s。Retimer 对 Beacon 的支持是可选的，超出了本规范的范围。

当使用 128b/130b 编码时，Retimer 将执行协议，以使每个链路段都进行独立的链路均衡，如 Section 4.3.6 中所述。

伪端口方向（上游或下游）是动态确定的，而 Link partner 处于 Configuration 状态。同时支持 crosslink 和常规链路。

4.3.1 Retimer Requirements

以下是对 Retimer 规则的高度总结：

- Retimer 必须符合 Chapter 8 中 Electrical Sub-block 中所述的所有电气规范。Retimer 必须以以下两种模式之一进行操作：
  - Retimer 的接收器以 8.0 GT/s 以上的速度运行，其阻抗满足为 2.5GT/s 定义的 Z_{GT-DC} 参数定义的范围。
  - Retimer 的接收器以 8.0 GT/s 以上的速度运行，其阻抗不满足为 2.5GT/s 定义的 Z_{GT-DC} 参数定义的范围。在此模式下，必须在接收 EIOS 或推断电气空闲的 1 ms 内以及接收器保持电气空闲的状态下在 1 ms 内满足 2.5 GT/s 的 Z_{RX-DC} 参数。
- 当多个 Lane 正在转发符号时（包括 upconfigure 的情况），必须始终对转发的符号进行 de-skewed 处理。
- Retimer 的 Port 的方向是动态判定的。
- 如果有必要，要执行极性反转。
- 当使用 128b/130b 编码时，在每个链路段上执行 Phase 2 和 Phase 3 的链路均衡过程。
- 在每个链路段上以 5.0 GT/s 的速度进行去加重协商互操作。
- Retimer 可以执行链路的 Upconfigure 互操作。
- 在 Loopback Master 和 Loopback Slave 之间传递环回数据。
  - 可选的在 Retimer 的一个伪端口上执行 Slave 的环回操作。
- 每个伪端口都可以生成 Compliance Pattern 。
  - 针对负载版。

- 当链路通过 TS1 Ordered Set 的 Compliance Receive 进入 Polling.Compliance 时，Retimer 要转发 Modified Compliance Pattern 。
- 当通过把 Upstream Port 和 Downstream Port 的 Link Control 2 寄存器的 Enter Compliance 字段设置为 1b，并且把 Retimer 的 Retimer Enter Compliance 设置为 1b 的方式进入 Polling.Compliance 时， Retimer 要转发 Compliance Pattern 或 Modified Compliance Pattern 。
- 与链路的上游端口和下游端口一起调整操作的数据速率。
- 根据链路的上游和下游端口调整链路宽度。
- 在 Configuration 状态捕获 Lane number 。
  - 将 128b/130b 编码用于加扰种子时，需要使用 Lane number 。
- 动态调整 Retimer 接收器阻抗以匹配终端组件接收器阻抗。
- 在所有数据速率推断进入电气空闲状态。
- 转发时修改有序集的某些字段。
- 通过添加或删除 SKP 符号来执行时钟补偿。
- 支持 L1 低功耗状态。
  - 可选的支持 L1 PM Substate 。
- 支持链路均衡到高数据速率。
- 支持 No Equalization Needed 模式。

4.3.2 Supported Retimer Topologies

Figure 4-36 显示了此规范中定义的 Retimer 支持的拓扑。链路的上游端口和下游端口之间可能有一个或两个 Retimer。每个 Retimer 都有两个伪端口，它们可以动态确定其下行/上行方向。每个 Retimer 都有一个上游路径和一个下游路径。在转发模式下，两个伪端口必须始终以相同的数据速率运行。因此，每个路径也将具有相同的数据速率。允许 Retimer 支持本规范定义的任何宽度选项作为其最大宽度。在每个高级操作模式下， Retimer 的行为是：

- 转发模式（Forwarding mode）：
  - 符号、电气空闲和从电气空闲退出；在每个上游和下游路径上转发。
- 执行模式（Execution mode）：
  - 上游伪端口充当组件的上游端口。下游伪端口充当组件的下游端口。 在以下情况下使用此模式：
    - Polling.Compliance 。
    - 链路均衡过程的 Phase 2 和 Phase 3 。
    - 可选的 Slave 环回。

公众号：芯片业者相承双

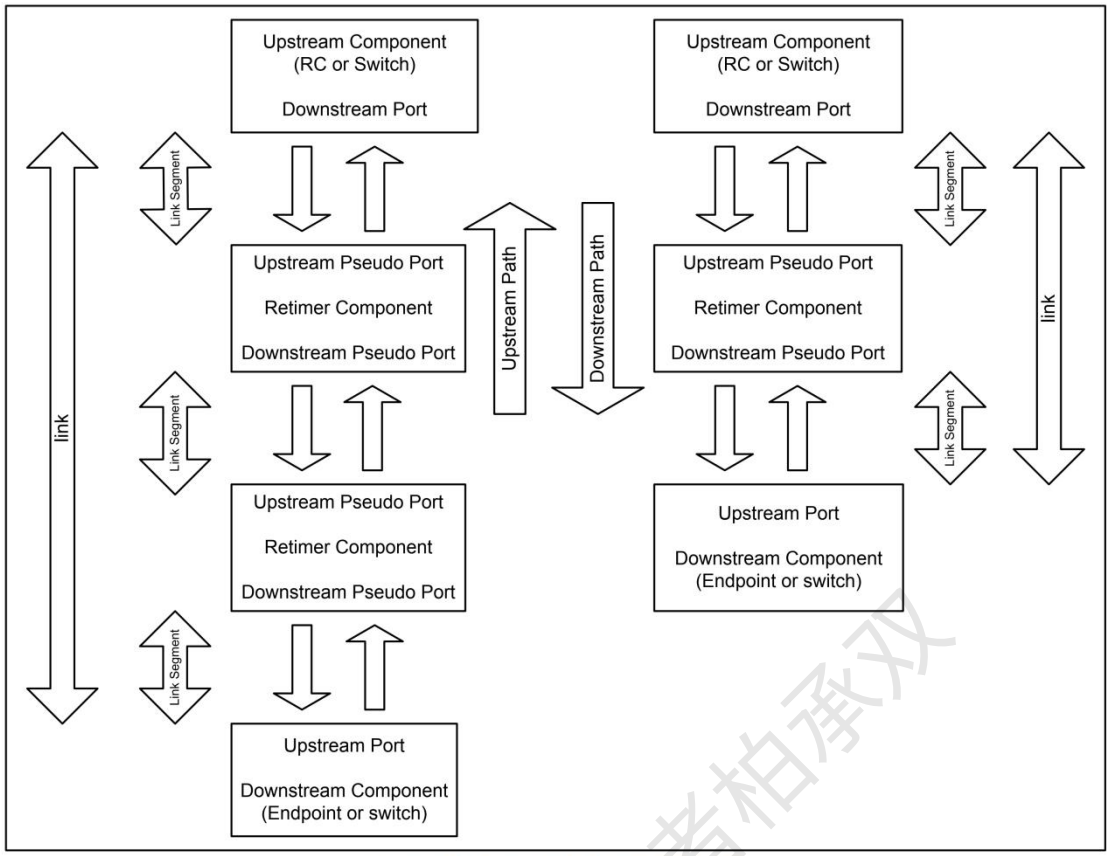

Figure 4-36 Supported Retimer Topologies

4.3.3 Variables

以下变量在基本复位后或 Retimer 接收到两个连续的 TS2 有序集上的 Link number 和 Lane number 等于 PAD 时设置为以下指定值，这两个连续的 TS2 有序集位于所有 Lane 上，这些 Lane 在距离 Retimer 最后一个符号 1µs 的时间窗内接收上游和下游伪端口上的 TS2 有序集第一条 Lane 上的第二个 TS2 有序集到最后一条 Lane 上的第二个 TS2 有序集的最后一个符号。

- RT_port_orientation = undefined
- RT_captured_lane_number = PAD
- RT_captured_link_number = PAD
- RT_G3_EQ_complete = 0b
- RT_G4_EQ_complete = 0b
- RT_G5_EQ_complete = 0b
- RT_LinkUp = 0b
- RT_next_data_rate = 2.5 GT/s
- RT_error_data_rate = 2.5 GT/s

公众号：芯片业者相承双

4.3.4 Receiver Impedance Propagation Rules

基本复位有效时，Retimer 发送器和接收器应满足 Section 4.2.4.9.1 的要求。当基本复位无效时，允许 Retimer 花 20 ms 的时间来开始主动确定其接收器阻抗。在此间隔内，接收器阻抗会在基本复位期间保持所需的状态。一旦此间隔到期，将按以下方式确定 Retimer 通道上的接收器阻抗：

- 在满足上游或下游端口的接收器 Z_{RX-DC} 参数的 1.0 ms 内，低阻抗将反向传播（即，Retimer 的接收器应满足另一个伪端口上相应通道上的 Z_{RX-DC} 参数）。每个 Lane 都独立运行，并且此要求始终适用。
- Retimer 必须将其发送器保持在电空闲状态，直到检测到 Z_{RX-DC} 状态为止。这适用于单个 Lane。

4.3.5 Switching Between Modes

Retimer 在两种基本模式下运行，即转发模式或执行模式。在这些模式之间切换时，必须同时在发送器的所有通道的有序集边界上进行切换。 当前模式下发送的最后一个有序集和新模式下发送的第一个符号之间不得有其他符号。

当使用 128b/130b 时，在模式之间切换时，发送器必须保持正确的加扰种子和 LFSR 值。

在转发模式和执行模式之间切换时，Retimer 必须确保在先前模式下发送的最后一个 EIEOS 和新模式下发送的第一个 EIEOS 之间发送至少 16 个 TS1 有序集和最多 64 个 TS1 有序集。

在执行链路均衡模式之间切换时，Retimer 必须确保发送器不会连续发送两个 SKP 有序集，并且在 SKP 有序集之间不超过最大允许间隔，请参见 Section 4.2.7.3。

4.3.6 Forwarding Rules

当 Retimer 处于转发模式时，这些规则适用。基本复位释放后，Retimer 处于转发模式。

- 如果 Retimer 的接收器检测到从通道上的电气空闲退出，则 Retimer 必须进入转发模式，并将该通道上的符号转发到相对的伪端口，如 Section 4.3.6.3 所述。
- Retimer 必须继续在给定通道上转发接收到的符号，直到它进入执行模式或直到接收到 EIOS，或者直到在该通道上推断出电气空闲为止。即使接收器丢失符号锁定或块对齐，此要求也适用。有关电气空闲进入的规则，请参见 Section 4.3.6.5。
- 除 Section 4.3.6.9 和 Section 4.3.6.7 所述外，Retimer 应原样转发所有符号。
- 当以 2.5 GT/s 数据速率运行时，如果伪端口的任何通道接收到 TS1 有序集，且 Link number 和 Lane number 设置为 PAD 达 5 ms 或更长时间，而另一个伪端口在在同一窗口中任何端口的 Lane 上均未检测到从电气空闲退出，将发生以下任一情况：
  - 发生以下过程：
    - 在接收 TS1 有序集的任何通道上都将接收到 EIOS。
    - 随后是一段电气空闲时间，持续时间少于 5 ms 。
    - 然后是根据 Section 4.3.6.3 无法转发的电气空闲退出。
    - 注意：这被解释为连接到接收器的端口进入电空闲状态，然后数据速率更改为 2.5 GT/s 以上的 Compliance Pattern 。
  - 在接收 TS1 有序集的任何通道上都以 2.5 GT/s 的速度收到 Compliance Pattern 。

公众号·芯片业者相承双

然后，Retimer 进入执行模式 CompLoadBoard 状态，并遵循 Section 4.3.7.1。

- 如果在使用 128b/130b 编码时，上游伪端口上的任何通道接收到两个连续的 EC 字段等于 10b 的 TS1 有序集，则 Retimer 将进入执行模式的均衡过程，并遵循 Section 4.3.7.2 。
- 如果将 Retimer 配置为支持执行模式的 Slave 环回，并且任一伪端口上的任何通道接收到两个连续的 TS1 有序集或两个连续的 TS2 有序集，且 Loopback 字段设置为 1b，则 Retimer 进入执行模式的 Slave 环回，并遵循 Section 4.3.7.3 。

4.3.6.1 Forwarding Type Rules

Retimer 必须确定其转发的符号类型。推断电气空闲的规则是 Retimer 正在转发的符号类型的函数。如果路径在任何一个通道上转发两个连续的 TS1 有序集或两个连续的 TS2 有序集，则该路径正在转发训练集。如果路径在所有正在转发符号的通道上转发了八个连续的空闲符号时间数据，则该路径正在转发非训练集。 当 Retimer 从转发训练集过渡到转发非训练集时，变量 RT_error_data_rate 设置为 2.5 GT/s。

4.3.6.2 Orientation and Lane Numbers Rules

当链路训练时，Retimer 必须动态确定端口方向，通道分配和通道极性。

- 当 RT_LinkUp = 0 时，第一个伪端口接收两个连续的 TS1 有序集，且在任何通道上都具有 non-PAD 的 Lane number，其 RT_port_orientation 变量设置为上游端口，另一个伪端口的 RT_port_orientation 变量设置为下游端口。
- Retimer 不会影响 Lane number 的确定。在 Configuration 状态的末尾，链路组件之间，Retimer 必须使用 RT_captured_lane_number 变量来捕获 Lane number。这在首次进入 Configuration 时适用，即 RT_LinkUp 设置为 0b 时。在 Link number 配置期间通过 Configuration 进行的后续过程不得更改 Lane number。使用 128b/130b 时，加扰种子依赖于 Lane number。在某些情况下，当 Retimer 处于执行模式时，需要 Link number。在 RT_LinkUp 变量设置为 0b 后，每当收到包含 non-PAD Lane number 和 non-PAD Link number 的前两个连续的 TS2 有序集时，就会使用 RT_captured_lane_number 和 RT_captured_link_number 变量捕获 Link number 和 Lane number。如果发生通道反转，则 Retimer 必须正常工作。当 Retimer 捕获了 Link number 和 Lane number 后，变量 RT_LinkUp 设置为 1b。另外，如果在上述两种情况下，TS2 有序集中的 Disable Scrambling 字段都设置为 1b，则 Retimer 将确定在使用 8b/10b 编码时禁用了加扰。
- 每当通道退出电气空闲并确定其以 2.5 GT/s 的速度锁定时，都会确定通道极性，如 Section 4.2.4.5 所述：
  - 如果确定了极性反转，则接收器必须反转接收到的数据。发送器绝不能反转发送的数据。

4.3.6.3 Electrical Idle Exit Rules

以 2.5 GT/s 以外的数据速率，在训练集中发送 EIEOS，以确保模拟电路检测到从电气空闲退出。使用 128b/130b 编码实现块对齐时，需要接收 EIEOS。当 Retimer 在检测到电气空闲退出后开始转发数据时，Retimer 开始在训练集边界上发送。当以高于 2.5 GT/s 的数据速率运行时，它转发的第一个训练集必须是 EIEOS。发送的第一个 EIEOS 将代替原本将转发的 TS1 或 TS2 有序集。

如果在伪端口上没有通道满足 Z_{RX-DC}，则会执行以下过程：

- 在该伪端口的任何通道上检测到从电气空闲退出。
- 然后，如果不是所有通道，都会通过该伪端口的 12 ms 窗口中没有从电气空闲退出而推断出电气空闲，而另一个伪端口不在同一 12 ms 窗口中的任何通道上接收到有序集。

然后，没有通道满足 Z_{RX-DC} 的同一个伪端口在所有通道上发送下面描述的电气空闲退出码型持续 5μs。

如果工作在 2.5GT/s 并发生了以下事件：

- 任何通道都检测到从电气空闲退出，
- 然后接收两个连续的 TS1 有序集，其通道和链路号等于 PAD，
- 而另一个伪端口在任何通道上均未接收到有序集。

然后，在未接收有序集的伪端口的所有通道上执行接收器检测。如果未检测到接收器，则：

- 如 Section 4.3.4 中所述，结果在 1.0 ms 内反向传播。
- 接收通道和链路号等于 PAD 的 TS1 有序集的同一个伪端口在所有通道上发送下面描述的电气空闲退出码型持续 5µs。

如果通道检测到从电气空闲退出，则在满足以下所有条件时，通道必须开始转发：

- 确定数据速率，请参见 Section 4.3.6.4，如果需要，将当前数据速率更改为 RT_next_data_rate。
- 确定通道极性，请参见 Section 4.3.6.2。
- 接收到两个连续的 TS1 有序集或两个连续的 TS2 有序集。
- 在检测到从电气空闲退出或发生最大 Retimer 退出延迟的所有通道上，将收到两个连续的 TS1 有序集或两个连续的 TS2 有序集，请参阅 Table 4-27。
- 在接收了两个连续的 TS1 或两个连续的 TS2 有序集的所有通道上都实现了通道 de-skew。
- 如果发生了数据速率变化，则自检测到电气空闲退出以来已经过去了 6 µs。

用于建立转发的所有有序集必须被丢弃。如 Section 4.3.4 中所述，仅考虑在另一个伪端口上检测到接收器的通道进行转发。

否则，在 3.0 ms 超时后，如果另一个伪端口未接收到有序集，则将在未接收有序集的伪端口的所有通道上执行接收器检测，结果将如 Section 4.3.4 中所述反向传播。如果未检测到接收器，则：

- 然后，在任何通道上均无法接收两个连续的 TS1 或 TS2 有序集的同一个伪端口在所有通道上发送下面描述的电气空闲退出码型持续 5µs。
- 否则，以下描述的电气空闲退出码型将在检测到电气空闲退出的所有通道上转发。
- 当使用 128b/130b 编码：
  - 一个 EIEOS 。
  - 32 个数据块，每个数据块的有效载荷为 16 个空闲数据符号（00h），已加扰，表示符号 0 到 13。
  - 每个数据块的符号 14 和 15 包含空闲数据符号（00h），已加扰或直流平衡，这是通过对这些数据块应用 Section 4.2.4.1 中的相同规则来确定的。
- 当使用 8b/10b 编码：
  - 错误状态符号设置为 00h 的 Modified Compliance Pattern。
- 现在，此路径正在转发电气空闲退出码型。在这种状态下，通过不存在电气空闲退出来推断电气空闲，请参阅 Table 4-28。路径将继续转发电气空闲退出码型，直到在任何通道上推断出电气空闲为止，否则会发生 48 ms 的超时。如果发生 48 ms 超时，则：
  - RT_LINK_UP 设置为 0b 。
  - 伪端口把它的发送器置于电气空闲状态。

- 两个伪端口的 RT_next_data_rate 和 RT_error_data_rate 都要设置为 2.5GT/s 。
- 接收器检测是在发送电气空闲退出码型并超时的伪端口上执行的，结果如 Section 4.3.4 中所述反向传播。

相对的伪端口上的发送器在发送电气空闲退出码型并超时后，将上述电气空闲退出码型发送了 5μs。

IMPLEMENTATION NOTE

Electrical Idle Exit

在 Retimer 无法解码训练集的错误情况下，会发生转发电气空闲退出的情况。上游端口和下游端口在 Polling.Compliance 和 Recovery.Speed 期间使用电气空闲退出（不解码任何符号）。如果 Retimer 没有转发电气空闲退出，则在某些情况下，上游端口和下游端口的行为将不正确。例如，这可能发生在速度更改为更高的数据速率之后。在这种情况下，需要转发电气空闲退出，以将上游端口和下游端口保持在 Recovery.Speed 的锁定步骤中，以便数据速率将返回到先前的数据速率，而不是从超时到检测到之间的链路断开条件。

当 Retimer 检测到从电气空闲状态退出并开始转发数据时，此时间称为 Retimer 退出延迟，并允许诸如数据速率更改（如果需要），时钟和数据恢复，符号锁定，块对齐， 通道到通道的偏移，接收器调整等。在以下几种情况下，指定了最大的 Retimer 退出延迟：

- 电气空闲和电气空闲退出检测之前和之后的数据速率不会改变。
- 数据速率更改为使用 8b/10b 编码的数据速率。
- 数据速率首次更改为使用 128b/130b 编码的数据速率。
- 数据速率并非首次更改为使用 128b/130b 编码的数据速率。
- 发生数据速率变化时，两个发送器都处于电气空闲状态的时间。

Retimer 允许在电气空闲状态下更改其数据速率，建议 Retimer 在电气空闲状态下开始更改数据速率，以最大程度地减少 Retimer 退出延迟。

Table 4-27 Maximum Retimer Exit Latency

Condition	Link in EI For X μs, where, X < 500 μs	Link in EI for For X ≥ 500 μs
No data rate change	4 μs	4 μs
When forwarding TS1 Ordered Sets at 2.5 GT/s with Lane and Link number equal to PAD.	1 ms	1 ms
Any data rate change to 8b/10b encoding data rate	504 - X μs	4 μs
First data rate change to 128b/130b encoding date rate	1.5 -X ms	1 ms
Subsequent data rate change to 128b/130b encoding date rate	504 -X μs	4 μs

4.3.6.4 Data Rate Change and Determination Rules

基本复位无效后，Retimer 的数据速率设置为 2.5 GT/s。

Retimer 的两个伪端口必须以相同的数据速率运行。如果伪端口将其发送器置于电气空闲状态，则刚刚完成发送的符号将确定变量 RT_next_data_rate 和 RT_error_data_rate 的值。仅当两个伪端口都在电气空闲状态下具有所有通道时，Retimer 才可以更改数据速率。如果两个伪端口都不能对这些变量进行相同的确定，则必须将两个变量都设置为 2.5 GT/s。

公众号：芯片业者相承双

- 如果两个伪端口都在转发非训练序列，则必须将 RT_next_data_rate 设置为当前数据速率。 RT_error_data_rate 必须设置为 2.5 GT/s。 注意：这涵盖了链路已从 L0 进入 L1 的情况。
- 如果两个伪端口都在将 speed_change 位设置为 1b 的情况下转发 TS2 有序集，若：
  - 转发这些 TS2 时，数据速率大于 2.5 GT/s，或者
  - 两个方向数据速率标识符中接收到的最高通用数据速率大于 2.5 GT/s，

  然后必须将 RT_next_data_rate 设置为最高通用数据速率，并将 RT_error_data_rate 设置为当前数据速率。注意：这涵盖了链路已从 Recovery.RcvrCfg 进入 Recovery.Speed 并根据最高通用数据速率更改数据速率的情况。
- 否则，必须将 RT_next_data_rate 设置为 RT_error_data_rate。 RT_error_data_rate 设置为 2.5 GT/s。请注意，这涵盖了两种错误情况：
  - 这表明链路无法以当前数据速率（大于 2.5 GT/s）运行，并且链接将以 2.5 GT/s 数据速率运行，或者，
  - 这表明该链路无法以新的协商数据速率运行，并且将恢复为从 L0 或 L1 进入 Recovery 时的旧数据速率。

4.3.6.5 Electrical Idle Entry Rules

转发模式下的电气空闲进入规则取决于 Retimer 是转发训练集还是非训练集。对此的确定在 Section 4.3.6.1 中描述。

除非另有说明，否则在发送器进入电气空闲之前，它必须始终发送电气空闲有序集序列（EIOSQ）。

如果 Retimer 正在转发训练集，则：

- 如果在通道上接收到 EIOS，则 EIOSQ 在该通道上转发，并且只有该通道将其发送器置于电气空闲状态。
- 如果在通道上推断出电气空闲，则在该通道上发送 EIOSQ 之后，该通道会将其发送器置于电气空闲中。

否则，如果 Retimer 正在转发非训练集，则：

- 如果在任何通道上都收到 EIOS，则 EIOSQ 会在当前正在转发符号的所有通道上转发，并且所有通道都将其发送器置于电气空闲状态。
- 如果在通道上推断出电气空闲，则该通道将其发送器置于电气空闲上，而 EIOSQ 不在该通道上传输。

Retimer 需要推断电气空闲。Table 4-28 中描述了 Retimer 推断电气空闲的标准。

公众号·芯片业者相承双

Table 4-28 Inferring Electrical Idle

State	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s or higher
Forwarding: Non Training Sequence	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window
Forwarding: Training Sequence	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval
Forwarding: Electrical Idle Exit Executing: Force Timeout	Absence of an exit from Electrical Idle in a 2000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval
Forwarding: Loopback Executing: Loopback Slave	Absence of an exit from Electrical Idle in a 128 µs window	N/A	N/A	N/A

4.3.6.6 Transmitter Settings Determination Rules

当数据速率更改为 32.0 GT/s 时，将按以下方式确定 Retimer 发送器设置：

- 如果变量 RT_G5_EQ_complete 设置为 1b：
  - 发送器必须使用适用于 32.0 GT/s 操作的最后一个均衡过程结束时协商的系数设置。
- 否则：
  - 上游伪端口必须使用从在发送器预设设置中以 16.0 GT/s 的速度运行时接收到的连续八个 128b/130b EQ TS2 有序集中注册的 128b/130b 发送器预设值，并且必须确保其满足 Section 4.2.3.2 中的预设定义。接收到保留或不支持发送器预设值的通道必须使用特定于实现的方法，以选择支持的发送器预设设置，以便在 32.0 GT/s 时开始传输。
  - 当下游伪端口开始以 32.0 GT/s 的速率传输时，将以实现特定方式确定其发送器设置。

当以下条件发生时，RT_G5_EQ_complete 变量会被设置为 1b：

- 在 32.0 GT/s 下收到了两个连续的 EC = 01b 的 TS1 有序集。

当以下任意条件发生时，RT_G5_EQ_complete 变量会被设置为 0b：

- RT_LinkUp 变量被设置为 0b 。
- 伪端口以 16.0 GT/s 的速度运行，并且在上游伪端口的任何通道上都接收到八个连续的 128b/130b EQ TS2 有序集。128b/130b Transmitter Preset 字段中的值已注册，以供以后以 32.0 GT/s 的速度用于该通道。

当数据速率更改为 16.0 GT/s 时，将按以下方式确定 Retimer 发送器设置：

- 如果变量 RT_G4_EQ_complete 设置为 1b：
  - 发送器必须使用适用于 16.0 GT/s 操作的最后一个均衡过程结束时协商的系数设置。
- 否则：
  - 上游伪端口必须使用从在发送器预设设置中以 16.0 GT/s 的速度运行时接收到的连续八个 128b/130b EQ TS2 有序集中

公众号：芯片业者相承双

注册的 128b/130b 发送器预设值，并且必须确保其满足 Section 8.3.3.3 中的预设定义。接收到保留或不支持发送器预设值的通道必须使用特定于实现的方法，以选择支持的发送器预设设置，以便在 16.0 GT/s 时开始传输。

- 当下游伪端口开始以 16.0 GT/s 的速率传输时，将以实现特定方式确定其发送器设置。

当以下条件发生时，RT_G4_EQ_complete 变量会被设置为 1b：

- 在 16.0 GT/s 下收到了两个连续的 EC = 01b 的 TS1 有序集。

当以下任意条件发生时，RT_G4_EQ_complete 变量会被设置为 0b：

- RT_LinkUp 变量被设置为 0b 。
- 伪端口以 16.0 GT/s 的速度运行，并且在上游伪端口的任何通道上都接收到八个连续的 128b/130b EQ TS2 有序集。128b/130b Transmitter Preset 字段中的值已注册，以供以后以 16.0 GT/s 的速度用于该通道。

当数据速率更改为 8.0 GT/s 时，将按以下方式确定 Retimer 发送器设置：

- 如果变量 RT_G3_EQ_complete 设置为 1b：
  - 发送器必须使用适用于 8.0 GT/s 操作的最后一个均衡过程结束时协商的系数设置。
- 否则：
  - 上游伪端口必须使用从在发送器预设设置中以 8.0 GT/s 的速度运行时接收到的连续八个 128b/130b EQ TS2 有序集中注册的 128b/130b 发送器预设值，并且必须确保其满足 Section 8.3.3 中的预设定义。接收到保留或不支持发送器预设值的通道必须使用特定于实现的方法，以选择支持的发送器预设设置，以便在 8.0 GT/s 时开始传输。上游伪端口可以选择使用它在那些 EQ TS2 有序集中注册的 8.0 GT/s Receiver preset hint 值。
  - 当下游伪端口开始以 8.0 GT/s 的速率传输时，将以实现特定方式确定其发送器设置。

当以下条件发生时，RT_G3_EQ_complete 变量会被设置为 1b：

- 在 8.0 GT/s 下收到了两个连续的 EC = 01b 的 TS1 有序集。

当以下任意条件发生时，RT_G3_EQ_complete 变量会被设置为 0b：

- RT_LinkUp 变量被设置为 0b 。
- 上游伪端口的任何通道上都接收到八个连续的 128b/130b EQ TS2 有序集。8.0 GT/s Transmitter Preset 和可选的 8.0 GT/s Receiver Preset Hint 字段的值已注册，以供以后以 8.0 GT/s 的速度用于该通道。

当数据速率更改为 5.0 GT/s 时，将按以下方式确定 Retimer 发送器设置：

- 在最新系列的 TS2 有序集合中，在进入“电气空闲”之前，根据在八个连续的 TS2 有序集合中接收到的 Selectable De-emphasis 字段，上游伪端口必须将其发送器设置为-3.5 dB 或-6.0 dB 。
- 下游伪端口以特定于实现的方式将其发送器设置为-3.5 dB 或-6.0 dB 。

4.3.6.7 Ordered Set Modification Rules

有序集在转发时可能根据以下规则修改某些字段：

- 除本规范中明确允许/要求修改的字段外，Retimer 均不得修改任何字段。
- LF：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 LF 字段。新的值由 Retimer 以实现特定方式确定。

公众号·芯片业务由承双

- FS：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 FS 字段。新的值由 Retimer 以实现特定方式确定。
- Pre-Cursor Coefficient：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 Pre-Cursor Coefficient 字段。新的值由当前发送器的设置确定。
- Cursor Coefficient：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 Cursor Coefficient 字段。新的值由当前发送器的设置确定。
- Post-Cursor Coefficient：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 Post-Cursor Coefficient 字段。新的值由当前发送器的设置确定。
- Parity：Retimer 会覆盖转发的 TS1 有序集中的 Parityt 字段。该位是 Symbol 6、7 和 8 的所有位以及 Symbol 9 的 6:0 的偶数奇偶校验。
- Transmitter Preset：Retimer 会覆盖两个方向上发送的 TS1 有序集中的 Transmitter Preset 字段。如果发送器正在使用发送器的预设设置，则该值等于当前设置，否则建议将 Transmitter Preset 字段设置为用于当前数据速率的最新发送器预设设置。

  Retimer 允许执行以下操作：

  - 覆盖任一方向上 EQ TS1 Order Set 中的 Transmitter Preset 字段的值。
  - 覆盖下行方向上 EQ TS2 Order Set 中的 8.0 GT/s Transmitter Preset 字段的值。
  - 覆盖下行方向上 128b/130b EQ TS2 Ordered Set 中的 128b/130b Transmitter Preset 值。

  8.0 GT/s Transmitter Preset 和 128b/130b Transmitter Preset 字段的新值由 Retimer 以实现特定方式确定。

  在均衡到 16.0 GT/s（即当前数据速率为 8.0 GT/s）的 Phase 0 或均衡到 32.0 GT/s（即当前数据速率为 16.0 GT/s）的 Phase 0 期间，允许 Retimer 在上游方向执行以下操作：

  - 转发收到的 TS2 Order Set 。
  - 把 TS2 Order Set 转化成 128b/130b EQ TS2 Ordered Set，128b/130b Transmitter Preset 的值由 Retimer 的实现决定。
  - 无修改的转发 128b/130b EQ TS2 Ordered Set，128b/130b Transmitter Preset 的值由 Retimer 的实现决定。
  - 把 128b/130b EQ TS2 Ordered Set 转化成 TS2 Order Set 。
- Receiver Preset Hint：Retimer 允许执行以下操作：

  - 覆盖任一方向上 EQ TS1 Order Set 中的 Receiver Preset 字段的值。
  - 覆盖下行方向上 EQ TS2 Order Set 中的 8.0 GT/s Receiver Preset 字段的值。

  8.0 GT/s Receiver Preset 和 128b/130b Receiver Preset 字段的新值由 Retimer 以实现特定方式确定。
- SKP Ordered Set：允许 Retimer 调整在两个方向上传输的 SKP 有序集的长度。Retimer 必须在所有通道上执行相同的调整。当使用 8b/10b 编码进行操作时，允许 Retimer 添加或删除 SKP 有序集的一个 SKP 符号。当使用 128b/130b 编码进行操作时，允许 Retimer 添加或删除 SKP 有序集的 4 个 SKP 符号。
- Control SKP Ordered Set：当 Retimer 处于 16.0 GT/s 或更高的转发模式时，Retimer 必须根据接收到的奇偶校验，修改 Control SKP Ordered Set 的 First Retimer Data Parity 或 Second Retimer Data Parity。接收的偶数奇偶校验是在每个通道上独立计算的，如下所示：

  - 当数据速率发生变化时，奇偶校验被初始化。
  - 当收到一个 SDS Ordered Set 时，奇偶校验被初始化。

公众号·芯片业者相承双

- 在执行解扰之前，用数据块有效负载的每一位更新奇偶校验。
- 收到 Control SKP Ordered Set，将初始化奇偶校验。但是，收到标准 SKP 有序集时，不会初始化奇偶校验。

如果伪端口在以 2.5 GT/s 的速度运行时检测到该伪端口最近接收到的两个连续 TS2 或 EQ TS2 有序集中的 Retimer Present 字段为 0b，则伪端口接收器会在转发 Control SKP Ordered Set 时修改 First Retimer Data Parity，否则伪端口接收器会在转发 Control SKP Ordered Set 时修改 Second Retimer Data Parity。

Retimer 必须按照 Section 4.2.13 中的描述，在上游方向上修改 Control SKP Ordered Set 的 Symbol 4 * N + 1、4 * N + 2 和 4 * N + 3 。

有关 Control SKP Ordered Set 的定义，请参见 Section 4.2.7.2 。

- Selectable De-emphasis：Retimer 可以覆盖两个方向上发送的 TS1 或 TS2 有序集中的 Selectable De-emphasis。新的值由 Retimer 以实现特定方式确定。
- The Data Rate Identifier： Retimer 必须将 Data Rate Identifier Symbol 的 Data Rate Supported 字段设置为与接收到的有序集中通告的数据速率及其自身的最大支持数据速率一致，即，它将所有 Symbol 4 的 Bit[5:0]中不支持的速率清零。 Retimer 必须支持低于并包括其最大支持数据速率的所有数据速率。 基本复位后，Retimer 将一次确定最大支持的数据速率。
- DC Balance：当使用 128b/130b 编码进行操作时，Retimer 将跟踪其伪端口发送器的 DC 平衡，并按照 Section 4.2.4.1 中的描述发送 DC Balance 符号。
- Retimer Present：当以 2.5 GT/s 的速度运行时，Retimer 必须将所有转发的 TS2 和 EQ TS2 有序集的 Retimer Present 字段设置为 1b。
- Two Retimers Present：如果 Retimer 支持 16.0 GT/s，则当以 2.5 GT/s 的速度运行时，如果 Retimer 接收到 Retimer Present 设置为 1b 的 TS2 或 EQ TS2 有序集合，则必须将所有转发的 TS2 和 EQ TS2 有序集合的 Two Retimers Present 位设置为 1。如果 Retimer 不支持 16.0 GT/s，则当以 2.5 GT/s 的速度运行时，如果 Retimer 接收到带有 TS2 或 EQ TS2 有序集的所有转发 TS2 和 EQ TS2 的 Two Retimers Present 位，则允许 Two Retimers Present 将其置位。
- Loopback：当有选择地在执行模式下支持 Slave 环回时，转发训练集时必须将 Loopback 位清零。
- Enhanced Link Behavior Control：如果 Retimer 支持 32.0 GT/s，则当以 2.5GT/s 的速率运行时，Retimer 必须如下设置所有转发的 TS1、TS2、EQ TS1 和 EQ TS2 有序集的 Enhanced Link Behavior Control 位：
  - 当 Retimer 支持 Modified TS1/TS2 Ordered Set，并且在收到的有序集中将 Enhanced Link Behavior Control 位设置为 11b 进行转发时，将其设置为 11b。
  - 当 Retimer 不支持均衡并且在接收到的用于转发的有序集中，Enhanced Link Behavior Control 位设置为 10b 时，设置为 10b。
  - 当 Retimer 支持均衡旁路到最高速率并且 Enhanced Link Behavior Control 字段在收到的用于转发的有序集合中设置为 01b 时，将其设置为 01b。
  - 其他情况设置为 00b 。

4.3.6.8 DLLP, TLP, and Logical Idle Modification Rules

除非另有说明，否则转发 DLLP、TLP 和逻辑空闲时不会对任何符号进行任何修改。

公众号：芯片业者的承义

4.3.6.9 8b/10b Encoding Rules

Retimer 应满足 Section 4.2.1.1.3 的要求，但以下情况除外：

- 当 Retimer 正在转发并且在接收到的数据中检测到 8b/10b 解码错误或不一致错误时，在已转发数据中具有错误不一致的 D21.3 符号将替换为带错误的符号。
- Section 4.2.1.1.3 中的该条款不适用：如果在与错误的运行差异相对应的列中找到了收到的符号，或者如果该符号与任一列都不对应，则物理层必须通知数据链路层： 收到的符号无效。这是一个接收器错误，会报告一个与端口相关的错误（请参见 Section 6.2）。

IMPLEMENTATION NOTE

Retimer Transmitter Disparity

Retimer 必须修改 TS1 和 TS2 有序集的某些字段（例如 Receiver Preset Hint、Transmitter Preset），因此 Retimer 必须重新计算运行的差异。简单地使用接收到的符号的差异可能会导致运行差异中的错误。例如，一些 8b/10b 编码具有 6 个 1 和 4 个零用于正差异，而其他代码则具有 5 个 1 个和 5 个零。

4.3.6.10 8b/10b Scrambling Rules

如 Section 4.3.6.2 所述，当使用 8b/10b 编码时，需要 Retimer 来确定是否禁用了加扰。

4.3.6.11 Hot Reset Rules

如果上游伪端口的任何通道接收到两个连续的 TS1 有序集，且 Hot Reset 位设置为 1b，Disable Link 和 Loopback 位均设置为 0b，则两个伪端口都将接收 EIOS 或推断任何通道上的电气空闲 ，即正在接收 TS1 有序集的设备，Retimer 将执行以下操作：

- RT_LinkUp 变量清 0 。
- 将两个伪端口的发送器都置于电气空闲状态。
- RT_next_data_rate 变量设置为 2.5GT/s 。
- RT_error_data_rate 变量设置为 2.5GT/s 。
- 在两个伪端口上的每个通道上等待从电气空闲退出。

Retimer 不在任何一个伪端口上执行接收器检测。

4.3.6.12 Disable Link Rules

如果上游伪端口的任何通道接收到两个连续的 TS1 有序集，且 Disable Link 位设置为 1b，并且 Hot Reset 和 Loopback 位均设置为 0b，则两个伪端口要么接收 EIOS，要么推断任何通道上的电气空闲 ，即正在接收 TS1 的有序集，Retimer 将执行以下操作：

- RT_LinkUp 变量清 0 。
- 将两个伪端口的发送器都置于电气空闲状态。
- RT_next_data_rate 变量设置为 2.5GT/s 。
- RT_error_data_rate 变量设置为 2.5GT/s 。

公众号·芯片业者相承双

- 在两个伪端口上的每个通道上等待从电气空闲退出。

Retimer 不在任何一个伪端口上执行接收器检测。

4.3.6.13 Loopback

如果任何通道接收到两个连续的 TS1 有序集，且其 Loopback 位等于 1b，并且 Hot Reset 和 Disable Link 位均设置为 0b，并且未以实现特定方式配置执行 Slave 环回的能力，则 Retimer 将遵循这些附加规则。这些规则的目的是当 Loopback Master 和 Loopback Slave 之间存在一个 Retimer（或两个 Retimer）时允许互操作。

- 接收到 TS1 有序集且 Loopback 位设置为 1b 的伪端口用作 Loopback Slave（另一个伪端口用作 Loopback Master）。上游路径定义为作为 Loopback Master 的伪端口到作为 Loopback Slave 的伪端口。另一个路径是下游路径。
- 建立后，如果通道失去维持符号锁定或块对齐的能力，则通道必须在此状态下继续传输符号。
- 当使用 8b/10b 编码并且丢失符号锁定时，Retimer 必须尝试重新实现符号锁定。
- 当使用 128b/130b 编码并且丢失块对齐时，Retimer 必须尝试通过 SKP 有序集重新实现块对齐。
- 如果在链路组件处于 Configuration.Linkwidth.Start 时进入了 Loopback 状态，则通过在任意两个连续的 TS1 有序集或两个连续的 TS2 有序集中接收到的数据速率，确定链路所支持的数据速率的最高公共数据速率。切换到 Forwarding.Loopback 时正在接收 TS1 或 TS2 有序集的 Lane。如果当前数据速率不是最高的通用数据速率，则：
  - 等待任何通道接收 EIOS，然后将发送器置于在该路径的电气空闲中。
  - 当所有发送器都处于电气空闲状态时，请按照先前确定的方法调整数据速率。
  - 如果新数据速率为 5.0 GT/s，则按与 Section 4.2.6.10.1 中所述相同的方式确定 Selectable De-emphasis 的值。
  - 如果新的数据速率使用 128b/130b 编码，则按与 Section 4.2.6.10.1 中所述相同的方式确定 Transmitter preset 值。
  - 在下游路径中，等待在每个通道上检测到电气空闲退出，然后在逐个通道的基础上收到两个连续的 TS1 有序集时开始转发。对于 Retimer 退出延迟，这被认为是首次达到此数据速率。
  - 在上游路径中，如果未确定将 Slave 定向到此状态的 TS1 有序集的 Compliance Receive 位，则等待在每个通道上检测到电气空闲退出，并在通道上接收到两个连续的 TS1 有序集时开始转发。 对于 Retimer 退出延迟，这被认为是首次达到此数据速率。
- 在上游路径中，如果将 Slave 定向到此状态的 TS1 有序集的 Compliance Receive 位设置为 1b，则等待在每个通道上检测到电气空闲退出，并立即开始逐个通道转发。对于 Retimer 退出延迟，这被认为是首次达到此数据速率。
- 如果在任何通道上收到四个 EIOS（如果当前数据速率为 2.5 GT/s，则为一个 EIOS），则：
  - 在未接收 EIOS 的伪端口上传输 TS1 有序集的每个通道上传输八个 EIOS，并将发送器置于电气空闲状态。
- 当两个伪端口都将它们的发送器都置于电气空闲状态时，则：
  - RT_next_data_rate 变量设置为 2.5GT/s 。
  - RT_error_data_rate 变量设置为 2.5GT/s 。
  - 除非再次满足进入本节的规则，否则 Loopback 的附加规则将不再适用。

公众号·芯片业者社区汉

4.3.6.14 Compliance Receive Rules

如果任一通道接收到八个连续的 TS1 有序集（或其补码），且 Compliance Receive 位设置为 1b，Loopback 位设置为 0b，则 Retimer 遵循这些附加规则。以下规则的目的是，当链路处于 Polling.Active 时，当上游端口或下游端口发送的 TS1 有序集中的 Compliance Receive 位被置位并且 Loopback 位被清除时，该链路支持通过 Retimer 进行链路操作。

- 伪端口 A 定义为第一个伪端口，该端口接收八个连续的 TS1 有序集（或它们的补码），并且 Compliance Receive 位置 1，Loopback 位清零。伪端口 B 定义为另一个伪端口。
- Retimer 通过检查在每个伪端口上接收的 TS1 有序集中的 data rate identify 以及 Retimer 支持的最大数据速率，来确定链路的最高公共数据速率。
- 如果最高公共数据速率等于 5.0 GT/s，则：
  - Retimer 必须按照 Section 4.3.6.4 中所述将其数据速率更改为 5.0 GT/s。
  - Retimer 伪端口 A 必须根据在八个连续的 TS1 有序集中接收到的 Selectable De-emphasis 位来设置其去加重。
  - Retimer 伪端口 B 必须以实现特定的方式设置其去加重。
- 如果最高公共数据速率等于 8.0 GT/s 或更高，则：
  - 如 Section 4.3.6.4 所述，Retimer 必须将其数据速率更改为合适的值。
  - Lane number 如 Section 4.2.11 所述确定。
  - Retimer 伪端口 A 必须将其每个通道上的发送器系数设置为在八个连续的 TS1 有序集的 Symbol 6 中通告的发送器预设值，并且该值必须由发送器使用（使用在那些 TS1 中通告的接收器预设提示值）。如果公共数据速率为 8.0 GT/s 或更高，则任何未接收到八个连续的带有发送器预设信息的 TS1 有序集的通道都可以以实现特定的方式使用任何受支持的发送器预设设置。
  - Retimer 伪端口 B 必须以实现特定的方式设置其发送器和接收器均衡。
- 当 Retimer 锁定到 Modified Compliance Pattern 时，它必须转发该 Pattern。这在每个方向上的每个通道上独立发生。如果通道的接收器丢失符号锁定或块对齐，则关联的发送器（即，相对的伪端口上的同一通道）将继续转发数据。
- 一旦锁定到 Pattern，Retimer 将对每个通道保留接收到的符号错误的内部计数。在每个通道的基础上，允许以实现特定的方式读取 Pattern 锁定和通道错误。
- 当使用 128b/130b 编码进行操作时，默认情况下不带任何错误地转发有错误的符号，或者可以选择对其进行纠正以消除错误污染。必须支持默认行为，选择可选行为的方法（如果支持）是特定于实现的。
- 当使用 8b/10b 编码进行操作时，默认情况下，带有错误的符号将替换为具有不正确差异的 D21.3 符号，或者可以选择对其进行纠正以消除错误污染。必须支持默认行为，选择可选行为的方法（如果支持）是特定于实现的。
- 默认情况下，使用 8b/10b 编码时的错误状态符号或使用 128b/130b 编码时的 Error_Status 字段将默认未经修改地转发，或者可以选择重新定义，因为它由 Retimer 发送。必须支持默认行为，选择可选行为的方法（如果支持）是特定于实现的。
- 如果任何一个通道在任一伪端口上接收到 EIOS，则：
  - 在未接收到 EIOS 的伪端口的每个通道上传输 EIOS，并将发送器置于电气空闲状态。将另一个伪端口的发送器置于电气空闲状态；EIOS 不会通过其他伪端口进行传输。
  - RT_next_data_rate 变量设置为 2.5GT/s 。

公众号·芯片业者相承双

- RT_error_data_rate 变量设置为 2.5GT/s 。
- 除非再次满足进入本节的规则，否则 Compliance Receive 的附加规则将不再适用。

4.3.6.15 Enter Compliance Rules

如果由于 Hot Reset 导致 Retimer 在进入电气空闲状态后退出电气空闲状态，并且 Retimer 中的 Retimer Enter Compliance 位被置位，则 Retimer 将遵循这些附加规则。 以下规则的目的是当 Link partner 由于链接组件中的 Link Control 2 寄存器中的 Enter Compliance 位都设置为 1b 以及链路上发生热复位时，通过 Retimer 支持链路操作。如果 Link partner 退出时检测到进入不是由热复位引起的，则 Retimer 不支持链路操作。

Retimer 必须以实现特定的方式支持以下寄存器字段：

- Retimer Target Link Speed
  - 每个 Retimer 都有这样一个字段。
  - Type = RWS
  - Size = 3 bit
  - Default = 001b
  - 编码为：
    - 001b = 2.5 GT/s
    - 010b = 5.0 GT/s
    - 011b = 8.0 GT/s
    - 100b = 16.0 GT/s
    - 101b - 32.0 GT/s
- Retimer Transmit Margin
  - 每个伪端口都有这样一个字段。
  - Type = RWS
  - Size = 3 bit
  - Default = 000b
  - 编码为：
    - 000b = 正常工作范围
    - 001b-111b = 如 Section 8.3.4 中所定义，并非所有编码都需要实现 。
- Retimer Enter Compliance
  - 每个 Retimer 都有这样一个字段。
  - Type = RWS
  - Size = 1 bit

公众号：芯片业者柏承双

- Default = 0b
- 编码为：
  - 0b = 不进入一致性模式
  - 1b = 进入一致性模式

- Retimer Enter Modified Compliance
  - 每个 Retimer 都有这样一个字段。
  - Type = RWS
  - Size = 1 bit
  - Default = 0b
  - 编码为：
    - 0b = 不进入修改的一致性模式
    - 1b = 进入修改的一致性模式
- Retimer Compliance SOS
  - 每个 Retimer 都有这样一个字段。
  - Type = RWS
  - Size = 1 bit
  - Default = 0b
  - 编码为：
    - 0b = 发送使用 8b/10b 编码的 Compliance Pattern 或 Modified Compliance Pattern 时，序列之间不发送任何 SKP 有序集。
    - 1b = 发送使用 8b/10b 编码的 Compliance Pattern 或 Modified Compliance Pattern 时，序列之间发送两个 SKP 有序集。
- Retimer Compliance Preset/De-emphasis
  - 每个伪端口都有这样一个字段。
  - Type = RWS
  - Size = 4 bit
  - Default = 0000b
  - Retimer 目标链路速度为 5.0 GT/s 时的编码为：
    - 0000b = -6.0dB
    - 0001b = -3.5dB
  - Retimer 目标链路速度为 8.0 GT/s 或更高时的编码为：Transmitter Preset。

公众号·芯片业者相承双

当 Retimer 退出热复位时，Retimer 必须检查上述寄存器中的值。 如果 Retimer Enter Compliance 位设置为 1b，则适用以下规则：

- Retimer 根据 Retimer Target Link Speed 定义调整其数据速率。在发生数据速率更改之前，不会转发任何数据。
- Retimer 根据 Retimer Compliance Preset/De-emphasis，在每个伪端口的基础上配置其发送器。
- 当 Retimer 已锁定到 Compliance Pattern 或 Modified Compliance Pattern 时，它必须转发该 Pattern。如果 Retimer Enter Modified Compliance 位被清 0，则 Retimer 必须检索 Compliance Pattern；如果 Retimer Enter Modified Compliance 位设置为 1，则 Retimer 必须检索 Modified Compliance Pattern。这在每个方向上的每个通道上独立发生。
- 使用 8b/10b 编码时，特定的 Lane 的接收器通过查找任何出现的 Modified Compliance Pattern 或 Compliance Pattern 来独立确定是否成功锁定了输入进来的的修改后的 Modified Compliance Pattern 或 Compliance Pattern。
  - 上述的事件定义为 Section 4.2.8 中定义的 8b/10b 符号序列。
  - 对于 Modified Compliance Pattern，错误状态符号将不用于锁定过程，因为它们在任何给定时刻都是未定义的。
  - 必须在收到 Modified Compliance Pattern 后的 1.0 ms 内实现锁定。
- 当使用 128b/130b 编码时，每个通道在达到块对齐时将独立判定 Pattern 锁定，如 Section 4.2.2.2.1 中所述。
  - 必须在收到 Modified Compliance Pattern 或 Compliance Pattern 后的 1.5 ms 内实现锁定。
- 使用 128b/130b 编码时，有错误的符号默认情况下未经修改就转发，或者可以选择进行纠正以消除错误污染。必须支持默认行为，选择可选行为的方法（如果支持）是特定于实现的。
- 当使用 8b/10b 编码时，默认情况下，带有错误的符号将替换为具有不正确差异的 D21.3 符号，或者可以选择对其进行纠正以消除错误污染。必须支持默认行为。
- 一旦锁定，Retimer 将在每个通道的基础上保留接收到的符号错误的内部计数。如果 Retimer 正在转发 Modified Compliance Pattern，则默认情况下将转发未经修改的使用 8b/10b 编码时的错误状态符号或使用 128b/130b 编码时的 Error_Status 字段，或者可以选择重新定义它，因为它由 Retimer 发送。必须支持默认行为，选择可选行为的方法（如果支持）是特定于实现的。允许 Retimer 以实现特定方式在每个通道的基础上提供 Pattern 锁定和通道错误信息。
- 如果在任何通道上都收到 EIOS，则：
  - 该方向上的所有通道都传输 8 个 EIOS，然后该方向上的所有发送器都被置于电气空闲状态。
  - 当两个方向都发送了 8 个 EIOS 并将其发送器置于电气空闲状态时，数据速率将更改为 2.5 GT/s。
  - RT_next_data_rate 变量设置为 2.5GT/s 。
  - RT_error_data_rate 变量设置为 2.5GT/s 。
  - Retimer Enter Compliance 和 Retimer Enter Modified Compliance 字段都设置为 0b 。
  - 除非再次满足进入本节的规则，否则以上附加规则将不再适用。

4.3.7 Execution Mode Rules

在执行模式下，Retimer 直接控制由伪端口传输的所有信息，而不是转发信息。

4.3.7.1 CompLoadBoard Rules

当 Retimer 处于 CompLoadBoard（一致性负载板）状态时，两个伪端口都将其作为常规端口执行协议，从而在每个端口的以下小节中指定生成符号，而不是从一个伪端口转发到另一个伪端口。

IMPLEMENTATION NOTE

Passive Load on Transmitter

当一个被动负载放置在一个伪端口上，而另一个伪端口正在接收业务流时，将进入此状态。

4.3.7.1.1 CompLoadBoard.Entry

- RT_LinkUp = 0b 。
- 收到 Compliance Pattern 的伪端口（Pseudo Port A）执行以下操作：
  - 数据速率保持为 2.5GT/s 。
  - 发送器被置于电气空闲状态。
  - 接收器忽略输入的符号。
- 另一个伪端口（Pseudo Port B）执行以下操作：
  - 数据速率保持为 2.5GT/s 。
  - 发送器被置于电气空闲状态。按 Section 8.4.5.7 所述在所有 Lane 上执行接收器检测。
  - 接收器忽略输入的符号。
- 如果 Pseudo Port B 的接收端检测确定在任何通道上都没有连接到接收端，则两个伪端口的下一个状态是 CompLoadBoard.Exit 。
- 否则两个伪端口的下一个状态是 CompLoadBoard.Pattern 。

4.3.7.1.2 CompLoadBoard.Pattern

当 Retimer 进入 CompLoadBoard.Pattern 时，将发生以下情况：

- Pseudo Port A 执行以下操作：
  - 发送器保持电气空闲状态。
  - 接收器忽略输入的符号。
- Pseudo Port B 执行以下操作：
  - 发送器在按照 Section 4.2.6.2.2 中确定的数据速率和去加重/预设级别在检测到接收器的所有通道上发送 Compliance Pattern（即，每次连续进入 CompLoadBoard 都会推进该模式），除了在 Polling.Configuration 期间 Setting 未设置为 Setting #1 的情况。如果自基本复位退出以来，Pseudo Port B 已收到 TS1 或 TS2 有序集（或其补码），则不使用 Setting #26 及更高版本。如果新的数据速率不是 2.5 GT/s，则在更改数据速率之前将发送器置于电气空闲状态。电气空闲时间段必须大于 1 ms，但不得超过 2 ms。

公众号：芯片业者柏承双

- 如果 Pseudo Port B 检测到任何检测到接收器的通道的电气空闲退出，则两个伪端口的下一个状态是 CompLoadBoard.Exit 。

4.3.7.1.3 CompLoadBoard.Exit

当 Retimer 进入 CompLoadBoard.Exit 时，会发生以下情况：

- 对于 Pseudo Port A：
  - 数据速率保持为 2.5GT/s 。
  - 发送器在 Pseudo Port B 上检测到电气空闲退出的通道上发送 Section 4.3.6.3 中描述的 Electrical Idle Exit pattern 持续 1 ms。然后将发送器置于电气空闲状态。
  - 接收器忽略输入的符号。
- 对于 Pseudo Port B：
  - 如果发送器以 2.5 GT/s 以外的速率发送，则发送器将发送八个连续的 EIOS。
  - 发送器被置于电气空闲状态。如果发送器以 2.5 GT/s 以外的速率发送，则电气空闲时间必须至少为 1.0 ms。
  - 如果尚未将数据速率更改为 2.5 GT/s，则将数据速率更改为 2.5 GT/s。
- 两个伪端口均处于转发模式。

IMPLEMENTATION NOTE
TS1 Ordered Sets in Forwarding mode

进入转发模式后，可能会发生以下两种情况之一：

- 接收 TS1 有序集并将其从伪端口的 B 接收器转发到伪端口的 A 发射器。链路训练继续进行。
- 或：未接收到 TS1 有序集，因为正在从一致性负载板上的一条通道上接收到 100 MHz 脉冲，从而进入了一致性模式。在这种情况下，当连接到伪端口 A 的设备从 Polling.Active 超时到 Polling.Compliance 超时时， Retimer 必须从转发模式转换为 CompLoadBoard。Retimer 将每个条目上的一致性模式推进到 CompLoadBoard。

4.3.7.2 Link Equalization Rules

在执行模式下执行链路均衡时，伪端口充当常规端口，在每个端口上生成符号，而不是从一个伪端口转发到另一个伪端口。当 Retimer 处于执行模式时，它必须使用存储在 RT_captured_lane_number 和 RT_captured_link_number 中的通道号和链路号。

按照 Section 4.3.5 中描述的切换到执行模式的过程，在链路上的上游端口和下游端口协商进入均衡过程的 Phase 2 时进入此模式。

4.3.7.2.1 Downstream Lanes

如果下游伪端口要调整上游端口的发送器，则必须存储在上游端口处于 Phase 1 时在两个连续的 TS1 有序集中接收到的 LF 和 FS 值，以便在 Phase 3 期间使用。

4.3.7.2.1.1 Phase 2

除以下内容外，发送器的行为与 Section 4.2.6.4.2.1.2 中所述的相同：

公众号：芯片业者柏承汉

- 如果操作的数据速率为 16.0 GT/s 或更高，则当上游伪端口状态为 Phase 2 Active 时，将发送的 TS1 有序集的 Retimer Equalization Extend 字段设置为 1b，而当上游伪端口状态为 Phase 2 Passive 时把 Retimer Equalization Extend 字段设置为 0b。
- 如果所有已配置的通道都接收到两个连续的 TS1 有序集且 EC = 11b，则下一阶段为 Phase 3 Active。
- 否则，下一个状态是在 32 ms 超时后的强制超时，容差为-0 ms 和+4 ms。

4.3.7.2.1.2 Phase 3 Active

如果操作的数据速率为 8.0 GT/s，则发送器的行为与 Section 4.2.6.4.2.1.3 中所述的相同，不同的是 24 ms 的超时为 2.5 ms，如下所示：

- 如果所有已配置的通道均在其最佳设置下运行，则下一阶段为 Phase 3 Passive 。
- 否则，下一个状态是 2.5 ms 超时后的 Force Timeout，容差为-0 ms 和+0.1 ms 。

如果数据操作速率为 16.0 GT/s 或更高，则发射机的行为与 Section 4.2.6.4.2.1.3 中所述的相同，不同的是 24 ms 超时为 22 ms，如下所示：

- 发送的 TS1 Order Set 中的 Retimer Equalization Extend 字段被设置为 0b 。
- 如果所有已配置的通道均以其最佳设置运行，并且所有已配置的通道接收到两个连续的 TS1 有序集，且 Retimer Equalization Extend 字段设置为 0b，则下一阶段为 Phase 3 Passive 。
- 否则，下一个状态是 22 ms 超时后的 Force Timeout，容差为-0 ms 和+1.0 ms。

4.3.7.2.1.3 Phase 3 Passive

- 发送器发送 EC = 11b，Retimer Equalization Extend = 0b 的 TS1 有序集，并且 Transmitter Preset 字段和 Coefficient 字段不得与在 Phase 3 Active 中发送的最终值有变化。
- 当上游伪端口退出 Phase 3 时，发送器切换到转发模式。

4.3.7.2.2 Upstream Lanes

如果上游伪端口要调整下游端口的发送器，则必须存储当下游端口处于 Phase 1 时在两个连续的 TS1 有序集中接收到的 LF 和 FS 值，以便在 Phase 2 中使用。

4.3.7.2.2.1 Phase 2 Active

如果操作的数据速率为 8.0 GT/ s，则发送器的行为与 Section 4.2.6.4.2.2.3 中所述的相同，除了 24 ms 超时为 2.5 ms，如下所示：

- 如果所有已配置的通道均在其最佳设置下运行，则下一个状态为 Phase 2 Passive 。
- 否则，下一个状态是在 2.5 ms 超时后的 Force Timeout，容差为-0 ms 和+0.1 ms 。

如果数据操作速率为 16.0 GT/s 或更高，则发送器的行为与 Section 4.2.6.4.2.2.3 中所述的相同，不同的是 24 ms 超时为 22 ms，如下所示：

- 发送的 TS1 Ordered Set 的 Retimer Equalization Extend 字段设置为 0b 。
- 如果所有已配置的通道均以其最佳设置运行，并且所有已配置的通道接收到两个连续的 TS1 有序集，且 Retimer Equalization Extend 字段设置为 0b，则下一阶段为 Phase 2 Passive 。

公众号：芯片业者相承双

- 否则，下一个状态是在 2.5 ms 超时后的 Force Timeout，容差为-0 ms 和+0.1 ms 。

4.3.7.2.2.2 Phase 2 Passive

- 发送器发送 EC = 10b，Retimer Equalization Extend = 0b 的 TS1 Ordered Set，并且 Transmitter Preset 字段和 Coefficient 字段不得与在 Phase 2 Active 中发送的最终值相同。
- 如果操作数据速率为 8.0 GT/s，则当下游伪端口已完成 Phase 3 Active 时，下一个状态为 Phase 3 。
- 如果操作数据速率为 16.0 GT/s 或更高，则当下游伪端口已启动 Phase 3 Active，下一个状态为 Phase 3。

4.3.7.2.2.3 Phase 3

发送器遵循 Section 4.2.6.4.2.2.4 中有关上游通道的 Phase 3 规则，但以下情况除外：

- 如果操作的数据速率为 16.0 GT/s 或更高，则当下游伪端口状态为 Phase 3 时，将发送的 TS1 有序集的 Retimer Equalization Extend 字段设置为 1b，并在下游伪端口状态下将其设置为 0b。端口状态为 Phase 3 Passive 。
- 如果所有已配置的通道都接收到两个连续的 TS1 有序集，且 EC = 00b，则 Retimer 切换到转发模式。
- 否则，下一个状态是在超时 32 ms 之后的 Force Timeout，其公差为-0 ms 和+4 ms 。

4.3.7.2.3 Force Timeout

- 两个伪端口均以当前数据速率至少在 1.0 ms 内传输 Section 4.3.6.3 中描述的 Electrical Idle Exit Pattern 。
- 如果在任何通道上，接收端通过未检测到电气空闲的退出而接收到 EIOS 或推断出电气空闲（请参阅 Table 4-28），则在相对的伪端口的所有通道上的发送器都会发送 EIOSQ，然后把发送器置于电气空闲状态。
- 如果两个路径都将其发送器置于电气空闲状态，则在两个伪端口上将 RT_next_data_rate 设置为 RT_error_data_rate，并将 RT_error_data_rate 设置为 2.5 GT/s，并且 Retimer 进入转发模式。
  - 在转发数据之前，两个伪端口的发送器必须处于电气空闲状态至少 6 s 。
- 否则，在 48 ms 超时后，两个伪端口上的 RT_next_data_rate 设置为 2.5 GT/s，RT_error_data_rate 设置为 2.5 GT/s，并且 Retimer 进入转发模式。

IMPLEMENTATION NOTE
Purpose of Force Timeout State

此状态的目的是确保两个链路组件同时处于 Recovery.Speed 中，以便它们返回到先前的数据速率。

4.3.7.3 Slave Loopback

Retimer 可以选择在执行模式下支持 Slave 环回。默认情况下，Retimer 配置为在 Loopback Master 和 Loopback Slave 之间转发环回。允许 Retimer 以特定于实现的方式进行配置，以充当任一伪端口上的 Loopback Slave。 另一个不是 Loopback Slave 的伪端口，将其发送器置于电气空闲状态，并忽略其接收器上的任何数据。

4.3.7.3.1 Slave Loopback.Entry

没有收到 Loopback 字段设置为 1b 的 TS1 有序集的伪端口将执行以下操作：

- 发送器被置于电气空闲状态。
- 接收器忽略输入的符号。

如 Section 4.2.6.10.1 所述，接收到 Loopback 字段设置为 1b 的 TS1 有序集的伪端口的行为就像 Section 4.2.6.10.1 中所述的 Loopback Slave 一样，但以下情况除外：

- 语句“ LinkUp = 0b（False）”被“ RT_LinkUp = 0b”代替。
- 将语句” If Loopback.Entry was entered from Configuration.Linkwidth.Start ”替换为” If Slave.Loopback.Entry was entered when RT_LinkUp =0b ”。
- 对 Loopback.Active 的引用将成为 Slave Loopback.Active。

4.3.7.3.2 Slave Loopback.Active

没有收到 Loopback 字段设置为 1b 的 TS1 有序集的伪端口将执行以下操作：

- 发送器保持电气空闲状态。
- 接收器忽略输入的符号。

收到 TS1 有序集且 Loopback 字段设置为 1b 的伪端口的行为与 Section 4.2.6.10.2 中所述的 Loopback Slave 相同，但以下情况除外：

- 对 Loopback.Exit 的引用将成为 Slave Loopback.Exit。

4.3.7.3.3 Slave Loopback.Exit

没有收到 Loopback 字段设置为 1b 的 TS1 有序集的伪端口必须执行以下操作：

- 发送器保持电气空闲状态。
- 数据速率设置为 2.5GT/s 。
- 接收器继续忽略输入的符号。

收到了将 Loopback 字段设置为 1b 的 TS1 或 TS2 有序集的伪端口的行为必须与 Section 4.2.6.10.3 中所述的 Loopback Slave 相同，但以下情况除外：

- 语句"The next state of the Loopback Master and Loopback Slave is Detect"替换为"The Data rate is set to 2.5 GT/s and then both Pseudo Ports are placed in Forwarding mode"。

4.3.8 Retimer Latency

本节定义了对允许的 Retimer Latency 的要求。

4.3.8.1 Measurement

延迟必须在 Retimer 处于转发模式且链路处于 L0 时进行测量，并定义为从在一个伪端口的输入管脚处接收到符号的最后一位到在另一个伪端口的输出管脚上发送等效位的时间。

强烈建议 Retimer 供应商在其数据表中指定 Retimer 的延迟。

公众号·芯片业者相承双

Retimer 允许在不同的数据速率下具有不同的延迟，在这种情况下，强烈建议为每个数据速率指定延迟。

4.3.8.2 Maximum Limit on Retimer Latency

当不在 SRIS 中运行时，Retimer 延迟应小于以下限制。

Table 4-29 Retimer Latency Limit not SRIS (Symbol times)

	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s	32.0 GT/s
Maximum Latency	32	32	64	128	256

4.3.8.3 Impacts on Upstream and Downstream Ports

Retimer 将增加通道延迟。当存在两个 Retimer 时，往返延迟是指定延迟的 4 倍。建议上游端口和下游端口的设计人员在确定以下特征时，请考虑 Retimer 延迟：

- Data Link Layer Retry Buffer size
- Transaction Layer Receiver buffer size and Flow Control Credits
- Data Link Layer REPLAY_TIMER Limits

可能需要额外的缓冲（Replay 或 FC）来补偿额外的通道延迟。

4.3.9 SRIS

Retimer 可以但不是必须要支持 SRIS。支持 SRIS 的 Retimer 必须提供一种机制来启用更高速率的 SKP 有序集传输，因为 Retimer 必须在执行模式下生成 SKP 有序集合。启用支持 SRIS 的 Retimer 将在接收和发送时钟域之间的弹性存储中引起额外的延迟。需要额外的等待时间来处理以下情况：发送 Max_Payload_Size TLP 而不发送已调度的 SKP 有序集。附加延迟是链路宽度和 Max_Payload_Size 的函数。Table 4-29 中未包含此额外的延迟。

具有 SRIS 功能的 Retimer 必须提供一种特定于实现的机制，以在 SRIS 中配置支持的 Max_Payload_Size，该机制必须配置为大于或等于伪端口正在接收的端口中的发送器的 Max_Payload_Size。对于带有 SRIS 的当前支持的 Max_Payload_Size，Retimer 延迟必须小于以下限制。

Table 4-30 Retimer Latency Limit SRIS (Symbol times)

Max_Payload_Size	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s	32.0 GT/s
128 Bytes	34 (max)	34 (max)	66 (max)	130 (max)	194 (max)
256 Bytes	36 (max)	36 (max)	68 (max)	132 (max)	196 (max)
512 Bytes	39 (max)	39 (max)	71 (max)	135 (max)	199 (max)
1024 Bytes	46 (max)	46 (max)	78 (max)	142 (max)	206 (max)
2048 Bytes	59 (max)	59 (max)	91 (max)	155 (max)	219 (max)
4096 Bytes	86 (max)	86 (max)	118 (max)	182 (max)	246 (max)

IMPLEMENTATION NOTE

Retimer Latency with SRIS Calculation:

公众号：芯片业者相承双

假设链路以 x1 链路宽度运行。最大延迟是 Table 4-29 与弹性存储中 SRIS 时钟补偿所需的额外延迟的总和。 下式描述了 SRIS 时钟补偿所需的符号时间中的 SRIS 附加的延迟：

$$2 * \left(\frac{\left(\left(\text{SRIS Link Payload Size} + \text{TLP Overhead} \right) / \text{Link Width} \right)}{\text{SKP\_rate}} \right)$$

Equation 4-1 Retimer Latency with SRIS

其中，

SRIS Link Payload Size

是编程到 Retimer 中的值。

TLP Overhead

表示消耗链路带宽的其他 TLP 组件（TLP Prefix，header，LCRC，framing Symbol），在这里被视为 28 个符号的恒定值。

Link Width

链路工作时的宽度。

SKP_rate

当使用 8b/10b 编码时，发送器调度 SKP 有序集的速率，请参见 Section 4.2.7.3。当使用 128b/130b 编码时，有效速率是相同的。

标称延迟将是 SRIS 附加延迟的 1/2，并且是弹性存储的标称填充量。这使情况更糟，即每个阻塞的 SKP 有序集都需要在弹性存储区中附加一个延迟的符号。传输最大有效负载大小 TLP 时，弹性存储的实际填充可能会变为零，或者是标称填充的两倍，具体取决于相对时钟频率。链路宽度 down configure 可能随时发生，例如，通道发生故障，并且这种 down configure 可能比 Retimer 能够调整其名义弹性存储的速度更快。默认情况下，无论当前的实际链路宽度如何，Retimer 都会根据 x1 链路宽度配置其标称填充。

可选地支持 SRIS 的 Retimer 可以可选地支持动态弹性存储。动态弹性存储会随着链路宽度的变化而更改标称缓冲区填充。允许 Retimer 延迟链路的 LTSSM 转换，仅当链路 down configure 时，在配置中，最多可延迟 40us。允许 Retimer 延迟 Configuration.Lanenum.Accept 和 Configuration.Comp 之间的 TS1 有序集到 TS2 有序集的转换。

4.3.10 L1 PM Substates Support

以下部分描述了支持可选 L1 PM 子状态的 Retimer 的要求。

当 CLKREQ＃被采样到无效时，Retimer 进入 L1.1，并发生以下事件：

- Retimer 的 REFCLK 关闭。
- PHY 保持供电。
- Retimer 将两个伪端口上的所有发送器都置于电气空闲状态（如果尚未处于电气空闲状态，则为预期状态）。发送器保持其共模电压。
- Retimer 必须忽略两个伪端口上所有接收器的任何电气空闲退出。

当 CLKREQ＃被采样到有效时，Retimer 退出 L1.1，并发生以下事件：

- Retimer 的 REFCLK 打开。

公众号·芯片业者怕承双

- 在 Retimer 的两个伪端口的所有通道上，都会恢复电气空闲退出电路的正常运行。
- 继续正常的从电气空闲退出的行为，请参阅 Section 4.3.6.3。

Retimer 不支持 L1.2，但是如果它们支持 L1.1 和移除参考时钟，则它们不得干扰所连接的组件进入 L1.2。

Retimer 供应商必须记录适用于 CLKREQ＃的特定实现要求。例如，不支持移除参考时钟的 Retimer 实现可能需要将 CLKREQ＃拉低。

IMPLEMENTATION NOTE

CLKREQ# Connection Topology with a Retimer Supporting L1 PM Substates

在此平台配置中，下游端口（A）仅具有单个 CLKREQ＃信号。上游端口和下游端口的 CLKREQ＃（A 和 C）与 Retimer 的 CLKREQB＃信号相互连接。在这种情况下，只要需要参考时钟，下游端口（A）就必须声明 CLKREQ＃信号。组件 A，组件 B 和 Retimer 同时删除/恢复了它们的 REFCLK。

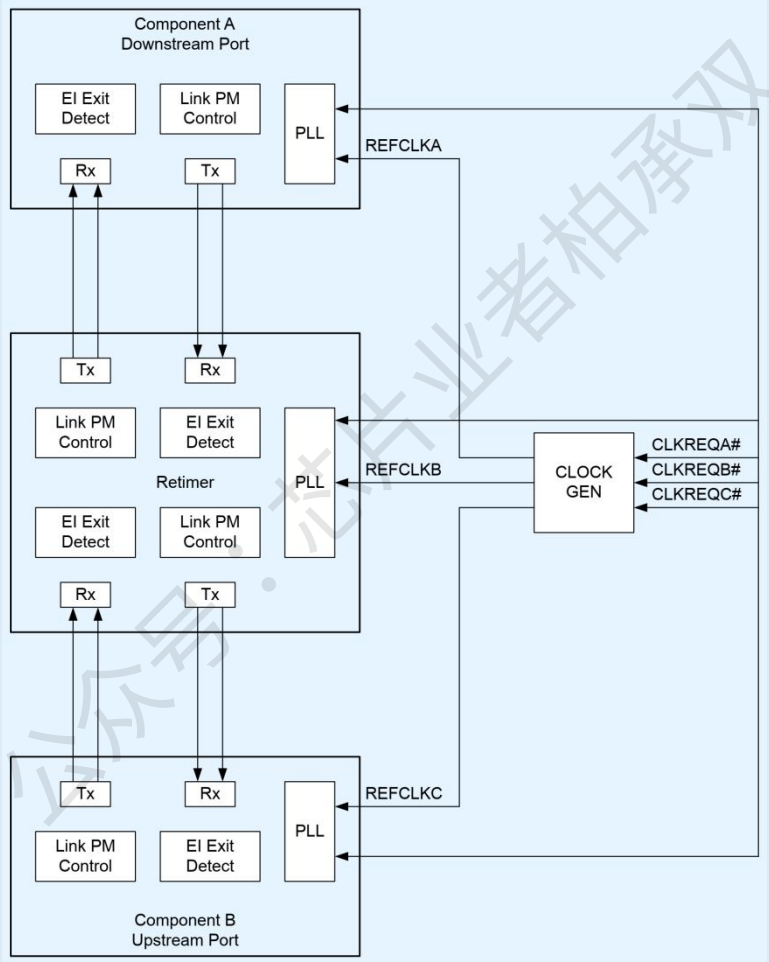

Figure 4-37 Retimer CLKREQ# Connection Topology

4.3.11 Retimer Configuration Parameters

Retimer 必须提供一种特定于实现的机制来配置本节中的每个参数。

这些参数分为两组：可为 Retimer 全局配置的参数和可为每个物理 Retimer 伪端口配置的参数。

如果每个伪端口参数仅适用于上游或下游伪端口，则不需要 Retimer 来提供特定于实现的机制来为其他类型的伪端口配置参数。

4.3.11.1 Global Parameters

- **Port Orientation Method。**这控制是根据 Section 4.3.6.2 中的说明动态确定端口方向，还是根据上游和下游伪端口的供应商分配静态确定端口方向。如果端口方向设置为静态，则不需要 Retimer 来动态调整端口方向，如 Section 4.3.6.2 中所述。默认行为是动态确定端口方向。

- **Maximum Data Rate。**这控制了 Retimer 在其发送的训练集的 Data Rate Identifier 字段中设置的最大数据速率。允许仅支持 2.5 GT/s 速度的 Retimer 不提供此配置参数。

- **SRIS Enable。**此参数控制在执行模式下，是否为 Retimer 启用了 SRIS，并以 SRIS 模式速率发送 SKP 有序集。不要求不支持 SRIS 和至少一个其他时钟体系结构的 Retimer 提供此配置参数。

- **SRIS Link Payload Size。**这控制了 Retimer 在 SRIS 中时支持的最大有效负载大小。该值必须可以在 Table 4-29 中显示的 Maximum Payload Size 中选择。此参数的默认值是支持 4096 字节的有效负载大小。不要求不支持 SRIS 的 Retimer 提供此配置参数。

  以下是一些示例的示例，在这些示例中，可能需要将 SRIS 链路有效负载大小配置为小于默认值：

  - Retimer 是具有根端口的主板的一部分，该根端口支持 Maximum Payload Size 小于 4096 字节。
  - Retimer 是附加卡的一部分，该附加卡的 Endpoint 支持的 Maximum Payload Size 小于 4096 字节。
  - Retimer 位于作为系统一部分集成的 Switch 的下游端口的下游，根端口芯片支持的 Maximum Payload Size 小于 4096 字节，并且系统不支持对等业务流。

- **Enhanced Link Behavior Control。**这控制了 Retimer 支持 32.0 GT/s 时将均衡器旁路到最高数据速率或完全旁路均衡的能力。

4.3.11.2 Per Physical Pseudo Port Parameters

- **Port Orientation。**仅当将 Port Orientation Method 配置为静态确定时，此方法才适用。这是为上游或下游设置的。必须为每个伪端口配置不同的方向，否则行为是不确定的。

- **Selectable De-emphasis。**当下游伪端口以 5.0 GT/s 的速度运行时，这将控制特定情况下链路到-3.5dB 或-6dB 的发送器去加重，以及下游伪端口传输的训练集中 Selectable De-emphasis 字段的值。详细使用信息见 Section 4.2.6。当链路段未以 5.0 GT/s 速度运行时，此位的设置不会产生任何影响。仅支持 2.5 GT/s 速度的 Retimer 不允许提供此配置参数。

- **Rx Impedance Control。**这可控制 Retimer 是动态使用并删除 50Ω终端电阻，还是静态存在 50Ω终端电阻。该值必须可以从 Dynamic，Off 和 On 中选择。默认行为是 Dynamic 。

- **Tx Compliance Disable。**这控制了 Retimer 是否在 CompLoadBoard.Pattern 状态下发送 Compliance Pattern。Retimer 的默认行为是在 CompLoadBoard.Pattern 状态下传输 Compliance Pattern。如果将 TX Compliance Pattern 设置为禁用，则 Retimer 发送器将保持电气空闲状态，并且不会在 CompLoadBoard.Pattern 中传输 Compliance Pattern——CompLoadBoard 状态下的所有其他行为都是相同的。

- **Pseudo Port Slave Loopback。**这可控制 Retimer 在链路环回期间是在转发模式下运行还是在伪端口上进入 Slave 环回。默认的行为是使 Retimer 在环回过程中以转发模式运行。允许不支持可选的 Slave 环回的 Retimer 不提供此配置参数。此配置参数仅应为一个物理端口启用。如果为多个物理端口启用了该参数，则 Retimer 行为是不确定的。

- **Downstream Pseudo Port 8GT TX Preset。**这可控制下游伪端口发送器用于 8.0 GT/s 传输的初始 TX 预设。 默认值是特定于实现的。该值必须可以从 Table 4-4 中的所有适用值中选择。

- **Downstream Pseudo Port 16GT TX Preset。**这可控制下游伪端口发送器用于 16.0 GT/s 传输的初始 TX 预设。 默认值是特定于

实现的。该值必须可以从 Table 4-4 中的所有适用值中选择。

- **Downstream Pseudo Port 32GT TX Preset。**这可控制下游伪端口发送器用于 32.0 GT/s 传输的初始 TX 预设。默认值是特定于实现的。该值必须可以从 Table 4-4 中的所有适用值中选择。
- **Downstream Pseudo Port 8GT Requested TX Preset。**这可控制由下游伪端口以 8.0 GT/s 的速率发送的 EQ TS2 Ordered Set 中使用的初始发送器预设值。默认值是特定于实现的。该值必须可以从 Table 4-4 中的所有值中选择。
- **Downstream Pseudo Port 16GT Requested TX Preset。**这可控制由下游伪端口以 16.0 GT/s 的速率发送的 128b/130b EQ TS2 Ordered Set 中使用的初始发送器预设值。默认值是特定于实现的。该值必须可以从 Table 4-4 中的所有值中选择。
- **Downstream Pseudo Port 32GT Requested TX Preset。**这可控制由下游伪端口以 32.0 GT/s 的速率发送的 128b/130b EQ TS2 Ordered Set 中使用的初始发送器预设值。默认值是特定于实现的。该值必须可以从 Table 4-4 中的所有值中选择。
- **Downstream Pseudo Port 8GT RX Hint。**这将控制由下游伪端口发送的 EQ TS2 Ordered Set 中使用的 Receiver Preset Hint 值，以 8.0 GT/s 的速度使用。默认值是特定于实现的。该值必须可以从 Table 4-5 中的所有值中选择。

4.3.12 In Band Register Access

以 16.0 GT/s 或更高速度运行的 Retimer 可以选择支持带内只读访问。16.0 GT/s 或更高速度的 Control SKP Ordered Set 通过 Margin Command ‘Access Retimer Register’提供了访问机制，请参见 Table 4-26。支持带内只读访问的 Retimer 必须在寄存器偏移 80h 和 84h 处为 DWORD 返回非零值。不支持带内只读访问的 Retimer 必须返回零值。

A0h 和 FFh 之间的寄存器偏移量指定为供应商定义的寄存器空间。

寄存器偏移量从 00h 到 7Fh 和 85H 到 9Fh 保留给 PCI-SIG 将来使用。

公众号·芯片业者相承双

5 Power Management

本章介绍 PCI Express 电源管理（PCI Express-PM）的功能和协议。

5.1 Overview

电源管理状态分以下两种：

- 用于管理特定 Function 的 D state。
  - D0 是正常工作的电源状态，它消耗最多电源。
  - D1 和 D2 是中间的省电状态，消耗电源也处于中间。
  - $D3_{Hot}$ 是一个功耗非常低的状态。
  - $D3_{Cold}$ 是电源关闭状态。
- 用于管理特定链路的 L state。
  - L0 是链路正常工作状态。
  - L0s、L1、L1.0、L1.1 和 L1.2 是低功耗状态。

其他规范定义了相关的电源状态（例如 S state）。本规范不描述这些状态与 D/L/B 状态之间的关系。

PCI Express-PM 提供一下 service：

- 识别给定 Function 的电源管理功能的机制。
- 将一个 Function 转换为某种电源管理状态的能力。
- 通告一个 Function 当前电源管理状态的能力。
- 在特定事件下唤醒系统的能力。

PCI Express-PM 与 *PCI Bus Power Management Interface Specification* 和 *Advanced Configuration and Power Interface Specification* 规范都兼容。本章也介绍本 PCI Express 扩展的电源管理功能。这在 *PCI Bus Power Management Interface Specification* 的范围内提供了额外的电源管理功能。

PCI Express-PM 定义了链路电源管理状态，该链接电源管理状态允许 PCI Express 物理链接进入，以响应软件驱动的 D-state 转换或活动链路电源管理状态的转换。PCI Express 链路状态对传统总线驱动程序软件不直接可见，而是从这些链路上驻留的组件的电源管理状态得出的。定义的链路状态为 L0，L0，L1，L2 和 L3。随着链路状态从 L0 过渡到 L3，节电效果会增加。

Component 可以使用后跟 PME 消息的唤醒机制来唤醒系统。PCI Express 系统可以提供从主电源关闭的状态进行唤醒操作所需的可选辅助电源（Vaux）。

与 Vaux 相关的特定定义和要求是特定于外形规格的，在本文档中，术语“auxiliary power”和“Vaux”应参考使用中的特定外形规格来理解。

PCI Express-PM PME 机制的另一个区别是其将以下两个 PME 任务分开：

- 重新激活（唤醒）相关资源（即，重新建立 PCI Express 组件的参考时钟和主电源）。

公众号·芯片业者相承双

■ 向 RC 发送 PME 消息。

ASPM（Active State Power Management）是一种基于硬件的自主活动状态机制，即使所连接的组件处于 D0 状态，也可以节省功耗。在一段时间的空闲链路时间后，ASPM 物理层协议将空闲链路置于低功耗状态。进入低功耗状态后，链路两端都可以发送事务传输，从而触发向 L0 状态的转换。ASPM 可以被软件禁用。有关 ASPM 的更多信息，请参见 Section 5.4.1。

5.2 Link State Power Management

PCI Express 定义了链路电源管理状态，替换了 *PCI Bus Power Management Interface Specification* 定义的总线电源管理状态。链路状态对 PCI-PM 兼容的软件是不可见的，而是从连接到该链路的相应组件的电源管理 D-states 或通过 ASPM 协议推导出来的（see Section 5.4.1）。

请注意，PCI Express 物理层可能会定义其他中间状态。有关每种状态以及物理层如何处理状态之间转换的更多详细信息，请参阅 Chapter 4。

PCI Express-PM 定义了下述链路电源管理状态：

- **L0** – Active state

ASPM 和 PCI-PM 兼容电源管理都需要支持 L0 状态。该状态下 PCI Express 正常传输数据。

- **L0s** – 一种恢复延迟低，低功耗的“standby”状态

L0s 对 ASPM 的支持是可选的，除非适用于链路的设备明确要求 L0s 支持。它不适用于 PCI-PM 兼容的电源管理。

在 L0s 期间，所有主电源，组件参考时钟和组件的内部 PLL 必须始终处于有效状态。对于链路位于 Tx_L0s 的端口，禁止 TLP 和 DLLP 传输。

物理层提供了从此状态到 L0 状态的快速转换的机制。当在链路的两端使用 common (distributed) reference clock 时，从 L0s 到 L0 的切换时间通常少于 100 个 Symbol Time。

链路上一个组件的发送器可能位于 L0s 状态，而链路上另一组件的发送器可能处于 L0 状态。

- **L1** – 一种恢复延迟高，低功耗的“standby”状态

PCI 兼容电源管理需要支持 L1。L1 对于 ASPM 是可选的，除非特定设备要求。

当把 L1 PM Substates Control 1 Register 中的一个或多个使能位置 1 来使能 L1 PM Substates 时，此状态称为 L1.0 子状态。

在 L1 期间，所有主电源必须保持活动状态。只要它们遵守通告的 L1 退出延迟，实现时就允许通过使用一些技术来降低功耗，例如周期性的而不是连续的检查电气空闲退出，或者检查一个通道上的电气空闲退出， 以及关闭不需要的电路。除非使能时由 Clock Power Management（使用 CLKREQ＃）允许，否则所有平台提供的组件参考时钟必须在 L1 期间保持有效状态。组件的内部 PLL 可能会在 L1 期间关闭，从而以增加退出延迟为代价来节省更多的功耗。例如，在 D3Hot 中禁用内部 PLL 可能是可取的，但在 D1 或 D2 中则不然。

只要将给定链路上的下游组件的所有 Function 编程为 D0 以外的 D-state，就进入 L1 状态。如果下游组件请求进入 L1（ASPM）并收到对该请求的确认，则也进入 L1 状态。

从 L1 退出是通过上游设备或下游设备发送传输事务引起的。从 L1 到 L0 的过渡时间通常为几微秒。

处于 L1 的链路禁止传输 TLP 和 DLLP（因为一旦传输会引起一个链路恢复过程）。

- **L1 PM Substates** - 指 PCI-PM 和 ASPM 在 L1 低功耗链路状态可选实现的 L1.1 和 L1.2 子状态。L1 时可以选择关断时钟，L1 PM Substates 时关闭 RX 侧电气空闲检测电路。L1.1 维护一个共模电压，L1.2 共模电压也被关掉。

公众号·芯片业者相承双

在 L1.1 子状态中，需要维护链路共模电压。当链路处于 L1.0 子状态并且满足进入 L1.1 子状态的条件时，进入 L1.1 子状态。有关详细信息，请参阅 Section 5.5.1 。

在 L1.2 子状态中，关闭链路共模电压。当链路处于 L1.0 子状态并且满足进入 L1.2 子状态的条件时，进入 L1.2 子状态。有关详细信息，请参阅 Section 5.5.1 。

当 CLKREQ# 信号被置 1 时，将启动从所有 L1 PM 子状态的退出过程（参阅 Section 5.5.2.1 和 Section 5.5.3.3 ）。

- **L2/L3 Ready** – 暂缓进入 L2 或 L3 的一个状态

需要支持 L2/L3 Ready 这个过渡状态。

L2/L3 Ready 是一种伪状态（对应于 LTSSM L2 状态），当准备从下游组件或两个相互连接组件中删除电源和时钟时，给定的链路进入该伪状态。在 PM 软件将设备转换为 D3 状态后启动这个过程，然后调用电源管理软件把电源和时钟移除。链路进入 L2/L3 Ready 状态后，即可准备卸下电源。断开主电源后，如果提供并使用了 Vaux，则链路将转换为 L2，如果没有提供 Vaux，则链路将转换为 L3。请注意，这些是链路的 PM 伪状态。在这种情况下，LTSSM 通常将仅使用主电源运行，因此将在关闭主电源后关闭电源。

在确认 PME_Turn_Off 消息之后（即，插入一个 PME_TO_Ack TLP），必须尽快开始 L2/L3 Ready 状态转换过程。下游组件通过发送 PM_Enter_L23 DLLP 来启动 L2/L3 Ready 的进入。有关电源管理消息的更多详细信息，请参见 Section 5.7。

L2/L3 Ready 状态是禁止 TLP 和 DLLP 传输的。

注意：L2/L3 Ready 返回到 L0 将通过中间 LTSSM 状态。有关详细信息，请参阅 Chapter 4。

- **L2** – 辅助电源链路，深度节能状态

L2 状态是可选实现的，并且它依赖于 Vaux 电源的存在。

在 L2 状态，组件将只消耗 Vaux 的功率，参阅 Section 5.6。

在 L2 状态，主电源和参考时钟都被关闭。

在 L2 状态，任何链路重新激活唤醒逻辑（Beacon 或 WAKE＃），PME 上下文以及任何其他“保持活动”的逻辑均由 Vaux 供电。

L2 状态是禁止 TLP 和 DLLP 传输的。

- **L3** – 链路断开状态

当没有任何电源时，链路处于 L3 状态。

- **LDn** – L0 之前的过渡 Link Down 伪状态

此伪状态与 LTSSM 的 Detect、Polling、Configuration、Disabled、Loopback 和 Hot Reset 有关。

有关进入和退出 L0 和 L2/L3 Ready（从 Chapter 4 角度来看为 L2.Idle）之间的每个 L-state 的更多详细信息，请参见 Section 4.2。 L2 状态是以存在辅助电源为特征用于 PM 目的的抽象，并且不要求 LTSSM 保持活动状态。

电气部分指定未加电时驱动器和接收器的电气特性。这是 L3 状态，但电气部分未提及 L3。

Figure 5-1 显示了在链路操作过程中可能发生的合法 L-state 的转换。

公众号：芯片业者相承网

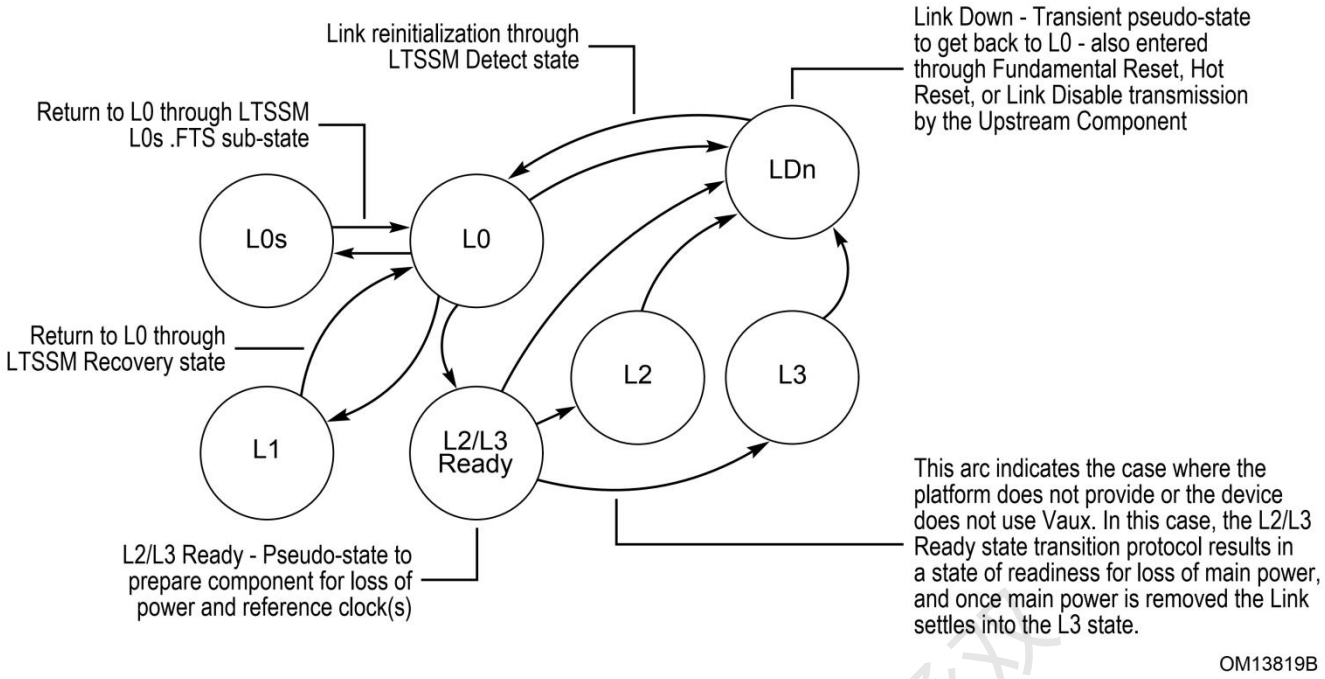

Figure 5-1 Link Power Management State Flow Diagram

L1 和 L2/L3 Ready 的进入需要在 L0 状态下协商进行。只有在协商完成后才能进入 L1 和 L2/L3 Ready。除非出现 LDn，否则链路电源管理将保持在 L0 中，直到协商过程完成。请注意，这些状态和状态转换并不直接对应于物理层 LTSSM 的操作。例如，在 Figure 5-1 中，L0 包含 LTSSM 的 L0，Recovery 以及在 LinkUp 期间的 Configuration 状态。同样，LTSSM 通常由主电源（不是 Vaux）供电，因此 LTSSM 不会在 L2 或 L3 状态下供电。

以下示例说明了导致进入系统睡眠状态的链路状态转换过程的多个步骤：

1. 系统软件导致一个下游组件的所有 Function 进入 $D3_{hot}$ 。
2. 然后下游组件发起把链路转换到 L1 状态的过程。
3. 然后系统软件引导 RC 广播 PME_Turn_Off 消息以准备切断主电源。
4. 此消息使链路转换回 L0，以便发送 PME_Turn_Off 消息并使下游组件能够响应 PME_TO_Ack。
5. 下游组件在发送完 PME_TO_Ack 后发起向 L2/L3 Ready 的转换过程。

   L0 → L1 → L0 → L2/L3 Ready

下面一个例子说明，不把所有 Function 置于 D3hot 也可以切断电源：

1. 系统软件引导 RC 广播 PME_Turn_Off 消息以准备切断主电源。
2. 下游组件响应 PME_TO_Ack。
3. 下游组件在发送完 PME_TO_Ack 后发起向 L2/L3 Ready 的转换过程。

   L0 → L2/L3 Ready

L1 进入协商（无论是通过 PCI-PM 还是 ASPM 机制调用）和 L2/L3 Ready 进入协商怎么映射到状态机会在稍后介绍。此状态机将重置为空闲状态。对于下游组件，状态机在离开空闲状态后采取的第一个操作是根据协商的类型开始发送适当的进入 DLLP。如果协商中断，则两个组件中的状态机都将重置为空闲状态。对于上游组件，这始终意味着进入空闲状态，并等待接收进入 DLLP。对于下游组件，这意味着进入空闲状态并继续发送进入 DLLP 以重新开始协商。

Table 5-1 总结了每个 L-state，描述了它们的使用时机以及与每个状态对应的平台和组件行为。

“Yes”表示需要支持（除非另有说明）。“On”和“Off”表示相应的电源和时钟需求。“On/Off”表示可以自行选择是否支持。

Table 5-1 Summary of PCI Express Link Power Management States

	L-State Description	Used by S/W Directed PM	Used by ASPM	Platform Reference Clocks	Platform Main Power	Component Internal PLL	Platform Vaux
L0	Fully active Link	Yes (D0)	Yes (D0)	On	On	On	On/Off
L0s	Standby state	No	Yes [1] (opt., D0)	On	On	On	On/Off
L1	Lower power standby	Yes (D1-D3Hot)	Yes (opt., D0)	On/Off [6]	On	On/Off [2]	On/Off
L2/L3 Ready (pseudo-state)	Staging point for power removal	Yes [3]	No	On/Off [6]	On	On/Off	On/Off
L2	Low power sleep state (all clocks, main power off)	Yes [4]	No	Off	Off	Off	On [5]
L3	Off (zero power)	n/a	n/a	Off	Off	Off	Off
LDn (pseudo-state)	Transitional state preceding L0	Yes	N/A	On	On	On/Off	On/Off
	L-State Description	Used by S/W Directed PM	Used by ASPM	Platform Reference Clocks	Platform Main Power	Component Internal PLL	Platform Vaux

Note：

1. 在具有独立参考时钟输入的链路配置中，L0s 的退出延迟将是最大的，该参考时钟输入用于连接到给定链路对端的组件（相对于公共的分布式参考时钟）。
2. 对于在此状态下内部关闭其 PLL 的组件，L1 退出等待时间将最大。
3. L2/L3 Ready 进入序列在 PME_Turn_Off / PME_TO_Ack 协议握手完成时启动。它既不直接与 D-state 也不与 ASPM 定义的过渡策略有直接关系。
4. 根据平台的实现，系统的睡眠状态可以使用 L2 状态，或者 L3 状态，也可以将链路保持在 L2/L3 Ready 状态。 L2/L3 Ready 状态转换协议由下游组件在收到 PME_Turn_Off TLP 消息并进行 TLP 确认后启动。 虽然平台对 L2 睡眠状态配置的支持是可选的（取决于 Vaux 的可用性），但需要组件协议支持才能将 Link 转换为 L2/L3 Ready 状态。
5. 仅通过 Vaux 的存在和使用，将 L2 与 L3 状态区分开。在完成 L2/L3 Ready 状态转换协议之后，并且在断开主电源之前，链路表示已准备好断开主电源。
6. 低功耗移动或手持设备可能会通过“clock request ”（CLKREQ＃）机制对参考时钟进行时钟门控来降低功耗。结果，针对这些器件的组件应能够承受在低功耗状态退出期间为参考时钟重新供电所需的额外延迟。

公众号·芯片业者相承双

5.3 PCI-PM Software Compatible Mechanisms

5.3.1 Device Power Management States (D-States) of a Function

虽然这些电源状态的概念对于系统中的所有 Function 都是通用的，但转换到给定电源管理状态时的含义或预期功能行为取决于 Function 的类型（或类）。

D0 电源管理状态是 Function 的正常运行状态。其他状态是降低功耗的各种级别，其中 Function 要么不运行，要么支持一组有限的操作。D1 和 D2 是中间状态，旨在为系统设计人员提供更大的灵活性，以平衡给定设备类别的节能、恢复时间和低功耗功能可用性权衡。例如，可以将 D1 状态支持为比 D2 稍多功耗的状态，但是这种状态产生的恢复时间比从 D2 恢复要快。

D3 电源管理状态是电源管理状态的特殊类型，因为 Function 可以通过软件或通过物理移除其电源来转换到 D3。从这个意义上说，两个 D3 变体被指定为 D3Hot 和 D3Cold，其中下标分别表示主电源的存在或不存在。通过写入 Function 的 PMCSR 寄存器，允许通过软件将处于 D3Hot 的 Function 转换为 D0 状态。通过重新提供主电源和执行基本复位，允许处于 D3Cold 状态的 Function 转换到 $D0_{uninitialized}$ 状态。

所有 Function 都必须支持 D0 和 D3 状态（D3Hot 和 D3Cold）。D1 和 D2 状态是可选实现的。

IMPLEMENTATION NOTE

Switch and Root Port Virtual Bridge Behavior in Non-D0 States

当与 Switch/Root Port（“虚拟桥”）关联的 Type 1 Function 处于 non-D0 电源状态时，它将在处理 Memory，I/O 和 Configuration Requests 和 Completion 时模拟常规 PCI 桥的行为。流向下游的所有 Memory 和 I/O 请求均作为 Unsupported Request 而终止。所有 Type 1 类型的配置请求都作为 Unsupported Request 而终止，但是 Type 0 配置请求处理不受虚拟桥 D-state 的影响。跨虚拟桥向任一方向流动的 Completion 不受虚拟桥 D-state 的影响。

请注意，消息的处理不受虚拟桥的 PM 状态影响。

5.3.1.1 D0 State

所有 Function 都必须支持 D0 状态。D0 分为两个不同的子状态，即“un-initialized”子状态和“active”子状态。当一个组件退出常规复位或 FLR 时，默认为 $D0_{uninitialized}$ 状态。处于这种状态的组件将由枚举过程枚举和配置。枚举和配置过程完成后，Function 进入 $D0_{active}$ 状态，即 PCI Express Function 的完全运行状态。只要系统软件启用了该 Function 的 Memory Space Enable、I/O Space Enable 或者 Bus Master Enable 字段中的任何一个或多个，该 Function 就进入 $D0_{active}$ 状态。注意，即使这些使能位随后被清 0，此 Function 仍保持 $D0_{active}$ 状态。

5.3.1.2 D1 State

D1 是可选支持的。当处于 D1 状态时，除了 Section 5.3.3 中定义的 PME 消息外，Function 不得在链路上发起任何 TLP 请求。配置和消息请求是处于 D1 状态的 Function 唯一接受的 TLP。所有其他收到的请求必须作为 Unsupported Request 处理，所有收到的 Completion 可以有选择地作为 Unexpected Completion 处理。如果在 D1 中检测到由接收到的 TLP 引起的错误（例如 Unsupported Request），并且启用了报告功能，则必须将链路返回到 L0（如果该链路尚未位于 L0 中），并且必须发送错误消息。如果在 D1 中检测到接收到的 TLP 以外的事件引起的错误（例如，完成超时），则在将功能编程回 D0 状态时，必须发送错误消息。

请注意，Function 的软件驱动程序参与了将 Function 从 D0 转换为 D1 的过程。它通过保存一些 Function 的状态（如有必要），并且做了一些把 Function 从 D0 转换到 D1 的准备工作。作为此过程的一部分，Function 的软件驱动程序必须确保在将控制权交给系统配置软件（此后将完成向 D1 的转换）之前，终止所有传输中的 TLP。

5.3.1.3 D2 State

D2 支持是可选的。当一个 Function 当前没有被使用并且可能在一段时间内不会被使用时，它可能会被置于 D2 状态。这种状态要求 Function 提供比较大的节能，同时仍然保留完全恢复到其先前状态的能力。当处于 D2 状态时，除 Section 2.2.8 中定义的消息外，Function 不得在链路上发起任何请求 TLP。配置和消息请求是 D2 状态下 Function 接受的唯一 TLP。所有其他收到的请求必须作为不受支持的请求处理，并且所有收到的完成可以选择作为意外完成处理。如果在 D2 中检测到由接收到的 TLP（例如，不支持的请求）引起的错误，并且启用了报告，则如果链路不在 L0 中，则必须将链路返回到 L0，并且必须发送错误消息。如果在 D2 中检测到由接收到的 TLP 以外的事件（例如，完成超时）引起的错误，则必须在功能编程回 D0 状态时发送错误消息。

请注意，Function 的软件驱动程序参与了将 Function 从 D0 转换为 D2 的过程。它通过保存一些 Function 的状态（如有必要），并且做了一些把 Function 从 D0 转换到 D2 的准备工作。作为此静默过程的一部分，Function 的软件驱动程序必须确保在将控制权交给系统配置软件（此后将完成向 D2 的转换）之前，终止所有传输中的 TLP。

系统软件必须将 Function 恢复到 D0active 状态，然后才能访问内存或 I/O 空间。只有在 Function 恢复到 D0active 后才能开始启动诸如总线主控和中断请求生成的操作。

对 Function 编程使得从 D2 到 D0 和向 Function 发出下一个请求之间的最短恢复时间要求为 200 µs。对于在此恢复时间窗口中收到的请求，其行为未定义（请参阅 Section 7.9.17 ）。

5.3.1.4 D3 State

D3 状态是必须支持的，包括 $D3_{cold}$ 和 $D3_{hot}$ 。支持从 D3 生成 PME 的功能必须同时支持 $D3_{cold}$ 和 $D3_{hot}$ 状态。

如果设置了 PMCSR 中的 No_Soft_Reset 字段，则必须由 $D3_{hot}$ 状态下的 Function 维护功能上下文。在这种情况下，从 $D3_{hot}$ 转换到 D0 后（Function 将处于 $D0_{active}$ 状态），不需要软件重新初始化 Function。如果清除 No_Soft_Reset，则 Function 不需要处于 $D3_{hot}$ 状态的功能上下文。结果，在这种情况下，由于 Function 将处于 $D0_{uninitialized}$ 状态，因此在转换到 D0 后需要软件完全重新初始化功能。

如果链路状态已转换为 L2/L3 Ready 状态，则无论 No_Soft_Reset 位的值如何，都将复位该 Function。

IMPLEMENTATION NOTE

Transitioning to L2/L3 Ready

如 Section 5.2 所述，由平台电源管理软件启动向 L2/L3 Ready 状态的转换，以开始从设备中移除主电源和时钟的过程。 结果，预计设备将在其链路转换到 L2/L3 Ready 之后不久将转换到 $D3_{cold}$，从而使仅适用于 D3hot 的 No_Soft_Reset 字段无关紧要。虽然不能保证 L2/L3 Ready 和 $D3_{cold}$ 之间存在这种相关性，但是系统软件应确保仅在断开设备主电源时才进入 L2/L3 Ready 状态。当退出 L2/L3 Ready 是因 DL_Down 造成，则需要设备 Function（包括那些能够在 $D3_{hot}$ 中保持功能上下文，即设置 No_Soft_Reset 字段的 Function），按 Section 2.9.1 所述重新初始化内部状态。

在访问该 Function 之前，在 $D3_{hot}$→D0 转换之后至少 10 ms 之后，系统软件必须允许最短的恢复时间。例如，该恢复时间可以在 $D3_{hot}$→D0 转换之后用于 boot 其接口（例如，从串行 ROM）。在恢复时间内尝试访问功能（包括配置请求数据包）将导致不确定的行为。

5.3.1.4.1 $D3_{hot}$ State

配置和消息请求是处于 $D3_{hot}$ 状态的 Function 唯一接受的 TLP。所有其他收到的请求必须作为 Unsupported Request 处理，所有收到的 Completion 可以有选择地作为 Unexpected Completion 处理。如果在 $D3_{hot}$ 中检测到由接收到的 TLP 引起的错误（例如，不

支持的请求），并且启用了报告功能，则必须将链路返回到 L0（如果该链路尚未位于 L0 中），并且必须发送错误消息。如果在 $D3_{hot}$ 中检测到接收到的 TLP 以外的事件引起的错误（例如，完成超时），则在将 Function 编程回 D0 状态时，必须发送错误消息。一旦 Function 进入 $D3_{hot}$ 则接下来可以由主机切断电源而进入 $D3_{cold}$ 。

请注意，Function 的软件驱动程序参与了将 Function 从 D0 转换为 $D3_{hot}$ 的过程。它通过保存一些 Function 的状态（如有必要），并且做了一些把 Function 从 D0 转换到 $D3_{hot}$ 的准备工作。作为此静默过程的一部分，Function 的软件驱动程序必须确保在将控制权交给系统配置软件（此后将完成向 $D3_{hot}$ 的转换）之前，终止所有传输中的 TLP。

请注意，$D3_{hot}$ 也是一种有用的状态，可用于减少本来正在运行的系统中的空闲组件的功耗。

$D3_{Hot}$ 中的 Function 允许通过软件（写入其 PMCSR PowerState 字段）转换为 D0active 状态或 D0uninitialized 状态。只要提供电源和时钟，D3Hot 中的 Function 就必须响应配置空间访问，以便它们可以通过软件返回到 D0。请注意，在从 $D3_{hot}$ 转换为 D0 期间或之后，不需要 Function 来生成内部硬件复位（请参阅 PMCSR 中 No_Soft_Reset 位的用法）。

如果不需要内部复位，则在完成 D3Hot 到 D0active 状态后，除了写入 PowerState 字段外，不需要额外的操作系统干预。如果需要内部复位，设备将返回 D0uninitialized 并且需要对设备进行完全重新初始化。完整的重新初始化序列将设备返回到 D0active。

如果设备支持 PME 事件，并且 PME_En 为 1，则必须在 D3Hot 中保留 PME 上下文。PME 上下文也必须保留在 PowerState 命令转换回 D0 中。

IMPLEMENTATION NOTE

Devices Not Performing an Internal Reset

总线控制器到非 PCIe 总线，并从作为非 PCIe 总线接口的 PCIe 总线上的 D3Hot 总线控制器恢复，（例如 CardBus、USB 和 IEEE 1394）是总线控制器的示例，它们将受益于不需要内部 从 D3Hot 恢复时重置。如果不需要此内部复位，总线控制器将不需要在其辅助（非 PCIe）总线上从 D3Hot 恢复时执行下游总线复位。

IMPLEMENTATION NOTE

Multi-Function Device Issues with Soft Reset

对于 Multi-Function Device（MFD），影响整体设备行为的某些控制设置有些由所有功能一起决定，有效只由 Function 0 决定。以下是一些关键示例：

- 对于 non-ARI MFD，Device Control register 和 Link Control register 的某些控制会关闭所有 Function 的整体设置。
- 对于 ARI MFD，Device Control register 和 Link Control register 的某些控制只关闭 Function 0 的设置。
- 对于所有 MFD，Device Control 2 register 和 Link Control 2 register 的某些控制只关闭 Function 0 的设置。

对任何 Function（特别是 Function 0）执行软复位可能会破坏 MFD 中其他 active Function 的正常运行。由于某些操作系统希望在不影响其他 Function 的情况下在 $D3_{hot}$ 和 D0 之间转换，因此强烈建议使用 Power Management Status/Control register 中的 No_Soft_Reset 字段来实现 MFD 中的每个 Function。这样，将给定 Function 从 $D3_{hot}$ 转换为 D0 不会破坏其他 active Function 的正常运行。

强烈建议 MFD 中的每个 Endpoint Function 都实现 FLR。FLR 可用于复位单个 Endpoint Function，而不会影响其他 Function，尤其是在那些 active Function。由于 FLR 的静默作用、错误恢复以及清除重用属性，因此也建议将 FLR 用于 single-Function Endpoint 设备。

公众号·芯片业者相承及

5.3.1.4.2 $D3_{cold}$ State

断开主电源后，Function 会转换为 $D3_{cold}$ 状态。上电过程及其相关的 Cold Reset 将 Function 从 $D3_{cold}$ 状态转换为 $D0_{uninitialityized}$ 状态。此时，软件必须对 Function 进行完全初始化，以便重新建立所有功能上下文，从而完成将 Function 恢复到其 $D0_{active}$ 状态的操作。

支持 $D3_{cold}$ 唤醒功能的 Function 必须在恢复过程中维护其 PME 上下文（在 PMCSR 中），以供 PME 服务例程软件进行检查。

IMPLEMENTATION NOTE

PME Context

PME 上下文的示例包括但不限于 Function 的 PME_Status 字段、请求代理的 Requester ID、Caller ID（如果调制解调器支持）、触发恢复事件的 IP 定向网络数据包的 IP 信息等。

当系统软件执行“写 1 清除”配置事务把 PCI-PM 兼容 PMCSR 的 Function 的 PME_Status 置为有效时，该 Function 的 PME 有效就被 Ack。

必须使用辅助电源来支持链路重新激活时该 Function 的 PME 事件检测，并保留 $D3_{cold}$ 中的 PME 上下文。 请注意，由于链路重新激活，一旦 I/O 层次结构回到完全通信状态，唤醒逻辑便会将 PME 消息传播到层次结构的根，指示 PME 事件的来源。有关 PME 的更多详细信息，请参阅 Section 5.3.3。

5.3.2 PM Software Control of the Link Power Management State

链路的电源管理状态由其下游组件的 D-state 决定。

Table 5-2 描述了组件（具有上游端口）的电源状态与其上游链路之间的关系。

Table 5-2 Relation Between Power Management States of Link and Components

Downstream Component D-State	Permissible Upstream Component D-State	Permissible Interconnect State
D0	D0	L0, L0s, L1 (1), L2/L3 Ready
D1	D0-D1	L1, L2/L3 Ready
D2	D0-D2	L1, L2/L3 Ready
$D3_{Hot}$	D0- $D3_{Hot}$	L1, L2/L3 Ready
$D3_{Cold}$	D0- $D3_{Cold}$	L2 (2), L3

Notes:

1. Requirements for ASPM L0s and ASPM L1 support are form factor specific.
2. If Vaux is provided by the platform, the Link sleeps in L2. In the absence of Vaux, the L-state is L3.

PCI-PM 兼容的电源管理规则如下：

- 处于 D0，D1，D2 和 $D3_{hot}$ 中的设备必须能通过发送 PME_TO_Ack 消息来响应对 PME_Turn_Off 消息的接收。
- 在任何设备 D-state 下，在执行 PME_Turn_Off/PME_TO_Ack 握手序列之后，下游组件必须使用 PM_Enter_L23 DLLP 请求将链路转换为 L2/L3 Ready。根据 L2/L3 Ready Entry 过渡协议，下游组件必须准备就绪以减少主电源和参考时钟。
- 单功能设备的 Upstream Port 必须通过编程为 D1，D2 或 $D3_{hot}$ 才能启动到 L1 的链路状态转换。对于 Switch，系统软件负责

公众号：芯片业者相承双

确保按照层次结构 PM 策略以兼容的方式对 Switch 的 Upstream Port 进行 D-state 编程(即,Upstream Port 不能编程为 D-state，其 active 要低于 Downstream Port 和与其相连接的更上层的 Function 的 active）。

- non-ARI multi-Function 设备的 Upstream Port 在将其所有功能都编程为非 D0 的 D-state 之前，不得启动向 L1 的链路状态转换（代表 PCI-PM）。
- ARI 设备的 Upstream Port 不得将链路状态转换为 L1（代表 PCI-PM），直到至少一个功能被编程为 non-D0 状态，并且其所有功能都处于 non-D0 状态或 $D0_{uninitialized}$ 状态。

5.3.2.1 Entry into the L1 State

Figure 5-2 描述了一个过程，在该过程中，电源管理软件将下游连接的组件编程为较低电源状态（D1，D2 或 D3hot 状态），以使链路转换为 L1 状态。该 Figure 和随后的描述概述了正被编程为 non-D0 状态的 single -Function Downstream component 的状态切换过程。

公众号·芯片业者相承双

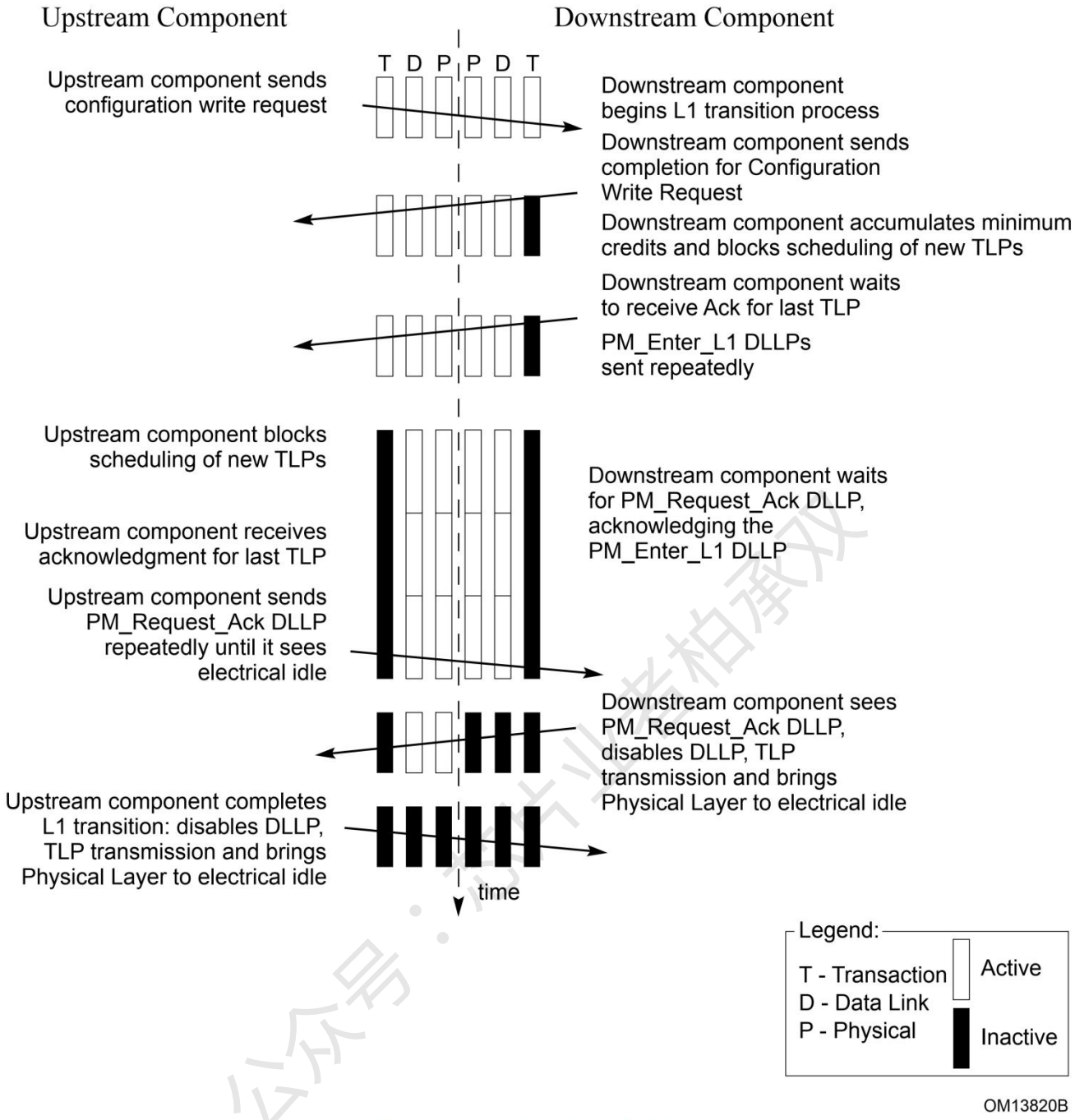

Figure 5-2 Entry into the L1 Link State

下述过程详细描述了 Figure 5-2。

PM Software Request:

1. PM 软件发送 CfgWr TLP 到 Downstream Function 的 PMCSR，以更改 Downstream Function 的 D-state（例如，从 D0 到 D1）。

Downstream Component Link State Transition Initiation Process:

2. Downstream component 将调度一个 CPL 响应并计算 credit 值。

3. 然后，Downstream component 必须等待，直到它至少累积了发送任何 FC 类型的最大可能数据包所需的最小信用数（如果尚未拥有此类信用）。然后，暂停所有事务层 TLP 调度。

4. 然后，Downstream component 等待直到上述 CPL 以及之前已发送的任何其他 TLP 被链路层确认（收到 Ack）。如果因数据

链路层规则要求，组件可能必须从其数据链路层 Retry buffer 中重传 TLP。

5. 一旦确认了所有 Downstream component 的 TLP，Downstream component 便开始传输 PM_Enter_L1 DLLP。Downstream component 以不超过四个（使用 8b/10b 编码时）或 16 个（使用 128b/130b 编码时）的 Symbol time 的时间重复发送该 DLLP，它们在两次 PM_Enter_L1 DLLP 传输之间的空闲时间为零。在 PM_Enter_L1 传输之间的任何时间都允许传输其他 DLLP 和 SKP 有序集，并且不影响此空闲时间限制。

   Downstream component 如上所述继续发送 PM_Enter_L1 DLLP，直到它接收到来自 Upstream component 的响应（PM_Request_Ack）。如果此时 Downstream component 需要在链路上启动传输事务，则它必须首先完成到 L1 的转换。一旦进入 L1，就可以启动退出 L1 来处理传输事务。

   Downstream component 必须继续接受来自 Upstream component 的 TLP 和 DLLP，并根据需要继续响应 DLLP，包括 FC update DLLP 和 Ack/Nak DLLP。必须保存任何被阻塞传输的 TLP（包括对收到的 TLP 的响应）以供以后传输，并且必须使 Downstream component 在 L1 进入后尽快启动 L1 退出。

Upstream Component Link State Transition Process:

6. 收到 PM_Enter_L1 DLLP 后，Upstream component 将阻塞所有 TLP 传输的调度。

7. 然后，Upstream component 必须等待，直到收到其先前发送的最后一个 TLP 被链路层确认为止。如果链路层规则要求这样做，则 Upstream component 必须从其链路层重试缓冲区重新传输 TLP。

8. 确认所有 Upstream component 的 TLP 后，不管挂起的任何事务，Upstream component 必须向下游发送 PM_Request_Ack DLLP。Upstream component 以不超过四个（使用 8b/10b 编码时）或 16 个（使用 128b/130b 编码时）的 Symbol time 的时间重复发送该 DLLP，它们在两次 PM_Enter_L1 DLLP 传输之间的空闲时间为零。在 PM_Request_Ack 传输之间的任何时间都可以传输 SKP 有序集，并且不影响此空闲时间限制。

   Upstream component 继续如上所述发送 PM_Request_Ack DLLP，直到观察到其接收通道进入电气空闲状态为止。有关物理层行为的更多详细信息，请参见 Chapter 4。

Completing the L1 Link State Transition:

9. 一旦 Downstream component 在其接收通道上捕获了 PM_Request_Ack DLLP（表明 Upstream component 已确认向 L1 请求的转换），它便禁用 DLLP 传输，并使上游定向的物理链路进入电气空闲状态。

10. 当 Upstream component 上的接收通道进入电气空闲状态时，Upstream component 将停止发送 PM_Request_Ack DLLP，禁用 DLLP 传输，并将其发送通道变为电气空闲，以完成链路到 L1 的切换。

    当由于 Downstream component 被编程为 non-D0 状态而导致两个组件的互连链路处于 L1 时，两个组件都将挂起其 Flow Control Update 和 Update FCP Timer（如果实现）的操作（see Section 2.6.1.2）。有关物理层行为的更多详细信息，请参见 Chapter 4。

参考 Section 5.2 获取 L1 协商过程。

L1 中链路任一端的组件可以选择禁用其内部 PLL，以节省更多能量。但是请注意，必须继续向 L1 链路两端的组件提供平台提供的主电源和参考时钟。

有关进入 L1 PM 子状态的信息，请参阅 Section 5.5 。

5.3.2.2 Exit from L1 State

L1 退出可由链路任一侧组件发起。

公众号·芯片业者相承双

从 L1 退出时，建议 Downstream component 在 L1 退出后 1us 内为所有已启用 VC 发送流控制更新 DLLP。

链路从 L1 切换回 L0 的物理机制在 Chapter 4 中介绍。

如果组件需要在链路上发送 TLP，则必须由该组件启动 L1 退出。即使 Upstream component 没有传输 TLP 所需的流量控制信用，也必须在其 Downstream Port 上发起 L1 退出。L1 退出后，Upstream component 必须等待从 Downstream component 接收所需的信用。Figure 5-3 概述了一个示例，该序列将触发一个 Upstream component 以启动链路到 L0 状态的转换。

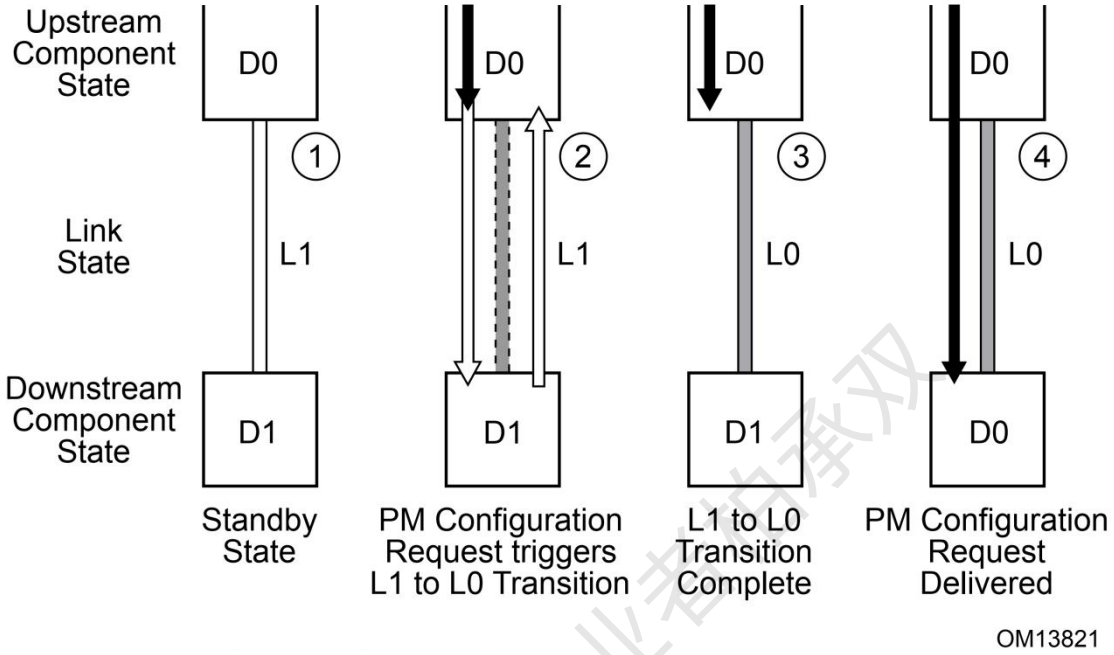

Figure 5-3 Exit from L1 Link State Initiated by Upstream Component

事件顺序：

1. 电源管理软件发送一个针对 PM 配置寄存器（在此示例中为 PMCSR 的 PowerState 字段）的配置请求（例如，使 Function 返回到 D0 状态）。
2. Upstream component 检测到配置请求是针对当前处于低功耗状态的链路的，因此，启动该链路到 L0 状态的转换。
3. 根据 Chapter 4 的定义，链路的两个方向都进入链路训练，从而导致链路转换为 L0 状态。
4. 一旦链路的两个方向都恢复为有效的 L0 状态，则配置请求才能正常发送。

5.3.2.3 Entry into the L2/L3 Ready State

切换到 L2/L3 Ready 状态的过程类似于 L1 进入过程。两者之间有一些细微的区别，说明如下。

- L2/L3 Ready 的进入转换协议不会立即导致链路切换为 L2 或 L3 状态。切换到 L2/L3 Ready 实际上是一次握手，可以确定 Downstream component 是否已准备好断电。当平台断开组件的电源和参考时钟时，最终将进入 L2 或 L3。
- L2/L3 Ready 进入转换的时间由 PME_Turn_Off/PME_TO_Ack 握手序列的完成指示。在启动向 L2/L3 Ready 转换之前，必须完成 Downstream component 为自己准备断电所必需的任何操作。一旦完成了有关断电和时钟的所有准备工作，则下游组件通过向上游发送 PM_Enter_L23 DLLP 来启动 L2/L3 Ready 的进入。
- L2/L3 Ready 进入转换通过 PM_Enter_L23 DLLP 实现。

  注意，PM_Enter_L23 DLLP 连续发送，直到收到确认或断电为止。

公众号：芯片业者的承双

- 如果切换到 L2/L3 Ready 的协商中断，请参阅 Section 5.2。

5.3.3 Power Management Event Mechanisms

5.3.3.1 Motivation

PCI Express PME 机制与 *PCI Bus Power Management Interface Specification* 定义的 PME 机制是软件兼容的。Power Management Event 是由 Function 生成的，表示发起一个 PM state 的切换。Power Management Event 通常用于从低功耗状态恢复系统或单个 Function。

电源管理软件可以将分层结构转换为低功耗状态，并将这些设备的 Upstream Link 转换为非通信的 L2 状态。因此，PCI Express PME 生成机制分为两个部分：

- 唤醒非通信的分层结构（wakeup）。仅当发起 PME 的设备的 Upstream Link 处于非通信的 L2 状态时，才需要执行此步骤，因为在该状态下，设备无法向上游发送 PM_PME 消息。
- 将 PM_PME 消息发送到分层结构的根。

来自 PCI Express Endpoint 或 PCI Express Legacy Endpoint 的 PME 以 TLP 消息的形式传播到 RC。PM_PME Message 通过 PME Message header 中的 Requester ID 标识分层结构中的请求。PM_PME Message 中的显式标识旨在促进更快的 PME 服务程序响应，从而缩短恢复时间。

如果实现了 Root Complex Event Collector，RCiEP 可选的在 Root Complex Event Collector 中报告 PME 事件。Root Complex Event Collector 必须明确声明支持的 RCiEP 作为其功能的一部分；每个 RC Integrated Endpoint 必须与一个以上的 Root Complex Event Collector 关联。Root Complex Event Collector 明确标识请求代理的逻辑位置，以促进更快的 PME 服务程序响应。

来自 Root Port 本身的 PME 通过同一 Root Port 报告。

5.3.3.2 Link Wakeup

链路唤醒机制提供了一种信令，以通知平台重新为其组件重建电源和参考时钟。有两种唤醒机制：Beacon 和 WAKE#。Beacon 机制使用带内信令来实现唤醒功能，请参见 Section 4.3.5.7。对于支持唤醒功能的组件，实现目标的设备规格规范确定了唤醒机制的支持要求。对于某些端口使用 Beacon 且其他端口使用 WAKE # 的 Switch 组件，必须适当地转换唤醒机制（请参阅“Example of WAKE# to Beacon Translation”的实现说明）。在 WAKE # 是唯一使用的唤醒机制的应用程序中，不需要 RC 来支持 Beacon 接收。

WAKE # 机制使用边带信令来实现唤醒功能。WAKE # 是一个“开漏（open drain）”信号，由要求唤醒的组件发出并由相关的电源控制器观察到。WAKE # 仅针对某些设备规格定义，并且 WAKE # 的详细规格包含在相关的设备规格中。特定的设备规格可能需要使用 Beacon 或 WAKE # 作为唤醒机制。

当将 WAKE # 用作唤醒机制时，一旦 WAKE # 被置为有效，置起该信令的 Function 必须继续将信号驱动为 low，直到主电源恢复到基本复位释放情况下的组件的功率为止。

系统不需要以 Endpoint 能够检测到信号已被另一个 Function 置起的方式路由或缓冲 WAKE # 。

在使用任何唤醒机制之前，必须通过将 PMCSR 中的功能的 PME_En 位置 1 来使能该功能。PME_Status 位是粘性的，如果有辅助电源并且启用了唤醒事件，则 Function 必须通过复位来保持 PME_Status 位的值（此要求也适用于 PMCSR 中的 PME_En 字段和 Device Control register 中的 Aux Power PM Enable 字段）。

当主系统电源轨关闭时，允许从 $D3_{cold}$ 状态生成 PME 的系统必须提供辅助电源以支持链路唤醒。如果软件启用了组件，则只能消耗辅助电源，如 Section 5.5.1 所述。需要使用软件来启用参与链路唤醒的所有组件（包括必须传播 Beacon 信令的所有组件）的辅助功耗。在存在旧版系统软件的情况下，这是系统固件的职责。

无论使用哪种唤醒机制，一旦重新激活了链路并对其进行了训练，请求代理便会将 PM_PME 消息上游传播到 RC。从电源管理的角度来看，这两种唤醒机制提供相同的功能，在本章的其他地方没有区别。

IMPLEMENTATION NOTE

Example of WAKE# to Beacon Translation

面向连接“Beacon domains”和“ WAKE# domains”应用的 Switch 组件必须适当地转换唤醒机制。Figure 5-4 显示了两个示例系统，每个系统都包含使用 WAKE # 唤醒机制的插槽。在 Case 1 中，WAKE # 直接输入到 Power Management Controller，不需要转换。在 Case 2 中，WAKE # 是 Switch 的输入，并且要响应 WAKE # ，这种情况下 Switch 必须生成一个 Beacon，将该 Beacon 传播到 RC/Power Management Controller。

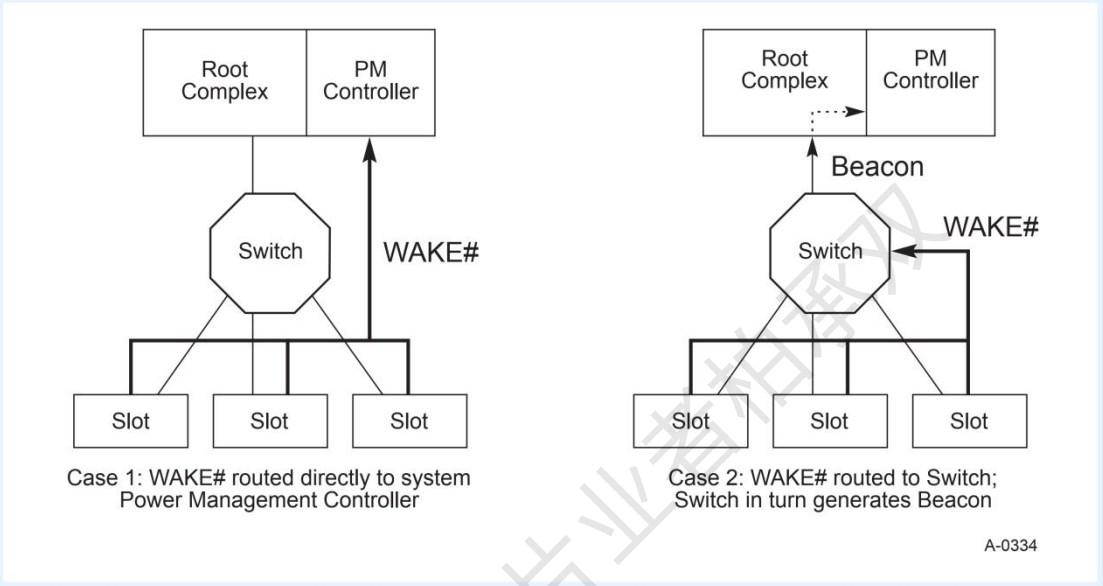

Figure 5-4 Conceptual Diagrams Showing Two Example Cases of WAKE# Routing

这里需要考虑的一个问题是作为 EP 设备，如果它支持 WAKE#或 Beacon 的唤醒方式，那么 EP 设备在什么样的场景下会触发控制器发出 WAKE#或 Beacon？Synopsys 的 IP 提供了一个一个 SII 信号，叫 outband_pwrup_cmd，每个 Function 一个，一旦此信号翻转，会触发一个 WAKE#或 Beacon。我的理解是此信号在集成的时候可以拉到 PAD 上，然后由 PCB 上其他器件触发此信号。

5.3.3.2.1 PME Synchronization

PCI Express-PM 引入了一种隔离机制，该机制可用于开启一个电源移除过程，同时还可协调平台电源管理控制器的行为以及 PCI Express 代理对 PME 的处理。

PME_Turn_Off Broadcast Message

在关闭主要组件电源和参考时钟之前，RC 或 Switchd 的 Downstream Port 必须发送一个广播消息，以指示该分层结构中下游的所有设备停止任何后续的 PM_PME 消息的初始化，此过程在收到 PME_Turn_Off 消息后立即生效。

每个 PCI Express 设备都需要用一个始终被路由到上游的 PME_TO_Ack TLP 消息来响应。在任何情况下，PME_TO_Ack 消息都必须在 PME_Turn_Off 消息的源点处终止。PME_Turn_Off 消息的源点可以是给定 RC（具有完整平台睡眠状态转换）的所有 Root Port、单个 Root Port 或 Switch 的 Downstream Port。

Switch 必须在每个 Downstream Port 接收到 PME_TO_Ack 消息后，才报告所有的确认信息。一旦 PME_TO_Ack 消息到达每个 Downstream Port，Switch 就必须在其 Upstream Port 上发送 PME_TO_Ack 数据包。如果发生以下任何一种情况，则必须重置聚合（aggregation）机制：从 Upstream Port 发送 PME_TO_Ack 消息；在 Upstream Port 接收到任何 TLP；切断 Switch 的主电源；或基本复位。

所有有 Upstream Port 的组件都必须接收并确认 PME_Turn_Off 消息，而不管关联设备的 D-state 或 multi-Function 设备的任何 Function 如何。一旦组件发送了 PME_TO_Ack 消息，它就必须准备通过启动向 L2/L3 Ready 状态的转换来移除其电源和参考时钟。

Switch 的所有 Downstream Port 都进入 L2/L3 Ready 状态后，它必须将其 Upstream Link 转换为 L2/L3 Ready 状态。

连接到发送 PME_Turn_Off 消息的始发者的链路是最后进入 L2/L3 Ready 状态的链路。此状态转换向 power delivery manager 指示，该分层结构中的所有链路已成功完成所有针对 PME_Turn_Off 消息的正在传输的 PME 消息，并已执行任何必要的本地条件以准备断电。

在此上下文中，power delivery manager 涉及对整个链路层次结构的控制，或者涉及对子层次结构（控制独立的电源和时钟分配）的子层次结构（直至单个链路和关联的端点）的控制。

为了避免在一个或多个设备不响应 PME_TO_Ack 消息而将其链路置于 L2/L3 Ready 状态的情况下出现死锁，电源管理器必须在等待一定时间后执行超时，就像收到了消息并且所有链路都进入 L2/L3 Ready 状态一样。此计时器的建议容差为 1ms 至 10ms。

IMPLEMENTATION NOTE

PME_TO_Ack Message Proxy by Switches

PME_Turn_Off/PME_TO_Ack 握手的关键作用之一是确保在关闭睡眠状态电源之前，将所有正在运行的 PME 消息从 PCI Express 结构中清除掉。由于 PME 消息和 PME_TO_Ack 消息都使用 VC0 中的的阻塞请求队列，因此可以保证上述情况发生，因此在 RC 收到 PME_TO_Ack 之前，所有先前注入的 PME 消息将对系统可见。一旦 RC 的所有下游端口收到 PME_TO_Ack 消息，RC 便可以向电源管理器发送信号，表明可以安全地断电而不会丢失任何 PME 消息。

Switch 会创建分层结构扩展点，因而它们必须等待所有已连接的下游端口接收到 PME_TO_Ack 消息，然后才能代表已创建下游的子层次结构向上游发送 PME_TO_Ack 消息。 可以使用常见的计分板技术非常简单地完成此操作。例如，一旦在 Switch 的下游广播了 PME_Turn_Off 广播消息，Switch 就简单地检查接收到 PME_TO_Ack 的每个下行端口。一旦其最后一个活动下游端口收到 PME_TO_Ack，Switch 便代表其整个子层次结构下游发送单个 PME_TO_Ack 消息上游作为代理。请注意，一旦下游端口接收到 PME_TO_Ack 消息并且 Switch 已对其到达进行评分，则该端口便可以从其内部队列中丢弃数据包，并释放相应的已发布请求队列 FC 信用。

5.3.3.3 PM_PME Messages

PM_PME 消息是 posted 的事务层数据包（TLP），用于通知电源管理软件分层结构中的哪个代理请求 PM 状态更改。像所有其他电源管理系统消息一样，PM_PME 消息必须使用通用 Traffic Class ： TC # 0。

PM_PME 消息始终按 RC 的方向路由。要在其上游链路上发送 PM_PME 消息，设备必须将链路转换为 L0 状态（如果链路尚未处于该状态）。除非另有说明，否则在发送 PM_PME 消息后，设备会将链接保持在 L0 状态。

5.3.3.3.1 PM_PME “Backpressure” Deadlock Avoidance

RC 通常使用本地缓冲来实现，以临时存储有限数量的 PM_PME 消息，这些消息可能同时在分层结构中传播。如果可以在 RC 中存储有限数量的 PM_PME 消息，则在超出此临时 PM_PME 消息缓冲区容量的情况下，可能会将反压应用于上游定向的已发布队列。

根据以下示例情况，可能会发生死锁：

1. 传入的 PM_PME 消息会填充 RC 的临时存储空间，而分层结构中还有其他 PM_PME 消息仍在上行。
2. RC 代表系统软件发出针对 PME 请求者的 PMCSR 之一的配置读取请求（例如，读取其 PME_Status 位）。

3. 根据生产者/消费者规则，需要相应的完成数据包将所有先前发布的 PM_PME 消息（在这种情况下是无处可去的 PM_PME 消息）推向其前面。

4. PME 服务程序无法取得进展； PM_PME 消息存储情况没有改善。

5. 死锁发生。

要排除潜在的僵局，RC 必须始终在这些情况下实现向前的进展。必须通过接受已发布队列流量控制信用允许的任何 PM_PME 消息，并丢弃任何会创建溢出条件的 PM_PME 消息来完成此操作。这种必需的行为可确保在这种情况下不会发生死锁。但是，PM_PME 消息将被丢弃，从而在此过程中丢失。

为了确保不会永久丢失 PM_PME 消息，所有能够生成 PM_PME 的代理都必须实现 PME 服务超时机制，以确保在合理的时间内为他们的 PME 请求提供服务。

如果在 100ms（+ 50％/-5％）之后，仍未清除请求代理的 PME_Status 位，则 PME 服务超时机制将到期，触发 PME 请求代理重新发送临时丢失的 PM_PME 消息。如果此时链路处于非通信状态，则在重新发送 PM_PME 消息之前，代理必须按照 Section 5.3.3.2 的定义重新激活该链路。

5.3.3.4 PME Rules

- 所有设备的 Function 都必须根据 PCI-PM 规范实现 PCI-PM Power Management Capabilities (PMC) register 和 PMCSR。这些寄存器采用 PCI-PM 兼容的 PCI 功能列表格式。
  - 具有 PME 功能的 Function 必须在其 PMCSR 中实现 PME_Status 字段以及基础功能行为。
  - 当 Function 启动链路唤醒或发出 PM_PME 消息时，它必须将其 PME_Status 字段置 1。
- Switch 必须将在任何 Downstream Port 上收到的 PM_PME 路由到其 Upstream Port 。
- 收到 PME_Turn_Off 消息后，设备必须阻止 PM_PME 消息的传输并向上游发送 PME_TO_Ack 消息。链路通过 LDn 返回到 L0 状态后，允许组件发送 PM_PME 消息。
- 在将链路或分层结构的一部分转换为非通信状态（即无法从其发出 PM_PME 消息的状态）之前，必须向下游广播 PME_Turn_Off 消息。

5.3.3.5 PM_PME Delivery State Machine

Figure 5-5 从概念上概述了 PM_PME 传递控制状态机。此状态机通过立即发出 PM_PME 与请求链路唤醒来确定链路服务 PME 事件的能力。

公众号·芯片业者相承双

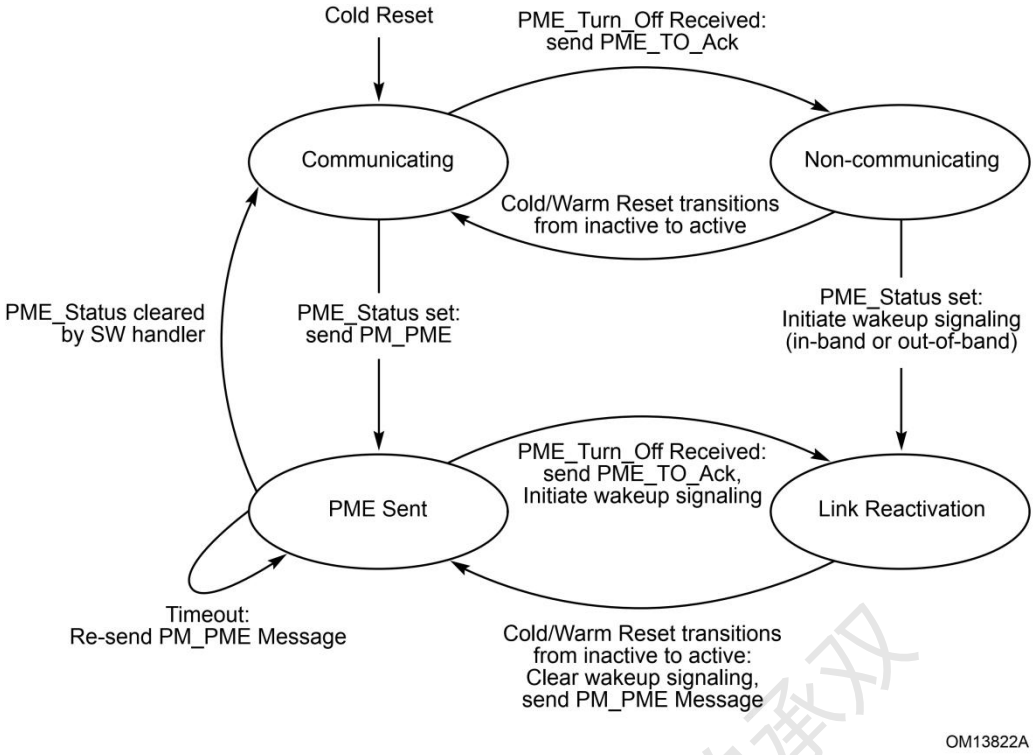

Figure 5-5 A Conceptual PME Control State Machine

Communicating State:

在初始加电和相关联的复位时，Upstream Link 进入通信状态。

- 如果 PME_Status（假定启用了 PME delivery）被置起，则会向上游发出 PM_PME 消息，该消息终止于分层结构的根。下一个状态是 PME Sent 状态。
- 如果收到 PME_Turn_Off 消息，则链路在确认该消息后并随后进入 L2/L3 Ready 状态后，将进入 Non-communicating state 。

Non-communicating State:

- 在恢复电源和时钟以及相关的复位之后，下一个状态是 Communicating State 。
- 如果 PME_Status 被置起，则链路将转换为链路重新激活状态，并激活唤醒机制。

PME Sent State

- 如果清除了 PME_Status，该 Function 将重新具有 PME 能力。下一个状态是 Communicating State 。
- 如果在 PME 服务超时到期之前 PME_Status 位未清除，则向上游重新发送 PM_PME 消息。有关超时机制的说明，请参见 Section 5.3.3.3.1。
- 如果已发布 PME 消息，但在链路即将转换为无法发送消息的状态（接收到 PME_Turn_Off 消息）时未通过软件清除 PME_Status，则在发送 PME_TO_Ack 消息后，链路将转换为链接重新激活状态。该设备还将激活唤醒机制。

Link Reactivation State

- 在恢复电源和时钟以及相关的复位之后，链路将恢复具有事务处理能力的状态。设备会在必要时清除唤醒信令，并发出 PM_PME 并转换为 PME Sent State。

公众号：芯片业者相承双

5.4 Native PCI Express Power Management Mechanisms

以下各节定义的电源管理功能需要新的软件服务程序。尽管在新的 PCI Express 设计中这些功能的存在不会破坏旧版软件的兼容性，但要充分利用这些功能，则需要新的代码来对其进行管理。

使用本规范 Chapter 7 所述的 PCI Express 本地配置机制可以枚举和配置这些功能。有关这些 PCI Express-PM 功能的寄存器定义请参见 Chapter 7。

5.4.1 Active State Power Management (ASPM)

与 Internal RC Link 或 system Egress Port 无关的所有端口都必须支持此处为 ASPM 定义的最低要求。 从最低要求的角度出发，必须将此功能视为与 PCI-PM 软件兼容功能正交。例如，RC 不受 PCI-PM 软件兼容功能的要求；但是，它必须执行 ASPM 的最低要求。

根据 Section 5.3.2 的定义，处于 D0 状态的组件通常会将其 Upstream Link 保持在 active L0 状态。ASPM 为处于 D0 状态的组件定义协议，以通过将其链路置于低功耗状态并指示链路的另一端进行同样操作来降低链路功耗。此功能可实现硬件自主的动态链路低功耗功能，其范围超出了仅由软件控制（即 PCI-PM 软件驱动）的电源管理所能达到的范围。

为 ASPM 定义了两个低功耗"standby"链路状态。L0s 低功耗链路状态针对短的进入和退出等待时间进行了优化，同时节省了大量功率。如果设备中启用了 L0s 状态，则建议该设备在不使用该链路时将其发送链路置于 L0s 状态（有关 L0s 启用策略的详细信息，请参阅 Section 5.4.1.1.1）。D0 设备状态中对 L0s 链路状态的组件支持是可选的，除非设备明确要求。

L1 链路状态经过优化，可最大程度地节省功耗，但需要更长的进入和退出等待时间。如果需要非常低的功耗并且可以接受更长的转换时间，则 L1 将链路功率降低到 L0s 状态以上。 除非设备明确要求，否则对 L1 链路状态的 ASPM 支持是可选的。

每个组件都必须在 ASPM Support 字段中报告其对 ASPM 的支持级别。实现时，每个组件还应报告其 L0s 和 L1 退出等待时间（指从 L0s 或 L1 状态转换到 L0 状态所需的时间）。Endpoint Function 还必须报告它们在承受风险之前可以承受的最坏情况下的延迟，例如，由于从 L0s 或 L1 到 L0 状态的转换延迟导致内部 FIFO 溢出。电源管理软件可以使用提供的信息来启用适当级别的 ASPM。

如果给定链路的对端的参考时钟是从同一来源提供的，或者是从不同来源传递给每个组件的，则 L0s 的退出延迟可能会大大不同。PCI Express-PM 软件通过 Capability structure's Link Control register 中的 Common Clock Configuration 字段报告每个设备的时钟配置。该字段用作设备报告的 L0s 退出延迟值的决定因素。默认情况下，ASPM 可以启用或禁用，具体取决于实现要求。软件可以使用 Section 5.4.1.3.1 描述的过程启用或禁用 ASPM。

电源管理软件通过对 ASPM Control 字段进行编程来启用或禁用组件每个端口中的 ASPM。请注意，当在旧版操作系统上运行时，新的 BIOS 代码可以有效地启用或禁用 ASPM 功能，但支持 PCI Express 的操作系统可能会选择覆盖 BIOS 配置的 ASPM 设置。

IMPLEMENTATION NOTE

Isochronous Traffic and ASPM

定时业务（Isochronous Traffic）需要有限的服务延迟。ASPM 可能会使同步事务的延迟增加到超出预期的限制。一种可能的解决方案是为配置了 Isochronous Virtual Channel 的设备禁用 ASPM。

对于 ARI 设备，ASPM 控制完全由 Function 0 中的设置决定，而与 Function 0 的 D-state 无关。组件将忽略其他 Function 中的 ASPM Control 设置。

non-ARI multi-Function 设备的 Upstream Port 可以在每个 Function 各自的 ASPM Control 字段中使用不同的值编程。此类组件的策略将由所有 D0 Function 中最活跃的设备根据以下规则决定：

- 在确定 ASPM 策略时会忽略处于 non-D0 状态（D1 和更深）的 Function 。

- 如果处于 D0 状态的任何 Function 的 ASPM 被禁用（ASPM Control = 00b），或者处于 D0 状态的 Function 至少有一个启用了 L0s（ASPM Control = 01b），并且处于 D0 状态的 Function 至少有一个启用了 L1（ASPM Control = 10b），然后需要对整个组件禁用 ASPM 。
- 否则，如果处于 D0 状态的 Function 至少有一个启用了 L0s（ASPM Control = 01b），则仅对启用了 L0s 的 Function 启用 ASPM。
- 否则，如果处于 D0 状态的 Function 至少有一个启用了 L1（ASPM Control = 10b），则仅对启用了 L1 的 Function 启用 ASPM。
- 否则，L0s 和 L1 状态均启用 ASPM。

请注意，随着设备 Function 进入和退出低功耗设备状态，组件必须能够在运行时更改其行为。例如，如果将 multi-Function 设备中的一个 Function 编程为禁用 ASPM，则必须在该功能处于 D0 状态时为该设备禁用 ASPM。一旦该 Function 转换为 non-D0 状态，如果所有其他 Function 启用了 ASPM，则该 Function 可以启用 ASPM。

5.4.1.1 L0s ASPM State

设备对 L0s 低功耗链路状态的支持是可选的，根据设备规格与需求来决定。

IMPLEMENTATION NOTE

Potential Issues With Legacy Software When L0s is Not Supported

在本规范的早期版本中，必须对 L0s 提供设备支持，并且软件可以合法地假定所有设备都支持 L0s。不支持 L0s 的较新硬件组件可能会遇到此类“旧版软件”的问题。此类软件甚至可能不会检查 Link Capability register 的 ASPM Support 字段，当然也可能无法识别为 ASPM Support 字段定义的值（00b 和 10b），或者可能仅在链路两侧都支持 L0s 的情况下才遵循启用 L0s 的策略。

ASPM Support 保留的值为 00b（No ASPM Support）的旧版软件（操作系统或固件）最有可能避免启用 L1，这是预期的行为。旧版软件也很可能会为该组件的发送器禁用 L0s（也是预期的行为），但目前尚不清楚此类软件是否还会为链路另一侧的组件禁用 L0s。如果软件在一侧启用 L0s，在另一层不支持 L0s，则结果是不确定的。可能需要通过更新旧版软件，诉诸“blacklists”或类似的机制来指导旧版软件不启用 L0s 或根本不支持有问题的系统配置，来处理由此产生的不可接受行为的情况。

在某些平台上，固件控制 ASPM，并且操作系统可以保留或覆盖由固件建立的 ASPM 设置。这将受到操作系统是否支持控制 ASPM 的影响，并且在某些情况下会受到固件是否允许操作系统控制 ASPM 的影响。 同样，通过使用热插拔操作的 ASPM 控制可能会受到是否使用本机 PCI Express 热插拔与 ACPI 热插拔的影响。要解决 L0s 的任何旧版软件问题，可能需要更新固件，操作系统或两者都要更新。

当组件不通告其支持 L0s 时，即组件的 ASPM Support 字段值为 00b 或 10b 时，建议组件的 L0s Exit Latency 字段值设为 111b，指示最大延迟范围。通告此最大等待时间范围可能帮助旧版软件禁用 L0s，从而可以避免因旧版软件错误地在此组件或链路另一侧的组件上启用 L0s 而导致的问题。

事务层和链路层计时器不受转换到 L0s 状态的影响（即，它们必须遵循各自章节中定义的规则）。

IMPLEMENTATION NOTE

Minimizing L0s Exit Latency

L0s 的退出等待时间主要取决于接收器快速获取比特同步和符号同步的能力。高频时钟解决方案存在不同的方法，这些解决方案的 L0s 退出延迟时间以及 ASPM 的效率可能存在显着差异。为了让 ASPM 获取最大的节能效能，L0s 退出延时必须选择合适的解决方案。

公众号·芯片业者相承双

5.4.1.1.1 Entry into the L0s State

对于链路的每个方向，分别管理进入 L0s 状态。链路两端的每个设备都有责任在其发送通道上启动进入 L0s 状态的过程。除非链路两侧的组件均支持 L0s，否则软件不得在给定链路上的任一方向上启用 L0s，因为这个操作的结果是不确定的。

被禁用 L0s 状态的端口不得将其发送通道转换为 L0s 状态。但是，如果端口通告它支持 L0s，则端口必须能够忍受其接收器端口通道进入 L0s（由于另一端的设备将其发送通道变为 L0s 状态），然后再返回 L0 状态。（注意理解这里的禁用和支持关键字。）

L0s Invocation Policy

如果在一段时间内满足下面定义的空闲条件（建议不超过 7μs），则使能了 L0s 进入的端口通常应将其传输通道转换为 L0s 状态。在此时间段内，端口用于确定何时进入 L0s 的策略是特定于实现的。不能强制发送器进入 L0s 状态。

Definition of Idle

关于"空闲"的定义 Upstream Port 因设备 Function 类别而异。仅当 multi-Function 设备的所有 Function 均处于空闲状态时，才将其 Upstream Port 视为空闲状态。

如果满足以下条件，则确定 non-Switch Port 为空闲状态：

- 没有 TLP 在等待传输，或者没有 FC credit 可用于传输任何 TLP。
- 没有 DLLP 在等待传输。

如果满足以下条件，则确定 Switch 的 Upstream Port 的 Function 为空闲：

- Switch 的 Downstream Port 的 Receive Lane 均未处于 L0、Recovery 或 Configuration 状态。
- 没有 TLP 在等待传输，或者没有 FC credit 可用于传输任何 TLP。
- 没有 DLLP 在等待传输。

如果满足以下条件，则确定 Switch 的 Downstream Port 为空闲状态：

- Switch 的 Upstream Port 的 Receive Lane 均未处于 L0、Recovery 或 Configuration 状态。
- 没有 TLP 在等待传输，或者没有 FC credit 可用于传输任何 TLP。
- 没有 DLLP 在等待传输。

有关物理层进入 L0 的详细信息，请参见 Section 4.2。

5.4.1.1.2 Exit from the L0s State

组件的发送器处于 L0s 状态，如果此时有 TLP 或 DLLP 在链路上传输，则必须启动 L0s 退出过程。请注意，从 L0s 发生的切换过程与 FC credit 是否可用无关。链路必须能够达到 L0 状态，并且可以在链路上交换 FC credit。例如，如果在链路进入 L0s 时消耗了某种类型的所有信用，那么当需要通过链路发送新信用时，链路两侧的组件仍必须能够将链路转换为 L0 状态。注意，组件可能需要预料到空闲状态的结束并启动 L0s 退出过程；例如，当收到 NP 类型的请求时。

Downstream Initiated Exit

如果组件的 Upstream Port 需要通过链路进行通信，则允许该 Upstream Port 在其传输链路上启动从 L0s 退出过程。如 Section 4.2 所述，该组件会在上游方向上向 Lane 发起向 L0 状态的转换。

如果上游组件是 Switch（即它不是 RC），则它必须在任何 Downstream Port 上检测到 L0s 退出时立即在其 Upstream Port 传输通

公众号·芯片业者相承双

道上启动向 L0 的转换（如果 Upstream Port 的传输通道处于低功率状态）。

Upstream Initiated Exit

如果 Downstream Port 需要通过链路进行通信，则允许其在其任何传输链路上启动从 L0s 的退出过程。如 Chapter 4 所述，该组件会在“下游”方向上启动到 L0 状态的转换。

如果下游组件包含一个 Switch，则一旦它检测到 Upstream Port 上的 L0s 退出，它就必须在当时位于 L0s 中的所有 Downstream Port 传输通道上启动转换。已经处于 L0 状态的链路不受此转换的影响。处于低功耗状态（即 $D1\text{-}D3_{hot}$ 状态）的下游组件的链路也不受此退出过程的影响。

例如，考虑一个 Switch，其 Upstream Port 位于 L0s 中，而下游设备处于 D1 状态。配置请求数据包向下游传输到 Switch，打算将下游设备从 D1 重新编程为 D0。Switch 的 Upstream Port 链路必须转换为 L0 状态，以允许数据包到达 Switch。连接到处于 D1 状态的设备的下行链路仍不会转换为 L0 状态。它将保持在 L1 状态。检查捕获的数据包，并将其路由到与 D1 中的下游设备共享链路的 Downstream Port。如 Section 4.2 所述，Switch 现在将下行链路转换为 L0 状态。请注意，转换为 L0 状态是由将数据包路由到特定的下游 L1 链路触发的，而不是由上游端口的链路转换为 L0 状态触发的。如果数据包的目的地是针对不同的下行链路，则该特定下行链路将保持 L1 状态。

5.4.1.2 L1 ASPM State

组件可以选择支持 ASPM L1 状态。这是一种可提供更大的功耗节省的状态，但以更长的退出等待时间为代价。L1 退出延迟对软件可见，并通过 L1 Exit Latency 字段进行报告。

IMPLEMENTATION NOTE

Potential Issues With Legacy Software When Only L1 is Supported

在此规范的早期版本中，是强制设备要支持 L0s，并且没有合理的 ASPM Support 字段值来指示在不支持 L0s 的情况下对 L1 的支持。仅支持 L1 的较新硬件组件可能会遇到“旧版软件”问题，即无法识别为 ASPM Support 字段定义的值。

遇到先前保留为值 10b（支持 L1）的旧版软件可能会禁止同时启用 L0s 和 L1，但不幸的是，避免将 L1 与仅支持 L1 的新组件一起使用。虽然这可能会导致消耗更多的功率，但不应引起任何功能异常。但是，在此 10b 情况下，与启用 L0s 的旧版软件有关的问题同样存在，如 Section 5.4.1.1 节“Potential Issues With Legacy Software When L0s is Not Supported”中所述。

受支持时，ASPM Control 字段中的 L1 进入功能是默认禁用的。仅当链路上的两个组件都支持 ASPM L1 时，软件才在下游组件上启用 ASPM L1。软件必须对 ASPM L1 的启用和禁用进行排序，以便在下游组件之前启用上游组件，在下游组件之后禁用上游组件。

5.4.1.2.1 Entry into the L1 State

如果组件上的 Upstream Port 使能了 L1 ASPM 进入过程，则它可以发起进入 L1 链路状态的过程。

IMPLEMENTATION NOTE

Initiating L1

该规范并未规定组件的 Upstream Port 何时必须启动到 L1 状态的转换。在本规范中定义了用于进出 L1 的互操作机制。但是，管理何时切换到 L1 的 ASPM 策略留给实现者。

一种可能的方法是，一旦下游设备的接收方和发送方都处于 L0s 状态（RxL0s 和 TxL0s）一段设定的时间，则下游设备将启动到 L1 状态的转换。另一种方法是，一旦链路在 L0 状态已闲置了一段设定的时间，下游设备便会启动到 L1 状态的转换。如果未启用 L0s 进入功能，则此功能特别有用。如果下游设备已完成其分配的任务，则另一种方法是使下游设备启动到 L1 状态的转换。

公众号：芯片业者也承双

请注意，这几个示例绝不限制组件的 L1 的启用策略。

支持 L1 ASPM 的电源管理消息有：

- PM_ Active_State_Request_L1 (DLLP)
- PM_ Request_Ack (DLLP)
- PM_Active_State_Nak (TLP)

使能了 ASPM L1 进入的下游组件与链路上的上游组件协商进入 L1 状态。

如果满足以下所有条件，则 Downstream Port 必须接受进入 L1 的请求：

- 该端口支持 APPM L1 状态，且 ASPM L1 进入已经使能。

  注：仅当也为上游组件启用了 ASPM L1 时，软件才必须为它启用 ASPM L1。

- 没有 TLP 正在调度传输。
- 没有 DLLP 的 Ack 和 Nak 正在调度传输。

如果满足以下所有条件，则 Switch 的 Upstream Port 可以在其链路上请求 L1 进入：

- Upstream Port 支持 ASPM L1 并且使能了这个功能。
- Switch 的所有 Downstream Port 的链路都处于 L1 状态（或 L2、L3）。
- 没有挂起的 TLP 要传送。
- 没有挂起的 DLLP 要传送。
- Upstream Port 的接收器空闲了特定的实现时间。

请注意，为 Switch 在其任何 Downstream Port 上启用 ASPM L1 链路状态，并且在其 Upstream Port 上禁用甚至不支持 ASPM L1 是合法的。在这种情况下，Downstream Port 可能会进入 L1 链路状态，但是 Switch 将永远不会在其 Upstream Port 上启动 ASPM L1 进入转换。

ASPM L1 Negotiation Rules (见 Figure 5-6 和 Figure 5-7)

- Downstream component 必须累积至少发送 FC type 的最大数目的数据包所需的最少信用数，才能启动 ASPM L1 进入过程。
- 在决定进入低功耗链路状态时，Downstream component 必须阻止所有 TLP 从事务层到数据链路层的移动(包括完成数据包)。
  - 如果在 L1 协商过程中可以从事务层获取任何 TLP 进行传输，则必须首先完成到 L1 的转换，然后下游组件必须启动到 L0 的返回。如果与 L1 的协商中断，请参阅 Section 5.2。
- Downstream component 必须等待，直到它收到其先前发送的最后一个 TLP 的链路层确认（即，重试缓冲区为空）。如果数据链路层规则要求，则组件必须从其数据链路层重试缓冲区中重传 TLP。
- 然后，Downstream component 通过发送 PM_Active_State_Request_L1 DLLP 来启动 ASPM 协商。Downstream component 以不超过四个（使用 8b/10b 编码时）或 16 个（使用 128b/130b 编码时）重复发送此 DLLP 的符号时间，它们在 PM_Active_State_Request_L1 DLLP 后续传输之间的空闲时间为符号时间。其他 DLLP 和 SKP 有序集的传输必须在 PM_Active_State_Request_L1 传输之间的任何时间根据需要进行，并且不影响此空闲时间限制。在 L1 进入期间发送 SKP 有序集要遵循 Section 4.2.7 中的时钟容限补偿规则。
- Downstream component 继续如上述发送 PM_Active_State_Request_L1 DLLP，直到它接收到来自 Upstream device 的响应（请参见下文）。Downstream component 停在此循环中，等待 Upstream component 的响应。

公众号：芯片业者相承双

- 在此等待期间，上游组件不得启动任何事务层传输。它仍然必须接受来自上游组件的 TLP 和 DLLP，存储所需的任何 TLP 响应以供以后传输。根据链路层协议的要求，它将继续使用 DLLP（包括 FC update DLLP）进行响应。
- 如果下游组件出于任何原因需要在链路上传输 TLP，则必须首先完成到低功耗链路状态的转换。 一旦处于低功耗链路状态，下游组件就必须启动退出低功耗链路状态以处理传输。如果与 L1 的协商中断，请参阅 Section 5.2。

- Upstream component 必须立即（同时遵守本规范中的所有其他规则）以接受或拒绝请求的方式响应请求。
  - 如果上游组件不能接受该请求，则它必须立即（同时遵守本规范中的所有其他规则）拒绝该请求。
- 如果 L1 的协商中断，请参阅 Section 5.2。

Rules in case of rejection:

- 在拒绝请求的情况下，上游组件必须通过将 PM_Active_State_Nak 消息发送到下游组件来尽快实现拒绝。发送 PM_Active_State_Nak 消息后，允许上游组件启动 TLP 或 DLLP 传输。
- 如果请求被拒绝，通常建议下游组件立即将其传输通道转换为 L0s 状态，前提是 L0s 使能开启并且满足了进入 L0s 的条件。
- 在传输与后续 ASPM L1 协商过程相关联的 PM_Active_State_Request_L1 DLLP 之前，下游组件要么在其发送器上进入和退出 L0s，要么等待与上一 ASPM L1 关联的 PM_Active_State_Request_L1 DLLP 的最后一次传输至少等待 10 s。这个 10µs 计时器只计算在 LTSSM L0 和 L0s 状态中花费的时间。计时器必须在 LTSSM Recovery 状态保持。如果链路断开并重新建立，则将忽略计时器，并允许组件在链路重新建立后发出新的 ASPM L1 请求。

IMPLEMENTATION NOTE

ASPM L1 Accept/Reject Considerations for the Upstream Component

当上游组件通过 PM_Request_Ack DLLP 响应下游组件的 ASPM L1 请求以接受 L1 进入请求时，ASPM L1 协商协议将明确地以链路进入 L1 结束。但是，如果上游组件响应 PM_Active_State_Nak 消息以拒绝 L1 进入请求，则 ASPM L1 协商协议的终止尚不清楚。因此，这两个组件都需要设计明确的终止协议。如果不这样做，则存在两个组件彼此不同步的风险，并且结果可能不确定。例如，考虑以下情况：

- 下游组件通过发送 PM_Active_State_Request_L1 DLLP 序列来请求 ASPM L1 进入。
- 由于临时原因，上游组件以 PM_Active_State_Nak 消息响应以拒绝 L1 请求。
- 在能够响应 PM_Active_State_Nak 消息之前，下游组件会继续发送 PM_Active_State_Request_L1 DLLP 一段时间。
- 同时，解决了先前导致上游组件拒绝 L1 请求的临时条件，并且上游组件错误地将连续的 PM_Active_State_Request_L1 DLLP 视为进入 L1 的新请求，并通过向下游发送 PM_Request_Ack DLLP 进行响应。

此时，结果是不确定的，因为下游组件将 L1 请求视为已拒绝并完成，但上游组件将情况视为已接受第二个 L1 请求。

为了避免这种情况，下游组件需要提供一种机制来区分一个 ASPM L1 请求和另一个 ASPM L1 请求。下游组件通过进入 L0s 或从与第一个 ASPM L1 请求关联的最后一个 PM_Active_State_Request_L1 DLLP 的传输开始等待至少 10 s 来完成此操作，然后再开始与第二个请求关联的 PM_Active_State_Request_L1 DLLP 的传输（如上所述）。

如果上游组件能够表现出上述行为，则上游组件必须通过检测其接收器上的 L0s 转换来识别 L1 请求序列的结尾，或下游组件在 ASPM L1 请求之间的间隔为 L0 / L0 或更长时间时，测量到的 9.5µs 的 PM_Active_State_Request_L1 DLLP 接收中断来判断。

如果存在歧义的可能性，则上游组件应拒绝 L1 请求，以免潜在地造成上述情况。

Rules in case of acceptance:

- 如果上游组件准备好接受请求，则它必须阻止调度来自事务层的任何 TLP。

- 然后，上游组件必须等待，直到收到其先前发送的最后一个 TLP 的数据链路层确认为止。如果数据链路层规则要求，则上游组件必须重新传输 TLP。
- 一旦所有 TLP 均得到确认，上游组件将向下游发送 PM_Request_Ack DLLP。上游组件以不超过四个的符号时间（使用 8b/10b 编码时）重复发送该 DLLP，两次 PM_Request_Ack DLLP 传输之间的空闲符号时间相同。SKP 有序集的传输必须在 PM_Request_Ack 传输之间的任何时间按要求进行，并且不影响此空闲时间限制。在 L1 输入期间发送 SKP 有序集要遵循 Section 4.2.7 中的时钟容限补偿规则。
- 上游组件继续如上所述发送 PM_Request_Ack DLLP，直到它观察到其接收通道进入电气空闲状态为止。 有关物理层行为的更多详细信息，请参见 Chapter 4。
- 如果上游组件出于某种原因需要在发送 PM_Request_Ack DLLP 之后在链路上传输 TLP，则它必须首先完成到低功耗状态的转换，然后启动从低功耗状态退出以进行处理，一旦链路返回到 L0，就开始传输。 如果与 L1 的协商中断，请参阅 Section 5.2。
  - 在这种情况下，即使上游组件不具备发送 TLP 所需的流量控制信用，也必须从 L1 退出。
- 当下游组件在其接收通道上检测到 PM_Request_Ack DLLP（表明上游设备已确认向 L1 请求的切换）时，下游组件将停止发送 PM_Active_State_Request_L1 DLLP，禁用 DLLP、TLP 传输并将其发送通道置于电气空闲状态。
- 当上游组件在其接收通道上检测到电气空闲（表明下游组件已进入 L1 状态）时，它将停止发送 PM_Request_Ack DLLP，禁用 DLLP、TLP 传输，并将链路的下游方向引入电气空闲状态。

Notes：

1. 事务层完成超时机制不受切换到 L1 状态的影响（即，它必须继续计数）。
2. 当链路处于 L1 状态时，将冻结流控制更新计时器，以防止计时器到期，从而不必要地将链路转换回 L0 状态。

公众号：芯片业者相承攻

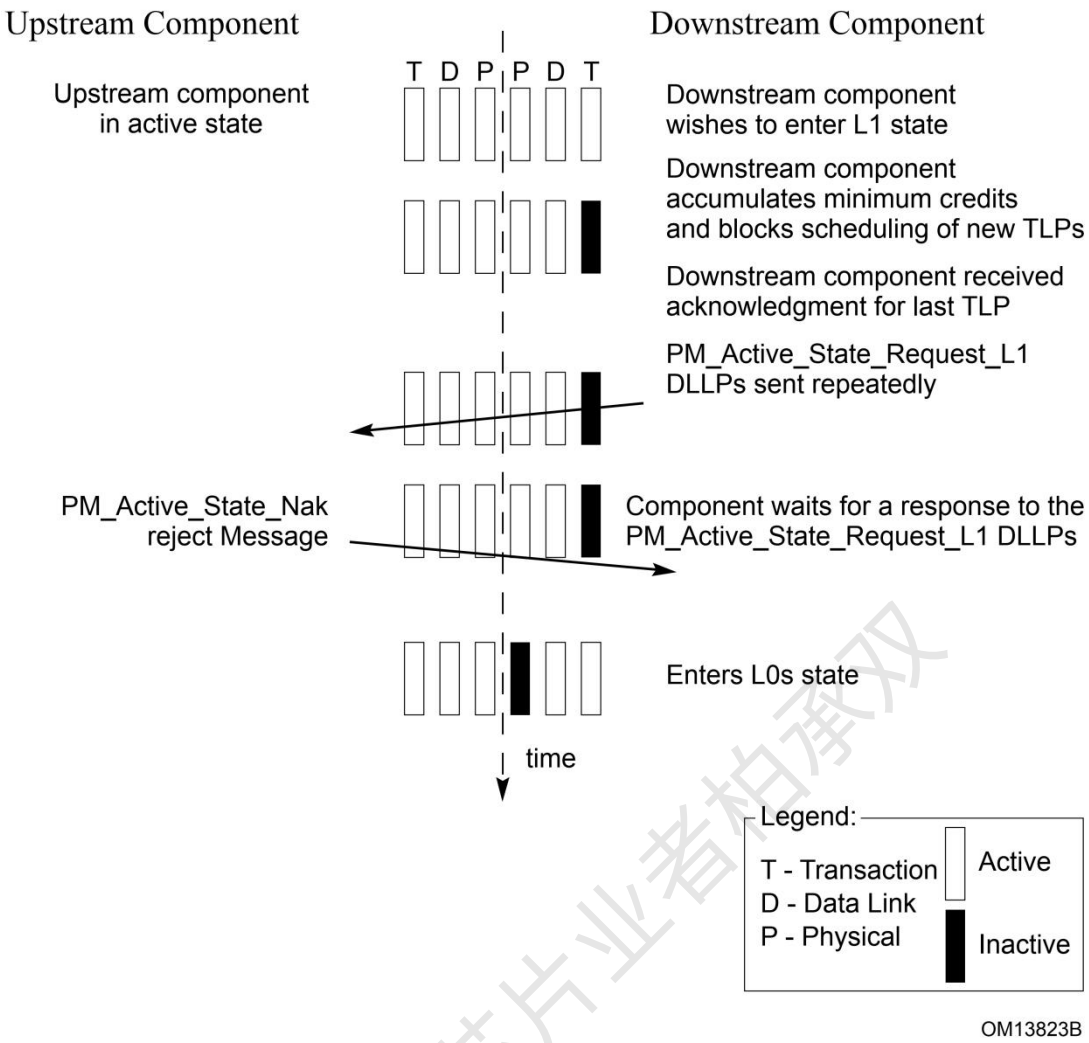

Figure 5-6 L1 Transition Sequence Ending with a Rejection (L0s Enabled)

公众号·芯片业者相承双

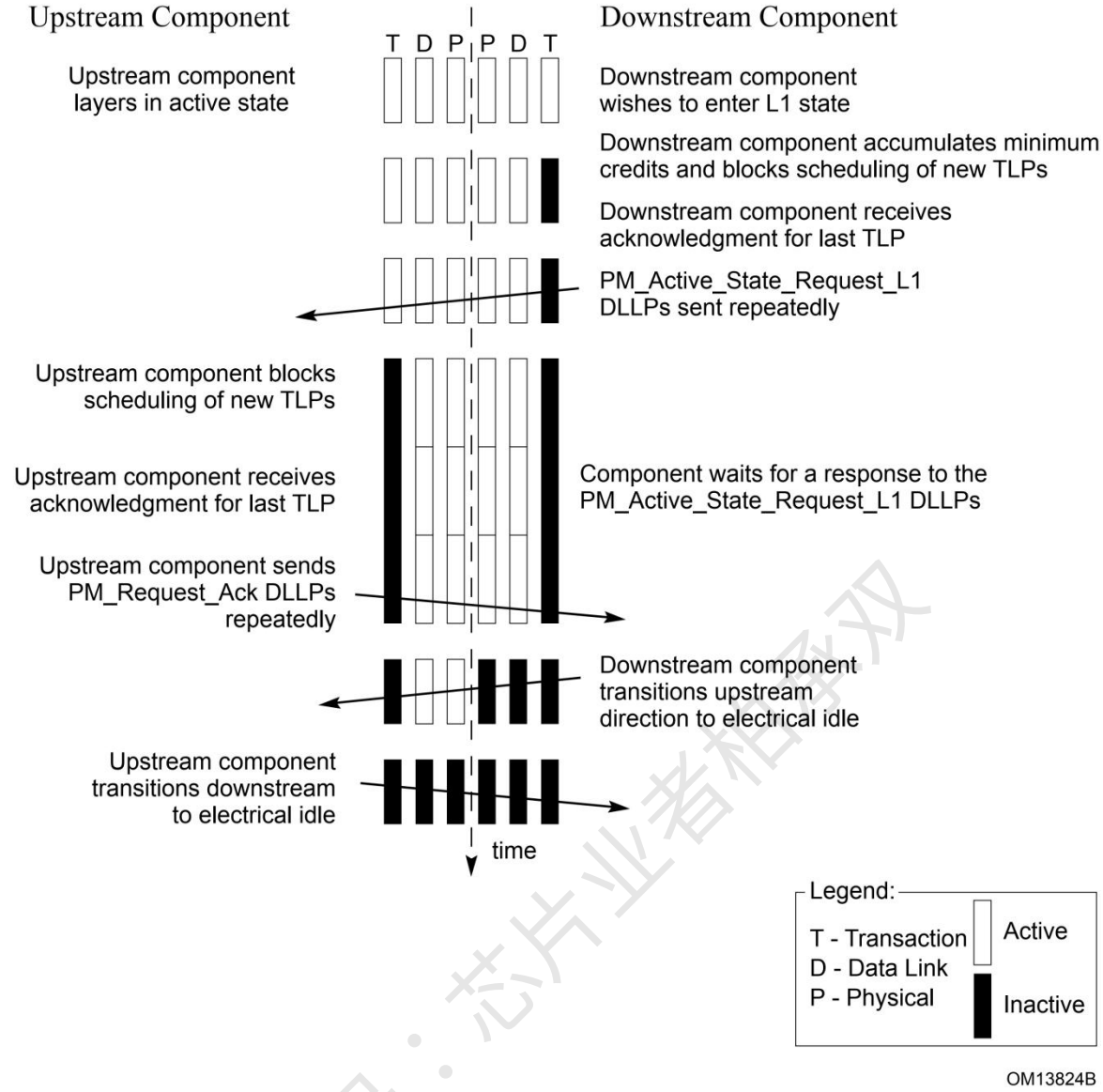

Figure 5-7 L1 Successful Transition Sequence

5.4.1.2.2 Exit from the L1 State

链路两端的组件可能会从 L1 链路状态启动退出。

如果从 L1 退出，建议下游组件从 L1 退出开始 1us 内发送所有使能 VC 和 FC type 的流控制更新 DLLP。

Downstream Component Initiated Exit

如果上游端口需要通过链路进行通信，则必须在其传输通道上从 L1 退出。如 Chapter 4 所述，该组件启动到 L0 状态的转换。上游组件必须通过发起其传输通道的类似转换来响应。

如果上游组件是 Switch 的下游端口（即它不是 RC 根端口），只要它在其任何下游端口链路上检测到 L1 退出活动，则 Switch 必须在其上游端口的传输通道上发起 L1 退出过程(如果上游端口的链路处于 L1 状态)。由于 L1 退出等待时间相对较长，因此 Switch 必须等到其下游端口链路完全退出到 L0 后才能在其上游端口链路上发起 L1 退出过程。等待下游链路完成 L0 转换，将导致一个 Message 在多个 Switch 之间传输，延迟为经历多个 Switch 的延迟的累积。

要求 Switch 在其任何下游端口链路上的 L1 退出开始后不超过 1μs 的时间内，在其上游端口链路上发起 L1 退出过程。有关 L1 退出期间物理层信令的详细信息，请参见 Section 4.2。

公众号·芯片业者相承双

考虑 Figure 5-8 的示例。 每个端口上附加的数字代表相应端口报告的传输通道 L1 退出延迟，以 us 为单位。

Link 1、2 和 3 都处于 L1 状态，并且 Endpoint C 在时间 T 启动到 L0 状态的转换。由于 Switch B 需要 32µs 的时间在其端口上退出 L1，因此 Link 3 将在 T + 32 时间后切换到 L0（最长的时间，端点 C 考虑 T + 8，而 Switch B 考虑 T + 32）。

要求 Switch B 在从 Link 3 上的 L1 状态开始不超过 1µs 的时间内，从其上游端口链路（Link 2）上发起从 L1 状态的转换。因此，Link 2 将在 T + 1 开始转换到 L0 状态。同样，Link 1 将在 T + 2 开始转换到 L0 状态。

按照上述步骤，Link 2 将在时间 T + 33 处完成向 L0 状态的转换（因为 Switch B 需要更长的转换时间，并且它在时间 T + 1 处开始）。Link 1 将在时间 T + 34 处完成其向 L0 状态的转换（因为 RC 需要 32µs 的转换时间并在时间 T + 2 处开始）。

因此，在 Link 1、2 和 3 中，最后完成到 L0 状态转换的链路延迟为 34 us。这是在 Endpoint C 中发起转换的数据包所经历的延迟。

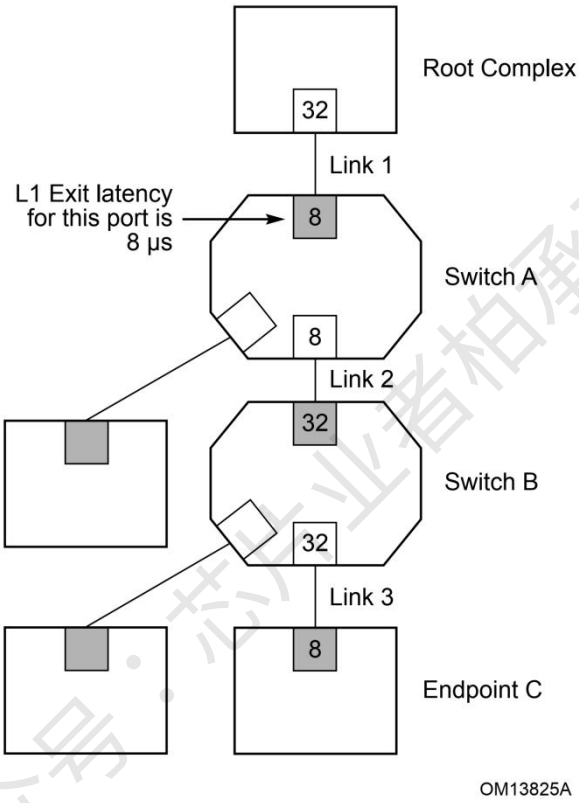

Figure 5-8 Example of L1 Exit Latency Computation

不需要 Switch 在其任何其他下游端口链路上启动 L1 退出过程。

Upstream Component Initiated Exit

RC 或 Switch 必须在其任何根端口或下游端口链路上启动从 L1 的退出（如果它需要通过该链路进行通信）。Switch 或 RC 必须能够启动 L1 退出过程，即使它没有传输给定 TLP 所需的流量控制信用。组件启动到 L0 状态的转换在 Chapter 4 中所述。下游组件必须通过在其传输通道上发起类似的转换来做出响应。

如果下游组件包含一个 Switch，则一旦它在其上游端口链路上检测到从 L1 状态退出，它就必须在其所有下游链路上发起转换（假定下游链路处于 ASPM L1 状态）。由于 L1 出口等待时间相对较长，因此 Switch 必须等到其上游端口链路完全退出到 L0 后才能在其下游端口链路上发起 L1 退出过程。如果真是这样，则通过多个 Switch 传播的消息在遍历每个 Switch 时会经历累积的延迟。

要求 Switch 在从其上游端口从 L1 状态开始切换不超过 1µs 的时间内，启动当前位于 L1 的所有下游端口链路从 L1 状态开始的切换。有关 L1 退出期间物理层信令的详细信息，请参见 Section 4.2。已经处于 L0 状态的下游端口链路不参与退出转换过程。下游端口处于低功耗 D-state（D1-D3hot）的下游端口链路也不受 L1 退出转换的影响（即，此类链路不得转换为 L0 状态）。

公众号：芯片业者相承双

5.4.1.3 ASPM Configuration

所有 Function 必须实现以下配置字段以支持 ASPM。有关配置寄存器分配和访问机制，请参见 Chapter 7。

每个组件在下面的 ASPM Support 字段中报告其对 ASPM 的支持级别。

Table 5-3 Encoding of the ASPM Support Field

Field	Description
ASPM Support	**00b** No ASPM support **01b** L0s supported **10b** L1 supported **11b** L0s and L1 supported

除非链路两侧的组件均支持 L0s，否则软件不得在给定链路的任一方向上启用 L0s。否则，结果不确定。

每个组件用其 Link Status register 中的 Slot Clock Configuration 字段报告其参考时钟的来源。

Table 5-4 Description of the Slot Clock Configuration Bit

Bit	Description
Slot Clock Configuration	This bit, when Set, indicates that the component uses the same physical reference clock that the platform provides on the connector. This bit, when Clear, indicates the component uses an independent clock irrespective of the presence of a reference on the connector. For Root and Switch Downstream Ports, this bit, when Set, indicates that the Downstream Port is using the same reference clock as the Downstream component or the slot. For Switch and Bridge Upstream Ports, this bit when Set, indicates that the Upstream Port is using the same reference clock that the platform provides. Otherwise it is Clear.

每个组件必须实现 Link Status register 中的 Common Clock Configuration 字段。软件写入该字段，以指示该设备是否与链路另一端的设备共享相同的时钟源。

Table 5-5: Description of the Common Clock Configuration Bit

Bit	Description
Common Clock Configuration	This bit when set indicates that this component and the component at the opposite end of the Link are operating with a common clock source. A value of 0b indicates that this component and the component at the opposite end of the Link are operating with separate reference clock sources. Default value of this bit is 0b. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies.

每个端口分别在 L0s Exit Latency 和 L1 Exit Latency 字段中报告 L0s 和 L1 退出等待时间（指它们的接收通道从 L0s 或 L1 状态转换到 L0 状态所需的时间）。如果端口不支持 L0s 或 ASPM L1，则相应的出口等待时间字段的值不确定。

公众号·芯片业者拍案叹

Table 5-6: Encoding of the L0s Exit Latency Field

Field	Description
L0s Exit Latency	000b – Less than 64 ns 001b – 64 ns to less than 128 ns 010b – 128 ns to less than 256 ns 011b – 256 ns to less than 512 ns 100b – 512 ns to less than 1 µs 101b – 1 µs to less than 2 µs 110b – 2 µs to 4 µs 111b – More than 4 µs

Table 5-7: Encoding of the L1 Exit Latency Field

Field	Description
L1 Exit Latency	000b – Less than 1 µs 001b – 1 µs to less than 2 µs 010b – 2 µs to less than 4 µs 011b – 4 µs to less than 8 µs 100b – 8 µs to less than 16 µs 101b – 16 µs to less than 32 µs 110b – 32 µs to 64 µs 111b – More than 64 µs

Endpoint 还报告由于从 L0s 状态或 L1 状态到 L0 状态的转换，它们可以吸收的其他延迟。这分别在 Endpoint L0s Acceptable Latency 和 Endpoint L1 Acceptable Latency 字段中报告。

电源管理软件使用分层结构中所有组件报告的延迟信息，可以通过比较从根到端点的每个给定路径的出口延迟与每个相应端点可以承受的可接受延迟来启用适当级别的 ASPM。

Table 5-8: Encoding of the Endpoint L0s Acceptable Latency Field

Field	Description
Endpoint L0s Acceptable Latency	000b – Maximum of 64 ns 001b – Maximum of 128 ns 010b – Maximum of 256 ns 011b – Maximum of 512 ns 100b – Maximum of 1 µs 101b – Maximum of 2 µs 110b – Maximum of 4 µs 111b – No limit

Table 5-9: Encoding of the Endpoint L1 Acceptable Latency Field

Field	Description
Endpoint L1 Acceptable Latency	000b – Maximum of 1 µs 001b – Maximum of 2 µs 010b – Maximum of 4 µs 011b – Maximum of 8 µs 100b – Maximum of 16 µs 101b – Maximum of 32 µs 110b – Maximum of 64 µs 111b – No limit

公众号：芯片业者柏承双

电源管理软件通过对 ASPM Control 字段进行编程来启用或禁用每个组件中的 ASPM。

Table 5-10: Encoding of the ASPM Control Field

Field	Description
ASPM Control	00b – Disabled 01b – L0s Entry Enabled 10b – L1 Entry Enabled 11b – L0s and L1 Entry enabled

ASPM Control = 00b

Port 的发送器不会进入 L0s。

下游链路终点上挂接的 Port 不能再其上游链路上发送 PM_Active_State_Request_L1 DLLP 。

上游链路终点上挂接的 Port 在收到 L1 请求时必须拒绝。

ASPM Control = 01b

如果满足所有条件，则端口必须使链路进入 L0s 状态。

下游链路终点上挂接的 Port 不得在其上游链路上发送 PM_Active_State_Request_L1 DLLP。

上游链路终点上挂接的 Port 在收到 L1 请求时必须拒绝。

ASPM Control = 10b

Port 的发送器不会进入 L0s。

下游链路终点上挂接的 Port 可以在其上游链路上发送 PM_Active_State_Request_L1 DLLP。

如果满足 Section 5.4.1.2.1 中的 RC 根端口或 Switch 下游端口的条件，则连接到链路上游端的端口必须以肯定的确认响应 L1 请求并转换为 L1。

ASPM Control = 11b

如果满足所有条件，则端口必须使链路进入 L0s 状态。

下游链路终点上挂接的 Port 可以在其上游链路上发送 PM_Active_State_Request_L1 DLLP。

如果满足 Section 5.4.1.2.1 中的 RC 根端口或 Switch 下游端口的条件，则连接到链路上游端的端口必须以肯定的确认响应 L1 请求并转换为 L1。

5.4.1.3.1 Software Flow for Enabling or Disabling ASPM

以下是示例软件算法，重点介绍了如何在组件中启用或禁用 ASPM。

- PCI Express 组件根据其 Slot Clock Configuration 字段中的值启动。初始化该字段的方法是特定于设备的。
- PCI Express 系统软件扫描每个链路两端组件中的 Slot Clock Configuration 字段，以确定两者是否使用相同的参考时钟源或来自不同源的参考时钟。如果两个设备中的 Slot Clock Configuration 字段都置 1，则它们都使用相同的参考时钟源，否则使用不同的参考时钟源。
- PCI Express 软件会更新每个链路两端组件中的 Common Clock Configuration 字段，以指示这些设备是否共享相同的参考时钟，

公众号：芯片业者相承双

并通过向上游组件的 Link Control register 中的 Retrain Link 字段写入 1b 来触发链路重新训练。

- 根据 Common Clock Configuration 字段的设置，设备必须在其 L0s/L1 Exit Latency 字段中反映适当的 L0s/L1 退出延迟。
- 然后，PCI Express 系统软件根据每个端口报告的延迟读取并计算每个端点的 L0/L1 退出延迟。有关示例，请参见 Section 5.4.1.2.2 。
- 对于具有一个或多个 Endpoint Function 的组件，PCI Express 系统软件检查每个 Endpoint Function 在其 Link Capabilities register 中报告的 Endpoint L0s/L1 Acceptable Latency，并根据分层结构中的一些或全部中间端口启用或禁用 L0s/L1 进入（通过 Link Control register 的 ASPM Control 字段）。

5.5 L1 PM Substates

L1 PM 子状态建立这样一种链路电源管理机制：它创建 L1 链路状态下的低功耗子状态（见 Figure 5-9），以及使用这些子状态的相关机制。L1 PM 子状态包括：

- L1.0 子状态
  - L1.0 子状态对应于传统的 L1 链路状态。每当链路进入 L1 时，就会进入该子状态。L1 PM 子状态机制定义了从该子状态到 L1.1 和 L1.2 子状态的转换。
  - Upstream 和 Downstream Port 必须开启对 Section 4.2.6.7.2 中要求的电气空闲退出的检测功能。
- L1.1 子状态
  - 保持链路共模电压。
  - 使用双向开漏时钟请求 (CLKREQ#) 信号进入和退出此状态。
  - Upstream 和 Downstream Port 不用开启对 Section 4.2.6.7.2 中要求的电气空闲退出的检测功能。
- L1.2 子状态
  - 不需要保持链路共模电压。
  - 使用双向开漏时钟请求 (CLKREQ#) 信号进入和退出此状态。
  - Upstream 和 Downstream Port 不用开启对 Section 4.2.6.7.2 中要求的电气空闲退出的检测功能。

L1.1 和 L1.2 子状态不得有参考时钟。

在 SRIS 模式下运行时，需要支持 L1 PM 子状态和 SRIS 模式的端口才能支持 L1 PM 子状态。在这种情况下，本节中定义的 L1 PM 子状态协议使用 CLKREQ# 信号，但与链路上任一端口使用的任何本地时钟没有关系，并且此类本地时钟的管理是特定于实现的。

在 ASPM L1 状态支持 L1.2 子状态的端口必须支持延迟容忍报告 (LTR)。

公众号·芯片业者相承双

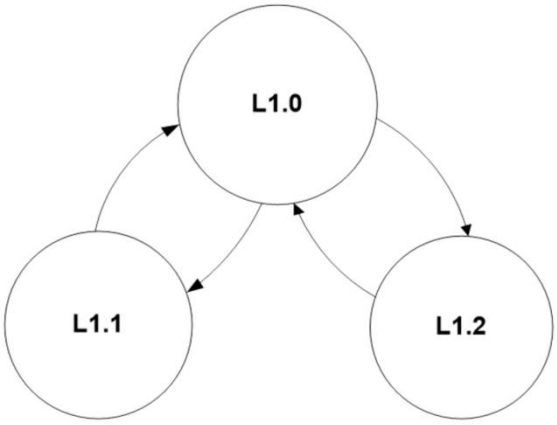

Figure 5-9 State Diagram for L1 PM Substates

- 当 L1 PM 子状态机制被使能，则以下附加要求会应用于 CLKREQ# 信号： CLKREQ# 信号必须作为双向开漏信号被链路的上游和下游端口支持。每个端口必须有一个唯一的信号实例，并且必须连接上游和下游端口的 CLKREQ# 信号。
- 当链路处于 PCI-PM L1 或 ASPM L1 状态时，或者当链路处于 L2/L3 Ready 伪状态时，允许上游端口把 CLKREQ#置为无效；当链路处于任何其他状态时，必须由上游端口把 CLKREQ#置为有效。
- 所有其他与 CLKREQ# 信号相关的、未由 L1 PM 子状态明确定义或修改的规范继续适用。

如果在特定系统中无法满足这些要求，则不得启用 L1 PM 子状态。

IMPLEMENTATION NOTE

CLKREQ# Connection Topologies

对于上游组件，CLKREQ# 信号的连接拓扑可能会有所不同。下面描述了 CLKREQ# 连接拓扑的几个示例。对于下游组件，这些情况本质上是相同的，但是从上游组件的角度来看，存在一些如下面所描述的关键差异。

示例 1：带有一个 PLL 的下游端口连接到一个上游端口（见 Figure 5-10）。

在此平台配置中，上游组件 (A) 只有一个 CLKREQ# 信号。上游和下游端口的 CLKREQ#（A 和 B）信号相互连接。在这种情况下，上游组件 (A) 只要需要参考时钟就必须把 CLKREQ# 信号置为 1。

公众号·芯片业者相承双

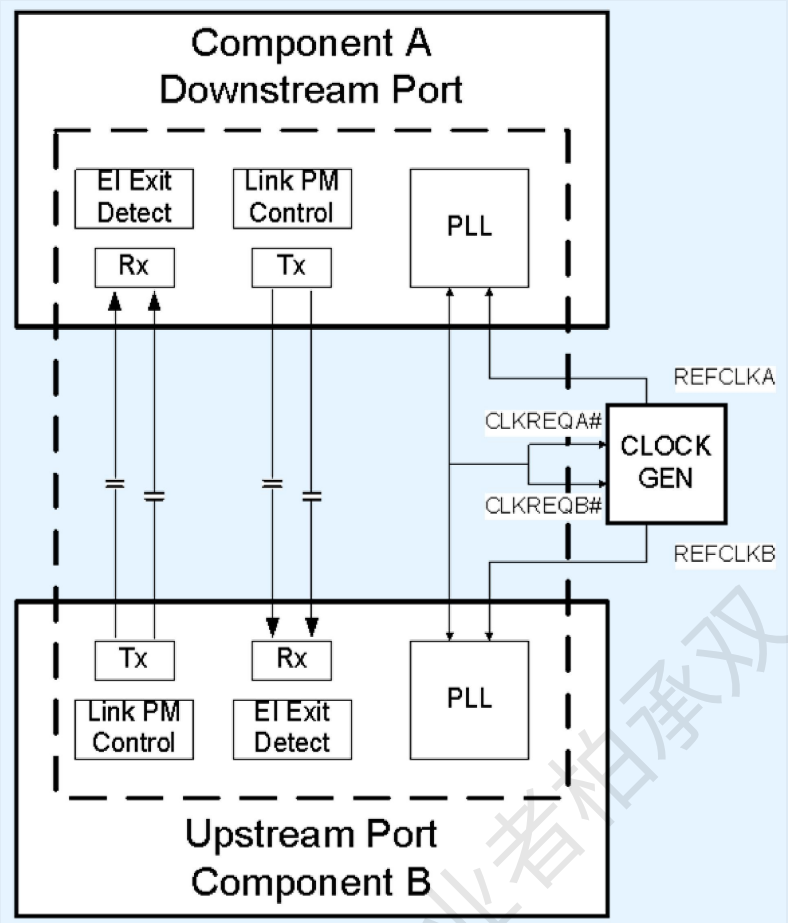

Figure 5-10 Downstream Port with a Single PLL

示例 2：具有多个下游端口的上游组件，具有公共的共享的 PLL，连接到单独的下游组件（见 Figure 5-11）。

在此示例配置中，上游组件 (A) 有 3 个 CLKREQ# 信号实例，每个下游端口一个，上游组件 (A) 有一个公共的共享的 CLKREQ# 信号。在此拓扑中，下游端口 CLKREQ#（CLKREQB#、CLKREQC#）信号用于连接到下游组件（B 和 C）的上游端口的 CLKREQ# 信号。上游组件的公共共享 CLKREQ# (CLKREQA#) 信号用于请求共享 PLL 的参考时钟。当两个下游端口都处于 L1.1 或 L1.2 子状态时，只能关闭上游组件 (A) 中的 PLL 控制逻辑并把 CLKREQA#置为无效，并且 PLL 的所有内部 (A) 消费者不需要一个时钟。

公众号：芯片业者相承双

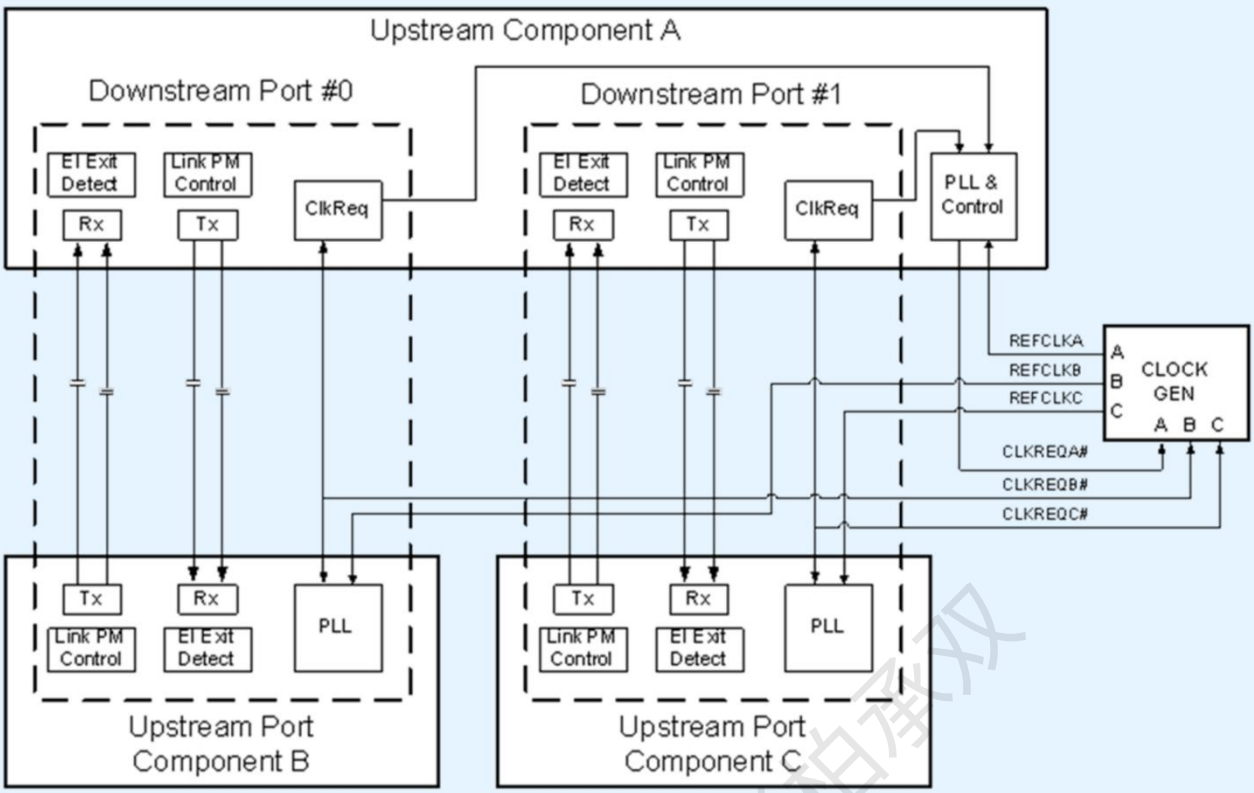

Figure 5-11 Multiple Downstream Ports with a shared PLL

IMPLEMENTATION NOTE

Avoiding Unintended Interactions Between L1 PM Substates and the LTSSM

通常情况下，节省功耗的实现技术也会增加恢复正常操作的延迟。在实现 L1 PM 子状态时，实现者必须确保任何增加的延迟不会与平台的其他元件产生负面影响。确保 LTSSM 超时条件不会被无意触发尤为重要。尽管典型的实现不会接近导致此类交互的延迟，但实现者有责任确保实现正确的整体操作。

5.5.1 Entry conditions for L1 PM Substates and L1.0 Requirements

当 L1 PM 子状态为 L1.0 且 LTSSM 通过 PCI-PM 兼容电源管理方式进入 L1 时，认为链路处于 PCI-PM L1.0。当 L1 PM 子状态处于 L1.0 且 LTSSM 通过 ASPM 进入 L1 时，链路被认为处于 ASPM L1.0。

以下规则定义了 L1.1 和 L1.2 子状态的进入方式：

- Upstream 和 Downstream Port 都必须监视 CLKREQ# 信号的逻辑状态。
- 当处于 PCI-PM L1.0 且 PCI-PM L1.2 Enable 字段被设置为 1 时，必须在 CLKREQ# 置为无效时进入 L1.2 子状态。
- 当处于 PCI-PM L1.0 且 PCI-PM L1.1 Enable 字段被设置为 1 时，必须在 CLKREQ# 置为无效且 PCI-PM L1.2 Enable 字段清 0 时进入 L1.1 子状态。
- 当处于 ASPM L1.0 且 ASPM L1.2 Enable 字段设置为 1 时，必须在 CLKREQ# 置为无效且以下所有条件为真时进入 L1.2 子状态：
  - 此端口上次发送或接收的报告的侦听 LTR 值大于或等于 LTR_L1.2_THRESHOLD Value 和 Scale 字段设置的值，或者没有

侦听服务延迟要求。

- 此端口最后发送或接收的报告的非监听 LTR 值大于或等于 LTR_L1.2_THRESHOLD Value 和 Scale 字段设置的值，或者没有非监听服务延迟要求。

- 当处于 ASPM L1.0 且 ASPM L1.1 Enable 字段设置为 1 时，当 CLKREQ# 置为无效且不满足进入 L1.2 子状态的条件时，必须进入 L1.1 子状态。

当满足 L1.2 的进入条件时，以下规则适用：

- 上游端口和下游端口都必须监测 CLKREQ# 输入信号的逻辑状态。
- 在链路进入 L1.0 之前，上游端口不得把 CLKREQ#置为无效。
- 允许任一端口把 CLKREQ# 置为有效以阻止链路进入 L1.2。
- 打算阻止进入 L1.2 的下游端口必须在链路进入 L1 之前把 CLKREQ#置为有效。
- 当 CLKREQ# 无效时，端口进入 L1.2 的 L1.2.Entry 子状态。

如果 Downstream Port 处于 PCI-PM L1.0 状态并且 PCI-PM L1.1 Enable 和/或 PCI-PM L1.2 Enable 被置为 1，或者如果 Downstream Port 处于 ASPM L1.0 状态且 ASPM L1.1 Enable 和/或 ASPM L1.2 Enable 被置为有效，并且 Downstream Port 在没有进入 L1.1 或 L1.2 的情况下启动退出到 Recoverry，则 Downstream Port 必须把 CLKREQ#置为有效，直到链路退出 Recoverry。

5.5.2 L1.1 Requirements

Upstream 和 Downstream Port 在实现时都允许关闭电气空闲 (EI) 退出检测和 Refclk 活动检测的机制，但是两个端口都必须保持共模。

5.5.2.1 Exit from L1.1

如果 Upstream 和 Downstream Port 需要启动从 L1.1 退出，它必须把 CLKREQ#置为有效，直到链路退出 Recovery 状态。Upstream Port 必须在进入 Recovery 状态时把 CLKREQ#置为有效，并且必须保持 CLKREQ#为有效，直到下一次进入 L1，或者到把 CLKREQ#置为无效的其他状态。

- 如果 CLKREQ# 被置 1，下一个状态是 L1.0。
  - Refclk 最终将按照 PCI Express Mini CEM 规范中的定义打开，这可能会根据上游端口通告的 LTR 进行延迟。

Figure 5-12 说明了 L1.1 的进入和由上游端口驱动的退出过程。

公众号·芯片业者相聚汉

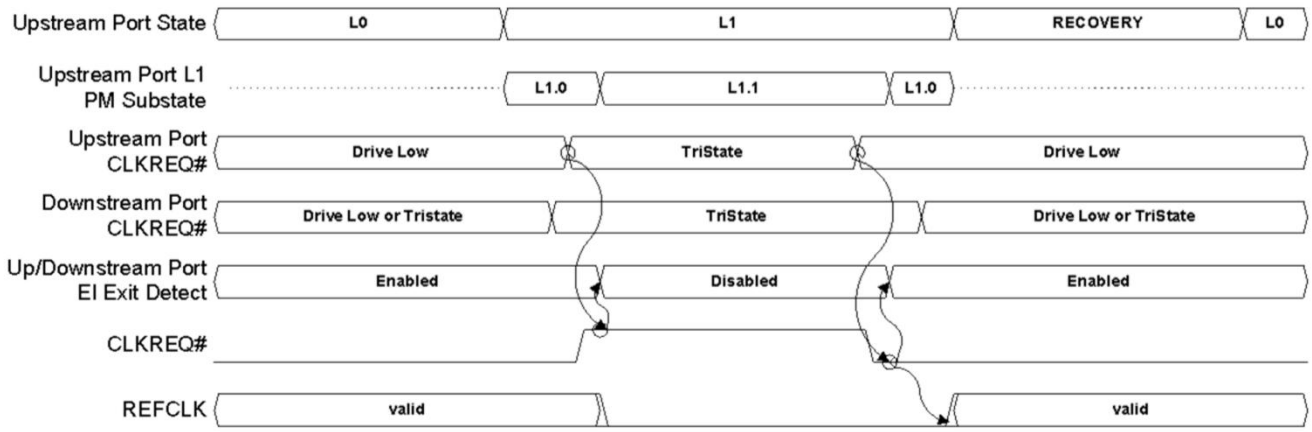

Figure 5-12 Example: L1.1 Waveforms Illustrating Upstream Port Initiated Exit

Figure 5-12 说明了 L1.1 的进入和由下游端口驱动的退出过程。

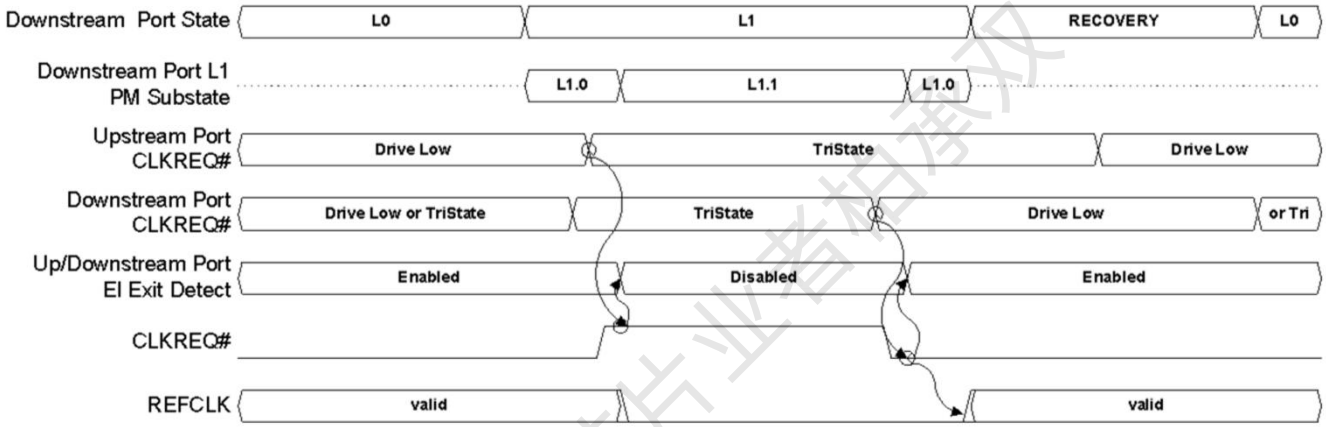

Figure 5-13 Example: L1.1 Waveforms Illustrating Downstream Port Initiated Exit

5.5.3 L1.2 Requirements

在 L1.2 期间必须保持所有链路和 PHY 状态，或者必须在退出时使用特定于实现的方式恢复，并且从 L1.2 退出时的 LTSSM 和相应的端口状态必须与 L1.0 LTSSM 和端口状态无法区分 。

L1.2 具有不适用于 L1.1 的附加要求，这些要求在本节中描述。

L1.2 具有三个子状态，定义如下（见 Figure 5-14）。

公众号·芯片业者招募及

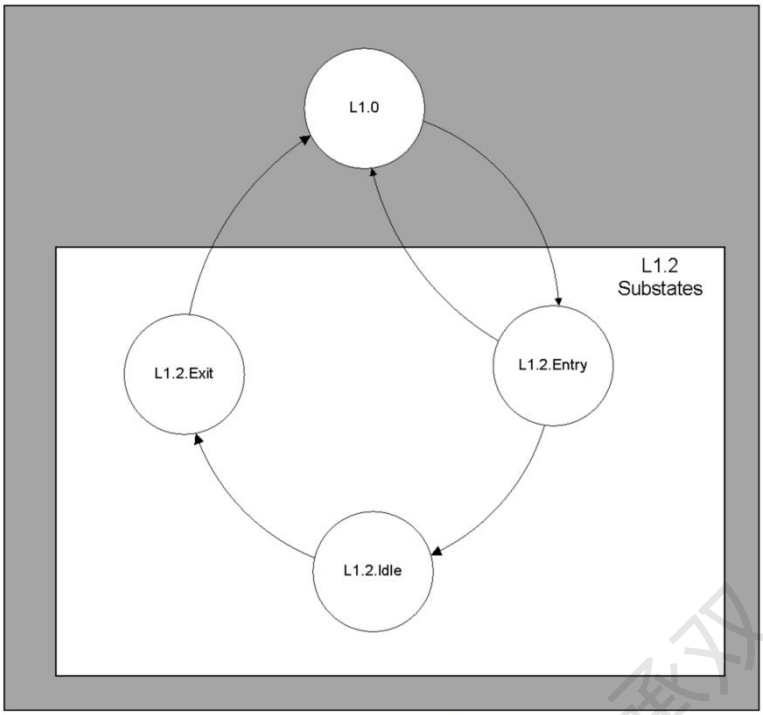

Figure 5-14 L1.2 Substates

5.5.3.1 L1.2.Entry

L1.2.Entry 是进入 L1.2 时的过渡状态，以允许 Refclk 有足够时间关闭，并确保两个端口都观察到 CLKREQ# 被置为无效。以下规则适用于 L1.2.Entry：

- Upstream 和 Downstream Port 都继续保持共模。
- Upstream 和 Downstream Port 都可以关闭其电气空闲 (EI) 出口检测电路。
- 在此状态下，Upstream 和 Downstream Port 不得把 CLKREQ#置为有效。
- Refclk 必须在 TL1O_REFCLK_OFF 内关闭。
- 如果 CLKREQ# 被置为有效，则下一状态为 L1.0，否则等待 TPOWER_OFF 后下一状态为 L1.2.Idle。

请注意，当一个端口在另一个端口将 CLKREQ# 置为无效后不久但在第一个端口观察到 CLKREQ# 置为无效之前将 CLKREQ# 置为有效时，可能会发生边界条件。这是一个不可避免的边界条件，实现必须要正确处理这种情况。这种情况的一个例子如 Figure 5-15 所示。

公众号·芯片业者相承双

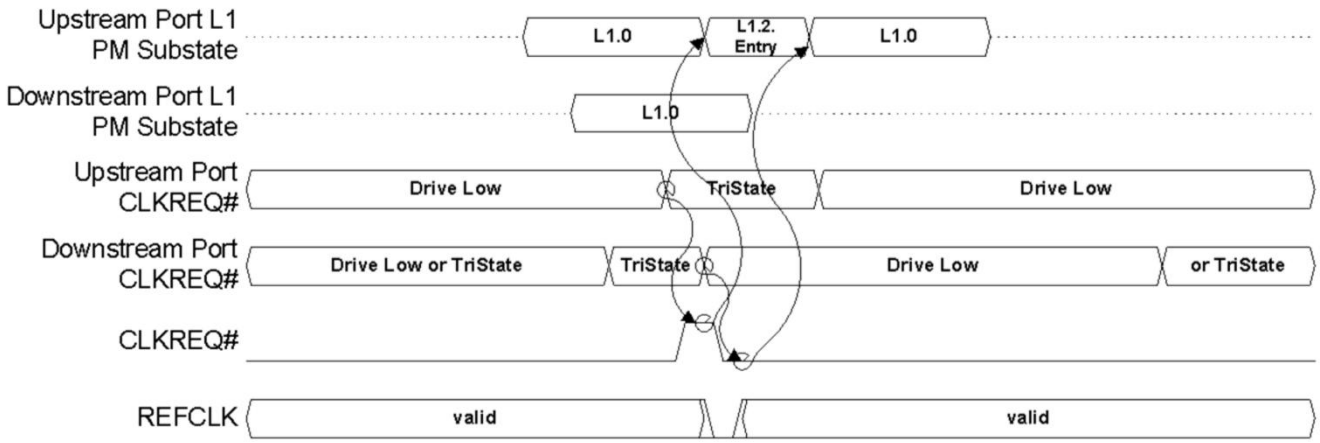

Figure 5-15 Example: Illustration of Boundary Condition due to Different Sampling of CLKREQ#

5.5.3.2 L1.2.Idle

当满足进入 L1.2.Idle 状态（见 Section 5.5.1 ）的要求时，端口进入 L1.2.Idle 子状态。以下规则适用于 L1.2.Idle：

- Upstream 和 Downstream Port 都可以关闭任何活动逻辑，包括维持共模所需的电路。
- Upstream 和 Downstream Port 的 PHY 都可能断电。

使用基于 CLKREQ# 的机制时，以下规则适用于 L1.2.Idle 状态：

- 如果 Upstream 和 Downstream Port 需要退出 L1.2，则必须在确保满足 TL1.2 后把 CLKREQ#置为有效。
- 如果 Downstream Port 启动从 L1 退出，它必须把 CLKREQ#置为有效，直到链路退出 Recovery 状态。Upstream Port 必须在进入 Recovery 状态时把 CLKREQ#置为有效，并且必须继续保持 CLKREQ#有效，直到下一次进入 L1，或者到把 CLKREQ#置为无效的其他状态。
- 如果 Upstream Port 启动从 L1 退出，它必须把 CLKREQ#置为有效，直到下一次进入 L1，或者到把 CLKREQ#置为无效的其他状态。
- Upstream 和 Downstream Port 都必须监视 CLKREQ# 输入信号的逻辑状态。
- 如果 CLKREQ# 被置为有效，下一个状态是 L1.2.Exit。

5.5.3.3 L1.2.Exit

这是从 L1.2 退出时的过渡状态，以允许两个设备有足够时间给电。在 L1.2.Exit 中，以下规则适用：

- 必须为 Upstream 和 Downstream Port 的 PHY 供电。
- 不能假定共模是保持的。

5.5.3.3.1 Exit from L1.2

以下规则适用于使用基于 CLKREQ# 的机制的 L1.2.Exit：

- Upstream 和 Downstream Port 都必须为 L1.0 所需的任何电路加电，包括维持共模所需的电路。
- 在此状态下，Upstream 和 Downstream Port 不得更改其 CLKREQ# 的驱动状态。

公众号·芯片业者相承双

- Refclk 的开启时间不得早于 TL10_REFCLK_ON 最短时间，并且可能需要根据端点通告的 LTR 所允许的时间量才能生效。
- 等待 TPOWER_ON 后，下一个状态是 L1.0。
  - 共模允许在 L1.0 期间被动建立，而在 Recovery 期间主动建立。为了确保共模已经建立， Downstream Port 必须保持一个定时器，并且 Downstream Port 必须继续发送 TS1 训练序列，直到 Downstream Port 开始传输 TS1 训练序列并检测到在任何已配置好的链路的 Lane 上的电气空闲退出。

Figure 5-16 说明了 L1.2 进入和 Upstream Port 发起退出相关的信号关系和时序约束。

Figure 5-17 说明了与 L1.2 进入和 Downstream Port 发起的退出相关的信号关系和时序约束。

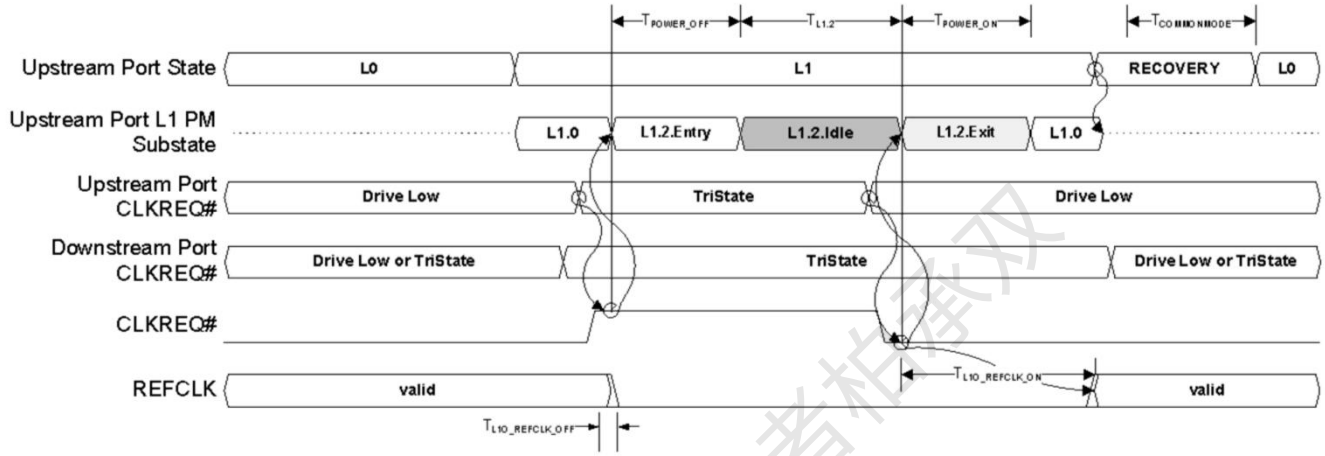

Figure 5-16 Example: L1.2 Waveforms Illustrating Upstream Port Initiated Exit

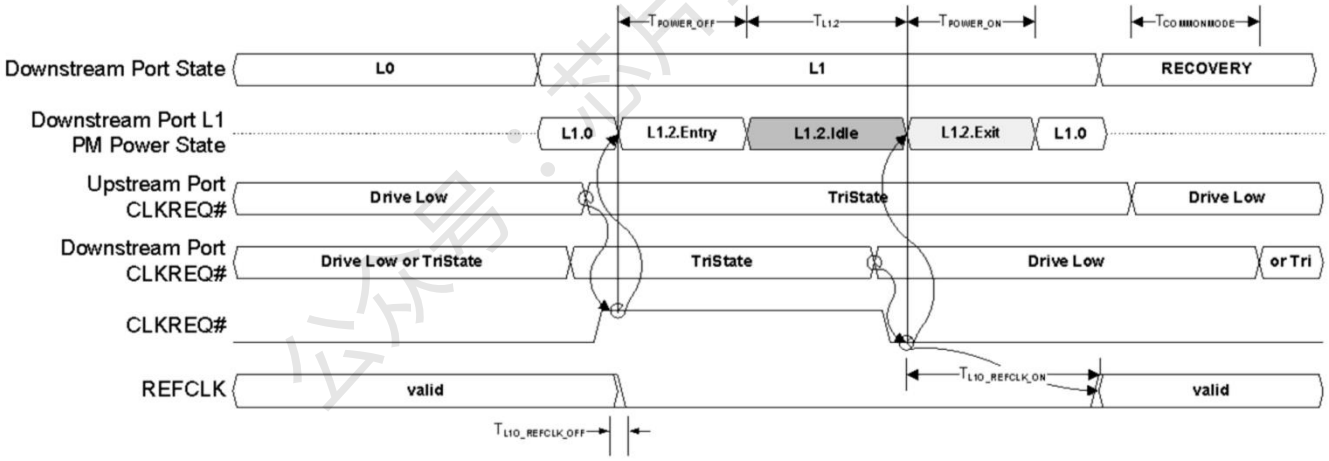

Figure 5-17 Example: L1.2 Waveforms Illustrating Downstream Port Initiated Exit

5.5.4 L1 PM Substates Configuration

当与该端口关联的 ASPM L1.1 Enable、ASPM L1.2 Enable、PCI-PM L1.1 Enable 和 PCI-PM L1.2 Enable 字段的任意组合被设置为 1 时，L1 PM 子状态被视为在端口上启用。

当该链路上的 Upstream 和 Downstream Port 都设置了相应的支持能力字段时，L1 PM Substate Enable 字段必须仅在该链路上的 Upstream 和 Downstream Port 中设置为 1，否则行为未定义。

任何使能位的设置都必须在 Downstream Port 进行，然后相应的字段才被允许在 Upstream Port 中设置。 如果稍后要清除任何 L1

PM 子状态使能位，则必须先清除 Upstream Port 中的使能位，然后才允许在 Downstream Port 中清除相应的使能位。

如果为 ASPM L1 PM 子状态设置一个或两个启用位，则在禁用 ASPM L1 时，两个端口都必须按本节所述进行配置。

如果为 PCI-PM L1 PM 子状态设置一个或两个启用位，则在 D0 中时，两个端口都必须按本节所述进行配置。

在为 L1.2 设置任一或两个启用位之前，必须要先编程 TPOWER_ON、Common_Mode_Restore_Time 的值，如果 ASPM L1.2 Enable 字段设置为 1，则还必须编程 LTR_L1.2_THRESHOLD Value 和 LTR_L1.2_THRESHOLD Scale 字段的值。

T_{POWER_ON} 和 Common_Mode_Restore_Time 字段必须根据组件和连接两个组件的 AC 耦合电容器编程为适当的值。这些值的确定是特定于设计实现的。

当 ASPM L1.2 Enable 和 PCI-PM L1.2 Enable 都被清 0 时，不需要对 TPOWER_ON、Common_Mode_Restore_Time 和 LTR_L1.2_THRESHOLD Value 和 LTR_L1.2_THRESHOLD Scale 字段进行编程，硬件不得依赖这些字段来获得任何特定的值。

编程 LTR_L1.2_THRESHOLD Value 和 LTR_L1.2_THRESHOLD Scale 字段时，必须在两个端口中编程相同的值。

5.5.5 L1 PM Substates Timing Parameters

Table 5-11 定义了与 L1.2 子状态机制相关的时序参数。

Table 5-11 L1.2 Timing Parameters

Parameter	Description	Min	Max	Units
T_{POWER_OFF}	CLKREQ# deassertion to entry into the L1.2.Idle substate		2	µs
$T_{COMMONMODE}$	Restoration of Refclk to restoration of common mode established through active transmission of TS1 training sequences (see Section 5.5.3.3.1)	Programmable in range from 0 to 255		µs
$T_{L1O\_REFCLK\_OFF}$	CLKREQ# deassertion to Refclk reaching idle electrical state when entering L1.2	0	100	ns
$T_{L10\_REFCLK\_ON}$	CLKREQ# assertion to Refclk valid when exiting L1.2	T_{POWER_ON}	LTR value advertised by the Endpoint	µs
T_{POWER_ON}	The minimum amount of time that each component must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface to ensure no device is ever actively driving into an unpowered component.	Set in the L1 PM Substates Control 2 Register (range from 0 to 3100)		µs
$T_{L1.2}$	Time a Port must stay in L1.2 when CLKREQ# must remain inactive	4		µs

5.5.6 Link Activation

Link Activation 是一种可选实现的机制，用于暂时禁用 L1 子状态。Link Activation 用于将链路转移出 L1.1/L1.2，避免潜在的停顿。此类停顿的一个示例是与执行 D3Hot 到 D0 转换的配置写入相关联的停顿。Link Activation 也可用于间接指示设备在延迟敏感或时间关键操作期间应避免长延迟内部电源管理。

以下规则适用于 Link Activation：

- Downstream Port 允许支持 Link Activation，这由 L1 PM Substates Capabilities Register 的 Link Activation Supported 字段指示。
- 除非设置了以下一位或多位，否则 Link Activation Control 字段不得对端口行为产生影响：

- PCI-PM L1.2 Enable
- PCI-PM L1.1 Enable

- 当 Link Activation Control 字段被设置为 1 时，即将进入 L1 的端口必须把 CLKREQ# 信号置为有效，并且在 L1 中要保持有效。
- 如果 Link Activation Control 字段为 0，则链路激活机制不会对 CLKREQ# 信号的状态强加任何额外要求。
- 如果端口启用了使用 MSI 或 MSI-X 的边沿触发中断信令，则每次以下条件的逻辑与从 FALSE 转换为 TRUE 时，都必须发送一条中断消息：
  - 关联向量未屏蔽（如果 MSI 不支持 PVM，则不适用）。
  - Link Activation Interrupt Enable 字段设置为 1。
  - Link Activation Control 字段设置为 1。
  - Link Activation Status 字段设置为 1。请注意，链路激活中断始终使用 PCI Express Capabilities Register 中的 Interrupt Message Number 字段指示的 MSI 或 MSI-X 向量。
- 如果使用 INTx 消息为电平触发中断信号启用端口，则只要满足以下条件，虚拟 INTx 线就必须置为有效：
  - Command Register 中的 Interrupt Disable 为 0。
  - Link Activation Interrupt Enable 字段设置为 1。
  - Link Activation Control 字段设置为 1。
  - Link Activation Status 字段设置为 1。
- 每次以下条件的逻辑 AND 从 FALSE 转换为 TRUE 时，必须把 Link Activation Control 字段设置为 1：
  - PCI-PM L1.2 Enable 或 PCI-PM L1.1 Enable（或两者）被设置为 1。
  - Link Activation Control 字段设置为 1。
  - 链路不在 L1 子状态。

5.6 Auxiliary Power Support

与辅助电源相关的特定定义和要求是特定于外形规格的，术语“auxiliary power”和“Vaux”应参考使用中的特定外形规格来理解。本规范中未定义用于提供辅助电源的具体机制。以下文本定义了适用于所有外形规格的要求。

PCI Express PM 在 Device Control register 中提供了一个新的 Aux Power PM Enable 字段，该字段提供了一种使 Function 能够汲取辅助电流的最大允许量的方式，而与它对 PME 生成的支持程度无关。

Function 通过在 PMC 寄存器的 Aux_Current 字段中指定非零值来请求 Aux power 分配。有关 Aux Power PM Enable register 的机制，请参阅 Chapter 7。

使用 Aux Power PM Enable 的辅助电源分配确定如下：

如果 Aux Power PM Enable = 1b ：

辅助电源根据 PMC 寄存器的 Aux_Current 字段中的请求分配，与 PMSCR 中的 PME_En 字段无关。PME_En 字段仍然控制着掌握 PME 的能力。

公众号：芯片业者柏承双

如果 Aux Power PM Enable = 0b ：

辅助电源的分配由 PME_En 字段控制，规则参阅 *PCI Bus Power Management Interface Specification* 规范。

Aux Power PM Enable 字段具有 sticky 性质（请参见 Section 7.4），因此其值可以在 $D3_{cold}$ 状态保留，并且不受从 $D3_{cold}$ 状态到 $D0_{Uninitialized}$ 状态转换的影响。

通常，需要辅助电源来支持 Beacon 唤醒机制，并且除非软件启用，否则支持 Beacon 的组件不得消耗辅助电源。为了实现更精细的电源控制，Root Port 和 Switch Port 应支持对每个端口的辅助电源进行独立控制。但是，允许对多个端口对辅助电源启用进行进行逻辑"或"，并组合这些端口的辅助电源控制，只要组合的辅助电源消耗不超过软件启用的总和即可。

5.7 Power Management System Messages and DLLPs

Table 5-12 定义了每个 PM 数据包在 PCI Express 堆栈中的位置。

Table 5-11: Power Management System Messages and DLLPs

Packet	Type
PM_Enter_L1	DLLP
PM_Enter_L23	DLLP
PM_Active_State_Request_L1	DLLP
PM_Request_Ack	DLLP
PM_Active_State_Nak	Transaction Layer Message
PM_PME	Transaction Layer Message
PME_Turn_Off	Transaction Layer Message
PME_TO_Ack	Transaction Layer Message

有关电源管理 DLLP 的结构的信息，请参阅 Section 3.5。

Power Management Message 遵循所有消息的一般规则。Power Management Message 各字段遵循以下规则：

- Length 字段保留。
- Attribute 字段设置为全 0b。
- Address 字段保留。
- Requester ID 字段参阅 Section 2.2.8.2 的 Table 2-21 。
- Traffic Class 字段使用 TC0。

5.8 PCI Function Power State Transitions

所有 PCI-PM 电源管理状态切换都由软件控制，但 Fundamental Reset 除外，它将所有 Function 置于 $D0_{uninitialized}$ 状态。Figure 5-18 显示了所有支持的状态转换。未标记的弧线表示软件启动的状态转换（设置电源状态操作）。

公众号：芯片生态伙伴

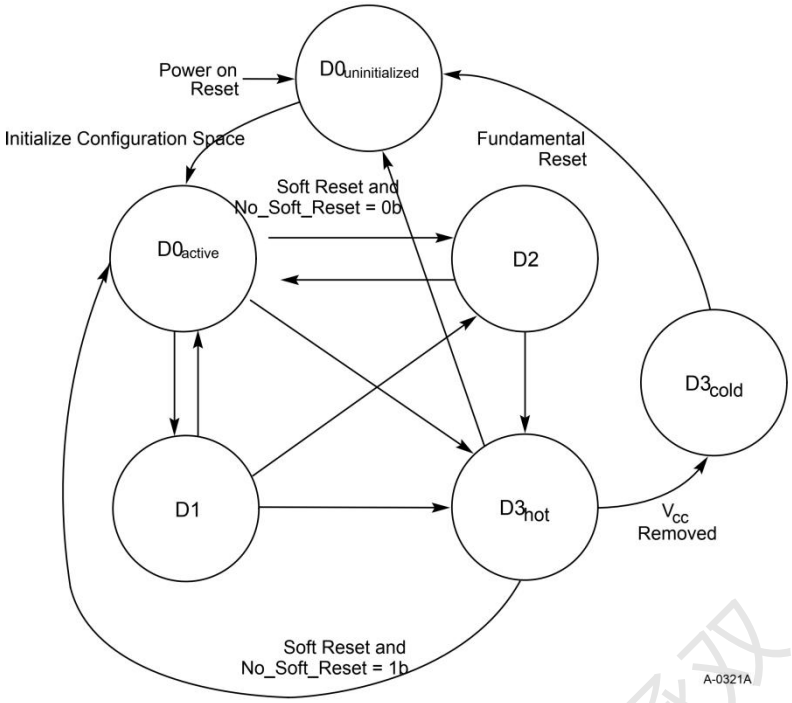

Figure 5-18 Function Power Management State Transitions

5.9 State Transition Recovery Time Requirements

Table 5-13 显示了系统软件必须允许的最短恢复时间，在 Function 被编程为改变状态的时间和 Function 下次访问的时间（包括配置空间）之间，除非使用 Readiness Notifications（参阅 Section 6.23 ） 向系统软件指示修改的值。对于 bridge Function，该延迟也构成了桥状态改变与逻辑总线上的任何 Function 可被访问之间的最小延迟。

Table 5-13 PCI Function State Transition Delays

Initial State	Next State	Minimum System Software Guaranteed Delays
D0	D1	0
D0 or D1	D2	200 ms
D0, D1 or D2	$D3_{Hot}$	10 ms
D1	D0	0
D2	D0	200 ms
$D3_{Hot}$	D0	10 ms

5.10 PCI Bridges and Power Management

在操作系统的指导下进行电源管理，每类 Function 都必须有明确定义的功能可用性标准，以及在每个电源管理状态下运行时必须保留的功能上下文。一些示例设备类规范已被提议作为 ACPI 规范的一部分，用于从音频到网络附加卡的各种 Function。虽然为大多数 Function 定义设备类特定的行为策略不在本规范的范围内，但为 PCI bridge Function 定义所需的行为在本规范的范围内。此处的定义适用于所有三种类型的 PCIe Bridge：

- Host bridge, PCI Express to expansion bus bridge, or other ACPI enumerated bridge
- Switches
- PCI Express to PCI bridge
- PCI-to-CardBus bridge

控制这些 Function 状态的机制根据所存在的始发设备的类型而有所不同。以下部分描述了这些机制如何用于三种类型的桥。

本节详细介绍 PCI Express Bridge Function 的电源管理策略。PCI Express Bridge Function 可以表征为一个发起设备，其下游有一个辅助总线。本节描述 Bridge Function 的电源管理状态与其辅助总线的电源管理状态的关系。

Figure 5-19 中的阴影区域说明了本节中讨论的内容。

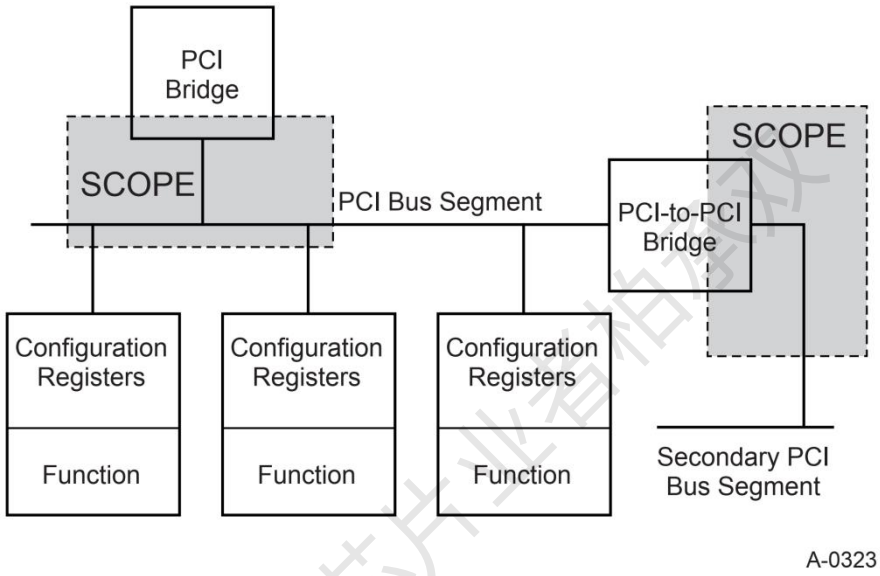

Figure 5-19 PCI Express Bridge Power Management Diagram

从 Figure 5-19 可以看出，从操作系统的角度来看，本章描述的 PCI Express Bridge 行为对 Host Bridge、Switch 以及 PCI Express to PCI bridge 是通用的。

系统软件有责任确保总线和下游 Function 电源管理状态的有效、可行组合仅用于给定总线和驻留在该总线上的所有 Function。

5.10.1 Switches and PCI Express to PCI Bridges

Switch 或 PCI Express to PCI bridge 的辅助总线的电源管理策略与为任何 Bridge Function 定义的相同。

Bridge Function 的 PMCSR_BSE 寄存器中的 BPCC_En and B2_B3# bus power/clock control 字段支持与任何其他桥相同的功能。

5.11 Power Management Events

电源管理事件有两种：

- Wakeup Events
- PME Generation

Wakeup Event 用于请求打开电源。

PME Generation Event 向系统提供标识，表示 Function 请求打开电源。

在传统 PCI 中，这两个事件都与 PME# 信号相关联。PME# 信号由 Function 置为有效以请求更改其电源管理状态。当 PME_En 被设置为 1 并且事件发生时，该 Function 设置 PME_Status 位为 1 并拉起 PME# 信号。它保持 PME# 信号有效，直到 PME_En 位或 PME_Status 被清 0（通常由软件清 0）。

在 PCI Express 中，唤醒事件与 WAKE# 信号相关联。如果支持，WAKE# 信号在相关的外形规格规范中定义，并且当 Function 处于 $D3_{Cold}$ 并且 PME_En 设置为 1 时，Function 使用它来请求更改其 PCI-PM 电源管理状态。

在 PCI Express 中，在恢复主电源并训练链路后，启动唤醒的 Function（例如，拉起 WAKE#）向 RC 发送 PM_PME 消息。PM_PME 消息向 RC 提供 requesting Function 的身份，而无需软件轮询设置的 PME_Status 位。

公众号·芯片业者相承双

6 System Architecture

6.1 Interrupt and PME Support

PCI Express 中断模型支持两种机制：

- INTx emulation
- Message Signaled Interrupt (MSI/MSI-X)

为了实现传统兼容性，PCI Express 提供了一种 PCI INTx 中断机制，以向系统中断控制器（通常是 RC 的一部分）发出中断信号。该机制与现有的 PCI 软件兼容，并提供与 PCI 中断信令机制相同的服务级别和类型，并且独立于系统中断控制器。这种传统的兼容性机制允许设备 boot 时无需复杂的 BIOS 级中断配置/控制服务栈。它使用带内信令机制虚拟化 PCI 物理中断信号。

除了兼容 PCI INTx 的中断之外，PCI Express 还要求支持 MSI 或 MSI-X 或两者都支持。 PCI Express MSI 和 MSI-X 机制与 PCI 本地总线规范中定义的机制兼容。

6.1.1 Rationale for PCI Express Interrupt Model

PCI Express 在中断支持方面采用了 PCI 的演进方法。

根据 PCI/PCI-X 中断机制的要求，每个设备的 Function 都需要区分 INTx 和 MSI 操作模式。相对 PCI/PCI-X 设备而言，支持这两种方案所需的设备复杂度差不多。这种方法的优点包括：

- 与现有 PCI 软件模型兼容。
- 直接支持 boot 设备。
- INTx 传统机制的寿命更短（EOL，Easier End of Life）。

现有软件模型会区分 INTx 与 MSI 操作模式。因此，PCI Express 不需要特殊的软件支持。

6.1.2 PCI Compatible INTx Emulation

PCI Express 支持 *PCI Local Bus Specification* 中定义的 PCI 中断，包括 PCI 设备的 Function 的 PCI 配置空间的 Interrupt Pin 和 Interrupt Line 寄存器。PCI Express 设备支持这些寄存器，以实现向后兼容。实际的中断信令使用带内消息，而不是使用物理引脚发出信令。

定义了两种类型的消息，分别是 Assert_INTx 和 Deassert_INTx，用于模拟 PCI INTx 信令，其中 x 是分别用于 PCI 中断信号的 A，B，C 和 D。这些消息用于提供“虚拟线路”，以通过链路发出中断信号。Switch 收集这些消息，并通过 Switch 的上游端口发送到 RC，该 RC 将消息映射到系统中断资源。设备必须成对使用 Assert_INTx 和 Deassert_INTx，以模拟 PCI 中断触发的信令。PCI Express INTx 模拟系统中断的实际映射是特定于实现的，就像传统 PCI 中物理中断信号的映射一样。

在此规范的将来版本中，可能不赞成使用传统的 INTx 机制。

6.1.3 INTx Emulation Software Model

传统 INTx 的软件模型与 PCI 的软件模型匹配。芯片组/平台中断映射的系统 BIOS 报告以及每个设备的 Function 的中断与 PCI 中断线的关联，其处理方式与常规 PCI 系统完全相同。传统软件会读取每个设备的 Function 的 Interrupt Pin 寄存器，以确定该 Function

公众号·芯片业者相承双

是否由中断驱动。介于 01h 和 04h 之间的值表示该 Function 使用模拟的中断引脚来生成中断。

请注意，类似于物理中断信号，INTx 机制可能潜在地导致必须由系统软件处理的虚假中断。

6.1.4 Message Signaled Interrupt (MSI/MSI-X) Support

PCI Express 设备需要支持 MSI/MSI-X 中断（对于 PCI 3.0 设备是可选的）。MSI 和 MSI-X 机制通过 MWr 事务来传递中断。MSI 和 MSI-X 是边沿触发的中断机制。PCI 本地总线规范和本规范均不支持级别触发的 MSI / MSI-X 中断。某些 PCI 设备及其驱动程序依赖于 INTx 类型的电平触发的中断行为（由 PCI Express 旧式 INTx 仿真机制解决）。为了利用 MSI 或 MSI-X 功能和边沿触发的中断语义，可能必须重新设计这些设备及其驱动程序。

Per Vector Masking with MSI/MSI-X

使用 MSI 或 MSI-X 的设备和驱动程序面临着在产生新的中断消息时准确进行协调的挑战。如果硬件未能发送软件期望的中断消息，则中断事件可能“丢失”。如果硬件发送软件不期望的中断消息，则可能导致“虚假”中断。

Per Vector Masking（PVM）是 MSI 和 MSI-X 的结构功能，可用于帮助进行这种协调。例如，当软件中断服务程序开始时，它可以屏蔽向量，以帮助避免“虚假”中断。中断服务程序为所有它知道的中断条件提供服务后，可以取消屏蔽该向量。如果仍然存在任何中断条件，则需要硬件来生成新的中断消息，以确保不会丢失任何中断事件。

PVM 是 MSI-X 的标准功能，MSI 也可以选择实现该功能。强烈建议 MSI 也实现 PVM。

实现 MSI 的 Legacy Endpoint 需要支持 32 位或 64 位消息地址格式的 MSI Capability structure。实现 MSI 的 PCI Express Endpoint 需要支持 64 位消息地址格式的 MSI Capability structure。

MSI/MSI-X 事务的请求者必须将 No Snoop 和 Relaxed Ordering 属性设置为 0b。如果启用了 IDO 属性，则允许 MSI/MSI-X 事务的请求者设置 ID-Based Ordering (IDO) 属性。

请注意，与 INTx 消息不同，MSI/MSI-X 事务不限于 TC0 业务类。

Synchronization of Data Traffic and Message Signaled Interrupts

允许 MSI/MSI-X 事务使用最适合设备编程模型的 TC。通常与用于传输数据的 TC 相同；对于传统 I/O，应使用 TC0。

如果设备使用多个 TC，则它必须确保在数据流量和不使用同一 TC 的中断消息之间保持适当的同步。确保此同步的方法是特定于实现的。一种选择是，设备在发出 MSI/MSI-X 事务之前，使用用于数据流量的每个其他 TC 发出零长度读取（如 Section 2.2.5 所述）。其他方法也是可以的。但是请注意，平台软件（例如设备驱动程序）通常仅能够使用 TC0 发出事务。

6.1.5 PME Support

PCI Express 支持与 PCI 设备的 PME 一样的电源管理事件。

如 Chapter 5 所述，使用带内事务处理 PME 消息（PM_PME）完成 PME 信令。

6.1.6 Native PME Software Model

PCI Express 感知软件可以让 RC 通过中断发送 PME。当配置为支持本机 PME 时，根端口会接收 PME 消息，并将其 Root Status 寄存器中的 PME Status 字段置 1。如果软件已将 Root Control 寄存器中的 PME Interrupt Enable 字段设置为 1b，则根端口将生成一个中断。

如果根端口可以使用 INTx 消息触发中断信令，则只要满足以下所有条件，就必须声明虚拟 INTx 连线：

- Command 寄存器中的 Interrupt Disable 字段设置为 0b。
- Root Control 寄存器的 PME Interrupt Enable 字段设置为 1b。
- Root Status 寄存器中的 PME Status 字段置 1b。

请注意，当请求服务时，同一 Function 内的所有其他中断源将声明相同的虚拟 INTx 线。

如果启用了根端口以使用 MSI 或 MSI-X 进行边沿触发的中断信令，则每次以下条件的逻辑 AND 从 FALSE 变为 TRUE 时，都必须发送中断消息：

- 相关的向量未屏蔽（如果 MSI 不支持 PVM，则不适用）。
- Root Control 寄存器的 PME Interrupt Enable 字段设置为 1b。
- Root Status 寄存器中的 PME Status 字段置 1b。

请注意，如 PCI Express Capabilities 寄存器中的 Interrupt Message Number 字段所示，PME 和热插事件中断（在同时实现时）始终共享相同的 MSI 或 MSI-X 向量。

该中断的软件处理程序可以通过读取根端口中 Root Status 寄存器中的 PME Requester ID 字段来确定哪个设备发送了 PME 消息。通过向 Root Status 寄存器中的 PME Status 字段写入 1b 来消除中断。有关更多详细信息，请参见 Section 7.8.14。

Root Complex Event Collector 为 RC Integrated Endpoint 提供了上述功能的支持。

6.1.7 Legacy PME Software Model

旧版软件不了解 PME 的机制。在存在旧版系统软件的情况下，RC 中的系统电源管理逻辑接收 PME 消息，并通过特定于实现的机制通知系统软件。RC 可以利用 PM_PME 中的 Requester ID 来通知系统软件哪个设备导致了电源管理事件。

由于 PME 是通过消息传递的，因此 PME 在 PCI Express 中具有边沿触发的语义，这不同于常规 PCI 中用电平触发的 PME 机制。RC 的职责是从系统软件中提取这种差异，以保持与常规 PCI 系统的兼容性。

6.1.8 Operating System Power Management Notification

为了保持与非 PCI Express 感知系统软件的兼容性，系统电源管理逻辑必须由固件配置为默认使用 PME 的旧机制。PCI Express 感知的系统软件必须在启用本地的，基于中断的 PME 信令之前通知固件。为了响应此通知，如果需要，系统固件必须重新配置 RC，以禁用 PME 的旧机制。该固件通知的详细信息超出了本规范的范围，但是由于它将在系统运行时执行，因此对此通知的响应不得干扰系统软件。 因此，在控制权移交给操作系统之后，固件不得写入操作系统所拥有的可用系统内存或任何 PCI Express 资源（例如，配置空间结构）。

6.1.9 PME Routing Between PCI Express and PCI Hierarchies

6.2 Error Signaling and Logging

在本文档中，描述了必须检查的错误以及可选择检查的错误。每个错误都与端口或特定设备（或 multi-Function 设备中的 Function）相关联，并且此关联关系与错误说明一起给出。本节将讨论如何分类和报告错误。

6.2.1 Scope

本节说明 PCI Express 组件的错误信令和日志记录的要求。这包括 PCI Express 接口本身发生的错误、在 PCI Express 上发起的事务发生的错误以及与 PCI Express 接口有关的组件内发生的错误。本节不重点讨论与 PCI Express 接口无关的组件中发生的错误。通过采用特定于设备的中断的专有方法，可以更好地处理这种类型的错误信令。

PCI Express 定义了两个错误报告模型：baseline capability 和 Advanced Error Reporting Capability。所有 PCI Express 设备都需要实现基本错误报告功能，并定义了最低错误报告要求。Advanced Error Reporting Capability 被定义用于更健壮的错误报告，并通过特定的 PCI Express Capability 结构实现（有关此可选功能的定义，请参阅 Chapter 7）。本节明确指出了 baseline 和 Advanced Error Reporting Capability 之间的所有错误处理差异。

通过将 PCI Express 错误映射到除 PCI Express 特定机制之外的现有 PCI 报告机制，所有 PCI Express 设备都支持现有的非 PCI Express 感知软件，用于错误处理。

6.2.2 Error Classification

PCI Express 错误可分为两种类型：不可纠正错误和可纠正错误。此分类将导致功能故障的错误与导致性能降低的错误分开。不可纠正错误可以进一步分为致命错误或非致命错误（请参见 Figure 6-1）。

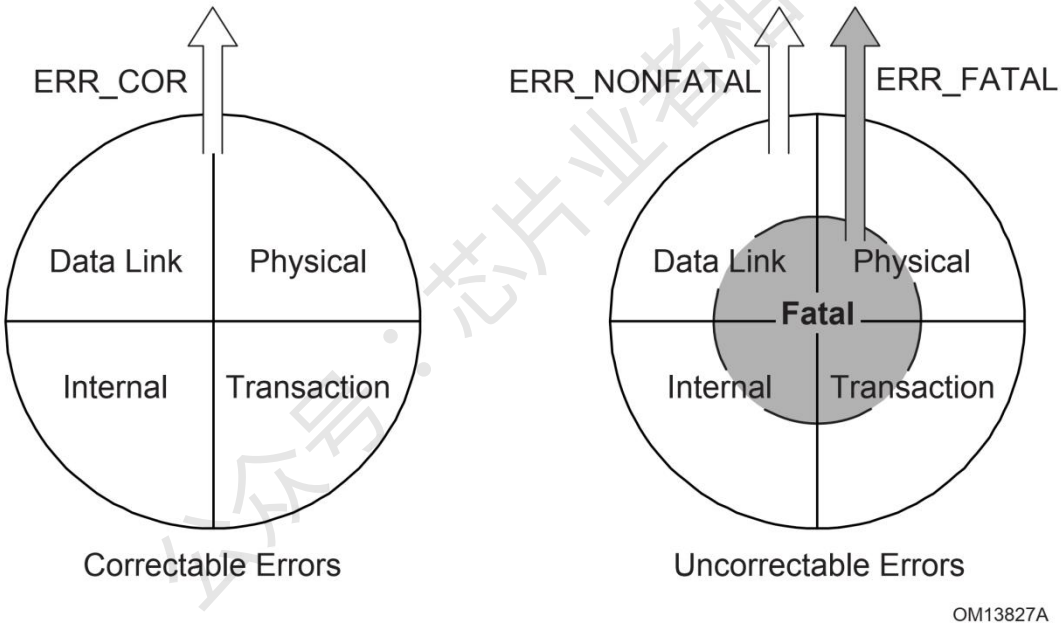

Figure 6-1: Error Classification

将错误严重性分类为致命、不可纠正和可纠正为平台提供了将错误映射到合适的处理机制的机制。例如，平台可能选择使用低优先级的性能监视软件来响应可纠正的错误。此类软件可以计算可纠正错误的频率，并提供链路完整性信息。另一方面，平台设计人员可能会选择将致命错误映射到系统角度的复位。平台设计者决定如何将 PCI Express 严重性级别映射到平台严重性级别。

6.2.2.1 Correctable Errors（可纠正错误）

可纠正错误包括硬件可以恢复而不会丢失任何信息的那些错误情况。硬件可以纠正这些错误，并且不需要软件干预。例如，可以通过数据链路级重试纠正的 TLP 中的 LCRC 错误被视为可纠正的错误。测量链路级别可纠正错误的频率可能有助于分析链路的完整性。

公众号·芯片业者相承双

可纠正错误还包括事务级别的情况，在这种情况下，一个引擎检测到 TLP 错误，但另一引擎负责根据需要采取任何恢复操作，例如用单独的后续事务重新尝试该操作。 检测引擎可以配置为将错误报告为可纠正的，因为恢复引擎可以纠正该错误。如果确实需要采取恢复措施，则如果恢复引擎决定不尝试恢复，则恢复引擎必须将错误报告为不可纠正。

6.2.2.2 Uncorrectable Errors（不可纠正错误）

不可纠正错误是影响接口功能的那些错误场景。本规范中没有定义纠正这些错误的机制。报告一个不可纠正的错误类似于在 PCI/PCI-X 中声明一个 SERR # 。为了使系统更可靠地处理错误，本规范进一步将不可纠正的错误分为致命错误和非致命错误。

6.2.2.2.1 Fatal Errors（致命错误）

致命错误是无法纠正错误的一种场景，会使链路和相关硬件不可靠。对于致命错误，可能需要复位链路上的组件才可以恢复可靠的操作。平台对致命错误的处理以及为限制这些错误的影响所做的任何努力都是特定于平台实现的。

6.2.2.2.2 Non-Fatal Errors（非致命错误）

非致命错误是无法纠正错误的一种场景，会导致特定事务传输不可靠，但是链路相对功能正常。将非致命错误与致命错误隔离开来，可为设备或系统管理软件中的请求者/接收者逻辑提供从错误中恢复的机会，而无需复位链路上的组件，而且不会干扰正在进行的其他事务。与错误事务无关的设备不受错误影响。

6.2.3 Error Signaling

有三种互补机制使得检测到错误的设备可以告知系统或另一设备已发生错误。第一种机制是 Completion Status，第二种机制是通过带内错误消息，第三种机制是通过错误转发（也称为数据中毒）。

Section 6.2.7 描述了所有错误以及检测到错误时要求硬件如何响应。

6.2.3.1 Completion Status

Completion header 中的 Completion Status 字段不是 Successful Completion 时表示相关的请求失败（请参阅 Section 2.2.9）。这是一种错误报告方法，可使请求者将错误与特定请求相关联。换句话说，由于直到完成返回后才认为 Non-Posted Request 是完整的，因此 Completion Status 字段使请求者有机会在某个更高级别的协议中"解决"该问题（超出本规范的范围）。例如，如果向可预取的内存空间发出读取，并且返回 Unsupported Request，则请求者如果选择重新发出读取请求是不违反规范的。请注意，从 PCI Express 的角度来看，重新发出的读请求是一个不同的请求。

6.2.3.2 Error Messages

Error Message 发往 RC 的对应 Root Port。 Root Port 自己的错误由其自己报告。

如果实现了 Root Complex Event Collector，则来自 RC Integrated Endpoint 的错误可以选择发往 Root Complex Event Collector。源自 RC Integrated Endpoint 的错误需要在与其位于同一逻辑总线上的 Root Complex Event Collector 中报告。Root Complex Event Collector 必须明确声明支持的 RC Integrated Endpoint 作为其功能的一部分；每个 RC Integrated Endpoint 必须与一个以上的 Root Complex Event Collector 关联。

当检测到多个错误具有相同严重性时，具有相同 Requester ID 的错误消息可能会合并。对于每个严重性级别的错误，必须至少发

送一条错误消息。但是请注意，在某些情况下，检测到给定的错误将排除某些错误的报告，请参阅 Section 6.2.3.2.3。另请注意 Section 6.2.4 中有关多功能设备中非特定功能错误的特殊规则。

Table 6-1: Error Messages

Error Message	Description
ERR_COR	This Message is issued when the Function or Device detects a correctable error on the PCI Express interface. Refer to Section 6.2.2.1 for the definition of a correctable error.
ERR_NONFATAL	This Message is issued when the Function or Device detects a Non-fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.2 for the definition of a Non-fatal, uncorrectable error.
ERR_FATAL	This Message is issued when the Function or Device detects a Fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.1 for the definition of a Fatal, uncorrectable error.

对于这些消息，RC 通过消息头的 Requester ID 标识消息的发起者。RC 将这些错误消息转换为平台级事件。

Use of ERR_COR, ERR_NONFATAL, and ERR_FATAL

在 1.0 和 1.0a 规范中，一个给定错误可能是可纠正的、非致命的或致命的。假设使能了信令功能，则可以通过 ERR_COR 发出可纠正的错误信令，通过 ERR_NONFATAL 发出非致命错误信令，并且通过 ERR_FATAL 发出致命错误信令。

在之后的 Role-Based Error Reporting 规范中，非致命错误有时候用 ERR_NONFATAL 发出信令，有时候用 ERR_COR 发出信令，有时不发出信令，这取决于检测到错误的设备的角色以及设备是否实现 AER（请参阅 Section 6.2.3.2.4）。在某些平台上，发送 ERR_NONFATAL 将阻止另一个设备尝试恢复或确定错误的最终处理方式。对于检测设备不是一个合适确定最终处理方式的设备的情况，实现了 AER 的设备会使用 ERR_COR 发出非致命错误的信令，用于对软件的通知。对于检测设备是合适的确定最终处理方式的设备的情况，设备会通过 ERR_NONFATAL 发出非致命错误信令。

对于非致命的不可纠正错误，如果软件希望避免在检测到该错误后继续进行层次结构操作，则软件可以配置实现 AER 的检测设备以将该错误的严重性升级为致命错误。 无论设备的角色如何，检测设备将始终通过 ERR_FATAL 发出致命错误信号。

软件应认识到多个设备可以使用不同类型的错误消息来发出单个事务的信令。 例如，中途的接收方可能使用 ERR_COR 发出中毒的 TLP 信号，而最终目标接收方可能使用 ERR_NONFATAL 发出信令。

6.2.3.2.1 Uncorrectable Error Severity Programming (Advanced Error Reporting)

对于实现了高级错误报告功能的设备的 Function，Uncorrectable Error Severity 寄存器允许将每个不可纠正错误编程为致命错误或非致命错误。不可纠正错误无法使用定义的 PCI Express 机制恢复。但是，某些平台或设备可能认为某个特定错误对链路或设备致命，而另一平台则认为该错误不是致命错误。Uncorrectable Error Severity 寄存器的默认值用作该规范的默认方式，但是如果设备驱动程序或平台软件需要更强大的错误处理，则可以重新编程该寄存器。

Baseline error handling 不支持严重性编程。

6.2.3.2.2 Masking Individual Errors

Section 6.2.7 列出了受此规范约束的所有错误，并描述了何时发送上述每个错误消息。使用 Device Control 寄存器的 Reporting Enable 字段（Section 7.8.4）或 PCI Command 寄存器的 SERR# Enable 字段（Section 7.5.1.1）按类别（可纠正，非致命，致命）传输这些错误消息。

对于实现高级错误报告功能的设备，Uncorrectable Error Mask 寄存器和 Correctable Error Mask 寄存器允许分别屏蔽每个错误条件。

如果 Device Control 寄存器和 PCI Command 寄存器中的组合设置未启用针对特定错误类别的消息，则无论相应掩码寄存器的值如何，都不会发送该类别的消息。

如果在检测到单个错误时将其屏蔽，则其错误状态位仍会受到影响，但是不会向 RC 发送任何错误报告消息，并且 Header Log 和 First Error Pointer 寄存器也不会被修改。

6.2.3.2.3 Error Pollution

如果给定事务的错误条件未隔离到最严重的事件，则可能发生错误污染。 例如，假设物理层检测到接收器错误。在物理层检测到此错误，并将错误报告给 RC。为避免此错误传播并在上层引起后续错误（例如，在数据链路层发生 TLP 错误），从而导致更难确定错误的根本原因，对于同一数据包发生的后续错误将不会由数据链路层或事务层报告。 同样，当数据链路层检测到错误时，事务层将不会报告在同一数据包中发生的后续错误。此行为仅适用于与特定数据包相关的错误。

已纠正内部错误是指那些被组件屏蔽或解决的错误。有关详细信息，请参见 Section 6.2.9。因此，已纠正内部错误不会导致错误污染，而且应该在检测到错误时进行报告。

对于在事务层中检测到的错误和不可纠正的内部错误，允许并建议对于单个接收到的 TLP 报告不超过一个错误，并且使用以下优先级（从最高到最低）：

- Uncorrectable Internal Error
- Receiver Overflow
- Flow Control Protocol Error
- ECRC Check Failed
- Malformed TLP
- AtomicOp Egress Blocked
- TLP Prefix Blocked
- ACS Violation
- MC Blocked TLP
- Unsupported Request (UR), Completer Abort (CA), or Unexpected Completion[62]
- Poisoned TLP Received

Completion Timeout 错误不在上面的优先级列表中，因为它不是由处理接收到的 TLP 检测到的。

这是优先级列表背后原理的示例。如果给定的 TLP 的 ECRC 检查失败，则 TLP 的全部内容（包括其标头）可能已损坏，因此，报告错误（如 Malformed TLP 或 Unsupported Request）几乎没有意义。

注 62：这些是互斥的错误，因此它们的相对顺序无关紧要。

6.2.3.2.4 Advisory Non-Fatal Error Cases

在某些情况下，非致命错误的检测器不是确定错误是否可恢复或根本不需要采取任何恢复措施的最合适设备引擎。例如，如果软件尝试执行对不存在的设备或 Function 进行配置读，则 Completion 的 UR 状态将导致向软件发出错误信令，并且软件也不需要为 Completer 另外发送一个 ERR_NONFATAL Message。实际上，在某些平台上，用 ERR_NONFATAL 发送错误会导致一个系统错

公众号：芯片业者相承汉

误，这会破坏正常的软件探测。

“建议性非致命错误”案例主要由检测引擎（请求者，完成者或接收者）的角色和特定错误确定。在这种情况下，实现了 AER 的引擎通过发送 ERR_COR 消息作为软件的参考建议而不是发送 ERR_NONFATAL 来发出非致命错误的信号。对于这些情况，未实现 AER 的引擎不会发送任何错误消息，因为接收 ERR_COR 的软件将无法区分建议性非致命错误情况和用于评估链路完整性的可纠正错误情况。

以下是建议性非致命错误的具体情况。请注意，单个 TLP 可能会出现来自相同或不同错误类别（可纠正，非致命，致命）的多个错误。例如，Unexpected Completion 也可能中毒。有关报告多个错误的要求和建议，请参阅 Section 6.2.3.2.3。对于前面的示例，建议报告 Unexpected Completion，并且不报告接收到的中毒 TLP。

如果软件希望实现了 AER 的引擎处理通常更严重的建议性非致命错误案例，则软件可以将不可纠正错误的严重性升级为致命，在这种情况下，引擎将用 ERR_FATAL 错误发出警告。

6.2.3.2.4.1 Completer Sending a Completion with UR/CA Status

Completer 发送状态为 Unsupported Request or Completer Abort（UR/CA）的 Completion 来为 Non-Posted Request 返回不可纠正错误 [63]。如果此 UR/CA 错误的严重性是非致命的 [64]，则 Completer 必须把这种情况处理为 Advisory Non-Fatal Error[65]。实现 AER 的 Completer 通过发送 ERR_COR 消息来通知一个非致命错误。没有实现 AER 的 Completer 在这种情况下不会发送任何错误消息。

即使此特定事务有一个不可纠正错误，Completer 也必须将此情况作为 Advisory Non-Fatal Error 来处理，因为请求者在收到具有 UR/CA 状态的完成信息后，将使用请求者的特定机制的方式报告此错误（请参见 Section 6.2.3.2.5）。

注 63：如果 Completer 正在返回完成中的数据，并且数据不正确或可疑，则允许完成者使用错误转发（数据中毒）机制来发信令通知错误，而不是将其视为 UR 或 CA。

注 64：Non-Posted Request 的某些其他错误（例如 ACS 违规）也会导致 Completer 发送具有 UR 或 CA 状态的完成。如果错误的严重性（例如 ACS 违规）不是致命的，则 Completer 也必须将这种情况作为 Advisory Non-Fatal Error 进行处理。

注 65：如果严重程度是致命的，则该错误就不能是 Advisory Non-Fatal Error，必须通过 ERR_FATAL 发出信令。

6.2.3.2.4.2 Intermediate Receiver

当并非作为 TLP 最终 PCI Express 目的地的接收器检测 [66] 到 TLP 出现非致命错误时，此“中间”接收器必须将这种情况作为 Advisory Non-Fatal Error 处理 [67]。实现了 AER 的接收器通过发送 ERR_COR 消息来指示错误（如果启用）。没有实现 AER 的接收器在这种情况下不会发送任何错误消息。以下指出了 RC（RC）的中间接收者的一个例外情况。

中间接收器发生这种情况的一种示例是 Switch，该 Switch 检测正在路由的 TLP 中的有毒或不良 ECRC。即使这在 TLP 路径的这一点上是不可纠正（但不是致命的）错误，但中间接收器会将其作为 Advisory Non-Fatal Error 进行处理，以便 TLP 的最终接收方根据其错误设置更适当地处理错误。例如，在内存写入请求中检测到中毒的给定完成者可能掩盖了错误（因此未发出信号），而同一层次结构中的其他完成者则可能使用 ERR_NONFATAL 发出该错误信号。

如果 RC 检测到一个 TLP 的非致命错误，则通常会在根端口之间对等转发，但是 RC 不支持使用以下方式传播与错误相关的信息（例如 TLP Digest、EP 字段或等效内容），如果转发了交易，则 RC 必须使用 ERR_NONFATAL 发出错误信令，并且也不得转发该事务。一个示例是 RC 需要在根端口之间对等转发中毒的 TLP，但是 RC 的内部结构不支持中毒指示。

注 66：如果接收方未实现 ECRC 检查或未启用 ECRC 检查，则接收方将不会检测到 ECRC 错误。

注 67：如果严重程度是致命的，则该错误不能是 Advisory Non-Fatal Error，必须通过 ERR_FATAL 发出信令。

注 68：有关中毒的“内存写请求”的特殊规则，请参见 Section 2.7.2.2。

6.2.3.2.4.3 Ultimate PCI Express Receiver of a Poisoned TLP

当最终 PCI Express 目标接收到中毒 TLP 时，如果严重性不是致命的，并且接收方以允许继续操作的方式处理中毒数据，则接收方必须将此情况作为 Advisory Non-Fatal Error 进行处理 [69]。实现了 AER 的接收器通过发送 ERR_COR 消息来指示错误（如果启用）。没有实现 AER 的接收器在这种情况下不会发送任何错误消息。 有关适用于中毒的内存写请求的特殊规则，请参见第 2.7.2.2 节。

一个示例是 RC，它接收针对主机内存的中毒的 Memory Write TLP。如果 RC 将中毒的数据及其指示信息传送到主机内存，它将使用 ERR_COR 发出错误信令（如果启用）。如果 RC 没有将中毒数据传送到主机内存，它将使用 ERR_NONFATAL 发出错误信令（如果启用）。

另一个示例是请求者，该请求者接收了中毒的 Memory Read Completion TLP。如果请求者在内部传播中毒的数据或像处理具有 UR/CA 状态的完成一样处理错误，则会使用 ERR_COR 发出错误信令（如果启用）。如果请求者未以允许继续操作的方式处理中毒，则会使用 ERR_NONFATAL 发出错误信令（如果启用）。

注 69：如果严重程度是致命的，则该错误不能是 Advisory Non-Fatal Error，必须通过 ERR_FATAL 发出信令。

6.2.3.2.4.4 Requester with Completion Timeout

当 Non-Posted Request 的 Requester 在等待相关的 Completion 时超时，则允许 Requester 通过发出单独的后续请求来尝试从错误中恢复。允许请求者尝试恢复零次，一次或多次（有限的）时间，但是如果不做进一步的恢复尝试，则必须用不可纠正的错误消息来报告此错误（如果已启用）。

如果完成超时的严重程度不是致命的，并且请求者选择通过发出新请求来尝试恢复，则请求者必须首先将当前错误情况作为 Advisory Non-Fatal Error 进行处理 [70]。实现了 AER 的接收器通过发送 ERR_COR 消息来指示错误（如果启用）。没有实现 AER 的接收器在这种情况下不会发送任何错误消息。

请注意，通常只有在 Non-Posted Request 没有副作用的情况下，请求者才能从完成超时中进行自动恢复，但也可能取决于本规范范围之外的其他考虑因素。

注 70：如果严重程度是致命的，则该错误不能是 Advisory Non-Fatal Error，必须通过 ERR_FATAL 发出信令。 强烈建议不要尝试请求恢复，因为发送 ERR_FATAL 通常会导致整个拓扑结构崩溃。

6.2.3.2.4.5 Receiver of an Unexpected Completion

当接收方收到 unexpected Completion 且该 Unexpected Completion 错误的严重性不是致命错误时，接收方必须将此情况作为 Advisory Non-Fatal Error 进行处理 [71]。实现了 AER 的接收器通过发送 ERR_COR 消息来指示错误（如果启用）。没有实现 AER 的接收器在这种情况下不会发送任何错误消息。

如果 unexpected Completion 是错误路由的结果，则最终将在相关的请求者处触发 Completion Timeout 机制，请求者可以选择尝试恢复。unexpected Completion 的接收方通过将错误作为 Advisory Non-Fatal Error 来处理，可以避免对请求者恢复的干扰。

注 71：如果严重程度是致命的，则该错误不能是 Advisory Non-Fatal Error，必须通过 ERR_FATAL 发出信令。

6.2.3.2.5 Requester Receiving a Completion with UR/CA Status

当请求者收到具有 UR/CA 状态的完成信息时，通常假定错误严重性在完成者上不是致命的，完成者会将错误作为 Advisory Non-Fatal Error 来处理（请参阅 Section 6.2.3.2.4.1）。请求者必须确定是否需要执行任何错误恢复操作，采取哪种类型的恢复操作以及是否报告错误。

如果请求者需要报告错误，则请求者必须仅通过特定于请求者的机制进行报告。例如，许多设备具有关联的设备驱动程序，可以将错误报告给软件。再举一个重要的例子，如果 Configuration Read Completion 携带 UR/CA 状态，则某些平台上的 RC 会将全 1

返回给软件。

不允许请求者使用 PCI Express 日志记录和错误消息信令报告错误。

6.2.3.3 Error Forwarding (Data Poisoning)

通过将 TLP 中的 EP 位置 1 来指示错误转发，也称为数据中毒，请参阅 Section 2.7.2。这是 PCI Express 中错误报告的另一种方法，它使 TLP 的接收方能够将错误与特定的请求或完成关联。与完成状态机制不同，错误转发可以与包含数据的请求或完成一起使用。此外，TLP 路径上的"中间"接收者（不仅仅是最终目的地的接收者）需要检测并报告（如果启用）接收中毒的 TLP。这可以帮助软件确定路径上的特定 Switch 是否为该 TLP 引入了"毒"。

6.2.3.4 Optional Error Checking

该规范包含许多可选的错误检查。除非另有说明，否则如果未执行可选的错误检查并发生错误，则行为是不确定的。

当可选错误检查涉及多个规则时，除非另有说明，否则每个规则都是独立可选的。一个实现可以检查所有规则、或不检查所有规则或检查任何规则的组合。

除非另有说明，否则在确定是否执行可选的错误检查时使用特定于实现的标准。

6.2.4 Error Logging

Section 6.2.7 列出了受此规范约束的所有错误，并为每个错误指定了 logging 的要求。不支持 AER 的 Device Function 仅记录检测到错误的 Device Status 寄存器字段。请注意，使用 PCI 兼容（Type 00h 和 01h）配置寄存器中的报告机制，也会报告一些错误。Section 7.5 描述了这些寄存器字段如何受到本节中描述的不同类型的错误条件的影响。

对于支持 AER 的 Device Function，Table 6-3，Table 6-4 和 Table 6-5 中的每个错误对应于 Uncorrectable Error Status 寄存器或 Correctable Error Status 寄存器中的特定字段。这些寄存器由软件更精确地定义发生了哪些错误以及其严重程度。对于特定的事务层错误和 Uncorrectable Internal Error，将记录相关的 TLP Header。

在 multi-Function device 中，与设备内任何特定 Function 无关的 PCI Express 错误记录在该设备中所有 Function 的相应状态和日志寄存器中。

以下 PCI Express 错误不是特定于 Function 的：

- 所有的物理层错误。
- 所有的数据链路层错误。
- 以下事务层错误：
  - ECRC Fail
  - UR, when caused by no Function claiming a TLP
  - Receiver Overflow
  - Flow Control Protocol Error
  - Unexpected Completion, when caused by no Function claiming a Completion
  - Unexpected Completion，当由 Switch 无法转发的完成引起，并且入口端口是与多功能设备关联的 Switch 上游端口时。

公众号 芯片业者相承双

- 一些内部错误：
  - 如何确定内部错误是否是特定于 Function 的是特定于实现的。

在检测到其中一种错误时，一个 multi-Function device 对于给定的严重性最多生成一个的错误报告消息，其中该消息必须报告该设备的该 Function 的 Requester ID。如果没有使能任何 Function 来发送一个错误报告消息，则设备不会发送报告消息。如果所有启用了报告功能的 Function 都为错误设置了相同的严重性级别，则仅发送一条错误消息。如果所有启用报告功能的 Function 均未为错误设置相同的严重性级别，则将为每个严重性级别发送一条错误消息。当软件检测到这些错误之一时，负责扫描 multi-Function device 中的所有 Function。

6.2.4.1 RC Considerations (Advanced Error Reporting)

6.2.4.1.1 Error Source Identification

除了上述 logging 功能外，支持 AER 的 Root Port 或 Root Complex Event Collector 还需要实现 Error Source Identification 寄存器，该寄存器记录 Root Port 或 Root Complex Event Collector 收到的 ERR_NONFATAL/ERR_FATAL（不可纠正错误）和 ERR_COR（可纠正错误）消息的 Requester ID。为支持 AER 而编写的系统软件可以使用 Root Error Status 寄存器来确定哪些字段包含有效信息。

如果实现了 Root Complex Event Collector，则可以选择在与 RC Integrated Endpoint 位于同一逻辑总线上的 Root Complex Event Collector 中报告来自 RC Integrated Endpoint 的错误。Root Complex Event Collector 必须显式声明受支持的 RC Integrated Endpoint 作为其功能的一部分。每个 RC Integrated Endpoint 必须与一个以上的 Root Complex Event Collector 关联。

对于 Root Port 或 Root Complex Event Collector，为了将接收到的错误消息或内部生成的错误消息记录在 Root Error Status 寄存器和 Error Source Identification 寄存器中，必须“发送”错误消息。有关如何转发和发送已接收消息的信息，请参见 Section 6.2.8.1。内部产生的错误消息通过 Command 寄存器中的 SERR # Enable 字段（只表示 ERR_NONFATAL 和 ERR_FATAL）或 Device Control 寄存器中的 Reporting Enable 字段（ERR_COR，ERR_NONFATAL 和 ERR_FATAL）被启用以进行传输。

6.2.4.1.2 Interrupt Generation

与基本根复杂功能生成对错误消息的响应的基本错误功能相比，Root Error Command 寄存器可以进一步控制 RC 对 Correctable、Non-Fatal 和 Fatal 错误消息的响应。Bit 字段使能或禁用三种错误消息类型的中断生成。可以通过 PCI Express Capability structure 禁用响应于错误消息的系统错误生成。

如果 Root Port 或 Root Complex Event Collector 使用 INTx 消息进行电平触发的中断信令，则只要满足以下所有条件，就必须声明虚拟 INTx 线：

- Command 寄存器的 Interrupt Disable 字段设置为 0b。
- Root Error Command 寄存器中的至少一个 Error Reporting Enable 字段及其在 Root Error Status 寄存器中的相关 Error Messages Received 字段都设置为 1b。

请注意，当请求服务时，同一 Function 内的所有其他中断源将声明相同的虚拟 INTx 中断线。

如果 Root Port 或 Root Complex Event Collector 使用 MSI 或 MSI-X 进行边沿触发的中断信令，则每当以下条件的逻辑 AND 从“FALSE”转换为“TRUE”时，都必须发送中断消息：

- 对应的向量未屏蔽（如果 MSI 不支持 PVM，则不适用）。
- Root Error Command 寄存器中的至少一个 Error Reporting Enable 字段及其在 Root Error Status 寄存器中的相关 Error Messages Received 字段都设置为 1b。

请注意，Advanced Error Reporting MSI/MSI-X 中断始终使用 Root Error Status 寄存器中 Advanced Error Interrupt Message Number 字段指示的向量作为中断向量。

6.2.4.2 Multiple Error Handling (Advanced Error Reporting Capability)

对于高级错误报告功能，Uncorrectable Error Status 寄存器和 Correctable Error Status 寄存器会累积该特定 PCI Express 接口的错误集合。这些字段将保持设置状态，直到被软件清除或复位为止。由于在 Uncorrectable Error Status 寄存器中可能设置了多个字段，因此 First Error Pointer 寄存器指向最先发生的不可屏蔽的不可纠正错误。当 Uncorrectable Error Status 寄存器的相应字段置 1 时，First Error Pointer 寄存器有效。如果未设置 Uncorrectable Error Status 寄存器的相应字段，则 First Error Pointer 的值就没有意义。

高级错误报告功能提供了记录错误的报头的功能。一个实现可以支持记录多个报头，但是必须支持记录至少一个报头的能力。记录多个报头的能力由 Multiple Header Recording Capable 字段指示，并由 Advanced Error Capabilities and Control 寄存器的 Multiple Header Recording Enable 字段来使能。当支持并使能多个报头记录时，将按照检测到相应错误的顺序记录报头。

当 First Error Pointer 寄存器有效时，Header Log 寄存器包含对应错误的已记录报头。将 1b 写入与 First Error Pointer 寄存器对应的 Uncorrectable Error Status 寄存器的字段，用于释放该记录的报头占用的空间，然后 Header Log 寄存器中报告下一个记录的报头，First Error Pointer 寄存器也要更新，以指向相应的 Uncorrectable Error Status 寄存器字段。如果没有其他记录的报头具有与最后一个报头相同的错误类型，则 Uncorrectable Error Status 寄存器中的对应字段写 1 清除；否则，它保持置位状态。当到达最后记录的报头并将 1b 写入相应的 Uncorrectable Error Status 寄存器字段时，First Error Pointer 寄存器对应字段将变为无效，并且 Header Log 寄存器的值未定义。

由于实现仅能够记录有限数量的报头，因此软件必须及时处理 First Error Pointer、Header Log 和 TLP Prefix Log 寄存器，以限制后续错误导致当前错误信息丢失的风险，这一点很重要。当检测到需要记录报头的错误，并且记录的错误报头的数量达到实现支持的数量，或者未设置 Multiple Header Recording Enable 字段并且 First Error Pointer 寄存器有效时，就会发生 Header Log Overflow。

实现可以有选择地检查这种情况并报告 Header Log Overflow 错误。这是报告的与检测功能相关的错误。

具有多个报头记录功能的设置和 Header Log Overflow 的检查是独立可选实现的。

6.2.4.3 Advisory Non-Fatal Error Logging

Section 6.2.3.2.4 介绍了 Advisory Non-Fatal Error 的情况，在这种情况下，实现了 AER 的引擎检测到非致命严重性的不可纠正错误时，会使用 ERR_COR 而不是 ERR_NONFATAL 发出错误消息（如果启用）。对于相同的情况，没有实现 AER 的引擎不会发送任何错误消息。本节的讨论是在实现 AER 引擎的背景下进行的。

对于 Advisory Non-Fatal Error 的情况，由于使用可纠正错误消息发出不可纠正错误的信令，因此适用涉及不可纠正和可纠正错误的 control/status/mask 字段。Figure 6-2 给出了该过程的流程图。以下是记录 Advisory Non-Fatal Error 的一些不一样的地方。

首先，不可纠正错误的严重性必须是非致命的，这由 Uncorrectable Error Severity 寄存器中的相关字段确定。如果严重程度是致命的，则该错误不属于 Advisory Non-Fatal Error，并且会通过 ERR_FATAL 发出信令（如果启用）。

接下来，特定的错误案例需要是 Section 6.2.3.2.4 中记录的 Advisory Non-Fatal Error 案例之一。如果不是，则该错误不属于 Advisory Non-Fatal Error，并且将通过不可纠正的错误消息发出信令（如果启用）。

接下来，在 Correctable Error Status 寄存器中设置 Advisory Non-Fatal Error Status 字段以指示建议性错误的发生，并检查 Correctable Error Mask 寄存器中的 Advisory Non-Fatal Error Mask 字段，如果该字段设置为 1b， 则无需进一步处理。

如果 Advisory Non-Fatal Error Mask 字段被清 0，则根据报告为建议性错误的特定不可校正错误，通过设置 Uncorrectable Error Status 寄存器中的对应字段来继续进行记录。如果 Uncorrectable Error Mask 寄存器中的相应的不可纠正错误字段被清除，并且该错误是需要报头记录的错误，则将记录前缀和报头，这将取决于资源的可用性。参见 Section 6.2.4.2。

最后，如果 Device Control 寄存器中的 Correctable Error Reporting Enable 字段被设置，则发送 ERR_COR 消息。

6.2.4.4 TLP Prefix Logging

对于任何同时支持 TLP Prefix 和 AER 的设备，根据与 Header Log 寄存器相同的规则，将与错误的 TLP 相关联的 TLP Prefix 记录在 TLP Prefix Log 寄存器中（这样，当 First Error Pointer 寄存器有效时，TLP Prefix Log 和 Header Log 寄存器记录的错误信息就总与该 First Error Pointer 寄存器对应）。

TLP Prefix Log Present 字段（see Section 7.10.7）表示 TLP Prefix Log 寄存器（see Section 7.10.12）是否包含有效信息。

AER 仅记录 End-End TLP Prefix。Local TLP Prefix 的记录可能会在其他地方使用前缀特定的机制进行（例如，涉及 MRI-IOV TLP Prefix 的错误记录在 MR-IOV 中，而不记录在 AER 中）。

End-End TLP Prefix 记录在 TLP Prefix Log 寄存器中。相应的 TLP Header 记录在 Header Log 寄存器中，但有两个例外：

- 如果设置了 Extended Fmt Field Supported 字段（see Section 7.8.15），则不支持 TLP Prefix 的 Function 收到包含 TLP Prefix 的 TLP 会被视为一个 Malformed TLP，并且 Header Log 寄存器记录 TLP 的前四个 DW 值（这是个 TLP Prefix，后跟尽可能多的 TLP Header）。
- 接收到的 TLP 包含的 End-End TLP Prefix 的数量超过该 Function 的 Max End-End TLP Prefixes 字段所指示的值，则必须将 TLP 作为错误处理（有关详细信息，请参阅 Section 2.2.10.2），并在 Header Log 寄存器中存储第一个溢出的 End-End TLP Prefix 的第一个 DW，Header Log 寄存器的其余部分未定义。

6.2.5 Sequence of Device Error Signaling and Logging Operations

Figure 6-2 显示了与发信令和记录设备检测到的错误有关的操作流程。

公众号·芯片业者相承双

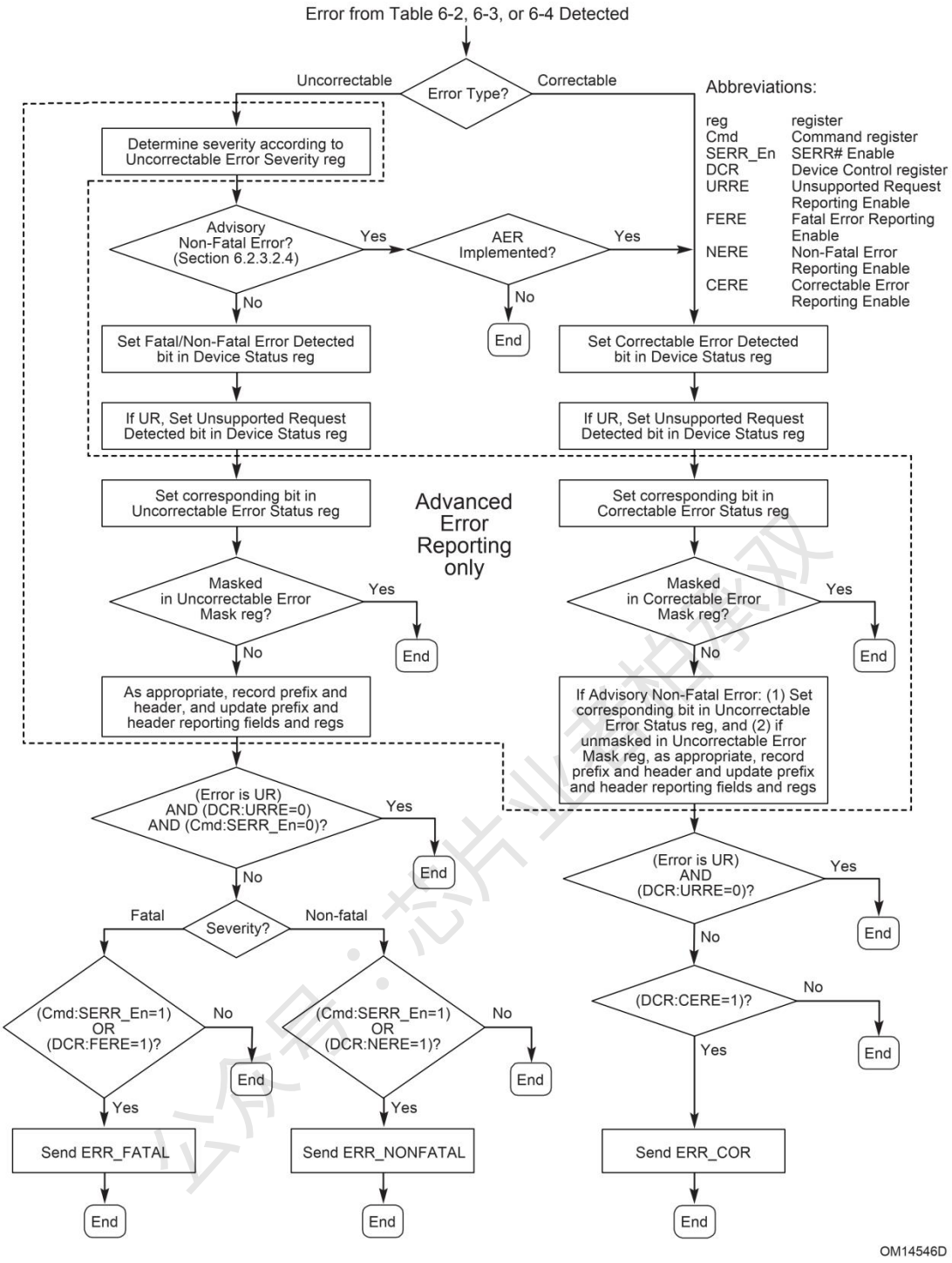

Figure 6-2: Flowchart Showing Sequence of Device Error Signaling and Logging Operations

6.2.6 Error Message Controls

错误消息具有一组复杂的关联控制和状态位。Figure 6-3 以伪逻辑图的形式提供了一个概念性简介，其中介绍了如何生成、记录、转发并最终将错误消息通知系统。并非所有记录的状态位都显示。此图中所示的逻辑门旨在传达一般概念，而不是直接实现。

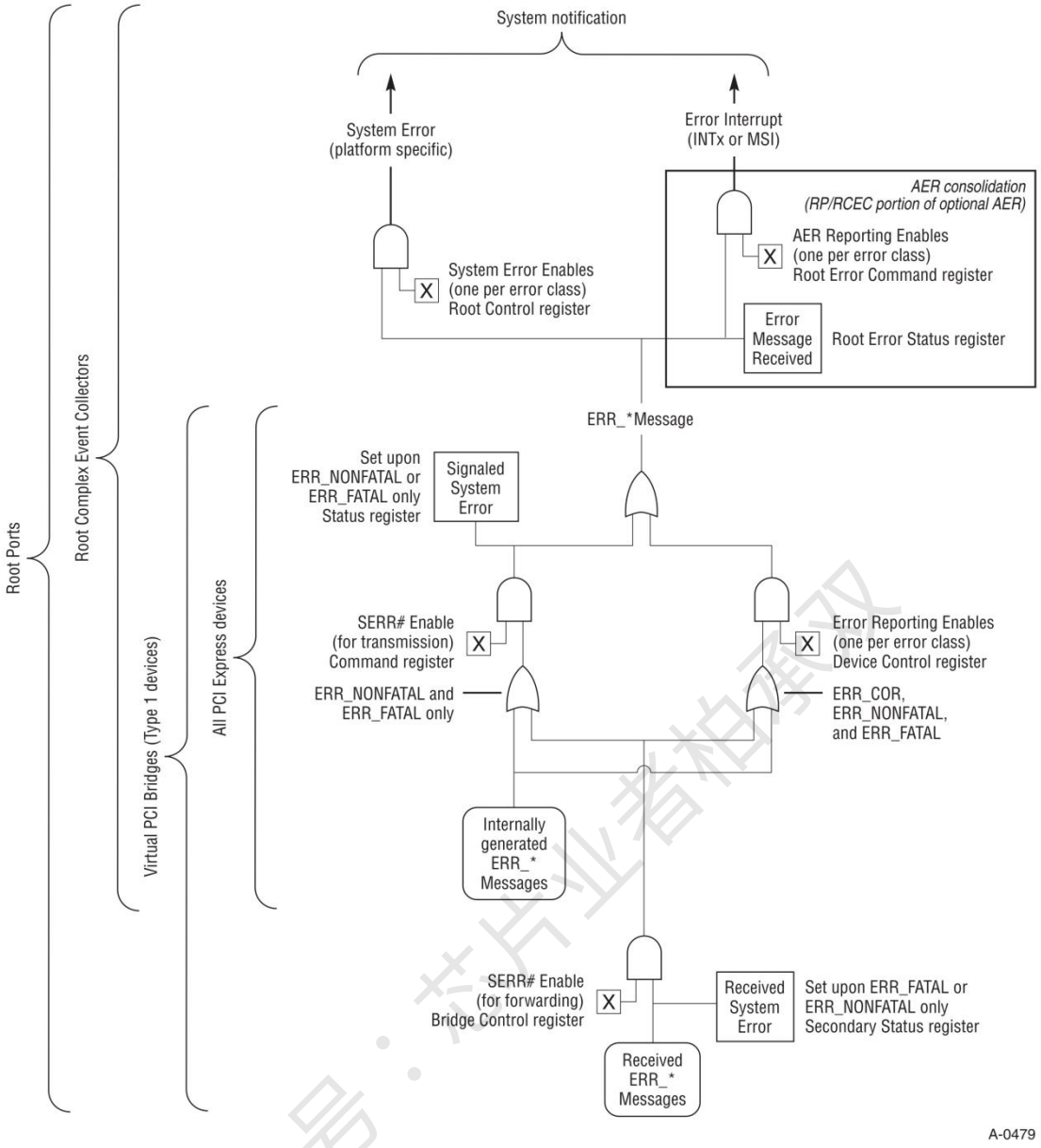

Figure 6-3: Pseudo Logic Diagram for Error Message Controls

6.2.7 Error Listing and Rules

Table 6-2 至 Table 6-4 列出了此规范定义的所有 PCI Express 错误。列出的每个错误都有一个简短的名称，在硬件中如何检测到该错误，该错误的默认严重性以及检测该错误的引擎采取的预期操作。这些操作构成了 PCI Express 错误报告和日志记录的规则。

Default Severity 指定错误的默认严重性，而无需重新编程任何软件。对于支持 AER 的 device Function，使用 Error Severity 寄存器可以将无法纠正的错误编程为致命或非致命错误。没有实现 AER 的 device Function 使用默认值，并且不可重新编程。

Table 6-2: General PCI Express Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[73]	References
Corrected Internal Error	Correctable (masked by default)	*Component:* Send ERR_COR to Root Complex.	Section 6.2.9
Uncorrectable Internal Error	Uncorrectable (Fatal and masked by default)	*Component:* Send ERR_FATAL to Root Complex. Optionally, log the prefix/header of the first TLP associated with the error.	Section 6.2.9
Header Log Overflow	Correctable (masked by default)	*Component:* Send ERR_COR to Root Complex.	Section 6.2.4.2

注 73：对于这些表，将给出检测引擎程序操作的方式，就好像所有使能位都设置为"启用"一样；对于高级错误处理，屏蔽字段被禁用，严重性字段设置为其默认值。必须根据这些字段的实际设置来修改操作。

Table 6-3: Physical Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[74]	References
Receiver Error	Correctable	*Receiver:* Send ERR_COR to Root Complex.	Section 4.2.1.1.3 Section 4.2.1.2 Section 4.2.4.7 Section 4.2.6

Table 6-4: Data Link Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[74]	References
Bad TLP	Correctable	*Receiver:* Send ERR_COR to Root Complex.	Section 3.5.3.1
Bad DLLP		*Receiver:* Send ERR_COR to Root Complex.	Section 3.5.2.2
Replay Timer Timeout		*Transmitter:* Send ERR_COR to Root Complex.	Section 3.5.2.1
REPLAY_NUM Rollover		*Transmitter:* Send ERR_COR to Root Complex.	Section 3.5.2.1
Data Link Protocol Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL to Root Complex.	Section 3.5.2.2
Surprise Down		If checking, send ERR_FATAL to Root Complex.	Section 3.2.1

公众号·志岩业者拍承双

Table 6-5: Transaction Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[74]	References
Poisoned TLP Received	Uncorrectable (Non-Fatal)	*Receiver:* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error cases described in Sections 6.2.3.2.4.2 and 6.2.3.2.4.3. Log the prefix/header of the Poisoned TLP.[75]	Section 2.7.2.2
ECRC Check Failed		*Receiver (if ECRC checking is supported):* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.2. Log the prefix/header of the TLP that encountered the ECRC error.	Section 2.7.1
Unsupported Request (UR)		*Request Receiver:* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the TLP that caused the error.	Section 2.2.8.6, Section 2.3.1, Section 2.3.2, Section 2.7.2.2, Section 2.9.1, Section 5.3.1, Section 6.2.3.1, Section 6.2.6, Section 6.2.8.1, Section 6.5.7, Section 7.3.1, Section 7.3.3, Section 7.5.1.1, Section 7.5.1.2
Completion Timeout		*Requester:* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.4.	Section 2.8
Completer Abort		*Completer:* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the Request that encountered the error.	Section 2.3.1

Error Name	Error Type (Default Severity)	Detecting Agent Action[74]	References
Unexpected Completion		*Receiver:* Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in Section 6.2.3.2.4.5. Log the prefix/header of the Completion that encountered the error.	Section 2.3.2
ACS Violation		*Receiver (if checking):* Send ERR_NONFATAL to Root Complex. Log the prefix/header of the Request TLP that encountered the error.	
MC Blocked TLP		*Receiver (if checking):* Send ERR_NONFATAL to Root Complex. Log the prefix/header of the Request TLP that encountered the error.	Section 6.14.4
AtomicOp Egress Blocked		*Egress Port:* Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the AtomicOp Request that encountered the error.	Section 6.15.2
Receiver Overflow	Uncorrectable (Fatal)	*Receiver (if checking):* Send ERR_FATAL to Root Complex.	Section 2.6.1.2
Flow Control Protocol Error		*Receiver (if checking):* Send ERR_FATAL to Root Complex.	Section 2.6.1
Malformed TLP		*Receiver:* Send ERR_FATAL to Root Complex. Log the prefix/header of the TLP that encountered the error.	Section 2.2.2, Section 2.2.3, Section 2.2.5, Section 2.2.7, Section 2.2.8.1, Section 2.2.8.2, Section 2.2.8.3, Section 2.2.8.4, Section 2.2.8.5, Section 2.2.9, Section 2.2.10, Section 2.2.10.1, Section 2.2.10.2, Section 2.3, Section 2.3.1, Section 2.3.1.1, Section 2.3.2, Section 2.5, Section 2.5.3, Section 2.6.1, Section 2.6.1.2, Section 6.2.4.4, Section 6.3.2

Error Name	Error Type (Default Severity)	Detecting Agent Action[74]	References
TLP Prefix Blocked		*Egress Port:* Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the TLP that encountered the error.	Section 2.2.10.2

对于上面列出的所有错误，必须在检测到错误后将相应的状态位设置为 1。对于不支持的请求（UR），可以使用额外的检测和

报告使能位（请参阅 Section 6.2.5）。

Device UR Reporting Compatibility with Legacy and 1.0a Software

对于没有实现 Role-Based Error Reporting 的 1.0a 的 device Function，Device Control 寄存器中的 Unsupported Request Reporting Enable 字段（如果清零）会阻止该 Function 发送任何错误消息来表示 UR 错误。使用 Role-Based Error Reporting 时，如果 Command 寄存器中的 SERR# Enable 字段被设置为 1b，则即使清除了 Unsupported Request Reporting Enable 字段，也会隐式使能该 Function 发送 ERR_NONFATAL 或 ERR_FATAL 消息来指示 UR 错误。这引起了与为 1.0a 设备编写的软件（或固件）的向后兼容性问题。

如果软件/固件设置了 SERR# Enable 字段，但 Unsupported Request Reporting Enable 字段和 Correctable Error Reporting Enable 字段，则如果 Non-Posted Request 遇到 UR 错误的 Role-Based Error Reporting 的 Function 将不会发送任何错误消息，如果请求是 Posted，则将通过 ERR_NONFATAL 发出错误信令。Non-Posted Request 的行为支持 PC 兼容的配置空间探测，而 Posted Request 的行为则恢复了与 PCI 和 PCI-X 的错误报告兼容性，从而避免了该区域潜在的数据损坏。因此，Role-Based Error Reporting 的设备向后兼容所设想的旧版和 1.0a 软件和固件。

6.2.7.1 Conventional PCI Mapping

为了支持传统的 PCI 驱动程序和软件兼容性，必须在适当的情况下将 PCI Express 错误条件映射到 PCI 的 Status 寄存器以进行错误报告。

换句话说，当检测到某些 PCI Express 错误时，将设置 PCI 的 Status 寄存器的合适的字段，以向传统 PCI 软件发出错误警报。当 PCI Express 错误导致设置 PCI Status 寄存器时，清除 PCI Status 寄存器不会导致清除 Uncorrectable Error Status 寄存器和 Correctable Error Status 寄存器中的字段。同样，清除 Uncorrectable Error Status 寄存器和 Correctable Error Status 寄存器中的字段也不会清除 PCI Status 寄存器。

PCI Command 寄存器具有控制 PCI 错误报告的字段。但是，PCI Command 寄存器不会影响 PCI Express 错误寄存器各字段的设置。

6.2.8 Virtual PCI Bridge Error Handling

虚拟 PCI 桥配置头与 RC 或 Switch 中的每个 PCI Express 端口相关联。对于这些情况，PCI Express 错误概念要求适当映射到 PCI 错误报告结构。

6.2.8.1 Error Message Forwarding and PCI Mapping for Bridge - Rules

通常，根据适用于 TLP 最终接收者的相同规则，将 TLP 从虚拟 PCI 桥的一侧传递到另一侧，或在桥的入口侧进行处理。以下规则涵盖与 PCI Express 特定的错误相关的情况。有关错误消息控制的概念性总结，请参见 Section 6.2.6。

- 如果 Request 未解决映射到桥内部空间或桥出口侧的空间的问题，则该 Request 会在入口侧终止为 Unsupported Request 。
- 中毒 TLP 按照与非中毒 TLP 相同的规则转发。
  - 向下游转发一个 Poisoned Request 时：
    - Status 寄存器的 Detected Parity Error 字段设置为 1b。
    - 如果 Bridge Control 寄存器的 Parity Error Response Enable 字段为 1b，则 Secondary Status 寄存器的 Master Data Parity Error 字段设置为 1b。
  - 向下游转发一个 Poisoned Completion 时：
    - Status 寄存器的 Detected Parity Error 字段设置为 1b。

公众号·芯片业者的承双

- 如果 Command 寄存器的 Parity Error Response 字段为 1b，则 Status 寄存器的 Master Data Parity Error 字段设置为 1b。

- 向上游转发一个 Poisoned Request 时：
  - Secondary Status 寄存器的 Detected Parity Error 字段设置为 1b。
  - 如果 Command 寄存器的 Parity Error Response 字段为 1b，则 Status 寄存器的 Master Data Parity Error 字段设置为 1b。
- 向上游转发一个 Poisoned Completion 时：
  - Secondary Status 寄存器的 Detected Parity Error 字段设置为 1b。
  - 如果 Bridge Control 寄存器的 Parity Error Response Enable 字段为 1b，则 Secondary Status 寄存器的 Master Data Parity Error 字段设置为 1b。

- 如果将 Bridge Control 寄存器中的 SERR# Enable 字段置 1，则 ERR_COR、ERR_NONFATAL 和 ERR_FATAL 从 secondary interface 转发到 primary interface。转发错误消息的桥不得在 Device Status 寄存器中设置相应的 Error Detected 字段。主接口对转发的错误消息的传输由多个字段控制，如 Figure 6-3 所示。
- 对于根端口，必须通过主接口把错误消息从二级接口转发到主接口以通过 Root Control 寄存器引起一个系统错误，如果实现了 AER，则通过 Root Error Command 寄存器进行报告，并在 Root Error Status 寄存器和 Error Source Identification 寄存器中记录。
- 对于 Root Complex Event Collector（技术上不是桥），从 RC Integrated Endpoint 接收的错误消息必须发送以通过 Root Control 寄存器引起一个系统错误，如果实现了 AER，则通过 Root Error Command 寄存器进行报告，并在 Root Error Status 寄存器和 Error Source Identification 寄存器中记录。

6.2.9 Internal Errors

内部错误是与 PCI Express 接口相关的错误，发生在组件内，并且可能不归因于 PCI Express 接口本身或代表 PCI Express 上发起的事务的数据包或事件。确定什么是内部错误是特定于实现的，不在本规范的范围之内。

内部错误可以分为 Corrected Internal Error 或 Uncorrectable Internal Error。Corrected Internal Error 是在硬件掩盖或解决的组件中发生的错误，没有任何信息丢失或不正确的操作。可能的 Corrected Internal Error 的一个示例是通过纠错码（ECC）纠正的内部数据包缓冲存储器错误。Uncorrectable Internal Error 是在组件内发生的错误，导致该组件的错误操作。 可能的 Uncorrectable Internal Error 的一个示例是无法通过 ECC 纠正的内存错误。从 Uncorrectable Internal Error 中恢复的唯一方法是复位或更换硬件。

Corrected Internal Error 和 Uncorrectable Internal Error 的报告是独立的可选操作。如果报告了任何一个，则必须实现 AER。

报头记录对于 Uncorrectable Internal Error 是可选的。记录报头时，报头是由于 Uncorrectable Internal Error 而丢失或损坏的第一个 TLP 的报头。 如果没有实现报头记录或报头不可用，则会记录所有报头。

可以与特定 PCI Express 接口相关联的内部错误由与该端口相关联的 Function 报告。上游端口报告在 Switch 内检测到的无法与特定 PCI Express 接口关联的内部错误。无法与所有其他 multiPort 组件（例如，RC）中的特定 PCI Express 接口相关联的内部错误的报告不在本规范的范围之内。

公众号：芯片业者相爱双

6.3 Virtual Channel Support

6.3.1 Introduction and Scope

虚通道机制为支持 PCI Express 架构内的差异化服务提供了基础。它可以部署独立的物理资源，并与业务标签一起用于优化处理各种业务。使用 Traffic Class 在 TLP 级别支持业务标签。业务区分策略由 TC/VC 映射以及 VC 仲裁、端口仲裁和 Function 仲裁机制确定。TC/VC 映射取决于平台应用程序要求。这些要求推动了仲裁算法的选择，仲裁器的可配置性/可编程性允许对业务服务策略进行详细调整。

Chapter 2 介绍了虚通道和相关的 Traffic Class 机制的定义。Section 7.11 和 Section 7.18 定义了 VC 配置/编程模型。

本节从系统角度介绍了 VC 机制。它描述了有关以下方面的问题：

- TC/VC 配置。
- VC 仲裁算法和规则。
- 业务流顺序问题。
- 定时传输模型。

6.3.2 TC/VC Mapping and Example Usage

当一个或多个 TC 与 VC ID 指定的物理资源相关联时，便建立了虚通道。结构内给定路径上支持的每个业务类别都必须映射到已启用的虚通道之一。每个端口必须支持默认的 TC0/VC0 映射。任何其他 TC 映射或其他 VC 资源启用都是可选的，并由系统软件使用 Section 7.11 和 Section 7.18 中描述的编程模型进行控制。

组件内分配的 VC 资源的数量或给定结构中使能的 VC 资源的数量会因实现、使用模型、具有不同资源功能的不同组件的热插拔以及系统软件限制可能启用哪些资源等因素而不同。

一些案例如下：

- 一组组件（RC，Endpoints，Switch）可能仅支持强制将 TC0 映射到 VC0。系统软件还可以根据应用程序使用要求，将一个或所有非零 TC 映射到 VC0。
- 一组组件可以支持两个 VC 资源，例如，VC0 和 VC1。系统软件必须实现 TC0/VC0 映射，此外，还可以将一个或所有非零 TC 标签映射到 VC0 或 VC1。如上所述，可以在结构内的任何或所有路径上启用这些映射。有关其他信息，请参考以下示例。
- Switch 可以实现为八个端口——具有两个 VC 资源的七个 x1 链路和具有一个 VC 资源的 x16 链路。系统软件可以启用 x1 链路上的两个 VC 资源，并为一个 VC 分配一个或多个其他 TC，从而允许 Switch 区分任何端口之间的流量。如果 x16 链路的业务要在此链路上流动，则也必须将 x16 链路配置为将任何非 TC0 业务映射到 VC0。注意：多端口组件（Switch 和 RC）每个端口需要独立支持 TC/VC 映射。

在以上示例中，系统软件都具有将 TC 的一个或全部或子集映射到给定 VC 的能力。如果系统软件希望限制可能流经给定链路的业务类别的数量，则它可能仅将 TC 的子集配置为启用的 VC 资源。未指定 TC/VC 映射的任何 TLP 必须视为 Malformed TLP。这称为 TC 过滤。该 TLP 的流控制信用将丢失，并且将生成不可纠正的错误，因此通常需要软件干预才能在 TC 过滤事件发生后恢复正常运行。

TC 过滤的图形示例如 Figure 6-4 所示，其中 TC(2:6)未映射到连接 Endpoint A 和 Switch 的链路。这意味着在 Switch 和 Endpoint A 之间不允许使用带有 TC(2:6)的 TLP。

公众号·芯片业者相承双

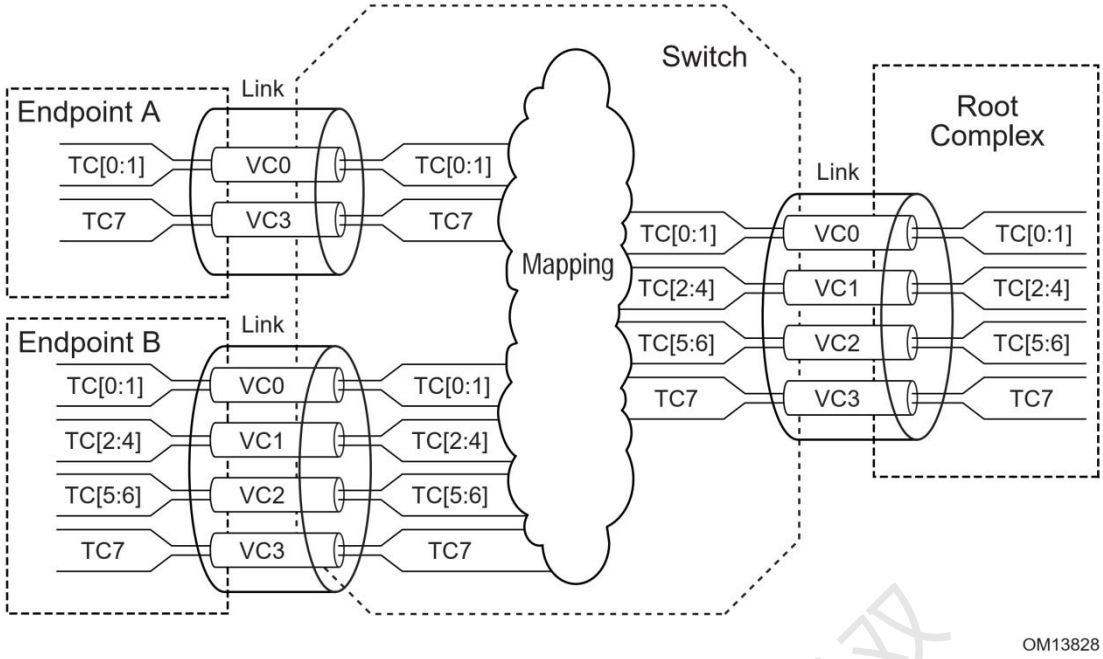

Figure 6-4: TC Filtering Example

Figure 6-5 给出了一个 TC 到 VC 映射的示例。具有一个下游端口和一个上游端口的简单 Switch 将 Endpoint 连接到 RC。在上游端口，两个 VC（VC0 和 VC1）通过以下映射启用：TC(0-6)/VC0，TC7/VC1。在下游端口，仅 VC0 被启用，并且所有 TC 都映射到 VC0。在此示例中，当 TC7 在下游端口映射到 VC0 时，它将重新映射到上游端口的 VC1。尽管端点仅支持 VC0，但当它标记具有不同 TC 的事务时，与 TC7 关联的来自/到 Endpoint 的事务可以利用在 Switch 和 RC 之间启用的第二个虚通道。

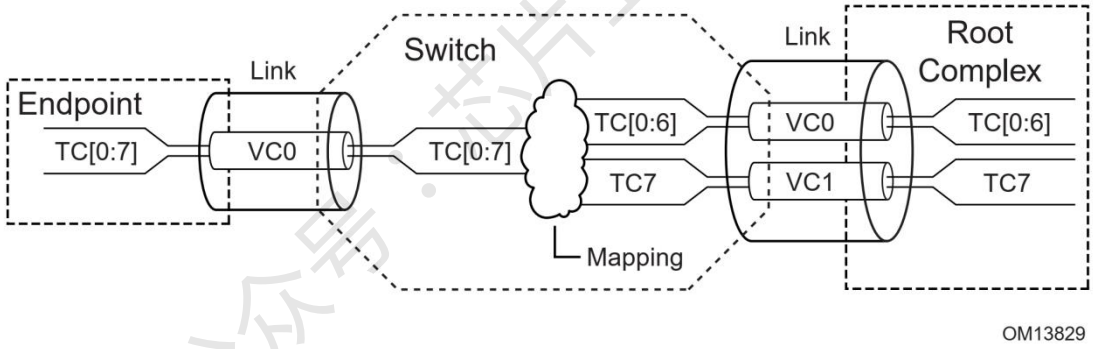

Figure 6-5: TC to VC Mapping Example

Multiple TCs Over a Single VC

单个 VC 实现可能会受益于使用多个 TC。TC 提供了排序域，可用于区分 Endpoint 或 RC 中的业务，而与所支持的 VC 数量无关。

在仅支持 VC0 的简单配置中，由于无法物理隔离不同的 TC，因此无法以最佳方式实现业务区分。但是，仍然可以利用承载多个 TC 的好处，尤其是在小型和“浅”拓扑中，在这些拓扑中，Endpoint 直接连接到 RC，而不是通过级联的 Switch 连接。在这些拓扑中，以 RC 为目标的业务仅需要遍历单个链路，并且基于 TC 的双方（Endpoint 和 RC），包的优化调度可能会比使用单个 TC 的情况显著改善。但是，无法通过具有完全独立的流控制和独立排序功能的单独资源来路由差异化的业务，使所有业务都面临潜在的 head-of-line blocking 情况。优化 Endpoint 内部架构以最大程度地减少阻塞条件的暴露可以降低这些风险。

6.3.3 VC Arbitration

仲裁是虚通道机制的关键之一，并且以特定应用程序可配置的方式进行定义。通常，基于 VC 的仲裁机制的定义受以下目标驱动：

- 为了防止错误的事务超时并确保数据流向前流动。
- 在结构内的数据流之间提供差异化服务。
- 在组件间提供确定性的带宽和尽量小的端到端延迟。

链路是双向的，即每个端口可以是 Ingress Port，也可以是 Egress Port，具体取决于业务的方向。这在 Figure 6-6 中以 3 端口的 Switch 为例进行了说明，其中 Switch 端口之间的业务路径用不同类型的线突出显示。在以下各节中，由于 Switch 从仲裁的角度表示功能超集，因此使用 Switch 仲裁模型定义 VC 仲裁。

另外，在描述中使用单向数据流。

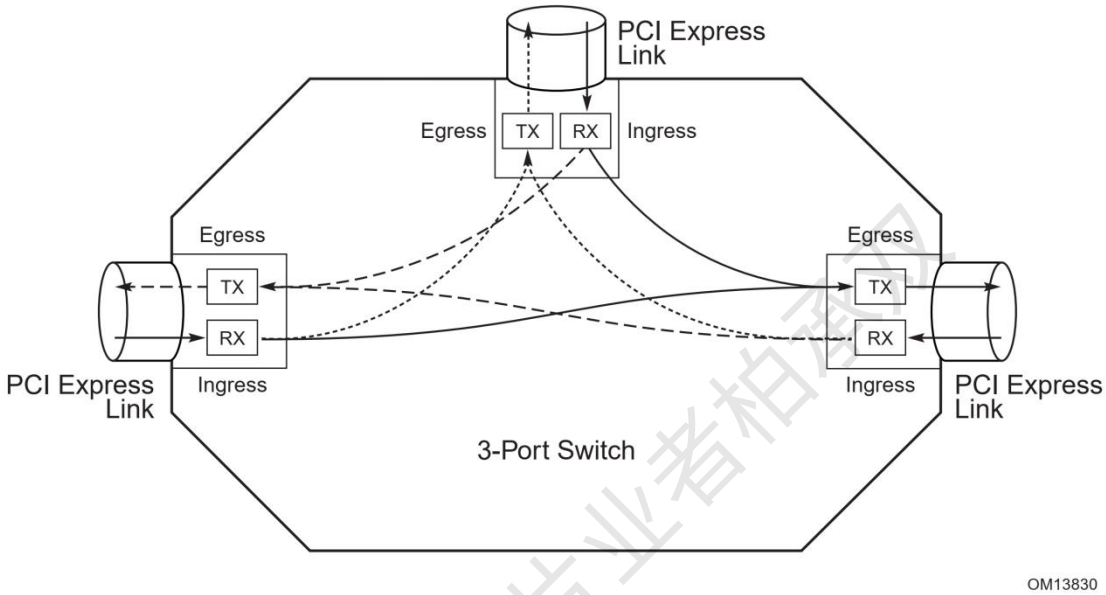

Figure 6-6: An Example of Traffic Flow Illustrating Ingress and Egress

6.3.3.1 Traffic Flow and Switch Arbitration Model

Figure 6-7 和 Figure 6-8 说明了通过 Switch 的业务流，并总结了仲裁的关键点。

公众号：芯片业者相关

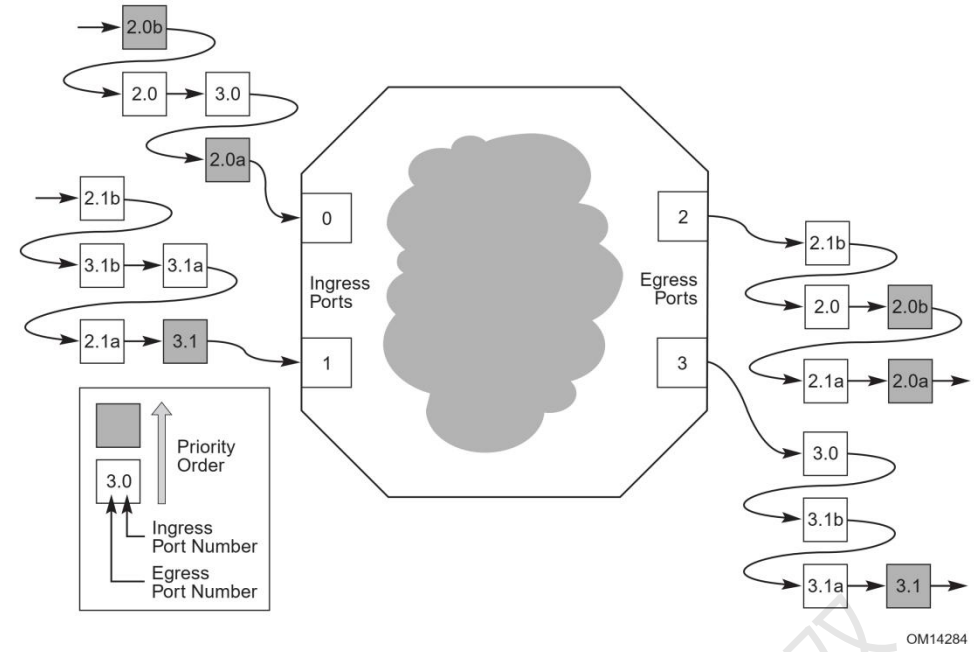

Figure 6-7: An Example of Differentiated Traffic Flow Through a Switch

在每个 Ingress Port，输入业务流在 Figure 6-7 中用小框表示。这些框代表要在不同 VC 中传输的数据包，这些数据包使用不同的灰度等级区分优先级。代表属于不同 VC 的数据包的每个框都包括 Ingress Port 和 Egress Port 的名称，以指示该数据包来自何处以及去向何处。 例如，名称“ 3.0”表示此数据包到达 Port #0 并去往 Port #3。在 Switch 内部，将根据 Switch 内部仲裁机制对数据包进行路由和提供服务。

Switch 仲裁模型定义了 Switch 内所需的仲裁的基础结构和功能。需要此功能来支持一组仲裁策略，这些策略控制来自多个 Ingress Port 的业务流对 Egress Port 的争用。

Figure 6-8 显示了一个 Switch 的概念模型，该模型突出显示了从入口到出口方向的资源和相关功能。请注意，Switch 中的每个端口都可以充当 Ingress Port 和 Egress Port 的角色。因此，此图仅显示了一种 4 端口 Switch 的特定情况，其中在 Port #0 和 Port #1 上具有流入业务流，该业务流将 Port #2 定位为 Egress Port。一个不同的示例可能显示不同的业务流，这意味着 Switch 上的端口具有不同的角色。PCI Express 体系结构允许通过 Switch 进行对等通信，因此，使用同一示例的可能方案可能包括多个单独且同时的入口到出口的业务流（例如，Port 0 到 Port 2 和 Port 1 到 Port 3）。

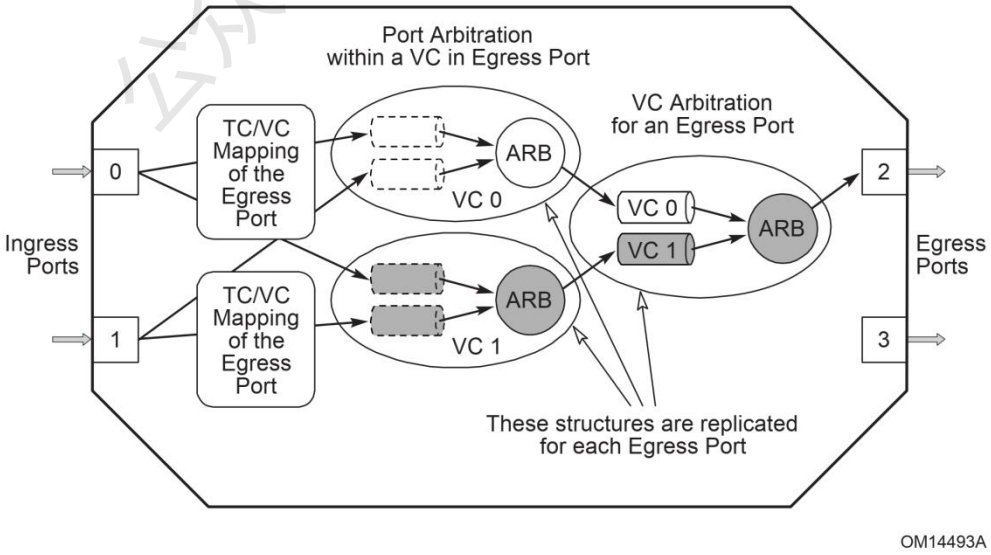

Figure 6-8: Switch Arbitration Structure

以下两个步骤从概念上描述了 Switch 在 Port 0 和 Port 1 上接收并定向到 Port 2 的业务流的路由。首先，基于 TLP header 中的地址/路由信息确定目标 Egress Port。然后，基于 Egress Port 的 TC/VC 映射来确定 Egress Port 的目标 VC。在将事务转发到 Egress Port 中的相应资源之前，必须对以 Egress Port 中相同 VC 为目标但来自不同 Ingress Port 的事务进行仲裁，这个仲裁过程称为端口仲裁（Port Arbitration）。

不同 Ingress Port 收到的 TLP 可能携带相同的 TC 值，这些有相同 TC 值的业务流在 Egress Port 中必须映射到同一个 VC，对这些来自不同 Ingress Port 且具有相同 VC 值的 TLP 的仲裁就是端口仲裁。

一旦业务流到达 Egress Port 中的目标 VC，就必须对共享的链路进行仲裁。从 Egress Port 的角度来看，该仲裁可以在概念上定义为一种简单的复用形式，其中复用控制是基于固定或可配置的仲裁策略。Egress Port 上不同 VC 之间的仲裁称为 Egress Port 的 VC 仲裁（VC Arbitration）。

与 VC 仲裁策略无关，与每个 VC 关联的管理/控制逻辑必须遵守事务排序和流控规则，然后才能对这些业务流进行仲裁。

VC Control Logic at the Egress Port

每个 Egress Port 上的 VC 控制逻辑要实现：

- VC 流控逻辑。
- VC 排序逻辑。

流控制信用只在连接到同一链路的两个端口之间进行交换。流控制信用的可用性是 VC 控制逻辑用来确定何时允许 VC 竞争共享链路资源（即数据链路层发送/重试缓冲区）要考虑的因素之一。如果由于缺少足够数量的流控制信用而无法提交候选数据包，则 VC 控制逻辑必须掩盖待处理数据包的存在，以防止其他 VC 阻塞业务流。请注意，由于每个 VC 都有用于 Posted Requests、Non-Posted Requests 和 Completion TLP 的缓冲资源，因此 VC 控制逻辑还必须考虑到特定候选数据包的流控制信用的可用性。此外，VC 控制逻辑必须遵守 Posted Requests、Non-Posted Requests 和 Completion 事务的排序规则（有关更多详细信息，请参见 Section 2.4），以防止死锁和违反生产者-消费者排序模型。

6.3.3.2 VC Arbitration - Arbitration Between VCs

该规范通过 VC ID 的分配定义了默认的 VC 优先级，即 VC ID 在 Virtual Channel Capability structure 或 Multi-Function Virtual Channel Capability structure 中的相对优先级以升序排列。Figure 6-9 的示例说明了一个端口，该端口支持八个 VC，其中 VC0 被视为最低优先级，而 VC7 被视为最高优先级。

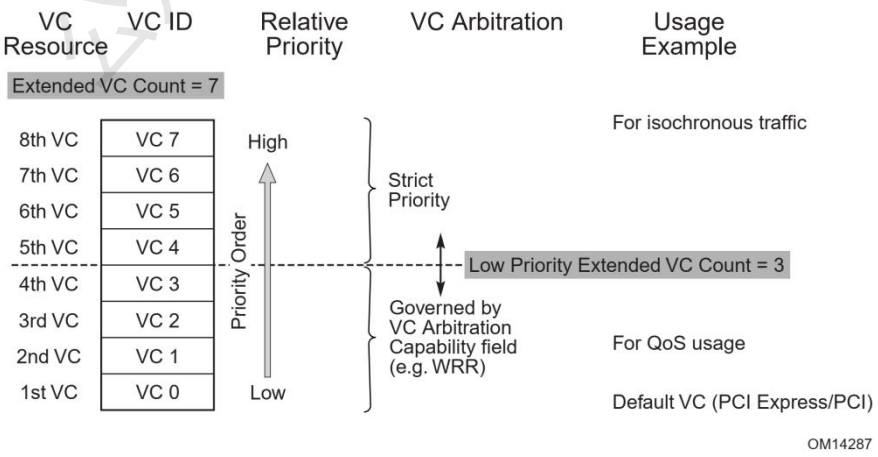

Figure 6-9: VC ID and Priority Order – An Example

常用的 VC 仲裁算法有：

- Strict Priority —— 严格优先级，即 VC0 = lowest，VC7 = highest 。
- Round Robin (RR) —— 轮询调度算法，所有 VC 拥有相同的优先级。
- Weighted RR —— 加权轮询调度算法，每个 VC 有可编程的加权优先级系数。

如果硬件为 VC 资源的子集实现严格优先级仲裁，则软件可以将 VC 配置为两个优先级组——优先级较低的组和优先级较高的组。优先级较高的组被视为严格优先级仲裁组，而较低的组仅在较高的组中没有要处理的数据包时才被仲裁。Figure 6-9 显示了一个示例配置，该配置支持将八个 VC 分为两组——低优先级的组由 VC0-VC3 组成，高优先级的组由 VC4-VC7 组成。可以将低优先级的组中的仲裁配置为支持的仲裁方法之一。Port VC Capability Register 1 中的 Low Priority Extended VC Count 字段指示一个组的大小。仲裁方法在 Port VC Capability Register 2 的 VC Arbitration Capability 字段中列出。有关详细信息，请参见 Section 7.11 和 Section 7.18。当 Low Priority Extended VC Count 字段设置为 0b 时，所有 VC 均遵循严格优先级 VC 仲裁；当该字段等于 Extended VC Count 时，所有 VC 均采用 VC Arbitration Capability 字段指示的 VC 仲裁机制。

6.3.3.2.1 Strict Priority Arbitration Model

严格优先级仲裁可为高优先级事务提供最小的延迟。但是，如果使用不正确则会造成带宽不足的潜在风险。 使用严格优先级要求根据最大峰值带宽和链路使用持续时间来监管所有高优先级流量。必须在事务注入 Port/Function 时或在随后的数据流争用公共链路的 Egress Port 内进行监管。系统软件必须配置业务流，以便以足够的速率为较低优先级的事务提供服务，以避免事务超时。

6.3.3.2.2 Round Robin Arbitration Model

循环仲裁用于为所有业务流提供均等的优先级。注意，该方案用于需要以相同优先级服务不同的无序业务流的情况。注意，这并不意味着在带宽使用方面的等同和公平。

在需要区分的情况下，可以使用加权循环仲裁方案。WRR 方案通常用于以下情况：业务流来源未强制执行带宽调节，因此，如果不使用优先级较低的业务流，就无法使用优先级方案。关键在于，该方案通过允许每个仲裁环路至少赢得一个仲裁获胜，从而在流量争用中提供了公平性。分配的权重在争用期间调节每个 VC 的最小允许带宽和最大突发性。这意味着它限制了来自不同 VC 的业务流的仲裁延迟。请注意，延迟还取决于映射到这些 VC 的业务流所允许的最大数据包大小。

WRR 方案的关键使用模型之一是对 QoS 策略的支持，其中可以使用不同的权重提供不同的 QoS 级别。

尽管可以为某些应用程序固定权重（通过硬件实现），但是要为不同的应用程序提供更通用的支持，建议使用支持 WRR 方案的组件来实现可编程 WRR。使用 Section 7.11 和 Section 7.18 中定义的寄存器控制 WRR 的编程。

6.3.3.3 Port Arbitration - Arbitration Within VC

对于 Switch，端口仲裁是指在映射到相同 VC 的其他 Ingress Port 的业务流之间在 Egress Port 进行的仲裁。 对于 Root Port，端口仲裁是指在来自其他根映射到同一 VC 的 Root Ingress Port 的对等业务流之间在 Root Egress Port 进行的仲裁。对于 RCRB，端口仲裁是指在来自映射到同一 VC 的根端口的业务流之间在 RCRB 上进行的仲裁（例如，对于主机内存）。在这种情况下，VC 之间固有的优先级仲裁方案不适用，因为这意味着对不同端口具有严格的仲裁优先级。可以使用以下支持的方案来仲裁来自不同端口的业务流：

- 基于硬件的固定仲裁策略，如 Round Robin。
- 可编程的 WRR 算法。
- 可编程的基于时间的 WRR 算法。

硬件固定的 RR 或者类似 RR 的方案是最简单的实现方法，因为它不需要任何可编程性。它使所有端口具有相同的优先级，这对于不需要软件管理的差异化或基于每个端口的带宽预算的应用程序是可以接受的。

可编程 WRR 具有灵活性，因为它可以作为平坦的 RR 运行，或者如果需要区分，则可以按照 Section 6.3.3.2 中所述的类似方式，将不同的权重应用于来自不同端口的业务流。该方案用于需要为不同端口提供不同带宽分配的场景。

基于时间的 WRR 用于不仅需要不同带宽分配，而且需要严格控制该带宽使用的应用场景。这种方案可以控制在一定的固定时间内可以从不同端口注入的业务流。对于某些应用（例如等时服务），这是必需的，其中业务流需要满足严格的时间要求。Section 6.3.4 节提供了支持等时应用的基本规则。有关基于时间的仲裁和作为该仲裁方案的使用模型的等时服务的更多详细信息，请参阅 Appendix A。

6.3.3.4 Multi-Function Devices and Function Arbitration

Multi-Function 仲裁模型定义了 Multi-Function 设备中的可选实现的仲裁逻辑和功能。需要此功能来支持一组仲裁策略，这些策略用以解决设备的多个 Function 对 Upstream Egress Port 的业务流争用。

Figure 6-10 给出了 Multi-Function 设备的概念模型，突出显示了资源和相关功能。请注意，每个 Function 可选地包含一个 VC Capability structure，该 structure 可用于管理该 Function 内的 TC/VC 映射、可选的端口仲裁和可选的 VC 仲裁。MFVC Capability structure 可管理设备的 Upstream Egress Port 的 TC/VC 映射、可选的 Function 仲裁和可选的 VC 仲裁。这些资源共同为上行请求启用了增强的 QoS 管理。但是，与 Switch 不同，Multi-Function 设备模型不支持针对 Function 之间的对等请求或下游请求的完整 QoS 管理。

公众号：芯片业者相关

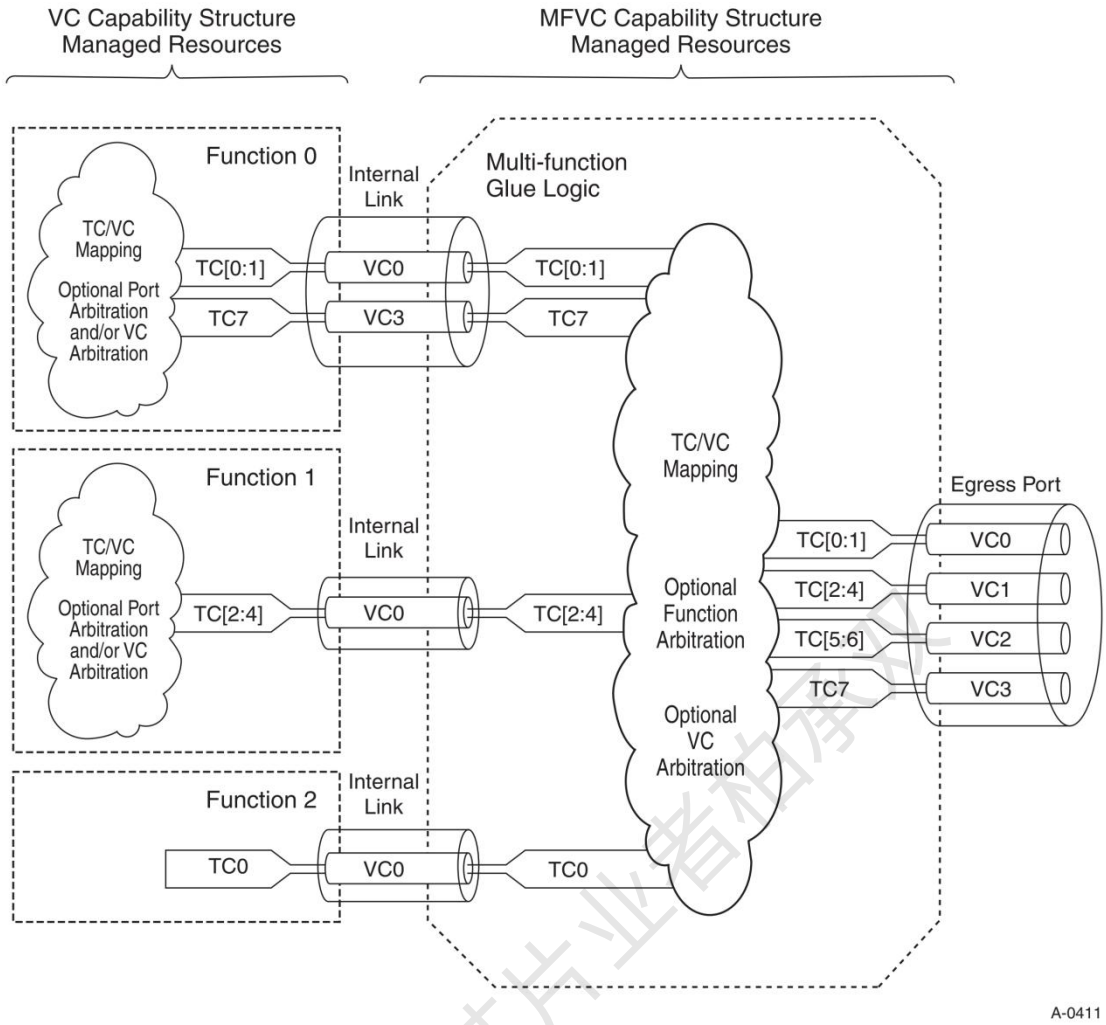

Figure 6-10: Multi-Function Arbitration Model

源自一个 Function 的上行请求的 QoS 按如下方式管理。首先，Function 逻辑将 TC 字段应用于 Request TLP。例如，设备驱动程序可能将 Function 配置为使用 TC7 标记其所有请求。

接下来，如果 Function 包含 VC Capability structure，则用它指定 TC/VC 映射到 Function 的某一 VC。此外，VC Capability structure 支持 Function 的 VC 资源的启用和配置。

如果 Function 是 Switch，并且目标 VC 资源支持端口仲裁，则此机制将控制 Switch 的多个 Downstream Ingress Port 如何为此 VC 资源进行仲裁。如果端口仲裁机制支持基于时间的 WRR，则这还将控制来自每个 Downstream Ingress Port 的请求的注入速率。

如果该 Function 支持 VC 仲裁，则此机制将管理该 Function 的多个 VC 资源如何仲裁与 MFVC 资源的概念性内部连接。

一旦请求数据包从概念上到达 MFVC 资源，TLP 报头中的地址/路由信息将确定请求是上行流还是对等访问到另一个 Function。对于 Function 到 Function 的对等访问，QoS 管理由设备特定机制负责。对于上游的情况，MFVC Capability structure 中的 TC/VC 映射确定请求将针对的 VC 资源。 MFVC Capability structure 还支持 multi-Function 粘合逻辑中 VC 资源的启用和配置。如果目标 VC 资源支持 Function 仲裁，则此机制控制多个 Function 如何为此 VC 资源进行仲裁。如果 Function 仲裁机制支持基于时间的 WRR，则它将控制每个 Function 请求到该 VC 资源的注入率。

最后，如果 MFVC Capability structure 支持 VC 仲裁，则该机制将控制 MFVC 的多个 VC 如何竞争设备的上游出口端口。与 VC 仲裁策略无关，与每个 VC 关联的管理/控制逻辑必须遵守事务排序和流控规则，然后才能对业务流使用仲裁机制。

Multi-Function Devices without the MFVC Capability Structure

如果 Multi-Function 设备未实现 MFVC Capability Structure，则来自 Multi-Function 设备不同 Function 的数据流仲裁不在本规范的范围内。但是，如果 Multi-Function 设备支持 TC0 以外的 TC 但是不实现 MFVC Capability Structure，则它必须在 Function 0 中实现单个 VC Capability structure，才能为链路提供可用的 TC/VC 映射。

6.3.4 Isochronous Support

等时数据传输服务（或叫定时业务）要求系统不仅要保证的数据带宽，而且还要提供确定的服务延迟。定义了等时机制的目的是确保等时业务在相关时间段内获取的分配的带宽，同时还防止系统中其他业务进入饥饿状态。等时机制适用于 Endpoint 与 RC 之间的通信以及对等通信。

等时服务是通过适当使用 TC 事务标记、VC 数据传输协议和 TC 到 VC 映射机制来实现的。端到端等时服务需要软件沿着请求者和完成者之间的路径设置正确的配置。 本节介绍了软件配置规则和硬件组件为提供端到端等时服务必须遵循的规则。有关等时应用程序和等时服务设计准则的更多信息和背景材料，请参见 Appendix A。

6.3.4.1 Rules for Software Configuration

系统软件必须遵守以下规则，以配置 PCI Express 用于等时通信：

- 软件必须为等时事务指定一个或多个 TC。
- 软件必须确保以同一 Completer 为目标的所有等时请求的 Attribute 字段是固定且相同的。
- 软件必须配置用于支持等时业务的所有 VC 资源，以所需的带宽和延迟来满足（仲裁）等时业务，以满足应用程序的目标。这可以通过使用严格的优先级，WRR 或硬件固定的仲裁来实现。
- 在给定的 VC 上，软件不应将等时业务与非等时业务混合在一起（意思是最好一个等时业务分配一个 VC）。
- 软件必须遵守端口或 RCRB 报告的最大时隙功能。
- 软件不能将所有链路带宽都分配给等时业务。这是确保其他非等时业务正常转发，以避免错误的事务超时。
- 软件必须限制每个支持等时的路径的 Max_Payload_Size 以满足等时延迟。例如，在从具有等时能力的设备到 RC 的路径上流动的所有业务流都应限于不超过满足等时延迟要求的 Max_Payload_Size 的数据包。
- 软件必须将有等时配置设备的 Max_Read_Request_Size 设置为不超过为设备设置的 Max_Payload_Size 的值。

6.3.4.2 Rules for Requesters

请求者的等时服务需要遵循以下规则：

- 读请求的 Length 值不能超过 Max_Payload_Size。
- 如果同步业务以 RC 为目标，并且 RCRB 指示在没有将所有事务的 No Snoop 字段（由设置 Reject Snoop Transactions 位指示）设置为 1b 的情况下它不满足等时带宽和等待延迟，而，则必须在 TLP header 内设置此位，否则事务将被拒绝。

6.3.4.3 Rules for Completers

完成者的等时服务需要遵循以下规则：

- 在正常操作条件下，完成者不应将流控制引起的反压应用于均匀注入的等时请求。

公众号：芯片业者拍承双

- 完成者必须在 VC Resource Capability 寄存器的 Maximum Time Slots 字段中报告其等时带宽能力。 请注意，完成者必须考虑部分写入。
- 完成者必须遵守最大等时事务延迟。
- RC 作为完成者必须至少实现一个 RCRB，并为关联的 VC 支持基于时间的端口仲裁。请注意，基于时间的端口仲裁仅适用于请求事务。

6.3.4.4 Rules for Switches and RCes

提供等时服务的 Switch 必须遵守以下规则。相同的规则适用于支持根端口之间对等传输的等时数据流的 RC，在本节中简称为“ P2P-RC”。

- 支持等时配置的 Switch 或 P2P-RC 端口在正常操作条件下不应将流控制引起的反压应用于均匀注入的等时请求。
- 支持等时配置的 Switch 或 P2P-RC 端口必须遵守最大等时事务延迟。
- Switch 或 P2P-RC 组件必须支持基于时间的端口仲裁。请注意，基于时间的端口仲裁适用于请求事务，但不适用于完成事务。

6.3.4.5 Rules for Multi-Function Devices

包含提供等时服务的 MFVC Capability structure 的 Multi-Function 设备必须遵守以下规则：

- 配置为等时运行的 MFVC 胶合逻辑（glue logic）不应在正常工作条件下对来自其 Function 的均匀注入的等时请求施加反压。
- MFVC Capability structure 必须为每个能够支持等式业务的 VC 支持基于时间的 Function 仲裁。 请注意，基于时间的 Function 仲裁仅适用于上游请求事务。它不适用于任何下游或对等请求事务，也不适用于任何完成事务。

缺少 MFVC Capability structure 的 Multi-Function 设备没有架构机制可同时为其多个 Function 提供等时服务。

6.4 Device Synchronization

系统软件需要一种“stop”机制，以确保系统中特定设备没有 outstanding transaction。例如，如果没有这种机制，则在系统运行期间对 Bus Number 重新编号可能会导致给定设备的 Requester ID（包括 Bus Number）在该设备的请求或完成仍在运行中时发生更改，因此可能由于 Requester ID 改变而变得无效。还希望能够确保在热插拔有序移除期间没有 outstanding transaction。

Stop 机制的实现取决于设备硬件、设备驱动程序和系统软件。但是，允许系统软件管理的结构的基本需求要包括以下能力：

- 能阻塞设备生成新的 Request。
- 能阻塞发到该设备的 Request 的生成。
- 能确定设备正在处理的所有 Request 均已完成。
- 能确定设备发出的所有 non-posted Request 均已完成。
- 能确定设备发出的所有 posted Request 均已到达目标设备。

驱动软件和/或系统软件可以利用每个设备的 Function 的 Command 寄存器（Section 7.5.1.1）中的 Bus Master Enable、SERR# Enable 和 Interrupt Disable 字段以及类似控制字段来阻塞新的 Request。

发送给设备的请求通常在驱动程序的直接控制下，因此系统软件可以通过指示驱动程序停止生成请求来阻止这些请求（此通信的详细信息是系统软件特定的）。同样，由设备提供服务的请求通常在设备驱动程序的控制下，因此确定此类请求的完成通常

很简单。

Transactions Pending 字段为每个 Function 提供了一种一致的方式，使软件可以确定设备发出的所有 non-posted Request 均已完成（请参见 Section 7.8.5）。

通过生成一个 transaction 来刷新任意 outstanding Request 可以确定 posted Request 是否到达目的地。使用 TC0 写系统内存的操作可以通过设备的主机读进行刷新，因此不需要显式的刷新协议。使用 TC0 以外的 TC 进行写系统内存需要某种类型的刷新同步机制。该机制本身的实现是特定于设备和其驱动程序的。但是，在所有情况下，设备硬件和软件实现者都应彻底理解 Section 2.4 中描述的排序规则。特别是在设备发起的任何请求中设置了 Relaxed Ordering 或 ID-Based Ordering 属性的情况下要尤为注意。

Flush Mechanisms

在一个简单的情况下，例如 Endpoint 仅通过 TC0 与主机内存通信，可以通过从 Endpoint 进行读来简单地实现"刷新"。如果 Endpoint 使用非 TC0 的 TC 写主存，则可以发射相应 TC 上的读主存来实现"刷新"。内存读需要在 Endpoint 使用的所有 TC 上执行。

如果使用内存读来"刷新"outstanding transaction，但是不需要实际读，则可能需要使用 Section 2.2.5 中描述的 zero-length read 语义。

设备之间的点对点交互，即使所有通信都是通过 TC0 进行的，也需要一个显式同步协议。 对于给定的系统，必须建立用于管理对等交互的模型。然后，系统软件以及设备硬件和软件必须符合此模型。阻止请求生成和确定请求完成的要求与非对等交互的要求相匹配，但是确定 Posted Request 已到达对等目标设备需要明确的同步机制。该机制本身的实现是特定于设备、其驱动程序以及用于建立和取消对等通信的模型的。

6.5 Locked Transactions

6.5.1 Introduction

锁定事务用于防止使用旧版软件的系统中的死锁导致对 I/O 设备的访问。请注意，某些 CPU 可能会因为执行隐式触发锁定的指令而导致生成一个锁定访问。即使不需要独占访问，某些旧版软件也会滥用这些事务并生成锁定序列。由于对 I/O 设备的锁定访问会导致潜在的死锁，而且还会导致严重的性能下降，因此禁止 PCI Express Endpoint 支持锁定访问，并且新软件不得使用会导致对 I/O 设备进行锁定访问的指令。Legacy Endpoint 仅支持与现有软件兼容的锁定访问。

只允许 RC 在 PCI Express 上发起锁定请求。不支持由 Endpoint 和 Bridge 发起的锁定请求。这与 *PCI Local Bus Specification*（Appendix F- Exclusive Accesses）中描述的锁定事务的使用规则一致。

本节描述主机 CPU 到 Legacy Endpoint 的锁定访问相关的规则，包括这些事务通过 Switch 和 PCI Express/PCI Bridge 时的规则。

6.5.2 Initiation and Propagation of Locked Transactions - Rules

锁定事务序列是由主机 CPU 生成的对同一位置的一次或多次读后跟一次或多次写。锁定后，将阻止所有其他业务使用 RC 和锁定的 Legacy Endpoint 或 Bridge 之间的路径。

- 锁定事务序列的 TLP 类型为 Read Request/Completion (MRdLk/CplDLk)并且用 Unlock Message 终止。
  - 以 Successful Completion 以外的状态完成的 Locked Request 不会建立锁定（在以下各节中有详细说明）。
  - 无论锁定序列的完成状态如何，所有锁定序列和尝试的锁定序列都必须通过发送 Unlock Message 来终止。
  - MRdLk、CplDLk 和 Unlock 只会以默认的 TC0 的方式发送。
  - 在单个层次结构域中，在给定时间只能进行一次锁定事务序列的尝试。

- Unlock Message 是从 RC 沿着锁定的事务路径发送到完成者的，并且可以从 RC 广播到所有 Endpoint 和 Bridge。
  - 锁定序列未涉及的设备都必须忽略此消息。
- 任何违反锁定事务发送和传播规则的行为都可能导致未定义的设备和/或系统行为。

通过 PCI Express 发送和传播锁定事务序列的过程如下：

- 一个锁定事务序列以一个 MRdLk Request 开始。
  - 锁定事务序列的任何后续读也必须使用 MRdLk Request。
  - 任何 MRdLk Request 的 Completion 都将 CplDLk 用于成功的请求，并将 CPlLk 用于不成功的请求。
- 如果与锁定序列相关的任何读未成功完成，则请求者必须假定不再确保锁定的原子性，并且不再锁定请求者和完成者之间的路径。
- 锁定序列的所有写操作均使用 MWr Request。
- Unlock Message 用来表示锁定序列的结束。
  - Switch 将 Unlock Message 传播到锁定的 Egress Port。
- 在收到 Unlock Message 后，如果 Legacy Endpoint 或 Bridge 处于锁定状态，则必须对其自身进行解锁。
  - 如果未锁定，或者如果接收器是不支持锁定的 PCI Express Endpoint 或 Bridge，则将忽略并丢弃 Unlock Message。

6.5.3 Switches and Lock - Rules

Switch 必须能将锁定序列事务与其他事务区分开，以防止其他事务干扰锁定事务并可能导致死锁。以下规则涵盖了此操作。请注意，锁定访问仅限于 TC0，它始终映射到 VC0。

- 当 Switch 将 MRdLk Request 从 Ingress Port（最靠近 RC）传播到 Egress Port 时，它必须阻止所有映射到默认虚通道（VC0）的请求向 Egress Port 传播。
  - 如果在此 Ingress Port 处寻址到另一个 Egress Port 的后续 MRdLk 请求，则 Switch 的行为未定义。

  Note：PCI Express 不支持这种拆分锁定访问，并且软件不得导致这种锁定访问。此类访问可能导致系统死锁。
- 当返回第一个 MRdLk Request 的 CplDLk 时，如果完成状态为 Successful Completion，则 Switch 必须阻止来自所有其他端口的所有请求传播到锁定访问所涉及的任一端口，但映射到 non-VC0 的 Egress Port 的 Request 除外。
- 锁定序列中涉及的两个端口必须如上所述保持阻塞状态，直到 Switch 接收到 Unlock Message（指在最初 MRdLk Request 的 Ingress Port 上）。
  - Unlock Message 必须转发到锁定的 Egress Port 上。
  - Unlock Message 有可能会广播到所有端口上。
  - 一旦 Unlock Message 到达，Ingress Port 解除阻塞，并且在从 Egress Port 中发出 Unlock Message 之后，被阻塞的 Egress Port 解除阻塞。
    - 未参与锁定访问的端口不受 Unlock Message 的影响。

公众号·芯片业者相承双

6.5.4 PCI Express/PCI Bridges and Lock - Rules

PCI Express/PCI Bridge 的要求与 Switch 相似，不同之处在于，由于 PCI Express/PCI Bridge 仅使用默认的虚通道和业务类别，因此在锁定访问期间将阻止所有其他业务。PCI Express/PCI Bridge 的 PCI 总线侧的要求与 PCI/PCI Bridge 的要求相匹配（请参阅 *PCI-to-PCI Bridge Architecture Specification, Revision 1.2* 和 *PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0*）。

6.5.5 RC and Lock - Rules

允许 RC 作为请求者支持锁定事务。如果支持锁定事务，则 RC 必须遵循 Section 6.5.2 描述的顺序来执行锁定访问。RC 用于将 PCI Express 连接到主机 CPU 的机制不在本文档范围之内。

6.5.6 Legacy Endpoints

Legacy Endpoint 可以支持锁定访问，虽然不建议这么做。如果 Legacy Endpoint 支持锁定访问，则它必须按以下方式处理它们：

- 当第一个锁定访问的 Read Request 返回了一个 Successful Completion 的 Completion，则 Legacy Endpoint 变成锁定状态。
  - 如果完成状态不是 Successful Completion，则 Legacy Endpoint 不会被锁定。
  - 一旦锁定，Legacy Endpoint 必须保持锁定，直到收到 Unlock Message。
- 锁定时，Legacy Endpoint 不得使用映射到默认虚通道（VC0）的 TC 发出任何请求。

  请注意，如果有多个可能的请求源，则此要求适用于 Endpoint 内所有可能的请求源。
  - 可以使用映射到默认虚通道以外的 VC 的 TC 发出请求。

6.5.7 PCI Express Endpoints

PCI Express Endpoint 不支持锁定。PCI Express Endpoint 必须将 MRdLk Request 视为 Unsupported Request（请参阅 Chapter 2）。

6.6 PCI Express Reset - Rules

本节介绍 PCI Express 复位机制。 本节介绍本文档中定义的体系结构机制与本文档中定义的复位机制之间的关系。PCI Express 常规复位与组件或平台复位之间的任何关系都是特定于组件或平台的（除非明确说明）。

6.6.1 Conventional Reset（常规复位）

Function-Level Reset 以外的所有复位机制都是常规复位。有两类常规复位：基本复位（Fundamental Reset）和非基本复位（non-Fundamental Reset）。

在所有组成形式和系统硬件配置中，都必须在某种程度上具有一种硬件机制，用于将所有端口状态设置或恢复为本文档中指定的初始条件–该机制称为"基本复位"。该机制可以采取系统提供给组件或适配卡的辅助信号的形式，该信号即所谓的 PERST＃，它必须符合 Section 4.2.4.8.1 指定的规则。当将 PERST＃提供给组件或适配器时，该信号必须由组件或适配器用作基本复位。如果未将 PERST＃提供给组件或适配器，则组件或适配器将自动生成基本复位，有关此操作的详细信息不在本文档的讨论范围之内。如果组件或适配器自动生成了基本复位，并且平台通过电源为组件/适配器供电，则如果所提供的功率超出了为设备/适配器指定

公众号：芯片业者相承双

的限制，则组件/适配器必须向其自身生成基本复位。

- 有三种不同的常规复位：cold、warm 和 hot。
  - Cold Reset 指的是在上电时必须要产生的一个基本复位。
  - 在某些情况下，基本复位机制可能由硬件触发，而无需移除组件并重新施加电源，这称为 Warm Reset。该规范没有规定产生 Warm Reset 的方法。
  - 通过带内机制在链路上传递复位信息导致设备复位的方式称为 Hot Reset（see Section 4.2.4.8.2）。

  有一种带内机制可以用软件将链路强制为电气空闲，从而“disabling”链路。Disable 状态在 Section 4.2.5.9 中描述，而 Link Disable 控制位在 Section 7.8.7 中描述。禁用了会导致下游组件进行 Hot Reset。

  还要注意，数据链路层报告 DL_Down 某些方面与 Hot Reset 相同（see Section 2.9）。
- 任意类型的常规复位（cold、warm 和 hot）在退出时，除 Section 7.4 定义的 sticky registers 外所有端口寄存器和状态机都将恢复为本文档中指定的初始值。

请注意，从设备角度来看，任何类型的常规复位（冷复位，暖复位，热复位或 DL_Down）在事务层及更高层均具有与常规 PCI 中的 RST # 相同的效果。

- 在退出基本复位时，物理层会尝试重新建立链路（see Section 4.2.5）。一旦链路上的两个组件都进入初始链路训练状态后，它们将继续进行物理层的链路初始化，然后进行 VC0 的流控制初始化，从而使数据链路层和事务层可以使用该链路。
  - 在 VC0 的流控制初始化之后，可以通过链路传输 TLP 和 DLLP。

常规复位后，某些设备可能需要更多时间才能响应收到的请求。特别是对于配置请求，组件和设备必须以确定性的方式运行，这是以下规则要解决的。

第一组规则解决了组件和设备的要求：

- 组件必须在基本复位结束后的 20 ms 内进入 LTSSM Detect 状态（Section 4.2.4 介绍了链路训练）。
  - Note：在某些系统中，链路上的两个组件有可能在不同的时间退出基本复位。每个组件都必须从自己的角度出发，在基本复位结束后 20ms 内遵守进入初始活动链路训练状态的要求。
- 在完成链路训练（进入 DL_Active 状态，see Section 3.2）后，组件必须能够接收和处理 TLP 和 DLLP。

第二套规则解决了对系统的要求：

- 为了允许组件执行内部初始化，系统软件必须在一个或多个设备的常规复位结束后等待指定的最短时间，然后才允许向这些设备发出配置请求。
  - 对于不支持大于 5.0 GT/s 的链路速度的 Downstream Port，软件必须等待至少 100ms，然后才能向该端口挂的设备发送配置请求。
  - 对于支持链路速度大于 5.0 GT/s 的 Downstream Port，在链路训练完成后，软件必须至少等待 100ms，然后才能向该端口挂的设备发送配置请求。软件可以通过检测 Data Link Layer Link Active 字段或通过设置相关的中断来确定链路训练何时完成（see Section 6.7.3.3）。
  - 系统必须确保在启动时对软件可见的组件都准备就绪，可以根据 RC 上的常规复位的结束，在适用的最短时间内接收配置请求-该操作超出了本规范的范围。
  - Note：仅当软件启用 CRS Software Visibility 时，软件才应等待 100 ms 时间。否则，可能会导致 Completion timeout，platform timeout 或冗长的处理器指令停顿。See the Configuration Request Retry Status Implementation Note in Section

公众号·芯片业者相随双

2.3.1 。

- RC 和/或系统软件必须在设备进行常规复位后至少等待 1.0 s，然后才能确定未能为有效配置请求返回 Successful Completion 状态的设备是损坏的设备。此时间段与链路训练的完成速度无关。
  - Note：此延迟类似于为 PCI/PCI-X 指定的 T_{rhfa} 参数，旨在为需要自我初始化的设备留出足够的时间。
- 尝试对 PCI Express/PCI（-X）桥上挂的 PCI 或 PCI-X 总线上的设备进行配置访问时，必须遵守时序参数 T_{rhfa} 的规范。

对于第二组规则，如果系统软件无法直接查看基本复位状态（例如热插拔；see Section 6.7），则软件必须在基本复位结束后基于这些时序参数来知道发生了什么事件。

当链路处于正常操作阶段，则需要满足以下规则：

- 如果出于某种原因，正常运行的链路断开，则事务层和数据链路层将进入 DL_Inactive 状态（请参见 Section 2.9 和 Section 3.2.1）。
- 对于任何根端口或 Switch 的下游端口，将 Bridge Control 寄存器的 Secondary Bus Reset 字段置 1 必须导致发送 Hot Reset（see Section 4.2.4.8.2）。
- 对于 Switch，下述操作会导致在所有的 Downstream Port 上发送 Hot Reset：
  - 将与对一个 Upstream Port 对应的 Bridge Control 寄存器的 Secondary Bus Reset 字段置 1。
  - Upstream Port 的数据链路层报告一个 DL_Down 状态。在支持链路速率大于 5.0 GT/s 的 Switch 中，Upstream Port 必须将每个 Downstream Port 的 LTSSM 引导到 Hot Reset 状态，但不能将 LTSSM 保持在该状态。这样允许每个 Downstream Port 在其 Hot Reset 完成后立即开始链路训练。建议所有 Switch 使用此行为。
  - Upstream Port 收到一个 Hot Reset。

本文档中描述了基本复位的某些方面，而其他方面则与特定平台、组成因素与特定实现有关。特定平台、组成因素与应用层接口可能需要对基本组件在系统组件之间的时序和/或顺序关系进行附加说明。例如，可能需要底座中的所有 PCI Express 组件同时（在一定的容差范围内）检测基本复位的复位与释放。在多底座环境中，可能需要指定包含 RC 的哪一个底座是最后一个退出基本复位的底座。

在提供电源和 PERST # 的情况下，必须定义以下参数：

- T_{pvperl} ——PERST # 必须在电源有效后至少保持有效这么长时间。
- T_{perst} ——当 PERST # 有效时，它必须至少保持这么长时间。
- T_{fail} ——PERST # 必须在电源失效后至少保持有效这么长时间。

可以指定其他参数。

在提供参考时钟的情况下，必须定义以下参数：

- $T_{perst-clk}$ ——PERST # 必须在任何提供的参考时钟稳定之后至少保持有效这么长时间。

可以指定其他参数。

6.6.2 Function-Level Reset (FLR)

FLR 机制可以使用软件在 Function-Level 的粒度来复位 Endpoint 硬件。以下三个示例的用法模型说明了此功能的好处：

- 在某些系统中，控制 Function 的软件实体可能会停止正常运行。为防止数据损坏，必须停止该 Function 正在执行的所有 PCI

Express 和外部 I/O（不是 PCI Express）操作。定义的其他复位操作不能保证停止外部 I/O 操作。

- 在一个大规模的并行处理系统中，系统软件使用分区的概念管理所有硬件资源，包括处理器资源和所有 IO 资源，这些 IO 资源中通常会包含 PCIe 设备。在这种处理器系统中，任务在指定的分区中运行，当这个任务执行完毕后，系统软件需要调整硬件资源的分区。此时受到影响的 PCIe 设备需要使用 FLR 方式复位内部的逻辑，以免造成对新的分区的资源污染，并保护之前任务的结果。
- 当系统软件拆除某个 Function 的软件堆栈然后重建该堆栈时，有时必须在重建该 Function 的软件堆栈之前使状态返回到未初始化状态。

FLR 的实现是可选的，但强烈建议实现这个功能。

FLR 应用于每个 Function，FLR 操作仅影响目标 Function。链路状态不受 FLR 的影响。

FLR 修改了本规范描述的 Function 的状态，如下所示：

- 必须将 Function 寄存器和特定于 Function 的状态机设置为本文档中指定的初始化值，以下各项除外：
  - sticky 类型的寄存器（ROS, RWS, RW1CS）。
  - Hwlnit 类型的寄存器。
  - 其他寄存器或字段：
    - Device Capabilities 的 Captured Slot Power Limit Value 字段。
    - Device Capabilities 的 Captured Slot Power Limit Scale 字段。
    - Device Control 的 Max_Payload_Size 字段。
    - Link Control 的 Active State Power Management (ASPM) Control 字段。
    - Link Control 的 Read Completion Boundary (RCB) 字段。
    - Link Control 的 Common Clock Configuration 字段。
    - Link Control 的 Extended Synch 字段。
    - Link Control 的 Enable Clock Power Management 字段。
    - Link Control 的 Hardware Autonomous Width Disable 字段。
    - Link Control 2 的 Hardware Autonomous Speed Disable 字段。
    - Link Status 2 的 Link Equalization Request 字段。
    - Secondary PCI Express Extended Capability 的 Link Equalization Control 寄存器。
    - Virtual Channel Capability 的所有寄存器。
    - Multi-Function Virtual Channel Capability 的所有寄存器。

注意，清除了使 Function 能够在 PCI Express 上发起请求的控件，包括 Bus Master Enable，MSI Enable 等，从而有效地使 Function 在链路上变为静默状态。

请注意，与链路功能关联的端口状态机（包括物理和数据链路层中的端口状态机）不会被 FLR 复位，并且 VC0 在 FLR 之后仍保持初始化状态。

- 在启动 FLR 之前，必须通过发送相应的 Deassert_INTx 消息将 Function 置起的任何 INTx 中断置为无效。

公众号：志先业者相承双

请注意，当 FLR 初始化 multi-Function 设备的一个 Function 时，如果另一个 Function 继续触发对应的 INTx，则不会发送 Deassert_INTx 消息。

软件可以通过查询 Device Capability Register 来查询该 PCIe 设备是否支持 FLR。

通过向 Initiate Function Level Reset 字段写入 1b 来启动 FLR 之后，该 Function 必须在 100 ms 内完成 FLR。 如果在 Transactions Pending 字段为 1b 时软件启动了 FLR，则软件必须等待足够的时间使任何关联的传输完成，或者有合理的把握确保任何剩余的事务确实无法完成，才可以初始化该 Function 。因此，建议软件为设备上的 Completion Timeout 所允许的时间尽可能多于 pre-FLR 值所提供的时间。如果在发出 FLR 时禁用了 Function 的 Completion Timeout，则延迟取决于系统，但必须不少于 100 ms。

注意，在接收到 FLR 时，设备 Function 可以清除所有事务状态，包括 Transactions Pending，或者将 Completion Timeout 设置为其默认值，以便所有待处理事务在 FLR 执行期间都将超时。无论如何，必须在 FLR 完成后清除 Transactions Pending 字段。

由于 FLR 修改了本规范未描述的 Function 状态（除了本规范所描述的状态之外），因此有必要使用一组标准来指定 FLR 的行为，这些标准应用于 Function 时，表明该 Function 满足 FLR 的要求。必须使用 Function-specific 的知识来应用以下条件，以评估 Function 对 FLR 的响应：

- 该 Function 不得在该 Function 控制的任何外部接口上暴露带有活动主机的初始化适配器。终止外部接口上的活动所需的步骤超出了本规范的范围。
  - 例如，网络适配器不得响应要求主机系统进行适配器初始化或与活动主机系统进行交互的查询，但可以执行设计为执行的操作而无需主机初始化或进行交互。如果网络适配器包括在同一外部网络接口上运行的多个功能，则此规则仅影响与由 FLR 复位的特定功能相关的那些方面。
- Function 必须将任何软件可读取的状态（可能包括加密信息等）打乱，换句话说，任何内部存储都必须被清零或者随机化。分配给 Function 的主机内存不得由 Function 修改。
  - 例如，具有内部存储器可被主机软件直接或间接读取的 Function 必须清除或随机分配该存储器。
- Function 必须可以被另一个 Device 配置为一般模式以工与该 Function 相关联的驱动程序使用。

启用 FLR 时，该 Function 必须具有以下行为：

- Function 必须为启动 FLR 的操作返回一个完成状态，然后启动 FLR。
- 如果正在执行 FLR：
  - 如果 Request 到达，则允许该 Request 被静默丢弃（在更新流控制信用之后），而无需将其记录为错误或用信号通知错误。
  - 如果 Completion 到达，则允许将该 Completion 作为意外完成处理或被静默丢弃（在更新流量控制信用之后），而无需将其记录为错误或用信号通知错误。
  - 虽然要求某个 Function 在上述时限内完成 FLR 操作，但随后的特定于 Function 的初始化序列可能需要更多时间。如果需要更多时间，则在上述时间限制后收到配置请求时，Function 必须返回 Configuration Request Retry Status (CRS) Completion Status。Function 以 CRS 以外的完成状态响应配置请求后，除非再次将其复位，否则不允许返回 CRS。

6.7 PCI Express Hot-Plug Support

PCI Express 结构旨在原生支持适热插拔，PCI Express 提供一种“toolbox”机制，这些机制允许使用自洽的基础结构来支持不同的用户/操作员模型。本节定义了热插拔机制集，并指定了在系统中实现热插拔的设备必须遵循的规则。

公众号·志毕业者相逢双

6.7.1 Elements of Hot-Plug

Table 6-6 列出了本规范中为支持热插拔模型所需要的物理元件。必须有一个 form factor 规范定义如何使用这些元件。对于给定的 form factor 规范，可能仅需要一些可用的热插拔元件，甚至不需要这些元件。在所有情况下，form factor 规范都必须通过选择包含的元件来定义对系统或用户的所有假设和限制。旨在仅与选定 form factor 规范一起使用的硅组件实现方式仅允许支持相关 form factor 规范所需的那些元件。

Table 6-6: Elements of Hot-Plug

Element	Purpose
Indicators	Show the power and attention state of the slot
Manually-operated Retention Latch (MRL)	Holds adapter in place
MRL Sensor	Allows the Port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of adapter from slot
Attention Button	Allows user to request hot-plug operations
Software User Interface	Allows user to request hot-plug operations
Slot Numbering	Provides visual identification of slots
Power Controller	Software-controlled electronic component or components that control power to a slot or adapter and monitor that power for fault conditions

6.7.1.1 Indicators （指示灯）

规范定义两个 indicators：一个 Power Indicator 和一个 Attention Indicator。他们有三种状态：ON、OFF 和 Blinking（闪烁）。热插拔系统软件通过写入与指示灯相关的命令寄存器来独占控制指示灯的状态（以下有一个例外）。

具有热插拔功能的端口控制指示灯的闪烁频率，占空比和相位。闪烁频率必须在 1 到 2 Hz 的频率下运行，占空比为 50％（+/- 5％）。端口之间的闪烁频率不需要同步或同相。

指示灯可能物理上位于底座或适配器上（有关指示灯的位置，请参阅相关的规格说明）。无论物理位置如何，指示灯的逻辑控制都是通过链路上上游组件的下游端口进行的。

除非软件发出命令，否则下游端口不得更改指示器的状态，但能够检测卡死（stuck-on）电源故障的平台除外（必须要实现电源控制器）。如果发生电源卡死故障，则允许平台覆盖下游端口并强制使电源指示灯亮（以指示不应卸下适配器）。系统软件对卡死故障的处理是可选的，因此在本规范中未进行描述。因此，平台供应商必须确保通过其他软件，平台手册或其他方式解决该问题。

6.7.1.1.1 Attention Indicator

Attention Indicator 必须是黄色或琥珀色的灯，用来指示出现了操作问题，或者表示热插拔已经被识别到，以便操作人员可以轻松识别它。

Table 6-7: Attention Indicator States

Indicator Appearance	Meaning
Off	Normal - Normal operation
On	Attention - Operational problem at this slot
Blinking	Locate - Slot is being identified at the user's request

Attention Indicator Off

Attention Indicator 处于 Off 状态，表示适配器（如果有）和热插拔插槽均不需要被关注。

Attention Indicator On

Attention Indicator 处于 On 状态，表示适配器或插槽存在操作问题。

操作问题指的是阻止适配器继续工作的意思。 操作系统或其他系统软件确定特定条件是否阻止适配器继续工作以及点亮 Attention Indicator 是否合适。操作问题包括与外部电缆，适配器，软件驱动程序和电源故障有关的问题。通常，处于 On 状态的 Attention Indicator 指示尝试操作并失败，或者发生了意外事件。

Attention Indicator 不用于报告在验证热插拔操作请求时检测到的问题。验证是一个术语，用于系统软件执行的任何检查，以确保请求的操作可行，允许且不会引起问题。 验证失败的示例包括拒绝执行热插拔操作，功率预算不足以及在接受热插拔请求之前可能检测到的其他条件。

Attention Indicator Blinking

Attention Indicator 处于 Blinking 状态，表示系统软件正在识别此插槽。此行为由用户控制（例如，从软件用户界面或管理工具）。

6.7.1.1.2 Power Indicator

Power Indicator 必须为绿色，用于指示插槽的电源状态。Table 6-8 列出了电源指示灯状态。

Table 6-8: Power Indicator States

Indicator Appearance	Meaning
Off	Power Off - Insertion or removal of the adapter is permitted.
On	Power On - Insertion or removal of the adapter is not permitted.
Blinking	Power Transition - Hot-plug operation is in progress and insertion or removal of the adapter is not permitted.

Power Indicator Off

处于 Off 的电源指示灯表示允许插入或卸下适配器。根据 form factor 需要，则关闭插槽的主电源。请注意，根据 form factor 的不同，即使关闭主电源并且电源指示灯熄灭，其他电源/信号也可能保持开启状态。在使用 PCI Express card form factor 的示例中，如果平台为热插拔插槽提供 Vaux 且 MRL 已关闭，则即使电源指示灯熄灭，由 MRL 切换的任何信号也会连接到插槽。打开 MRL 时，由 MRL 切换的信号将断开。当插槽未上电和/或允许插入或卸下适配器时，系统软件必须关闭插槽的电源指示灯。有关详细信息，请参考适当的 form factor。

公众号·芯片业者相承双

Power Indicator On

处于 On 状态的电源指示灯表示热插拔操作已完成，并且插槽的主电源已打开，并且不允许插入或卸下适配器。

Power Indicator Blinking

电源指示灯闪烁表示插槽正在加电或断电，并且不允许插入或卸下适配器。

当按下 Attention 按钮或通过热插拔软件界面启动热插拔操作时，闪烁的电源指示灯还会向操作员提供视觉反馈。

6.7.1.2 Manually-operated Retention Latch (MRL)

用于固定插卡的手动置留闩。

MRL 是一种手动操作的固定机制，可将适配器固定在插槽中，并防止用户卸下设备。MRL 将适配器牢固地固定在插槽中，以便可以连接电缆而不会产生间歇性接触的风险。在不提供 MRL 传感器的平台中，允许同时按住两个或多个适配器的 MRL。

6.7.1.3 MRL Sensor

MRL Sensor 是一种开关、光学设备或其他类型的传感器，可将插槽的 MRL 的信息报告给下游端口。当 MRL 完全关闭并在所有其他时间打开（指的是 MRL 完全打开或处于中间位置），则 MRL 传感器报告一个关闭。

如果为插槽实现了电源控制器，则当 MRL 传感器指示 MRL 已打开时，必须自动从插槽中拔出插槽主电源。如果 MRL 切换了 Vaux 和 SMBus 之类的信号，则当 MRL 传感器指示 MRL 打开时，这些信号必须从插槽中自动删除，并且当 MRL 传感器指示 MRL 重新关闭时必须将其恢复到插槽中。请参考合适的 form factor 规范以识别由 MRL 切换的信号。

请注意，热插拔控制器不会根据 MRL 传感器的变化自动更改 Power Indicator 或 Attention Indicator 的状态。

MRL Sensor Handling

在缺少 MRL 传感器的情况下，对于某些 form factor，可以使用交错的存在检测引脚来处理开关信号。在这种情况下，当在线状态引脚断开接触时，切换的信号将自动从插槽中移除。

如果实现的 MRL 传感器在热插拔控制器上没有相应的 MRL 传感器输入，则建议将 MRL 传感器路由到热插拔控制器的电源故障输入。这将在打开 MRL 时关闭活动适配器的电源。

6.7.1.4 Electromechanical Interlock（机械锁扣）

机械锁扣是一种将适配器或 MRL 物理锁定到位的机制，直到系统软件释放它为止。机械锁扣的状态由软件设置，除非响应随后的软件命令，否则不得更改。特别是，即使断开热插拔插槽的电源，也必须保持机械锁扣的状态。

机械锁扣的当前状态必须始终反映在 Slot Status 寄存器中的 Electromechanical Interlock Status 字段中，该状态必须在任何命令更改后的 200 ms 内进行更新。在发出用于切换机械锁扣状态的命令之后，软件必须等待至少 1 秒钟，然后才能发出另一个用于切换状态的命令。系统可以选择扩展对机械锁扣的控制，以提供适配器的物理安全性。

机械锁扣，用于防止在插槽加电的情况下移除附加卡。

6.7.1.5 Attention Button

Attention Button 是一个瞬时接触按钮开关，位于每个热插拔插槽附近或位于适配器上，用户按下该按钮即可在该插槽上启动热

插拔操作。无论按钮的物理位置如何，信号都会通过对应插槽的下游端口相关联的热插拔硬件进行处理并显示给软件。

Attention Button 必须允许用户启动热插入和热移除操作，而不管按钮的物理位置如何。

如果系统软件接受 Attention Button 发起的请求，电源指示灯（如果存在）通过闪烁向操作员提供视觉反馈。 电源指示灯开始闪烁后，存在 5 秒钟的中止间隔，在此期间，第二次按下 Attention Button 将取消操作。

如果由 Attention Button 启动的操作由于任何原因而失败，建议系统软件通过软件用户界面显示一条错误消息，说明失败的原因，或者将错误消息添加到系统日志中。

6.7.1.6 Software User Interface

系统软件提供了一个用户界面，该界面允许启动热插入和热移除，并允许监视占用的插槽。热插拔用户界面的详细讨论是特定于操作系统的，因此不在本文档的讨论范围之内。

在具有多个热插拔插槽的系统上，系统软件必须允许用户在每个插槽上启动操作，而与所有其他插槽的状态无关。因此，允许用户使用软件用户界面或 Attention Button 在一个插槽上启动热插拔操作，而在另一个插槽上进行热插拔操作时，则无论使用哪个界面来启动第一个操作 。

6.7.1.7 Slot Numbering

物理插槽标识符（Physical Slot Identifier）（在 *PCI Hot-Plug Specification, Revision 1.1*, Section 1.5 中定义）由可选的底座号和插槽的物理插槽号组成。物理插槽号是插槽的底座唯一标识符。系统软件根据端口中的寄存器确定物理插槽号。Chassis number 0 保留用于主底座。其他底座的 Chassis number 必须是从 PCI-to-PCI Bridge 的 Chassis Number 寄存器中获得的非零值（请参阅 *PCI-to-PCI Bridge Architecture Specification, Revision 1.2*，Section 13.4）。

6.7.1.8 Power Controller

电源控制器是由一个或多个分立组件组成的元件，在软件的控制下起作用，用于根据特定的规格设置热插拔插槽的电源状态。电源控制器还必须监视插槽，以查看插槽的主电源轨和辅助电源轨（如果支持）上出现的电源故障情况（如相关的规格说明中所定义）。

如果不存在电源控制器，则热插拔控制器必须根据插槽中适配器的变化自动设置热插拔插槽的电源状态。

电源控制器独立监视主电源和辅助电源故障。如果电源控制器在热插拔插槽中检测到主电源故障，则它必须自动设置其内部主电源故障闩锁，并从热插拔插槽中断开主电源（不影响辅助电源）。同样，如果电源控制器在热插拔插槽上检测到辅助电源故障，则它必须自动设置其内部辅助电源故障闩锁，并从热插拔插槽中断开辅助电源（不影响主电源）。只要电源故障情况保持锁定状态，电源就必须保持关闭状态，无论软件为打开热插拔插槽打开电源而进行的任何写入操作。当软件关闭热插拔插槽的电源时，主电源故障闩锁将被清除。清除辅助电源故障闩锁的机制是特定于规格的，但是通常需要从热插拔插槽中断开辅助电源。例如，一种 form factor 可能会在插槽的 MRL 打开时消除辅助电源，而另一种则可能要求从插槽中物理移除适配器。有关特定要求，请参阅相关的 form factor。

由于 Slot Control 寄存器中的 Power Controller Control 字段反映的是最后写入的值，而不是电源控制器的实际状态，因此，这意味着 Power Controller Control 字段的值与电源故障情况下的插槽的电源控制器的电源状态可能不一致。要确定插槽是否由于电源故障而关闭，软件必须使用电源故障软件通知来检测电源故障。 为了确定请求的加电操作否则失败，软件必须使用 Section 6.7.3.3 所述的热插拔插槽加电超时机制。

软件做不到通过写入 Slot Control 寄存器来更改热插拔插槽的电源状态，从而立即导致电源状态转换。打开电源后，软件必须等待数据链接层状态更改事件，如 Section 6.7.3.3 所述。关闭电源后，软件必须等待至少 1 秒钟，然后才能执行任何依赖于已从热

插拔插槽中拔下电源的操作。例如，不允许软件在完成 1 秒等待时间之前关闭电源指示灯（如果有）或尝试打开电源控制器。

6.7.2 Registers Grouped by Hot-Plug Element Association

本节中描述的寄存器按热插拔元件分组。与每个具有热插拔插槽的下游端口相关的寄存器位于 PCI Express Capability structure 中的 Device Capabilities、Slot Capabilities、Slot Control 和 Slot Status 寄存器中（see Section 7.8）。报告与适配器上的设备功能相关联的热插拔元件存在的寄存器位于 Device Capabilities 寄存器中。

6.7.2.1 Attention Button Registers

Attention Button Present (Slot Capabilities and Device Capabilities) —— 该字段指示 Attention Button 是由底座（Slot Capabilities）还是由适配器（Device Capabilities）进行电气控制。

Attention Button Pressed (Slot Status) —— 当按下由底座的电气控制的 Attention Button 时，此位置 1。

Attention Button Pressed Enable (Slot Control) —— 设置为 1 时，此字段在 Attention Button Pressed 事件发生时进行软件通知（请参阅 Section 6.7.3.4）。必须在 BIOS 中设定好 Attention Button Pressed Enable 字段，表示使能在热插拔事件中可以产生中断和 wakeup 信号。

6.7.2.2 Attention Indicator Registers

Attention Indicator Present (Slot Capabilities and Device Capabilities) —— 该字段指示 Attention Indicator 是由底座（Slot Capabilities）还是由适配器（Device Capabilities）进行电气控制。

Attention Indicator Control (Slot Control) —— 当要写时，将底座控制的 Attention Indicator 设置为写状态。

6.7.2.3 Power Indicator Registers

Power Indicator Present (Slot Capabilities and Device Capabilities) —— 该字段指示 Power Indicator 是由底座（Slot Capabilities）还是由适配器（Device Capabilities）进行电气控制。

Power Indicator Control (Slot Control) —— 当要写时，将底座控制的 Power Indicator 设置为写状态。

6.7.2.4 Power Controller Registers

Power Controller Present (Slot Capabilities) —— 该字段为 1 表示实现了电源控制器。

Power Controller Control (Slot Control) —— 根据写入的值关闭或打开电源控制器。

Power Fault Detected (Slot Status) —— 当适配器或插槽检测到一个电源故障该字段置为 1。

Power Fault Detected Enable (Slot Control) —— 置位时，此字段使能电源故障事件的软件通知（请参见 Section 6.7.3.4）。

6.7.2.5 Presence Detect Registers

Presence Detect State (Slot Status) —— 该字段表示插槽中存在一个适配器（卡）。

公众号：芯片业者相承见

Presence Detect Changed (Slot Status) —— 当检测到存在检测状态改变时，该位置位。

Presence Detect Changed Enable (Slot Control) —— 设置为 1 时，该字段使能在状态检测更改事件时的软件通知（请参见 Section 6.7.3.4）。

6.7.2.6 MRL Sensor Registers

MRL Sensor Present (Slot Capabilities) —— 该字段表示实现了一个 MRL Sensor。

MRL Sensor Changed (Slot Status) —— 当 MRL Sensor 状态改变时该字段置 1。

MRL Sensor Changed Enable (Slot Control) —— 置位时，该字段使能有关 MRL 传感器更改事件的软件通知（请参见 Section 6.7.3.4）。

MRL Sensor State (Slot Status) —— 如果实现了，该字段报告 MRL 传感器的状态。

6.7.2.7 Electromechanical Interlock Registers

Electromechanical Interlock Present (Slot Capabilities) —— 该字段表示是否实现了一个机械锁扣。

Electromechanical Interlock Status (Slot Status) —— 该字段指示当前机械锁扣的状态。

Electromechanical Interlock Control (Slot Control) —— 每当设置为 1b，此字段可切换机械锁扣的状态。

6.7.2.8 Command Completed Registers

No Command Completed Support (Slot Capabilities) —— 当设置为 1b 时，表示在热插拔控制器完成发出的命令后，此插槽不生成软件通知。

Command Completed (Slot Status) —— 当设置为 1b 时，表示热插拔控制器完成了发出的命令并准备好接收下一个命令。

Command Completed Interrupt Enable (Slot Control) —— 设置为 1b 时，当热插拔控制器完成命令时，此字段使能软件通知（请参见 Section 6.7.3.4）。

6.7.2.9 Port Capabilities and Slot Information Registers

Slot Implemented (PCI Express Capabilities) —— 置位时，该字段表示与此下游端口关联的链路已连接到插槽。

Physical Slot Number (Slot Capabilities) —— 这是个硬件初始化的字段，表示连接到端口的物理插槽号。

Hot-Plug Capable (Slot Capabilities) —— 置位时，该字段表示该插槽能够支持热插拔。

Hot-Plug Surprise (Slot Capabilities) —— 设置为 1 时，此位表示对于相关的 form factor，允许在不事先通知的情况下将适配器从系统中卸下。

6.7.2.10 Hot-Plug Interrupt Control Register

Hot-Plug Interrupt Enable (Slot Control) —— 当置位时，此位使能在发生热插拔事件时生成热插拔中断。

公众号：芯片业者相承双

6.7.3 PCI Express Hot-Plug Events

具有热插拔功能的下游端口支持以下热插拔事件：

- Slot Events
  - Attention Button Pressed
  - Power Fault Detected
  - MRL Sensor Changed
  - Presence Detect Changed
- Command Completed Events
- Data Link Layer State Changed Events

这些事件中的每一个都有一个状态字段和一个使能字段，状态字段指示事件已发生但尚未由软件处理，而使能字段指示该事件是否会触发启用一个软件通知。 某些事件还具有 capability 字段，该字段指示端口是否支持事件类型。在 Section 6.7.2 中列出了按事件类型对这些字段进行的分组，在 Section 7.8 中介绍了每个单独的字段。

6.7.3.1 Slot Events

具有热插拔功能的下游端口会监视它控制的插槽，以查看上面列出的插槽事件。当检测到这些插槽事件之一时，端口会通过设置与事件关联的状态字段来指示事件已发生。此时，事件待处理，直到软件清除状态字段为止。

一旦某个插槽事件在特定插槽上挂起，该类型的所有后续事件将在该插槽上被忽略，直到事件被清除。端口必须继续监视所有其他插槽事件类型的插槽，并在发生时报告它们。

如果通过相关的使能字段进行使能，则插槽事件必须生成软件通知。如果端口未如关联的 capability 字段所指示的那样支持该事件，则软件不得为该事件使能软件通知。在 Section 6.7.3.4 中描述了将此通知报告给软件的机制。

6.7.3.2 Command Completed Events

由于更改某些热插拔元件时其状态可能不会立即发生，因此 PCI Express 支持热插拔命令和命令完成事件。 需要所有支持热插拔的端口来支持热插拔命令，如果通告了此功能，则还必须支持命令完成事件。

软件通过发出针对端口 Slot Control 寄存器任何部分的写事务，向具有热插拔功能的下游端口发出命令。 即使写入影响 Slot Control 寄存器中的多个字段，对 Slot Control 寄存器的单次写入也被视为单个命令。为响应此事务，端口必须执行请求的操作，然后为命令完成事件设置相关的状态字段。即使发出命令时状态字段已设置，端口也必须正常处理命令。如果单个命令导致启动多个动作，则未指定执行动作的顺序。与单个命令执行相关的所有动作的时间不得超过 1 秒。

如果不支持命令完成事件，即 Slot Capabilities 寄存器的 No Command Completed Support 字段中的值 1b，则具有热插拔功能的端口必须依赖于以前的插槽控制写入处理针对该端口的 Slot Control 寄存器的任何部分的写事务。允许软件按顺序发出多个插槽控制写操作，而两次写操作之间没有任何延迟。

如果支持命令完成事件，那么在发出下一个命令之前，软件必须等待命令完成。但是，如果在执行命令的 1 秒限制后未设置状态字段，则允许软件重复执行该命令或发出下一条命令。如果软件在端口完成上一条命令的处理之前且在 1 秒时间限制到期之前发出写操作，则允许端口接受或放弃写操作。 这种写操作被认为是编程错误，并且可能导致 Slot Control 寄存器和热插拔元件状态之间出现差异。为了从这样的编程错误中恢复并使控制器返回一致状态，软件必须向符合命令完成规则的 Slot Control 寄存

公众号·芯片业者相承及

器发出写操作。

如果通过相关的使能字段进行使能，则命令的完成必须生成软件通知。该规则的例外是由于写入 Slot Control 寄存器而导致的命令发生，该命令禁用了命令完成事件的软件通知。必须如上所述处理此类命令，但不得生成软件通知。

6.7.3.3 Data Link Layer State Changed Events

数据链路层状态更改事件表示 Link Status 寄存器中的 Data Link Layer Link Active 字段的状态已更改。对于支持热插拔的下游端口，需要支持数据链路层状态更改事件以及这些事件的软件通知。如果支持此事件，则当 Data Link Layer Link Active 字段的值更改时，端口将设置与事件关联的状态字段。

此事件使软件可以间接确定何时向新热插拔的适配器供电。在启动对热插入设备的配置访问之前，软件必须在读到 Data Link Layer Link Active 字段为 1b 之后等待 100 ms，然后才能进行配置访问（请参见 Section 6.6）。 在读到 Data Link Layer Link Active 字段为 1b 之后，软件必须等待 1 秒钟，然后才可以确定为有效配置请求返回 Successful Completion 失败的热插拔设备是损坏的设备（see Section 6.6）。

数据链路层状态更改事件必须在启动热插入事件的 1 秒内发生。如果支持电源控制器，则从软件发出写入 Slot Control 寄存器以打开电源的时间开始计算超时间隔。如果不支持电源控制器，则从检测插槽事件发生开始测量超时间隔。 如果 1 秒钟内未发生数据链路层状态更改事件，则允许软件在热插入操作上超时。 此类超时发生后，软件采取的操作是特定于实现的。

6.7.3.4 Software Notification of Hot-Plug Events

具有热插拔功能的下游端口必须支持在热插拔事件时产生中断。如 Section 6.7.3.1 和 Section 6.7.3.2 中所述，每个热插拔事件都具有一个用于产生中断的使能位和一个状态位，用于指示何时发生事件但尚未由软件处理。Slot Control 寄存器中还有一个热插拔中断允许位，用作所有热插拔事件的主要的使能/禁用位。

如果为端口启用了使用 INTx 消息进行电平触发的中断信令的功能，则只要满足以下条件，就必须置起虚拟 INTx 线：

- Command 寄存器的 Interrupt Disable 字段为 0b。
- Slot Control 寄存器的 Hot-Plug Interrupt Enable 字段为 1b。
- Slot Status 寄存器中至少一个热插拔事件状态字段及其在 Slot Control 寄存器中的相关使能字段均被设置为 1b。

请注意，当请求服务时，同一 Function 内的所有其他中断源将声明相同的虚拟 INTx 线。

如果使用 MSI 或 MSI-X 为端口启用边沿触发的中断信令，则每当以下条件的逻辑 AND 从 FALSE 变为 TRUE 时，都必须发送中断消息：

- 相关的向量未屏蔽（如果 MSI 不支持 PVM，则不适用）。
- Slot Control 寄存器的 Hot-Plug Interrupt Enable 字段为 1b。
- Slot Status 寄存器中至少一个热插拔事件状态字段及其在 Slot Control 寄存器中的相关使能字段均被设置为 1b。

请注意，如 PCI Express Capabilities 寄存器中的 Interrupt Message Number 字段所示，PME 和 Hot-Plug Event 中断（在同时实现时）始终共享相同的 MSI 或 MSI-X 向量。

当在中断生成被禁用的情况下发生热插拔事件，端口可以选择发送 MSI，并随后使能中断生成。

如果相关的 form factor 规范要求生成一个唤醒，则具有热插拔功能的下游端口必须支持为系统处于睡眠状态或端口处于 D1、D2 或 $D3_{Hot}$ 下的热插拔事件生成唤醒事件（使用 PME 机制）。

公众号·芯片业者相承汉

软件通过使能一个 Section 6.7.3.1 所述软件通知可以使热插拔事件生成唤醒事件。请注意，为了使软件在保持唤醒生成使能的同时禁用中断生成功能，必须要把 Hot-Plug Interrupt Enable 字段置为 0b。对于支持唤醒生成的 form factor，如果同时满足以下三个条件，则必须生成唤醒事件：

- 已使能事件的状态寄存器从 0b 变成 1b。
- 端口处于 D1、D2 或 $D3_{Hot}$ 状态。
- 端口的 Power Management Control/Status 寄存器中的 PME_En 字段置 1。

请注意，热插拔控制器代表热插拔设备生成唤醒，并且该设备不必具有辅助（或主）电源。

6.7.4 Firmware Support for Hot-Plug

在支持本地热插拔的 ACPI 兼容操作系统可用之前，已释放包含具有热插拔功能的根端口和 Switch 的某些系统可以使用 ACPI 固件传播热插拔事件。如果使用具有本机支持的操作系统，则必须禁用对热插拔寄存器的固件控制。提供 ACPI 固件来传播热插拔事件的平台还必须提供一种将控制权转移到操作系统的机制。此方法的详细信息在 *PCI Firmware Specification* 中进行了描述。

操作系统对 PCIe 热插拔的支持，主要涉及到 ACPI 部分。当 BIOS 中已经将热插拔的功能做好初始化之后，进入 OS，接下来的动作就是有 ACPI 中的 method 来完成了。BIOS 会将系统的中断导入 GPE，这样 ACPI 可以响应 PCIe 的热插拔的事件，包括热插入和热移除。

6.8 Power Budgeting Capability

由于增加了适配器的热插拔功能，因此需要系统能够正确地为添加到系统中的任何新设备分配电源。此功能是与电源管理分开的独立功能，需要一些基本的支持才能确保系统正常运行。电源预算（Power Budgeting）概念将构建模块放置在适当的位置，这些模块允许设备与系统交互以实现这些目标。系统可以有多种方式实现实际的电源预算功能，因而电源预算的实现超出了本规范的范围。

对于以不需要热插拔支持的尺寸外形实现的设备或集成在系统板上的设备，实现 Power Budgeting Capability 是可选的。尺寸外形可能需要支持电源预算。设备和/或适配器需要保持在相应的机械规范中指定的配置功率限制内，直到它们已由系统配置和启用。在启用适配器之前，系统应确保已正确预算了电源。

6.8.1 System Power Budgeting Process Recommendations

建议系统固件为电源预算管理引擎提供以下信息：

- 系统总电源预算（供电信息）。
- 系统固件（系统板设备）分配的总电源。
- 插槽总数和插槽类型。

系统固件负责为主板上所有不具备电源预算功能的设备分配电源。 固件可能包括也可能不包括连接到标准电源轨的设备。当固件为实现 Power Budgeting Capability 的设备分配电源时，必须将 Power Budget Capability 寄存器中的 System Allocated 字段设置为 1b，以表明已正确分配了电源。电源预算管理器负责分配所有 PCI Express 设备，包括具有 Power Budgeting Capability 且 System Allocated 字段清 0 的系统板设备。电源预算管理器负责确定是否可以在系统中预算和启用热插拔设备。

存在可以提供相同功能的替代方法，并且不需要以这种方式实现电源预算过程。

6.9 Slot Power Limit Control

PCI Express 提供了一种机制，用于通过软件控制每个插槽上 PCIe 适配器可以消耗的最大电源限制。该机制的关键点是：

- RC 或 Switch 的 Downstream Port 要实现 Slot Capabilities 寄存器的 Slot Power Limit Value 字段 Scale 字段。
- Endpoint、Switch 或 PCI Express-PCI Bridge Function 要在其 Upstream Port 上实现 Device Capabilities 寄存器的 Captured Slot Power Limit Value 字段 Scale 字段。
- Downstream Port（RC 或 Switch 的）通过 Set_Slot_Power_Limit Message 将 Slot Capabilities 寄存器的 Slot Power Limit Value 字段 Scale 字段的值传送给相连的组件的 Upstream Port 的 Device Capabilities 寄存器的 Captured Slot Power Limit Value 字段 Scale 字段。

平台上的电源限制通常由了解平台细节的软件（例如，平台固件）控制，例如：

- 平台分区，包括用于使用适配器进行 I/O 扩展的插槽。
- 供电能力。
- 散热能力。

该软件负责正确编程连接到插槽的 Downstream Port 的 Slot Capabilities 寄存器的 Slot Power Limit Value 字段 Scale 字段。将该值写入 Downstream Port 内的寄存器后，使用 Set_Slot_Power_Limit Message 将其传送到适配器（请参阅 Section 2.2.8.5）。该 Message 的接收者必须使用 Message 中的有效数据负载值来限制整个适配器的电源使用，除非适配器永远不会超过相应的尺寸外形规范中指定的最小值。要求与适配器关联的设备驱动程序能够（通过读取 Device Capabilities 寄存器的 Captured Slot Power Limit Value 字段 Scale 字段的值）配置适配器的硬件，以确保适配器不会超出规定的限制范围。如果平台施加的限制低于适当操作所需的最小值，则设备驱动程序应该能够将此差异传达给更高级别的配置软件。需要配置软件，以基于平台的能力将插槽功率限制设置为为相应外形规格指定的最大值之一。

插槽电源限制控制机制的规则如下：

对于适配器：

- 直到并且除非收到 Set_Slot_Power_Limit 消息，除非该消息指示的插槽功率限制值大于适配器的规范中指定的最小值，否则适配器消耗的电源不得超过指定的最小值。
- 适配器的电源绝不能超过最近接收到的 Set_Slot_Power_Limit 消息中指定的功率或相应的外形规格中指定的最小值（以较高者为准）。
- 旨在集成到一个适配器上且该适配器的总功耗低于为目标外形规格规范的最低限制的 Endpoint, Switch, or PCI Express-PCI Bridge Function 组件，可以忽略 Set_Slot_Power_Limit 消息，并为 Device Capabilities 寄存器的 Captured Slot Power Limit Value 字段 Scale 字段返回一个 0。
    - 这些组件必须仍能够无错误地接收 Set_Slot_Power_Limit 消息，它只是丢弃 Message 值。

对于作为插槽源端的 RC 和 Switch：

- 配置软件不得对 Set_Slot_Power_Limit 值进行编程，该值指示的限制低于插槽的外形规格指定的最小值。

Example Adapter Behavior Based on the Slot Power Limit Control Capability

以下电源限制方案是基于插槽电源限制控制功能的适配器必须如何工作的示例。外形规格限制是一种表示形式，不应视为实际要求。

注意：Form factor #1 的最大功率要求为 40 W 和 25 W；Form factor #2 的最大功率要求为 15W。

公众号：芯片业者程承双

Scenario 1: An Adapter Consuming 12 W

- 如果将该适配器插入 Form factor #1 的 40 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。
- 如果将该适配器插入 Form factor #1 的 25 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。
- 如果将该适配器插入 Form factor #2 的 15 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。

在以上所有情况下，由于适配器在所有 Form factor 范围内均可正常运行，因此它可以忽略任何插槽电源限制消息。

Scenario 2: An Adapter Consuming 18 W

- 如果将该适配器插入 Form factor #1 的 40 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。
- 如果将该适配器插入 Form factor #1 的 25 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。
- 如果将该适配器插入 Form factor #2 的 15 W 插槽，则将遵循插槽功率限制控制机制，并且适配器必须向下调整至 15 W 或禁止工作。对于给定的 Form factor，未在任何功率限制范围内扩展的适配器将始终以该 Form factor 被禁用，并且不应使用。

在这种情况下，如果仅以 Form factor #1 使用适配器，则它可以忽略任何插槽功率限制消息。为了在 Form factor #2 中有用，适配器应该能够缩放到 Form factor #2 的功率极限。

Scenario 3: An Adapter Consuming 30 W

- 如果将该适配器插入 Form factor #1 的 40 W 插槽，则将遵循插槽功率限制控制机制，并且适配器可以正常运行。
- 如果将该适配器插入 Form factor #1 的 25 W 插槽，则将遵循插槽功率限制控制机制，并且设备必须缩放至 25 W 或禁止工作。
- 如果将该适配器插入 Form factor #2 的 15 W 插槽，则将遵循插槽功率限制控制机制，并且适配器必须向下调整至 15 W 或禁止工作。对于给定的 Form factor，未在任何功率限制范围内扩展的适配器将始终以该 Form factor 被禁用，并且不应使用。

在这种情况下，由于适配器的功耗超过了插槽的最低功率限制，因此适配器必须能够扩展或禁用以防止系统故障。必须避免以超出其所插入插槽容量的功率水平运行适配器。

Slot Power Limit Control Registers

通常，RC 或 Switch 的 Downstream Port 内的 Slot Power Limit 寄存器各字段将由平台特定的软件进行编程。一些实现可以使用硬件方法来初始化这些寄存器的值，因此不需要软件支持。

旨在集成到一个适配器上且该适配器的总功耗低于为目标外形尺寸定义的最低限制的 Endpoint, Switch, or PCI Express-PCI Bridge Function 组件，可以忽略 Set_Slot_Power_Limit 消息。请注意，采用此实现方法的组件可能与将来定义的 Form factor 不兼容。 这样的 Form factor 可能会施加较低的功率限制，该功率限制低于基于现有组件的新适配器所需的最低功率。

6.10 RC Topology Discovery

RC 可能会为配置软件提供以下拓扑之一：

- 单个不透明的 RC，这样软件就看不到 RC 的内部操作。从软件角度来看，所有根端口都彼此独立；不存在任何机制来管理和区分服务的各个根端口之间的仲裁。
- 单个 RC 组件，这样软件就对 RC 组件的内部操作具有可见性和控制力。如 Figure 6-11 所示，软件将 Root Port 视为该组件的 Ingress Port。用于将业务从 RC 的 internal Port 传递到 Egress Port 或内部 Sink 部件由 RCRB 表示。通过 RCRB 中的 Virtual

Channel Capability structure 提供差异化服务控制。

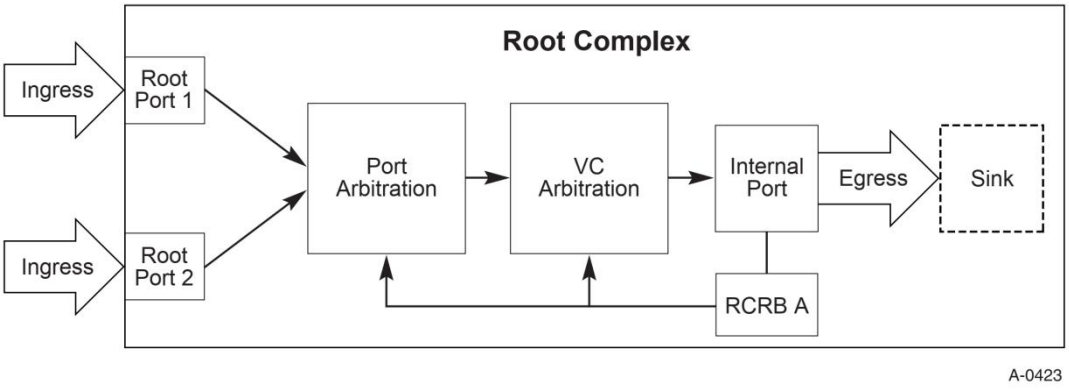

Figure 6-11: Root Complex Represented as a Single Component

- 多个 RC 组件，这样，软件不仅对给定 RC 组件的内部操作具有可见性和控制力，而且还具有发现和控制不同 RC 组件之间的仲裁的能力。如 Figure 6-12 所示，软件将 Root Port 视为该组件的 Ingress Port。RCRB 结构控制从组件到其他 RC Component（RCRB C）或内部 Sink 部件（例如内存）（RCRB A）的出口。另外，RCRB 结构（RCRB B）也可能存在于给定的组件中，以控制来自其他 RC 组件的业务。通过分别位于 RCRB 中的 Virtual Channel Capability structure，提供了差异化服务控制。

公众号：芯片业者相关

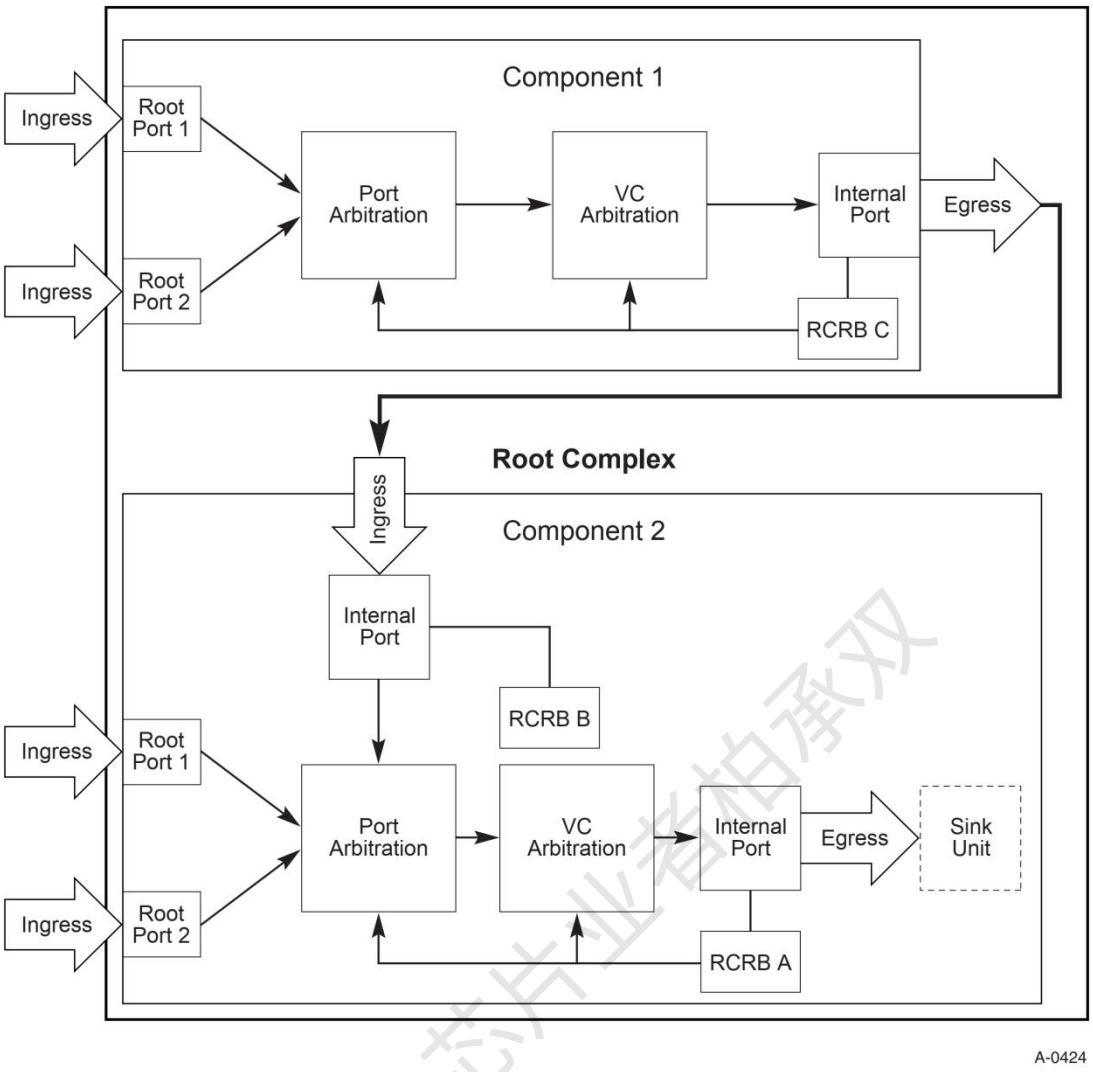

Figure 6-12: Root Complex Represented as Multiple Components

当然也存在更多的拓扑可能。

RC 的拓扑可以表示为逻辑 RC 组件的集合，这样每个逻辑组件都具有：

- 一个或多个 Ingress Port。
- 一个 Egress Port。
- 如果 RC 支持虚通道，则可选的关联虚通道功能位于配置空间（for Root Port）或 RCRB（for internal Ingress/Egress Port）中。
- RC 中可选集成 device/Function。

为了使软件能够正确编程 PCI Express 差异化服务的仲裁和其他控制参数，软件必须能够发现 RC 的内部拓扑。RC 拓扑声明发现是通过 RC Link Declaration Capability 来实现的，如 Section 7.13 所述。

6.11 Link Speed Management

本节介绍如何在 LTSSM（Section 4.2.6）和软件链路检测和控制机制（Section 7.8.6、7.8.7、7.8.8、7.8.18、7.8.19 和 7.8.20）之间协调链路速度管理。

公众号·芯片业者怕承双

Link Control 2 寄存器中的 Target Link Speed 字段设置了链路速率的上限。除下述内容外，上游组件必须尝试以目标链路速度或链路上两个组件都支持的最高速率来维持链路（在训练有序集中通告，请参见 Section 4.2.4.1）。

如果把 Link Control 2 寄存器中的 Hardware Autonomous Speed Disable 字段清为 0，则组件可以自主调整链路速率。

如果链路的可靠性太低，则允许任一组件通过从组件传输的训练集中通告的支持速率列表中删除不可靠的链路速率来降低链路速率。确定链路可靠性的标准是特定于实现的，并且不取决于 Hardware Autonomous Speed Disable 字段的设置。

在任何给定的速度协商期间，其中一个或两个组件都可能会通告支持的所有速率的子集，以此作为限制协商后链路速率的一种方法。允许组件在不设置 speed change 字段的情况下将链路驱动到 Recovery 以改变它通告的链路速率的设置而不用发起一个链路速率切换的请求。

当组件尝试协商到一个特定链路速率失败，则不允许该组件再次尝试协商到该链路速率或更高链路速率，知道从失败返回到 L0 之后经过 200ms 为止，或者直到链路上的其他组件通过其传输的训练集（有或没有携带请求更改链路速度的信息）通告对更高链路速率的支持，这两者以先到者为准。

允许软件通过设置上游组件中 Target Link Speed 字段中的值来限制链路操作的最大速率并设置首选的链路速率。修改 Target Link Speed 字段中的值后，软件必须通过将 1b 写入 Retrain Link 字段来触发链路重新训练。通过 Link Bandwidth Notification Mechanism，可将链路速率的任何变化（以及链路宽度的任何变化）通知给软件。

软件可以通过把两端组件的 Link Control 2 寄存器的 Target Link Speed 字段和 Enter Compliance 字段设置为 1 来让链路进入 Polling.Compliance 状态，然后上游组件在链路上发送 Hot Reset。软件需要将相同的 Target Link Speed 值写入上游组件和下游组件。请注意，这将使链路进入 DL_Down 状态，因此对于使用该链路的其他软件无法透明地完成。清除 Enter Compliance 字段后，Downstream Port 将返回到 Polling.Active 状态。

6.12 Access Control Services (ACS)

ACS 在 PCI Express 拓扑中定义了一组控制点，以确定 TLP 是否应该正常路由、阻塞或重定向。ACS 适用于 RC、Switch 和 multi-Function 设备。

注意：此 multiFunction 设备中的可应用的 Function 具体包括 PCI Express Endpoint、Switch Upstream Port、Legacy PCI Express Endpoint 和 RC Integrated Endpoint 。

ACS 提供了下述类型的访问控制机制：

- ACS Source Validation (V)
- ACS Translation Blocking (B)
- ACS P2P Request Redirect (R)
- ACS P2P Completion Redirect (C)
- ACS Upstream Forwarding (U)
- ACS P2P Egress Control (E)
- ACS Direct Translated P2P (T)

在以下小节中，将讨论每种方法的具体要求。每种类型后面带括号的字母是在 Section 7.16 中定义的功能和控制寄存器相关字段的缩写。

ACS 硬件功能默认情况下处于禁用状态，并且仅由支持 ACS 的软件启用。除 ACS Source Validation 外，ACS 访问控制不适用于 Multicast TLP（Section 6.14），并且对它们没有任何影响。

公众号·志告业者的承双

6.12.1 ACS Component Capability Requirements

ACS 功能通过 ACS Extended Capability structure 进行管理。允许 PCI Express 组件根据 Function 的需求实现部分、全部或不实现 ACS Extended Capability structure 中的 Capability。通过每个 ACS Extended Capability structure 中的 Capability 字段来传达实现的范围。一个给定的有 ACS Extended Capability structure 的 Function 可能需要或禁止某些 Capability，具体取决于 Function 的特定类型以及它是否是 multi-Function 设备的一部分。

ACS 永远不适用于 PCI Express to PCI Bridge Function 或 Root Complex Event Collector Function，并且此类 Function 绝不能实现 ACS Extended Capability structure。

6.12.1.1 ACS Downstream Ports

本部分适用于实现 ACS Extended Capability structure 的 Root Port 和 Downstream Switch Port。本节适用于 single-Function 设备和 multi-Function 设备的 Downstream Port Function。

- ACS Source Validation：上述所有设备类型必须实现 ACS Source Validation。

  启用该功能后，Downstream Port 会根据端口接收到的每个上游请求的 Requester ID 来测试 Bus Number，以确定它是否在端口的 Bus Number “aperture（孔）”内——此孔由 Secondary Bus Number 寄存器和 Subordinate Bus Number 寄存器定义。

  如果请求的 Requester ID 不在此“孔”内，相关的接收端口要报告一个 ACS Violation 错误，参阅 Section 6.12.4。

  Completion 永远不会受 ACS Source Validation 影响。

- ACS Translation Blocking：上述所有设备类型必须实现 ACS Translation Blocking。

  启用该功能后，Downstream Port 将检查端口接收到的每个上游的内存请求的 Address Translation（AT）字段。如果 AT 字段不是默认值，则这将报告与接收端口相关的错误（ACS Violation）（请参见 Section 6.12.4）。此错误必须优先于 ACS Upstream Forwarding 和任何 ACS P2P 控制机制。

  Completion 永远不会受 ACS Translation Blocking 影响。

- ACS P2P Request Redirect：支持与其他 Root Port 进行对等传输的 Root Port[80] 必须实现 ACS P2P Request Redirect；Switch 的 Downstream Port 必须实现 ACS P2P Request Redirect。

  ACS P2P Request Redirect 必须与 ACS P2P Egress Control 和 ACS Direct Translated P2P 机制（如果实现）进行交互。有关更多信息，请参阅 Section 6.12.3。

  在 Switch 的 Downstream Port 中启用 ACS P2P Request Redirect 后，必须将对等请求向上游重定向到 RC。

  在 Root Port 中启用 ACS P2P Request Redirect 后，必须将对等请求发送到 RC 内的重定向请求验证逻辑（Redirected Request Validation），该逻辑确定该请求是“反射”回下游返回到原始目标，还是作为 ACS Violation 进行阻塞。本规范未设计用于进行此确定的算法和特定控件。

  Downstream Port 从不重定向正在向下游传输的请求。

  Completion 永远不会受 ACS P2P Request Redirect 影响。

  注 80：ACS P2P Request Redirect 或 ACS P2P Completion Redirect 的 Root Port 指示并不表示 RC 支持对等传输，或者根本不支持对等传输。

- ACS P2P Completion Redirect：实现了 ACS P2P Request Redirect 的 Root Port 必须实现 ACS P2P Completion Redirect；Switch 的 Downstream Port 必须实现 ACS P2P Completion Redirect。

公众号·芯片业者相建双

ACS P2P Completion Redirect 的目的是避免在重定向请求时在完成和请求之间对排序规则进行排序。 有关更多信息，请参见 Section 6.12.5。

ACS P2P Completion Redirect 不与管理请求的 ACS 控制逻辑交互。

在 Switch 的 Downstream Port 中启用 ACS P2P Completion Redirect 时，必须将 Relaxed Ordering Attribute 字段为 0b 的 peer-to-peer Completion[81] 重定向到 RC 上游。否则，必须正常路由该 peer-to-peer Completion。

如果在 Root Port 中启用了 ACS P2P Completion Redirect，则必须处理 Relaxed Ordering Attribute 字段为 0b 的对等完成，使得它们不传递发送到 RC 中重定向请求验证逻辑的请求。最终，此类完成必须向其原始对等目标发送给下游，而无需进行额外的 ACS 访问控制检查。

Downstream Port 从不重定向正在向下游传输的 Completion。

Request 永远不会受 ACS P2P Completion Redirect 影响。

注 81：这包括 Read Completion，AtomicOp Completion 以及带数据或不带数据的其他 Completion。

- ACS Upstream Forwarding：如果 RC 支持 Redirected Request Validation，则其 Root Port 必须实现 ACS Upstream Forwarding；Switch 的 Downstream Port 必须实现 ACS Upstream Forwarding。

  如果在 Switch 的 Downstream Port 中启用了 ACS Upstream Forwarding，并且其 Ingress Port 收到针对该端口自己的 Egress Port 的上游请求或完成 TLP，则该端口必须改为将 TLP 上游转发到 RC。

  如果在 Root Port 中启用了 ACS Upstream Forwarding，并且其 Ingress Port 收到针对该端口自己的 Egress Port 的上游请求或完成 TLP，则该端口必须按以下方式处理该 TLP。对于请求，Root Port 必须采用与 ACS P2P Request Redirect 机制一致的方式进行处理。对于完成，Root Port 必须采用与 ACS P2P Completion Redirect 机制一致的方式进行处理。

  如果在 Downstream Port 上未启用 ACS Upstream Forwarding，并且其 Ingress Port 收到针对该端口自己的 Egress Port 的上游请求或完成 TLP，则该 TLP 的处理是不确定的。

- ACS P2P Egress Control：上述设备可选择是否实现该机制。

  ACS P2P Egress Control 必须与 ACS P2P Request Redirect 和 ACS Direct Translated P2P（如果实现）机制进行交互。有关更多信息，请参见 Section 6.12.3。

  可以选择配置支持 ACS P2P Egress Control 的 Switch，以阻塞其 Downstream Port 之间的对等请求。软件可以将 Switch 配置为仅允许其 Downstream Port 中的一个或仅一部分允许其对等请求发送到其他 Downstream Port。这是在每个 Downstream Port 的基础上配置的。

  支持 ACS P2P Egress Control 的 RC 可以有选择地配置为阻塞其 Root Port 之间的对等请求。软件可以将 RC 配置为仅允许 Root Port 层次结构中的任何一个或仅一部分将其对等请求发送到其他 Root Po 层次结构。这是基于每个 Root Port 进行配置的。

  对于 Downstream Port 中的 ACS P2P Egress Control，Ingress Port（“发送”端口）中的控件将确定是否阻塞对等请求，如果阻塞，则 Ingress Port 将根据 Section 6.12.4 处理 ACS Violation 错误。

  Completion 永远不会受 ACS P2P Egress Control 影响。

- ACS Direct Translated P2P ：支持 ATS（Address Translation Services ）且至此与其他 Root Port[82] 对等传输的 Root Port 必须实现 ACS Direct Translated P2P；Switch 的 Downstream Port 必须实现 ACS Direct Translated P2P。

  在 Downstream Port 中启用 ACS Direct Translated P2P 时，无论 ACS P2P Request Redirect 和 ACS P2P Egress Control 如何设置，AT 字段设置为 1（表示这是一个转换地址）的 peer-to-peer Memory Request 都必须正常（“直接”）路由到对等的 Egress Port。所有其他对等请求仍必须遵守 ACS P2P Request Redirect 和 ACS P2P Egress Control 的设置。

公众号·芯片业者相承双

Completion 永远不会受 ACS Direct Translated P2P 影响。

注 82：Root Port 支持 ACS Direct Translated P2P 并不表示 RC 支持对等传输，或者根本不支持对等传输。

6.12.1.2 ACS Functions in Multi-Function Devices

本节适用于 Multi-Function 设备的 ACS Function，但 Downstream Port Function 除外，Downstream Port Function 在上一节中进行了介绍。

- ACS Source Validation：Multi-Function 设备的 ACS Function 不实现 ACS Source Validation 机制。
- ACS Translation Blocking：Multi-Function 设备的 ACS Function 不实现 ACS Translation Blocking 机制。
- ACS P2P Request Redirect：支持与其他 Function 进行对等传输的 Function 必须实现 ACS Translation Blocking 机制。

  ACS P2P Request Redirect 必须与 ACS P2P Egress Control 和 ACS Direct Translated P2P（如果实现）机制进行交互。有关更多信息，请参见 Section 6.12.3。

  在 Multi-Function 设备中启用 ACS P2P Request Redirect 后，必须将对等请求（在设备 Function 之间）重定向到上游 RC。

  Completion 永远不会受 ACS P2P Request Redirect 影响。

- ACS P2P Completion Redirect：实现 ACS P2P Request Redirect 的 Function 必须实现 ACS P2P Completion Redirect。

  ACS P2P Completion Redirect 的目的是避免在重定向请求时在完成和请求之间对排序规则进行排序。 有关更多信息，请参见 Section 6.12.5。

  ACS P2P Completion Redirect 不与管理请求的 ACS 控制逻辑交互。

  在 Multi-Function 设备中启用 ACS P2P Completion Redirect 后，必须将未设置 Relaxed Ordering 字段的 peer-to-peer Read Completion 向上游重定向到 RC。否则，必须正常路由 peer-to-peer Read Completion。

  Request 永远不会受 ACS P2P Completion Redirect 影响。

- ACS Upstream Forwarding：Multi-Function 设备的 ACS Function 不实现 ACS Upstream Forwarding 机制。
- ACS P2P Egress Control：该机制对 Multi-Function 设备的 ACS Function 是可选实现的；这基于 Function Number 或 Function Group Number；控制 Multi-Function 设备中不同 Function 之间的对等请求。

  ACS P2P Egress Control 必须与 ACS P2P Request Redirect 和 ACS Direct Translated P2P（如果实现）机制进行交互。有关更多信息，请参见 Section 6.12.3。

  Multi-Function 设备中每个支持 ACS P2P Egress Control 的 Function 可以选择用来阻塞与设备内其他 Function 或 Function Group[83] 之间的对等通信。这是基于每个 Function 配置的。

  对于 Multi-Function 设备中的 ACS P2P Egress Control，发送相关的控制逻辑确定是否阻塞了请求，如果是，则“发送”根据 Section 6.12.4 处理 ACS Violation 错误。

  在 ARI 设备中启用 ACS Function Group 时，将基于每个 Function Group 强制实现 ACS P2P Egress Control，而不是基于每个 Function。参见 Section 6.13。

  Completion 永远不会受 ACS P2P Egress Control 影响。

  注 83：对于实现了 ACS P2P Egress Control 的 ARI 设备，ACS Function Group 功能是可选的。

- ACS Direct Translated P2P：如果 Multi-Function 设备支持 ATS 并且支持与其他 Function 进行对等传输，则必须实现该机制。

公众号：芯片业者相承双

在 Multi-Function 设备中启用 ACS Direct Translated P2P 时，无论 ACS P2P Request Redirect 和 ACS P2P Egress Control 如何设置，AT 字段设置为 1（表示这是一个转换地址）的 peer-to-peer Memory Request 都必须正常（“直接”）路由到对等的 Function。所有其他对等请求仍必须遵守 ACS P2P Request Redirect 和 ACS P2P Egress Control 的设置。

Completion 永远不会受 ACS Direct Translated P2P 影响。

6.12.1.3 Functions in Single-Function Devices

Single-Function 设备不支持 ACS capability，也必须不能实现 ACS Extended Capability structure。

6.12.2 Interoperability

下列规则用于管理 ACS 和 non-ACS 组件之间的互操作：

- 当不使用 ACS P2P Request Redirect 和 ACS P2P Completion Redirect 时，ACS 和 non-ACS 组件可能在拓扑结构中混合在一起，并且将完全互操作。可以在 ACS 组件的子集中启用 ACS，而不会影响互操作性。
- 当使用 ACS P2P Request Redirect 或 ACS P2P Completion Redirect 或同时使用这两者时，PCI Express 层次结构中的某些组件必须支持 ACS Upstream Forwarding。特别地：
  - 关联的 Root Port[84] 必须支持 ACS Upstream Forwarding。否则，Root Port 如何处理向上游发送的 Request TLP 或 Completion TLP 是不确定的。RC 还必须实现 Redirected Request Validation。

  注 84：不适用于 multi-Function RC Integrated Endpoint 的 Function 之间的 ACS 重定向。

  - 在启用了 P2P TLP 重定向的每个 ACS 组件及其关联的 R Port 之间，任何中间 Switch 都必须支持 ACS Upstream Forwarding。否则，此类 Switch 如何处理向上游发送的 TLP 是不确定的。

6.12.3 ACS Peer-to-Peer Control Interactions

对于每个 peer-to-peer Request，多个 ACS 控制机制可以进行交互以确定该请求是直接路由到其对等目标、还是作为 ACS Violation 立即阻塞、还是重定向到上游 RC 以进行访问验证。Peer-to-peer Completion 重定向完全由 ACS P2P Completion Redirect 机制确定。

如果在 Port/Function 中启用了 ACS Direct Translated P2P，则其 AT 字段设置为 1b 的 peer-to-peer Memory Request 必须正常（“直接”）路由到对等的 Port/Function，而与 ACS P2P Request Redirect 和 ACS P2P Egress Control 的设置无关。否则，此类请求以及无条件的所有其他对等请求必须服从 ACS P2P Request Redirect 和 ACS P2P Egress Control 设置。具体来说，Egress Control Vector 字段和 ACS P2P Egress Control Enable（E）字段以及 ACS P2P Request Redirect Enable（R）字段决定如何处理请求。必须注意，如果 ACS 对等请求重定向以传统设备位置作为锁定访问的目标，则不能保证访问的原子性。有关这些控制字段的说明，请参见 Section 7.16。Table 6-9 指明了这种交互。

公众号：芯片业者怕承双

Table 6-9: ACS P2P Request Redirect and ACS P2P Egress Control Interactions

Control Bit E (b)	Control Bit R (b)	Egress Control Vector Bit for the Associated Egress Switch Port, Root Port, Function, or Function Group	Required Handling for Peer-to-Peer Requests
0	0	X – Don't care	Route directly to peer-to-peer target
0	1	X – Don't Care	Redirect Upstream
1	0	1	Handle as an ACS Violation
1	0	0	Route directly to peer-to-peer target
1	1	1	Redirect Upstream
1	1	0	Route directly to peer-to-peer target

6.12.4 ACS Violation Error Handling

ACS Violation 可能是由于硬件或软件缺陷/故障引起的。为了帮助进行故障隔离和根本原因分析，建议在 ACS 组件中实现 AER。AER prefix/header logging 和 Prefix Log/Header Log 寄存器可用于确定有问题的请求的 prefix/header。ACS Violation Status、Mask 和 Severity 字段提供对错误的标识，并增强了对错误记录和信令的控制。

当检测到 ACS 违规时，充当 Completer[85] 的 ACS 组件必须执行以下操作：

- 对 Non-Posted Request，Completer 必须生成一个 Completion Status 为 Completer Abort（CA）的 Completion。
- Completer 必须记录日志并发信号通知 ACS Violation，如 Figure 6-2 所示。请注意以下几点：
  - 即使 Completer 在发送 Completion 时使用 CA 完成状态，Completer 也必须记录 ACS Violation 错误，而不是 Completer Abort 错误。
  - 如果 ACS Violation 的严重程度不是致命的，并且 Completer 发送具有 CA 完成状态的 Completion，则此情况必须按照 Section 6.2.3.2.4.1 中所述作为 Advisory Non-Fatal Error 错误进行处理。
- Completer[86] 必须在其 Status 寄存器或 Secondary Status 寄存器中适当地设置 Signaled Target Abort 字段。

注 85：除一种情况外所有检测到 ACS Violation 的 ACS 组件也充当 Completer。例外情况是当 RC Redirected Request Validation 逻辑禁止重定向请求时。如果重定向的请求通过一个 Root Port，则该 Root Port 必须作为 Completer 运行。如果重定向的请求来自 RC Integrated Endpoint，则关联的 Root Complex Event Collector 必须充当 Completer。

注 86：同样，如果请求是 Non-Posted 的，则当请求者收到带有 CA 完成状态的完成时，请求者必须在其 Status 寄存器或 Secondary Status 寄存器中适当地设置 Signaled Target Abort 字段。请注意，对于 multi-Function 设备因其 Function 之间的对等请求而发生 ACS Violation 错误的情况，同一 Function 可能同时充当请求者和完成者。

6.12.5 ACS Redirection Impacts on Ordering Rules

启用 ACS P2P Request Redirect 后，某些或所有对等请求将被重定向，在某些情况下，这可能会导致违反排序规则。本节探讨了这些情况，以及在 RC 中实现"Request Retargeting"作为强制执行对等访问控制的替代机制的类似情况。

6.12.5.1 Completions Passing Posted Requests

当一个 peer-to-peer Posted Request 被重定向时，直接路由的后续的 peer-to-peer non-RO Completion 可以有效地传递重定向的 Posted Request，这违反了 non-RO Completion 不得传递 Posted Request 的排序规则。有关更多信息，请参见 Section 2.4.1。non-RO

指的是 TLP 的 Relaxed Ordering Attribute 字段没有设置为 1b。

ACS P2P Completion Redirect 可用于避免违反此排序规则。启用 ACS P2P Completion Redirect 后，所有 peer-to-peer non-RO Completion 都将被重定向，从而采用与重定向的 peer-to-peer Posted Request 相同的路径。在直接路由 peer-to-peer Posted Request 时，启用 ACS P2P Completion Redirect 不会导致任何排序规则冲突，因为除具有相同 Transaction ID 的另一个 Completion 之外，任何 TLP 都可以传递给定的 Completion。

作为用于执行对等访问控制的 ACS P2P Request Redirect 的替代机制，一些 RC 实现了"Request Retargeting"，其中 RC 为"peer-to-peer"传输实现了一个特殊地址范围，并且 RC 将把验证过的上游请求重定向到对等设备。 收到针对特定地址范围的上游请求后，RC 会验证请求，将地址转换为目标对等设备，然后将请求发送回下游。对于 Non-posted 的重定向请求，如果 RC 不修改 Requester ID，则 Completion 的结果直接将对等事务返回原始请求者，从而导致 non-RO Completion 有效地传递重定向的可能性，这违反与使用 ACS P2P Request Redirect 时相同的排序规则的请求。ACS P2P Completion Redirect 也可以用来避免此类违反排序规则的情况。

如果未使用 ACS P2P Request Redirect 和 RC P2P Request Retargeting，则启用 ACS P2P Completion Redirect 没有预期的好处，并且建议不要这样做，因为这可能会影响性能。

Performance Impacts with ACS P2P Completion Redirect

尽管使用 ACS P2P Completion Redirect 可以避免通过传递 Posted Request 的 Completion 导致的排序违规，但它也可能会影响性能。具体来说，所有重定向完成都必须从重定向点返回到 RC，这会带来额外的延迟，并可能增加链路和 RC 的拥塞。

由于永远不会重定向设置了 Relaxed Ordering 字段的 peer-to-peer Completion（因此避免了性能影响），因此强烈建议请求者以最大程度地正确使用 Relaxed Ordering，并且该软件可以设置 Device Control 寄存器中的 Enable Relaxed Ordering 字段使请求者可以通过以下方式使用 Relaxed Ordering。

如果软件启用了 ACS P2P Request Redirect、RC P2P Request Retargeting 或同时启用这两种功能，并且软件确定 peer-to-peer[88] Posted Request 的 peer-to-peer non-RO Completion 不会损害正常操作，则建议该软件禁用 ACS P2P Completion Redirect，以避免其性能受到影响。

注 88：这些通过 ACS P2P Request Redirect 机制重定向的 peer-to-peer Request，以及路由到 RC 的"logically peer-to-peer" Request，然后 RC 将其重定向到对等设备。

6.12.5.2 Requests Passing Posted Requests

当某些 peer-to-peer Request 被重定向而其他 peer-to-peer Request 被直接路由时，存在违反 Non-posted Request 或 non-RO Posted Request 不得传递 Posted Request 的排序规则的可能性。有关更多信息，请参见 Section 2.4.1。

仅当同时启用 ACS P2P Request Redirect 和 ACS Direct Translated P2P 时，才存在这些违反排序规则的可能性。 除非可以确定不会发生此类排序规则冲突，否则，软件不应启用这两种机制，或者如果确实发生此类排序规则冲突，则不会损害正常的操作。

Ensuring Proper Operation with ACS Direct Translated P2P

ACS Direct Translated P2P 的目的是在将地址转换服务（ATS）与对等通信（其访问控制由 RC 实施）一起使用的环境中优化性能。允许直接发送具有转换地址的对等请求可以避免与重定向相关的性能影响，这会带来额外的延迟，并可能增加链路和 RC 拥塞。

对于允许使用 ATS 的 peer-to-peer Request 但将 non-ATS 的 peer-to-peer Request 作为 ACS Violation 而被阻塞的使用模型，建议软件启用 ACS Direct Translated P2P 和 ACS P2P Request Redirect，并配置 RC 中的 Redirected Request Validation 逻辑以阻塞具有未翻译地址的重定向请求。此配置没有与传递 Posted Request 的请求关联的排序规则违规。

对于这样的使用模型：其中一些请求者仅将 peer-to-peer Request 与 ATS 一起使用，而某些请求者仅将 peer-to-peer Request 与 non-ATS 一起使用，并且两类请求者之间不会进行对等通信，通过直接的 peer-to-peer Request（具有转换的地址）传递重定向对等请求（具有未转换的地址）不太可能损害正常操作。建议除非软件确定不会因所导致的排序规则冲突而损害正常操作，否则

不要启用 ACS Direct Translated P2P。

对于单个请求者同时使用带 ATS 的和不带 ATS 的 peer-to-peer Request 的使用模型，再次建议软件不要启用 ACS Direct Translated P2P，除非软件确定正确的操作不会因所导致的排序规则冲突而受到损害。这需要对正在使用的对等通信模型进行详细分析，这超出了本规范的范围。

6.13 Alternative Routing-ID Interpretation (ARI)

Routing ID、Requester ID 和 Completer ID 一般都是由三个字段组成的 16 比特标识符组成：8-bit Bus Number、5-bit Device Number 和 3-bit Function Number。使用 ARI 技术后，将 16 比特标识符解释为两个字段：8-bit Bus Number 和 8-bit Function Number。这种新的解释使 ARI 设备最多支持 256 个 Function，而不是 8 个 Function。

ARI 由一组新的可选功能和控制寄存器控制。他们提供以下功能：

- 软件必须能够检测组件是否支持 ARI。
- 软件具有配置下游端口的 ARI 的能力，因此确定何时将 Type 1 Configuration Request 转换为 Type 0 Configuration Request 的逻辑不再强制传统 Device Number 字段为 0。
- 软件具有配置 ARI 设备以将每个 Function 分配给一个 Function Group 的能力。当不需要基于各个 Function 的更精细的控制时，基于 Function Group 的控制可能更可取。
  - 如果支持并启用 Multi-Function VC arbitration，则可以选择基于 Function Group 而不是单个 Function 进行仲裁。
  - 如果支持并启用了 ACS P2P Egress Control，则访问控制可以选择基于 Function Group 而不是单个 Function。

下面说明了启用这些功能的示例流程，并提供了有关其用法的更多详细信息：

1. 软件会枚举 PCI Express 层次结构，并确定是否支持 ARI 功能。
   a. 对于一个 ARI 下游端口，该功能通过 Device Capabilities 2 寄存器进行确定。
   b. 对于 ARI 设备，该功能通过 ARI Capability structure 进行确定。
   c. ARI 对当今平台中使用的基本枚举算法没有影响。
2. 软件可在每个组件中启用 ARI 功能。
   a. 在 ARI 设备正上方的 ARI 下游端口中，软件会将 Device Control 2 寄存器中的 ARI Forwarding Enable 字段置 1。设置此字段可确保确定何时将 Type 1 Configuration Request 转换为 Type 0 Configuration Request 的逻辑不再强制将传统 Device Number 字段设置为 0。
   b. 在 ARI 设备中，始终隐式启用 Extended Function。但是，一旦 ARI 感知软件在 ARI 设备正上方的下游端口中启用 ARI 转发，则 ARI 感知软件必须发现并配置 Extended Function 。
   c. 如果 ARI 设备通过 Function arbitration 实现 Multi-Function VC Capability structure，并且还实现 MFVC Function Group，则 ARI 感知软件会将 Function 分类到 Function Group 中。
      1) 每个 Function 被分配给以 Function Group Number 表示的 Function Group。
      2) 最大可以配置 8 个 Function Group。
      3) 在 Multi-Function VC Arbitration Table 中，在每个仲裁时隙中使用 Function Group Number 代替 Function Number。
         1. 仲裁基于 Function Group 而不是单个 Function 进行。

2. Multi-Function VC Arbitration 的所有其他方面均保持不变。有关更多详细信息，请参见 Section 7.18.10。

4) 每个 Function Group 内的 Function arbitration 是特定于实现的。

d. 如果 ARI 设备支持 ACS P2P Egress Control，则可以选择实现在 Function Group 的基础上实施访问控制。

e. 为了提高枚举性能并创建更具确定性的解决方案，软件可以通过 Function Number 的链表来枚举 Function。下一个链表元素通过每个 Function 的 ARI Capability 寄存器进行确定。

1) Function 0 充当 Function Number 的链表的开头。软件将 ARI Capability 寄存器中非零的 Next Function Number 字段检测为链表中的下一 Function。软件使用设备捕获的 Bus Number 和从 ARI Capability 寄存器派生的 Function Number 发出配置尝试，以定位下一个相关 Function 的配置空间。

2) Function Number 在 ARI 设备的使用中可能是稀疏的，也可以是非顺序的。

对于 ARI 设备，必须将每个 Function 的 Device Capabilities 寄存器的 Phantom Functions Supported 字段（请参见 Section 7.8.3 和 Table 7-12）设置为 00b，以指示不支持 Phantom Function。Extended Tag Field Enable 字段（参见 Table 7-13 和 Section 7.8.4）仍可用于使每个 Function 支持最多 256 个 outstanding Request。

Figure 6-13 显示了一个示例系统拓扑，其中包含两个 ARI 设备，一个在 Root Port 下方，另一个在 Switch 下方。要访问 ARI Device X 的 Extended Function， Root Port A 必须支持 ARI 转发并由软件启用。要访问 ARI Device Y 中的 Extended Function，Switch 的 Downstream Port D 必须支持 ARI 转发并由软件启用。使用此配置，建议软件不要在 Root Port B 或 Switch 的 Downstream Port C 中启用 ARI 转发。

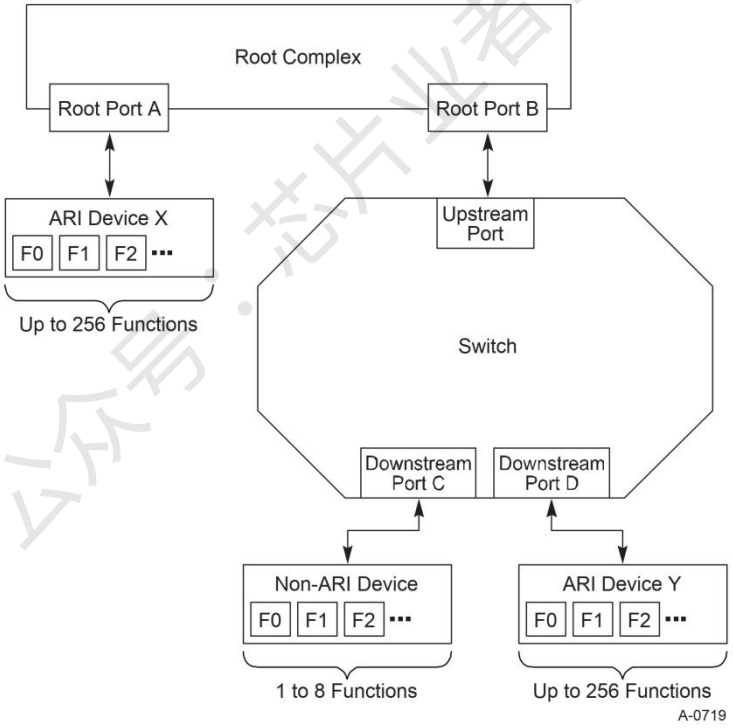

Figure 6-13: Example System Topology with ARI Devices

ARI Forwarding Enable Being Set Inappropriately

强烈建议仅在软件确定下游端口下方的设备是 ARI 设备时，才在下游端口中设置 ARI Forwarding Enable 字段。如果在有 non-ARI 设备的情况下将该字段置 1，则该 non-ARI 设备可以解释为不同 Device Number，并用于响应配置空间访问，而且这相当于其 Function 在多个 Device Number 下使用了别名，通常这会导致不期望的行为。

在下游端口下方发生热插拔事件之后，强烈建议软件清除下游端口中的 ARI Forwarding Enable 字段，直到软件确定新添加的组件

实际上是 ARI 设备为止。

ARI Forwarding Enable Setting at Firmware/OS Control Handoff

强烈建议在控制权移交给操作系统时，固件不要将下游端口的 ARI Forwarding Enable 字段设置为 1，除非固件知道操作系统支持 ARI。此字段设置为 1 后，不支持 ARI 的操作系统可能能够发现并枚举下游端口下方 ARI 设备中的 Extended Function，但是这样的操作系统通常将无法成功管理 Extended Function，因为它将把这种情况解释为下游端口下方有多个设备，而不是单个 ARI 设备。比如可能造成影响的一个示例：与 INTx 虚拟线绑定的中断将与不支持 ARI 感知的操作系统期望的不一致。

6.14 Multicast Operations

Multicast Capability structure 定义了一个多播地址范围，将该范围分割为数量相等的 N 个多播窗口，并将每个多播窗口与一个 Multicast Group（MCG）关联。组件中支持多播的每个 Function 都实现了一个 Multicast Capability structure，该 structure 为通过或到达该 Function 的 TLP 的每个 MCG 提供路由方向和权限检查。 Multicast Group 是一个宽度最多 6-bit 的字段，嵌入在从 MC_Index_Position 开始的地址中，如 Section 7.21.4 中所定义。

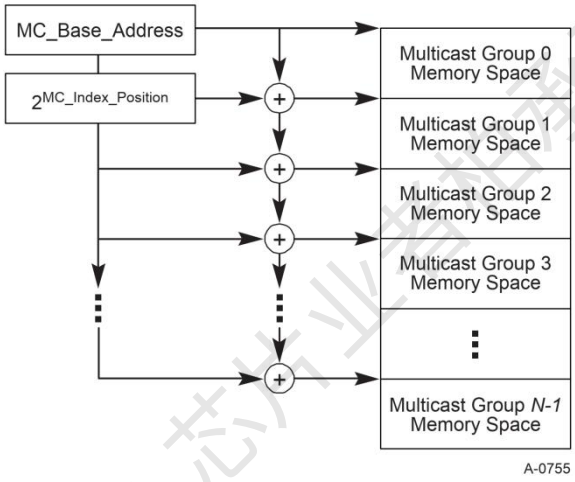

Figure 6-14: Segmentation of the Multicast Address Range

6.14.1 Multicast TLP Processing

如果下述所有条件都满足则会发生一盒多播命中（Multicast Hit）：

- MC_Enable 字段设置为 1。
- TLP 是一个 Memory Write 或 Address Routed Message，两者均为 Posted Request。
- $Address_{TLP}$ >= MC_Base_Address 。
- $Address_{TLP}$ < (MC_Base_Address + ($2^{MC\_Index\_Position}$ * (MC_Num_Group + 1)))

在此步骤中，每个 Switch 的 Ingress Port 和其他组件都使用其任何 Function 中的 MC_Enabl、MC_Base_Address、MC_Index_Position 和 MC_Num_Group 的值。需要软件来配置 Switch 的所有 Function 和 Multi-Function 所有 Function 的 Upstream Port，以使这些字段中的每个字段具有相同的值，如果不是这种情况，则结果不确定。

如果 Non-Posted Memory Request 中的地址在多播窗口中命中，则不会发生多播命中，并且将按照基本规范（即，作为单播）正常处理 TLP。

如果发生多播命中，则唯一仍可应用的 ACS 访问控制是 ACS Source Validation。特别是，无论是 ACS redirection 还是 ACS Egress

Control vector 都不会影响多播匹配期间的操作。

如果发生多播命中，则正常的地址路由规则不适用。相反，TLP 的处理如下：

使用任意 Function 的 MC_Base_Address 和 MC_Index_Position 值从 TLP 中的地址中提取 Multicast Group。具体算法为：

MCG = ((AddressTLP – MC_Base_Address) >> MC_Index_Position) & 3Fh

在此过程中，组件可以将任何 Function 的 MC_Base_Address 和 MC_Index_Position 值。使用哪个 Function 的值是设备指定的。

组件接下来检查与提取的 MCG 对应的 MC_Block_All 和 MC_Block_Untranslated 字段。Switch 和 Root Port 使用与 Ingress Port 关联的 MC_Block_All 和 MC_Block_Untranslated 寄存器检查其 Ingress Port 中的多播 TLP。Endpoint Function 使用其 MC_Block_All 和 MC_Block_Untranslated 寄存器检查它们准备发送的多播 TLP。如果设置了与提取的 MCG 对应的 MC_Block_All 字段，则将 TLP 视为 MC Blocked TLP。如果设置了与提取的 MCG 对应的 MC_Block_Untranslated 字段，并且 TLP 包含未翻译的地址，则 TLP 也将作为 MC Blocked TLP 处理。

MC_Block_Untranslated and PIO Writes

因为没有一种软件架构可以控制 Address Type（AT）字段发出 PIO 请求，因此对内存空间的 Programmed I/O (PIO) Write 通常有未转换的地址。因此，如果给定的 Switch 需要组播任何 PIO 写操作，则软件应确保该 Switch 的 Upstream Port 中的相应 MC_Block_Untranslated 为清除状态。否则，Switch 的 Upstream Port 可能会阻塞合理的命中多播窗口的 PIO 写入。出于 PIO Write 的目的，软件可能需要清除 Switch 的 Upstream Port 中的 MC_Block_Untranslated 字段，因此强烈建议对具有地址转换功能的 RC 执行以下操作：

- 所有 Integrated Endpoint 均实现一个 Multicast Capability structure，以提供用于发送未翻译的组播 TLP 的访问控制。
- 所有具有对等访问功能的根端口均实现 Multicast Capability structure，以为转发给对等访问的未转换多播 TLP 提供访问控制。

出于类似的原因，对于具有多播功能的 Switch 组件（其中 Upstream Port 是 multi-Function 设备中的 Function），强烈建议该 multi-Function 设备中的所有 Endpoint 均实现 Multicast Capability structure。

Multicast Window Size

组播 TLP 的每个最终接收者可能具有不同的组播窗口大小要求。在一个极端情况下，可能需要多播窗口来覆盖设备内实现的一系列内存。另一方面，它可能仅需要覆盖 FIFO 寄存器所在的特定偏移量。 Endpoint 使用 Multicast Capability 寄存器中的 MC_Window_Size_Requested 字段来通告其所需的 Multicast Window 的大小。

除非可用地址空间受到限制，否则资源分配软件将能够将每个请求视为最小，并通过 MC_Index_Position 设置多播窗口大小以容纳最大的请求。 在某些情况下，可以通过配置较小的窗口大小并将相同的成员资格分配给多个连续的 MCG 来满足较大窗口大小的请求。

Multicast, ATS, and Redirection

ACS P2P Request Redirection 和 ACS Direct Translated P2P 机制提供了一种方法，其中可以将携带未转换地址的 P2P 请求重定向到 RC（RC）进行访问控制检查，而可以将携带转换地址的 P2P 请求“直接”路由到其 P2P 的目标以提高性能。组播 TLP 没有相应的重定向机制。

为了实现类似的功能，可以将 RC 配置为提供不在多播地址范围内的一个或多个目标内存空间范围，但是 RC 映射到“受保护”的多播窗口。具有或不具有 ATS 功能的多播 TLP 发送者然后将这些 RC 内存空间范围作为目标，以便间接访问受保护的多播 Windows。当任一类型的发送者通过“存储器写”将这些范围作为目标时，满足访问控制检查的每个 TLP 都将由 RC 向下反射，其转换地址以受保护的“组播窗口”为目标。具有 ATS 功能的发送者可以使用 RC 内存空间范围来请求和缓存转换后的地址，然后将这些转换后

的地址用于直接针对受保护的多播 Windows 的内存写入，并且可以在不经过 RC 的情况下进行多播。

对于强制使用转换地址直接针对受保护的多播窗口的硬件实现，软件在整个平台的所有适用功能中的 MC_Block_Untranslated 寄存器中设置相应的 MCG 字段。设置了每个 MCG 的字段，将导致其关联的多播窗口受到保护，以免使用未转换的地址直接访问。

如果 TLP 没有在 Switch 或 RC 中被阻塞，则将其转发给所有所有与提取的 MCG 对应的 MC_Receive 字段设置为 1 的端口（Ingress Port 除外）。在一个 Endpoint 中，所有与提取的 MCG 对应的 MC_Receive 字段设置为 1 的 Function 都会接收。如果没有端口转发 TLP 或没有 Function 使用它，则 TLP 将被静默丢弃。

为了防止循环，即使根端口或 Switch 的端口由与该端口关联的 MC_Receive 寄存器指定，也禁止根端口或 Switch 的端口将 TLP 转发回其 Ingress Port。前面的实现说明中描述的情况是一个例外，其中 RC 反映了从 Ingress Root Port 进入多播窗口的单播 TLP。在这种情况下，当由与该 Ingress Root Port 关联的 MC_Receive 寄存器指定时，要求 RC 将反射的 TLP 发送到它最初进入的相同根端口。

组播命中会暂停正常的地址路由，包括 Switch 中的默认上游路由。当发生多播命中时，TLP 将被转发到与从 TLP 的地址中提取的 MCG 关联的 MC_Receive 设置为 1 的 Egress Port。如果 TLP 中的地址没有使用常规地址解码器解码为向任何 Downstream Port 传输，则只有在 Upstream Port 的 MC_Receive 寄存器指定的情况下，TLP 才会复制到 Upstream Port。

6.14.2 Multicast Ordering

没有定义新的排序规则来处理多播 TLP。所有多播 TLP 都是 Posted Request，并遵循 Posted Request 排序规则。相对于组件从复制点进入的流中的其他 TLP，按常规排序规则对多播 TLP 进行排序。一旦复制到出口流中，组播 TLP 就会与流中的其他 Posted Request 遵循相同的顺序。

6.14.3 Multicast Capability Structure Field Updates

Multicast Capability Structure 的某些字段可能随时更改。除非组件的每个 Function 中的 MC_Enable 字段清为 0，否则其他结果将无法更改，并具有可预测的结果。后一组包括 MC_Base_Address 和 MC_Index_Position。

可以随时随软件更改的字段包括 MC_Enable、MC_Num_Group、MC_Receive、MC_Block_All 和 MC_Block_Untranslated。这些字段的更新需要本身安排好。例如，考虑按顺序到达相同 Ingress Port 和有相同 TC 的 TLP A 和 TLP B。如果 TLP A 对这些字段之一使用值 X，则 B 必须使用相同的值或较新的值。

对于 Multi-Function Upstream Switch Port 的 Multicast TLP，由一个 Switch 接收或由一个 Endpoint Function 发送的 Multicast TLP 将呈现给其他并行的 Endpoint Function 以及其他并行 Switch 的 Downstream Switch Port（如果 Function 在同一 Switch 中，则视为并行）设备。当任何 Upstream Switch Function 把 MC_Receive 字段设置为 1 时，单个组播 TLP 就会向上游转发。

6.14.4 MC Blocked TLP Processing

当 TLP 被 MC_Block_All 或 MC_Block_Untranslated 机制阻塞时，该 TLP 将被丢弃。阻塞 TLP 的 Function 充当完成者。完成者必须记录并发信号通知此 MC Blocked TLP 错误，如 Figure 6-2 所示。此外，完成者必须在其 Status 寄存器或 Secondary Status 寄存器中设置 Signaled Target Abort 字段。为了帮助进行故障隔离和根本原因分析，强烈建议在具有组播功能的 Function 中实现 AER。

在 RC 和 Switch 中，如果 Ingress Port 接收到的 TLP 发生错误，则该 Ingress Port 将报告错误。如果在准备发送 TLP 的 Endpoint Function 中发生错误，则该 Endpoint Function 将报告错误。

公众号：芯片业者相承及

6.14.5 MC_Overlay Mechanism

提供了 MC Overlay 机制，以允许不包含 Multicast Capability structure 的 Endpoint 中的单个 BAR 用于多播和单播 TLP 的接收。软件可以通过在 Downstream Port 中设置 MC_Overlay_BAR 来配置 MC_Overlay 机制，以启用此效果，以便将多播地址范围或其中的一部分重新映射（覆盖）到 Endpoint 的 BAR 所接受的内存空间范围。在 Switch 的 Upstream Port，该机制可用于将一部分组播地址范围覆盖到与主机内存关联的内存空间范围内。

Downstream Port 的 MC_Overlay 机制适用于离开该端口的 TLP。Upstream Port 的 MC_Overlay 机制适用于离开 Switch heading Upstream 的 TLP。端口的 MC_Overlay 机制不适用于该端口接收的 TLP，也不适用于目标为该端口内内存空间的 TLP，同样也不适用于 Multi-Function Upstream Port 中 Function 之间的对等路由的 TLP。

启用后，覆盖操作将指定多播 TLP 地址中的字段（其比特数等于或高于 MC_Overlay_Size 字段）被 MC_Overlay_BAR 中的相应字段替换。即：

If (MC_Overlay_Size < 6)

Then Egress_TLP_Addr = Ingress_TLP_Addr;

Else Egress_TLP_Addr = {MC_Overlay_BAR[63:MC_Overlay_Size], Ingress_TLP_Addr[MC_Overlay_Size-1:0]};

如果 AT 为 1 的 TLP 包含可选的 ECRC，则几乎可以肯定的是，未修改的 ECRC 将指示错误。如果将包含 ECRC 的 TLP 组播复制到已启用 MC_Overlay 的 Egress Port，则应采取的措施取决于是否实现了对 ECRC 再生的可选支持。Table 6-10 概述了所有措施。如果未启用 MC_Overlay，则将 TLP 未经修改地转发。如果启用了 MC_Overlay，并且 TLP 没有 ECRC，则将修改该 TLP（地址替换为上一段中的指定地址），然后转发该 TLP。如果 TLP 具有 ECRC，但不支持 ECRC 再生，则转发已修改的 TLP 并丢弃 ECRC，并且清除 header 中的 TD 位以指示未附加 ECRC。如果 TLP 具有 ECRC 并且支持 ECRC 再生，则在转发 TLP 之前将执行 ECRC 检查。如果通过了 ECRC 检查，则将 TLP 与重新生成的 ECRC 一起转发。如果 ECRC 检查失败，则将 TLP 与反向生成的 ECRC 一起转发。

Table 6-10: ECRC Rules for MC_Overlay

MC_Overlay Enabled	TLP has ECRC	ECRC Regeneration Supported	Action if ECRC Check Passes	Action if ECRC Check Fails
No	x	x	Forward TLP unmodified	
Yes	No	x	Forward modified TLP	
Yes	Yes	No	Forward modified TLP with ECRC dropped and TD bit clear	
Yes	Yes	Yes	Forward modified TLP with regenerated ECRC	Forward modified TLP with inverted regenerated ECRC

MC_Overlay and ECRC Regeneration

Switch 和 RC 端口可以选择支持 ECRC 再生。如果支持 ECRC 再生，则强烈建议通过尽量减少检查原始 TLP 的 ECRC 与将其替换为在修改的 TLP 上计算的 ECRC 之间的时间来稳健地这样做。在此期间，TLP 不受保护，如果在同一流水线阶段未完成预检查和重新生成，则会留下数据完整性漏洞。

通过启用了 MC_Overlay 但不支持 ECRC 再生的 Endpoint 从多播 TLP 中剥离 ECRC 允许接收 Endpoint 启用 ECRC 检查。 在这种情况下，Endpoint 将享受到非多播 TLP 上的 ECRC 的好处，而无需检测到由 MC_Overlay 机制修改的多播 TLP 上的 ECRC。

如果支持多播 ECRC 再生，并且在修改 TLP 之前检测到 ECRC 错误，则对再生的 ECRC 进行反相可确保该再生过程不会掩盖 ECRC 错误。

Multicast to Endpoints That Don’t Have Multicast Capability

不包含 Multicast Capability structure 的 Endpoint Function 无法区分多播 TLP 和单播 TLP。系统设计人员可能会利用这一事实，将 Endpoint 用作多播目标。这样做的主要要求是，将设备上方的 Switch 端口中 virtual PCI to PCI Bridge 的基址和限制寄存器配置为与多播地址范围的至少一部分重叠，就像采用了 MC_Overlay 机制一样。再扩展一下这种推理，甚至有可能在 PCI Express to PCI/PCI-X Bridge 的 PCI/PCI-X 侧上放置单个多播目标 Function。

如果将没有 Multicast Capability structure 的 Endpoint 用作多播目标，而未使用 MC_Overlay 机制，则可能有必要使用与多播 TLP 相同的地址从 Endpoint 的存储空间中读取。因此，在“多播窗口”中命中的内存读取不一定是错误。根据标准地址路由规则，在多播窗口中命中且在 RC Integrated Endpoint 或 Switch 的 Downstream Port 中也未命中的内存读取将被向上路由，并在此处作为 UR 处理。

Multicast in a RC

如果一个 RC 有多个支持多播的根端口，则可以实现尽可能多的 Multicast Capability structure。如果实现了多个，则软件应确保 Section 6.14.3 中指定的某些字段配置相同。为了支持多播到 RC Integrated Endpoint，实现需要将通过 MC_Base_Address 寄存器标识为多播的所有 TLP 暴露给其中集成的所有潜在多播目标 Endpoint。然后，每个此类 Integrated Endpoint 都会使用其 Multicast Capability structure 中的 MC_Receive 寄存器来确定是否应接收 TLP。

Multicast and Multi-Function Devices

所有潜在的多播目标 Port Function 和 Endpoint Function 都需要实现 Multicast Capability structure，以便每个都有自己的 MC_Receive 向量。在单个组件内，软件应相同地配置这些 Multicast Capability structure 的 MC_Enable、MC_Base_Address、MC_Index_Position 和 MC_Num_Group 字段。在这种情况下，仅在组件中的多播 BAR 的一个实例上实现地址解码逻辑就足够了。

Congestion Avoidance

组播的使用将 Switch 的输出链路利用率提高到与所使用的组播组的大小以及组播业务占总业务的比例成比例的程度。使用组播时，这会增加拥塞和拥塞扩散的风险。

为了减轻这种风险，应将旨在用作多播目标的组件设计为以线速使用多播 TLP。打算用作多播源的组件应考虑添加速率限制机制。

在许多应用程序中，应用程序的多播数据流将具有固有的速率限制，并且可以容纳而不会引起拥塞。其他人将需要一种明确的机制来限制注入速率，选择具有足以容纳必需的多播业务突发而无需使用流量控制的缓冲区的 Switch，或者选择能够以所需速率下沉多播流量的多播目标组件。系统设计者有责任选择适当的机制和组件来为应用程序服务。

The Host as a Multicast Recipient

对于通用系统，预计通常不会将多播地址范围配置为与直接映射到主机内存的内存空间重叠。如果要将主机内存作为多播接收者包括在内，则 RC 可能需要具有某种 I/O 内存管理单元（IOMMU），该单元能够将多播窗口的某些部分重新映射到主机内存，也许具有页面级粒度 。或者，可以使用 Switch 的上游端口中的 MC_Overlay 机制将多播地址范围的一部分覆盖到主机存储器上。

对于缺少 IOMMU 的嵌入式系统，可以配置与直接映射到主机内存的内存空间重叠的多播窗口，从而避免使用 IOMMU。这种方法的具体细节超出了本规范的范围。

公众号·芯片业者柏承双

6.15 Atomic Operations (AtomicOps)

原子操作（AtomicOp）是单个 PCI Express 事务，它以内存空间中的某个位置为目标，读取该位置的值，有可能将新值写回到该位置，并返回原始值。该位置的“读取-修改-写入”过程是原子执行的。AtomicOps 包括以下内容：

- FetchAdd (Fetch and Add)：该请求包括单个操作数，即要 add 的值。
  - 从目标位置读取值。
  - 使用二进制补码运算法则把该读取值加上要“加”的值，而忽略任何进位或溢出。
  - 把加后的结果写回目标位置。
  - 返回最初从目标位置读取的值。
- Swap (Unconditional Swap)：该请求包括单个操作数，即要 swap 的值。
  - 从目标位置读取值。
  - 把交换后的值写入该目标位置。
  - 返回最初从目标位置读取的值。
- CAS (Compare and Swap)：该请求包括两个操作数，一个 Compare 值和一个 swap 值。
  - 从目标位置读取值。
  - 把该读取值与要“比较”的值作比较。
  - 如果相等，则把要“交换”的值写入目标位置。
  - 返回最初从目标位置读取的值。

给定的 AtomicOp 事务具有相关的操作数大小，并且目标位置访问和返回值使用相同的操作数大小。 FetchAdd 和 Swap 支持 32 和 64 位的操作数大小。CAS 支持 32、64 和 128 位的操作数大小。

AtomicOp 功能是可选的规范。允许 Endpoint 和 Root Port 实现 AtomicOp Requester 的功能。具有内存空间 BAR 和所有 Root Port 的 PCI Express Function 允许实现 AtomicOp Completer 功能。路由元素（Switch 以及支持 Root Port 之间对等访问的 RC）需要实现 AtomicOp 路由功能才能路由 AtomicOp 请求。 AtomicOps 是为设备到主机，设备到设备以及主机到设备事务而设计的。在每种情况下，请求者，完成者和所有中间路由元素都必须支持关联的 AtomicOp 功能。

PCI Express to PCI/PCI-X Bridge 不支持 AtomicOp 功能。如果需要，锁定事务可用于此类桥下的设备。 AtomicOps 和锁定事务可以在同一层次结构上同时运行。

软件通过 Device Capabilities 2 寄存器中的三个新的字段枚举特定的 AtomicOp Completer 功能（请参见 Section 7.8.15）。为了提高互操作性，需要 Root Port 来实现某些 AtomicOp Completer 功能（如果有的话）（请参阅 Section 6.15.3.1）。软件通过 Device Capabilities 2 寄存器中的 AtomicOp Routing Supported 字段来枚举 AtomicOp 路由功能。软件对 AtomicOp Requester 功能的枚举不在本规范的范围内，但是软件必须在 Function 可以启动 AtomicOp 请求之前将 Function 的 Device Capabilities 2 寄存器中的 AtomicOp Requester Enable 字段设置为 1（请参见 Section 7.8.16）。

借助路由元素，软件可以逐个端口设置 AtomicOp Egress Blocking 字段（请参见 Section 7.8.16），以避免 AtomicOp 请求转发至不应接收的组件，并且可能会将每个 AtomicOp 请求处理为 Malformed TLP，默认情况下这是致命错误（Fatal Error）。每个被阻塞的请求都被视为一个 AtomicOp Egress Blocked error ，默认情况下这被视为是一个 Advisory Non-Fatal Error。

AtomicOps 是内存事务，因此应用于管理内存空间访问的现有标准机制（例如，Bus Master Enable、Memory Space Enable 和 Base

公众号：芯片业者相承双

Address 寄存器）。

6.15.1 AtomicOp Use Models and Benefits

AtomicOps 支持高级同步机制，当有多个生产者和/或多个消费者需要以非阻塞方式进行同步时，该机制特别有用。例如，多个生产者可以在没有任何显式锁定的情况下安全地加入一个公共队列。

AtomicOps 还支持与锁无关的统计计数器，例如，设备可以原子地递增计数器，而主机软件可以原子地读取和清除计数器。

PCI Express 支持三个 AtomicOps，可以更轻松地将现有的高性能 SMP 应用程序迁移到使用 PCI Express 作为紧耦合的加速器（accelerator），协处理器（co-processor）或 GP-GPU 互连的系统。例如，使用 PCI Express 加速器的移植应用程序可能使用与早期 SMP 应用程序相同的同步算法和数据结构。SMP 指对称多处理器。

通常，针对给定目标的 AtomicOp 操作所产生的延迟时间与对同一目标的内存读取相当。在单个层次结构中，可以同时“运行”多个 AtomicOps。AtomicOps 通常对其他 PCI Express 业务造成的影响可忽略不计。

与锁定事务相比，AtomicOps 提供更低的延迟，更高的可伸缩性，先进的同步算法，并且对其他 PCI Express 业务的影响显着减少。

6.15.2 AtomicOp Transaction Protocol Summary

有关 AtomicOps 的详细协议规则和要求在本规范的其余部分中进行了阐述，这里给出一些简短的总结以及一些特殊的要求。

- AtomicOp 是 Non-Posted Memory Transaction，支持 32 bit 和 64 bit 地址格式。
- FetchAdd、Swap 和 CAS 的编码类型不同。
- Completer 从 Length 字段值和 AtomicOp 请求中的类型编码推断操作数大小。
- AtomicOp 的 Completer 用于在目标位置读取和写入的数据的字节序格式是特定于实现的，它可以是 Completer 指定的适合目标内存的任何形式（例如，little-endian、big-endian 等）。参见 Section 2.2.2。
- 如果 AtomicOp 请求者支持地址转换服务（ATS），则仅当翻译的地址具有适当的访问权限时，才允许请求者在 AtomicOp 请求中使用翻译的地址。具体来说，必须把 Read (R)和 Write (W)字段设置为 1，并且 Untranslated access only (U)字段必须清 0。参见 Section 2.2.4.1。
- 如果支持访问控制服务（ACS）的组件支持 AtomicOp 路由或 AtomicOp 请求者功能，则在遵循 ACS 功能的前提下，该组件处理 AtomicOp Request 和 AtomicOp Completion 与处理 Memory Request 和 Memory Completion 的方式相同。
- No Snoop 属性可以应用于 AtomicOp，但是无论 No Snoop 属性值如何，都必须保证原子性。
- Relaxed Ordering 属性可以应用于 AtomicOp，它会影响 Request 及其 Completion 的顺序。
- AtomicOp Request 的排序要求与 Non-Posted Write Request 一致。因此，如果请求者要确保完成者在后续的 Posted 或 Non-Posted Request 之前检测到 AtomicOp 请求，则请求者必须在发送后续的请求之前等待 AtomicOp Completion。
- AtomicOp Completion 的排序要求与 Read Completion 一致。
- 除非存在更高的优先级错误，否则完成者必须将 Poisoned AtomicOp Request 处理为一个 Poisoned TLP Received error，并且还必须返回一个 Unsupported Request (UR) 的 Completion。See Section 2.7.2.2。目标位置的值必须保持不变。
- 如果 AtomicOp 请求的完成者在访问目标位置或执行原子操作时遇到不可纠正的错误，则完成者必须将其作为 Completer Abort (CA) 处理。目标位置的值是否改变是特定于实现的。

公众号：芯片业者柏承双

- 完成者需要处理任何形式不正确的 AtomicOp 请求，如果这些请求的类型或操作数大小是不支持的，则为该请求返回一个 Unsupported Request (UR)的 Completion。如果 AtomicOp 请求中的 Length 字段包含架构无关的值，则该请求必须作为 Malformed TLP 处理。请参阅 Section 2.2.7。

- 如果一个 RC 的任意 Root Port 支持 AtomicOp 路由功能，则转发的 AtomicOp 请求可访问的 RC 中的 RC Integrated Endpoint 必须对格式正确的 AtomicOp 请求进行解码，并将不支持的任何内容作为 Unsupported Request (UR)处理。

- 如果一个 AtomicOp 请求具有支持的类型和操作数大小，则要求完成者响应该请求或者将其作为 Completer Abort (CA) 处理。允许完成者根据编程模型的需要在其目标存储空间的子集上支持 AtomicOp 请求（请参阅 Section 2.3.1）。除非在结构说明中明确说明，否则不要求将 PCI Express 定义或继承的内存空间（例如 MSI-X 表结构）作为 AtomicOp 的目标位置。

- 对于 Switch 或 RC，当在 Egress Port 中的 AtomicOp Egress Blocking 使能打开，并且 AtomicOp 请求以该 Egress Port 为目标时，Egress Port 必须将该请求作为 AtomicOp Egress Blocked error 处理（请参见 Figure 6-2），并且还必须返回一个 Unsupported Request (UR)的 Completion。如果 AtomicOp Egress Blocked error 的是非致命的，则此情况必须按照 Section 6.2.3.2.4.1 中所述作为 Advisory Non-Fatal Error 处理。

  注：尽管通过返回具有 UR 状态的 Completion 来处理 AtomicOp Egress Blocked error，但该错误不会作为 Unsupported Request 处理。例如，它不会将 Device Status 寄存器中的 Unsupported Request Detected 字段置成 1。

6.15.3 RC Support for AtomicOps

RC 对于 AtomicOp 功能具有独特的要求和考虑因素。

6.15.3.1 Root Ports with AtomicOp Completer Capabilities

Root Port 的 AtomicOp Completer capability 表明，Root Port 支持在其 Ingress Port 接收针对以 Root Port BAR 分配的主机内存或内存空间为目标的 AtomicOp Request。这独立于具有 AtomicOp Completer capability 的任何 RC Integrated Endpoint。

如果根端口实现了用于主机内存访问的任何 AtomicOp Completer 功能，则它必须实现所有 32 位和 64 位 AtomicOp Completer 功能。实现 128 位 CAS Completer 功能是可选的。

如果 RC 具有一个或多个实现了 AtomicOp Completer 功能的根端口，则 RC 必须确保代表给定 AtomicOp 请求对目标位置的主机内存访问是针对每个主机处理器或设备对该目标位置的访问原子地执行的范围。

如果主机处理器通过其指令集体系结构支持原子操作，则 RC 还必须确保代表给定 AtomicOp 请求的主机存储器访问保留任何主机处理器原子操作的原子性。

6.15.3.2 Root Ports with AtomicOp Routing Capability

与其他 PCI Express 事务一样，对 AtomicOp 请求和根端口之间的完成的对等路由的支持是可选的，并且取决于实现。如果 RC 支持两个或多个根端口之间的 AtomicOp 路由功能，则它必须通过 Device Capabilities 2 寄存器中的 AtomicOp Routing Supported 字段来指示每个关联的根端口中的功能。

AtomicOp Routing Supported 字段设置为 1 的 RC 不需要支持成对的 Root Port 之间的 AtomicOp 路由。需要在不支持的 Root Port 之间进行路由的 AtomicOp 请求必须作为 Unsupported Request (UR)处理，并由“发送”端口报告。

如果 Root Port 支持转发由主机软件或 RC Integrated Endpoint 发起的 AtomicOp 请求，则该 Root Port 的 AtomicOp Routing Supported 字段必须设置为 1。如果 Root Port 支持转发 Ingress Port 接收到的 AtomicOp 请求到 RC Integrated Endpoint，则该 Root Port 的 AtomicOp Routing Supported 字段必须设置为 1。

公众号·芯片业者相承双

6.15.3.3 RCs with AtomicOp Requester Capabilities

允许 RC 实现主机软件或 RC Integrated Endpoint 发起 AtomicOp 请求。

AtomicOp Requester 功能的软件枚举不在本规范的范围之内。

如果 RC 支持软件发起的 AtomicOp 请求者功能，则用于在主机处理器上运行的软件如何使 RC 生成 AtomicOp 请求的特定机制不在本规范范围内。

Generating AtomicOp Requests via Host Processor Software

如果主机处理器指令集体系结构（ISA）支持一个或多个与 PCI Express AtomicOps 直接对应的原子操作指令，则 RC 处理针对 PCI Express 内存空间的原子事务，就像它处理内部读事务导致一个处理器加载指令一样。但是，RC 不会将内部读事务作为 PCI Express 内存读取请求“导出”，而是将内部原子事务作为 PCI Express AtomicOp 请求导出。即使 RC 对某些 AtomicOp 类型和操作数大小使用“导出”方法，也无需对所有原子操作都使用此方法。

对于 RC 不使用“导出”方法的 AtomicOp 类型和操作数大小，RC 可能会使用基于 RC 寄存器的机制，就像某些 PCI 主机桥使用 CONFIG_ADDRESS 和 CONFIG_DATA 寄存器来生成配置请求的机制。有关详细信息，请参阅《*PCI Local Bus Specification*》。

“导出”方法可能允许大量并发的 AtomicOp 请求，而不会受到 RC 寄存器的限制。使用此方法从用户空间软件支持 AtomicOp 请求生成也可能会更容易。

即使主机处理器 ISA 不支持相应的原子指令，基于 RC 寄存器的机制也可以为所有 AtomicOp 类型和操作数大小工作。它还可能支持轮询模式以等待 AtomicOp 完成，而不是在等待完成时暂停处理器。

6.15.4 Switch Support for AtomicOps

如果 Switch 的任何端口都支持 AtomicOp 路由功能，则它的所有端口都必须支持。

6.16 Dynamic Power Allocation (DPA) Capability

管理功耗的常用方法是通过设备驱动程序，操作系统和应用程序之间的协商。预计通过这种规范范围之外的软件机制，为此类设备添加动态功率分配将作为该协商的扩展。某些设备没有特定于设备的驱动程序来有效地管理电源。DPA 功能提供了一种为这些类型的设备动态分配功率的机制。DPA 是适用于 Endpoint Function 的可选规范功能，可以从动态分配功率中受益，并且没有替代机制。

处于 D0 状态时，DPA 功能使软件能够主动管理和优化 Function 的电源使用情况。DPA 不适用于电源状态 D1-D3，因此 DPA 功能由 PCI-PM 功能独立管理。

DPA 定义了一组功率子状态，每个子状态都有一个关联的功率分配。每个 Function 最多可以定义 32 个子状态。Substate 0（默认子状态）指示该 Function 可以消耗的最大功率。

子状态必须从 0 到 Substate_Max 连续编号，如 Section 7.24.2 所定义。每个子状态的功率分配都小于或等于先前子状态的功率分配。例如，可以将具有四个子状态的 Function 定义如下：

1. Substate 0（默认状态）定义了 25W 的功率分配。
2. Substate 1 定义了 20W 的功率分配。
3. Substate 2 定义了 20W 的功率分配。

公众号：芯片业者怕承双

4. Substate 3 定义了 10W 的功率分配。

Function 初始化之后，它在 Substate 0 的功率分配内运行。不需要软件就可以进入中间子状态。随着时间的流逝，软件可以动态配置 Function，使其以其选择的任何顺序在任何子状态下运行。在 Function 完成先前启动的子状态转换之前，允许软件将 Function 配置为在任何子状态下运行。

子状态转换完成后，Function 必须将其子状态与配置的子状态进行比较。如果 Function 的子状态与配置的子状态不匹配，则 Function 必须开始转换到配置的子状态。Function 允许发送配置请求动态更改子状态转换，以便 Function 在新的子状态下运行。

在先前的示例中，软件可以将 Function 配置转换为 Substate 4，然后转换为 Substate 1，然后转换为 Substate 3，依此类推。因此，当软件配置相关的控制字段时，Function 必须能够在任何子状态之间转换。

Substate Control Enabled 字段提供了一种机制，该机制允许将 DPA 功能与上述软件协商机制结合使用。置位时，功率分配由 DPA 功能控制。清除时，将禁用 DPA 功能，并且不允许 Function 根据 Substate Control 寄存器字段的配置直接启动子状态转换。在适当的时间点，参与上述软件协商机制的软件将清除该位，从而有效地接管该 Function 的功率分配控制。

要求 Function 在任何子状态时都能对配置空间的访问作出响应。

在任何时候，Function 所消耗的功率绝不能超过其子状态所指示的功率。当 Function 配置为从较高功率的子状态转换为较低功率的子状态时，Function 的子状态必须在转换过程中指示较高功率的子状态，并且在完成转换后指示较低功率的子状态。当 Function 配置为从较低功率的子状态转换为较高功率的子状态时，Function 必须在转换期间以及完成转换后指示较高功率的子状态。

由于应用的多样性和给定 Function 所需的最大功率范围很广，因此任何子状态之间所需的转换时间都是特定于实现的。为了使软件能够构建电源管理策略（超出本规范的范围），该 Function 定义了两个切换延迟值。 每个 Function 子状态将其最大切换延迟与切换延迟值之一相关联，其中最大切换延迟是 Function 从任何其他子状态进入已配置子状态所花费的时间。允许 Function 完成子状态转换的速度快于子状态的最大切换延迟。

6.16.1 DPA Capability with Multi-Function Devices

允许 Multi-Function 设备的某些或全部 Function 实现 DPA 功能。Multi-Function 设备的电源分配是由 DPA 功能为每个 Function 设置的电源分配总和。Function 的 DPA 功能允许包含 Function 本身的功率分配，也可以考虑未实现 DPA 功能的其他 Function 的功率分配。DPA 的多个 Function 之间的关联是特定于实现的，超出了本规范的范围。

6.17 TLP Processing Hints (TPH)

TPH 功能使我们可以控制来自 EP 的 TLP 是否可以直接对 DDR 或 L3 缓存进行读/写。由于读/写流的属性不同，例如，在 DMA 写入 DDR/Cache 之后，CPU 会很快读取一些数据，对于这种数据，最好将它们直接放入缓存。

从物理上讲，RP 将对 DDR /Cache 进行读/写操作。PCIe TLP 中的 TH，PH，ST 将用于控制上述操作。TLP 中的 TH 位指示是否启用 TPH 功能。TLP 中的 PH（2 bits）指示它是哪种流。ST 是特定于硬件的设计，可以存储在 MSI-X Table 或 TPH capability 中。

所以 TPH 的软件接口为：

```
    PR: Device Capability 2(offset 0x24): TPH Completer Supported(12,13bit RO)
    EP: TPH Requester Capability: Header, capability register(RO), controler register(RW),
        ST table.(ST table can be in MSI-X table)
    EP's DMA descriptor: must have a place to offer PH related information.
```

在我们的系统中，我们实现上述寄存器，ST table 位于 EP 的 MSI-X table 中。此外，我们在 RP 中实现了一个 stash table。我们在 EP 中用 8 位 ST entry 实现 ST table，该 table 指示一个流预期进入哪个 core/cluster。stash table 是一个自定义表，用于将 PH/ST

信息转换为 stash 信息。

因此，从软件角度来看，我们可以通过 EP 的 DMA descriptor，ST table，stash table 来控制 TPH。现在 ST table 和 stash table 已经建立了联系，很难根据我们芯片的 ST table 定义来修改 PCIe 设备驱动程序，这基于以下三点考虑：

但是，目前在 PCIe 规范和 ARM 规范中尚无关于 ST 的明确定义。这是第一个考虑因素，这意味着将来我们可能需要专用补丁才能进行优化。

第二个考虑因素是，stach table 将在 BIOS 中实现，很难修改，我们只能使用 ST 和 DMA 描述符来控制 TPH 行为。

第三点是，我们应该首先配置可缓存属性，然后启用 TPH 将数据直接放入 L3 缓存，例如：我们应该首先为 RP 启用 CCA = 1，然后启用 TPH 功能（例如，我们可以启用 SMMU 和 CCA = 1 启用可缓存属性。

TLP Processing Hints 是一项可选功能，可在 Request TLP 的 header 中提供提示信息，以促进针对目标内存空间的 Request 的优化处理。这些 Processing Hint 使系统硬件（例如 RC 和/或 Endpoint）能够基于每个 TLP 优化平台资源，例如系统和内存互连。TPH 机制定义了 Processing Hint，这些 Hint 提供有关 Endpoint 和 RC 之间的通信模型的信息。Steering Tag 是特定于系统的值，用于标识请求者明确指定的处理资源。系统软件发现并识别 TPH 功能，以确定支持 TPH 的每个 Function 的 Steering Tag 的分配。

6.17.1 Processing Hints

Requester 向 RC 或其他目标提供有关主机和/或设备对数据和数据结构的预期用途的提示。提示信息由 Requester 提供，该 Requester 具有即将到来的 Request pattern 的知识，而完成者将无法准确度很高的自动推断。可以通过此类提示区分感兴趣的案例包括：

DWHR：设备写然后主机读。

HWDR：设备读取认为主机最近已写入的数据 。

D*D* ：设备写/读，然后设备读/写，包括 DWDW、DWDR、DRDW 和 DRDR 。

Bi-Directional ：主机和设备共享并具有相同读/写访问权限的数据结构。

使用模型到 Processing Hint 编码的映射如 Table 6-11。

Table 6-11: Processing Hint Mapping

PH[1:0] (b)	Processing Hint	Usage Model
00	Bi-directional data structure	Bi-Directional shared data structure
01	Requester	D*D*
10	Target	DWHR HWDR
11	Target with Priority	Same as target but with temporal re-use priority

6.17.2 Steering Tags

旨在将 TLP 定向到特定处理资源（例如主机处理器或系统缓存层次结构）的 Function 需要目标缓存的拓扑信息（例如，哪个主机缓存）。Steering Tags 是特定于系统的值，提供有关系统缓存层次结构中的主机或缓存结构的信息。这些值用于将平台内的处理元素与 Request 的处理相关联。

将使用的软件可编程 Steering Tag 值存储在 ST 表中，该 ST 表允许位于 TPH Requester Capability（请参见 Section 7.26）中，或与 MSI-X 表组合（请参见 Section 7.7），但不能同时存在于两者之中。当 ST 表与 MSI-X 表组合时，每个 MSI-X 表条目的 Vector Control

寄存器的 2 个最高有效字节用于包含 Steering Tag 值。

ST 表位置的选择是特定于实现的，并且可以通过软件发现。每个 ST 表条目均为 2 个字节。TPH Requester Capability 中指示了 ST 表的大小。

对于某些使用模型，不需要或不提供 Steering Tag，在这种情况下，允许 Function 在 ST 字段中使用全零的值表示没有 ST。每个请求与 ST 表条目的关联是特定于设备的，不在本规范范围之内。

6.17.3 ST Modes of Operation

TPH Requester Capability 中的 ST Table Location 字段指示该 Function 在 ST 表的实现位置（如果有的话）。如果实现了 ST 表，则软件可以使用系统特定的 Steering Tag 值对其进行编程。

Table 6-12: ST Modes of Operation

ST Mode Select [2:0] (b)	ST Mode Name	Description
000	No ST Mode	The Function must use a value of all zeroes for all Steering Tags.
001	Interrupt Vector Mode	Each Steering Tag is selected by an MSI/MSI-X interrupt vector number. The Function is required to use the Steering Tag value from an ST Table entry that can be indexed by a valid MSI/MSI-X interrupt vector number.
010	Device Specific Mode	It is recommended for the Function to use a Steering Tag value from an ST Table entry, but it is not required.
All other encodings	Reserved	Reserved for future use.

在 No ST Mode 操作模式下，Function 的每个 Steering Tag 必须使用全零的值，从而可以使用软件无关的 Steering Tag 来启用 Processing Hint。

在 Interrupt Vector Mode 操作模式下，使用 MSI/MSI-X 中断向量编号从 ST 表中选择 Steering Tag。对于启用了 MSI 的 Function，要求该 Function 在 MSI Capability 中的 Multiple Message Enable 字段指定的范围内选择 Steering Tag。对于启用了 MSI-X 的 Function，要求该 Function 在 MSI-X 表大小范围内选择 Steering Tag。

在 Device Specific Mode 操作模式下，将 Steering Tag 分配给 Request 的实现是特定于设备的。无论 ST 表的位置如何，该 Function 使用的 Steering Tag 的数量都可以与为该 Function 分配的中断向量的数量不同，并且 Request 中使用的 Steering Tag 值不需要来自架构的 ST 表。

能够生成 TPH Request 的 Function 需要支持 No ST Mode 操作模式。支持其他 ST 操作模式是可选的。通过编程 ST Mode Select，一次只能选择一种 ST 操作模式。

MPLEMENTATION NOTE

ST Table Programming

为确保在请求中使用确定的 Steering Tag 值，建议软件在执行 ST 表更新的过程中停止 Function 或禁用 TPH 请求者的相关能力。否则可能会导致在 ST 表更新期间使用不确定的 ST 值。

6.17.4 TPH Capability

TPH 功能是可选的规范。每个可以生成带 TPH 的 TLP 请求的 Fution 都需要实现一个 TPH Requester Capability。 支持处理带 TPH 的 TLP 的 Completer 需要在 Device Capabilities 2 中指明 TPH Completer 的能力。TPH 被设计为应用于以内存空间为目标的事务，并且应用于设备到主机、设备到设备和主机到设备之间的事务流。在每种情况下，要支持 TPH，Requester、Completer 和所有中间路由元素必须支持相关的 TPH 功能。

软件通过 TPH Requester Capability 来枚举 Requester 的 TPH 能力，并通过 Device Capabilities 2 寄存器枚举 Completer 的 TPH 能力（请参见 Section 7.8.15）。软件必须对 TPH Requester Capability 中的 TPH Requester Enable 字段进行编程，以使 Function 能够发起 TPH 请求。

TPH 仅提供额外信息以实现对以内存空间为目标的请求的优化处理，因此用于管理内存空间访问的现有机制和规则（例如，Bus Master Enable、Memory Space Enable 和 Base Address 寄存器）不会改变。

6.18 Latency Tolerance Reporting (LTR) Mechanism

延迟容忍报告（LTR，Latency Tolerance Reporting）机制使 Endpoint 能够报告其对 RC 的内存读取和写入的服务延迟要求，以便可以对中央平台资源（例如主内存，RC 内部互连和监听资源）在进行电源管理时可以考虑端点服务要求的实现。LTR 机制不会直接影响链路电源管理或 Switch 内部电源管理，尽管可能会产生间接影响。

在不同的设备类型和实现之间，“延迟容忍”的含义将有很大不同。当实现这种机制时，通常期望考虑服务等待时间是否影响功能或仅影响性能，如果对性能影响是线性的，设备就有可能需要使用缓冲或其他技术来补偿延迟时间敏感性。

RC 不需要遵守请求服务延迟，但是强烈建议您提供最坏情况的服务延迟，该延迟不超过 LTR 机制指示的延迟。

LTR 支持是 Chapter 7 中介绍的报告和控制寄存器发现和启用的。除非 RC 和所有中间 Switch 都支持 LTR，否则软件不得在 Endpoint 中启用 LTR。请注意，并不需要所有 Endpoint 都支持 LTR 才能在支持它的那些 Endpoint 中启用 LTR。在层次结构中启用 LTR 机制时，必须首先启用最接近根端口的设备。

如果在不支持 LTR 的 Downstream Port 上收到 LTR Message，或者如果未启用 LTR，则必须将该消息视为不支持的请求。

No-Snoop Latency and Snoop Latency：如 Figure 6-15 所示，寄存器包含一个 Requirement 字段，该字段指示设备是否对给定类型的 Request 有延迟要求。对于任何 LTR Message 传输，允许设备为 No-Snoop 请求，仅 Snoop 请求或两种类型的请求都报告一个延迟需求。

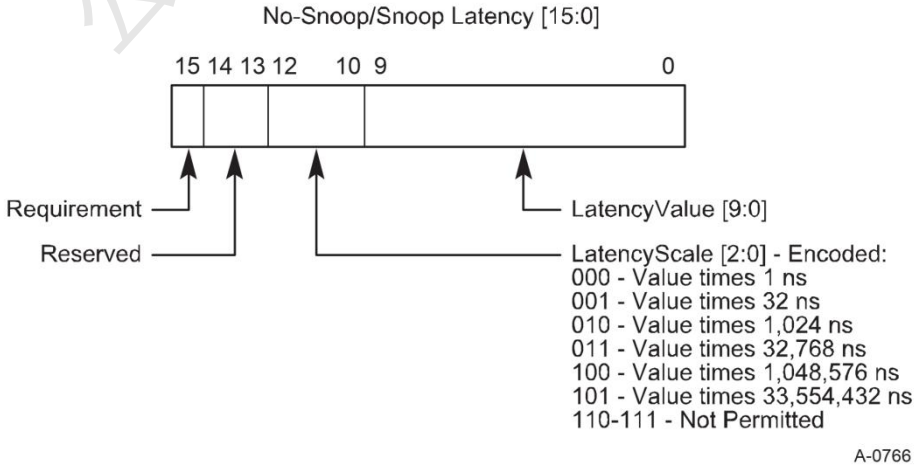

Figure 6-15: Latency Fields Format for LTR Messages

还允许设备通过清 Requirement 字段来表明对两种业务都不用报告延迟需求。

寄存器还包括 LatencyValue 和 LatencyScale 字段。LatencyValue 乘以 LatencyScale 可得出绝对时间值，可以在 1 ns 至 2^{25} *（2^{10}-1）= 34,326,183,936 ns 的范围内表示。

将 LatencyValue 和 LatencyScale 字段设置为全 0 表示该设备将受到任何延迟的影响，并且要求尽最大可能提供最佳服务。

如果设备对特定类型的业务没有实现或没有服务要求，则必须为关联的延迟字段清除 Requirement。

当通过写 PMCSR 寄存器将状态改变为 non-D0 时，如果设备最近发送的 LTR Message（自上一次 DL_Down 到 DL_Up 转换以来）报告了一个或两个 Requirement 位置 1 的延迟字段，则它必须在过渡到 non-D0 状态之前发送同时清除了两个 Requirement 的新 LTR Message。

LTR Mechanism Enable 字段清 0 后，如果设备最近发送的 LTR Message（自上一次 DL_DOWN 到 DL_Up 转换以来）报告了任意 Requirement 位置 1 的延迟容限值，则它必须发送一个清除所有 Requirement 位的新 LTR Message。

从设备的角度来看，来自设备的 LTR Message 反映了设备可容忍的延迟，为此平台必须考虑服务延迟本身，以及适用时使用时钟功率管理（CLKREQ＃）增加的延迟。服务延迟本身定义如下：

- 如果设备发出 Read Request，则延迟为从 Request TLP 的 END 符号发送到该请求的第一个 Completion TLP 的 STP 符号接收之间的时间。
- 如果设备发出一个或多个 Write Request，导致由于流控制反压而无法发出另一个 Write Request，则延迟为从用完 FC 信用的 TLP 的 END 符号的发送到接收到为该请求返回更多的信用的 DP 的 SDP 符号之间的时间。

如果使用了 Clock Power Management，则从设备置起 CLKREQ＃到设备接收到有效时钟信号之间的与平台实现相关的时间段构成了平台服务延迟的额外组成部分，当配置了平台电源管理策略时，平台必须能理解这个延迟。

建议 Endpoint 在使能 LTR 后尽快向上游发送 LTR Message。

强烈建议 Endpoint 在任意 500 µs 的时间内发送不超过两个 LTR Message，除非规范要求。如果在 500 µs 的时间内接收到两个以上的 LTR Message，则 Downstream Port 不得产生错误，并且必须正确处理所有 LTR Message，而不管它们之间的时间间隔如何。

与 Upstream Port 相关联的 Multi-Function 设备（MFD）必须根据以下规则向上游传输一个“综合（conglomerated）”的 LTR Message：

- MFD 向上游发送的消息的可接受延迟值必须反映 Function 中的最低值。
  - 允许在综合消息中报告的 snoop 和 no-snoop 延迟与不同的 Function 相关联。
  - 如果所有 Function 均未报告对某种类型的业务的 snoop 和 no-snoop 的要求，则 MFD 发送的消息不得设置与该类型的业务相对应的 Requirement 位。
- 当 MFD 的 Function 更改其内部已报告的值时，MFD 必须向上游发送新的 LTR Message，以更改 MFD 先前报告的综合值。

Switch 必须从使能了 LTR 机制的 Downstream Port 收集这些消息，并根据以下规则向上游传输“综合”的消息：

- 如果 Switch 支持 LTR 功能，则其 Upstream Port 和所有 Downstream Port 必须支持该功能。
- 仅当 Switch 的 Upstream Port 的 LTR Mechanism Enable 字段置 1 或如上述所述在软件中清除其 LTR Mechanism Enable 字段之后不久，才允许 Switch 的 Upstream Port 发送 LTR Message。
- Switch 上游发送的消息的可接受延迟值必须计算如下：
  - 如果 Downstream Port 没有收到包含对某种业务类型 snoop/no-snoop 的要求的 LTR Message，则 Switch 发送的任何 LTR Message 都不得设置与该业务类型相对应的 Requirement 位。
  - 将 LTRdnport [N]定义为在 Downstream Port N 收到的 LTR Message 中报告的值，并进行以下调整：

- 如果清除了 Requirement 位或使用了 Not Permitted LatencyScale 值，则 LTRdnport [N]实际上是无穷大的。
- 如果 Requirement 位为 1，且 LatencyValue 字段全为 0，则无论 LatencyScale 值如何，LTRdnport[N]必须为 0。

- 将 LTRdnportMin 定义为所有 Downstream Port 上 LTRdnport [N]的最小值。
- 将 Lswitch 定义为 Switch 引起的所有延迟。
  - 如果 Lswitch 根据 Switch 的操作模式动态变化，则 Switch 不得允许 Lswitch 超出 LTRdnportMin 的 20%，除非 Lswitch 的最低延迟模式更大，在这种情况下，必须使用最低延迟状态。
- 计算向上游发送的 LTRconglomerated = LTR dnportMin – Switch，如果该值小于 0，则 LTRconglomerated = 0。
- 如果 LTRconglomerated 为 0，则在综合的 LTR 消息中 LatencyValue 和 LatencyScale 字段都必须全部为 0。

- 当 Switch 的 Downstream Port 变为 DL_Down 状态时，必须将该端口记录的先前延迟视为无效。如果结果更改了整体延迟，则必须向上游发送新的 LTR Message。
- 如果将 Switch 的 Downstream Port 的 LTR Mechanism Enable 字段清零，则必须将为该端口记录的延迟容忍值视为无效，并且如果将综合延迟变的更高，则要更新该延迟值并向上游传输新的综合消息。
- 当任何 Downstream Port/Function 更改其已报告的延迟时，该 Switch 必须向上游发送新的 LTR Message，以改变 Switch 报告的综合延迟。
- 除非被上述事件之一触发，否则 Switch 不得向上游发送 LTR Message。

只要 RC 满足设备的服务要求，就可以延迟其对设备请求的 TLP 的处理。

当在一系列请求中更新了延迟要求时，要求在服务后续请求之前，由 RC 理解更新的延迟情况。在所有场景下，更新的延迟值必须在等于或小于先前报告的延迟要求的时间段内生效。允许 RC 早于此限制理解更新的延迟信息。

Optimal Use of LTR

建议每次 Endpoint 的服务要求更改时，Endpoint 都发送更新的 LTR 消息。如果减小了延迟容限，建议在有新要求的第一个请求之前发送更新的 LTR 消息，并允许记录先前发送的 LTR 消息中指示的时间。如果容限值在增加，则更新过程应立即跟在有最终延迟容限值的最终请求之后。

典型情况下，链路位于 ASPM L1 中，并且如果支持 Clock Power Management (Clock PM)，则当 Endpoint 到达内部触发点（导致 Endpoint 向 RC 发起请求）时，CLKREQ＃将置为无效。下面的文字显示了在这种情况下如何应用 LTR 的示例。关键时间点如 Figure 6-16 所示。

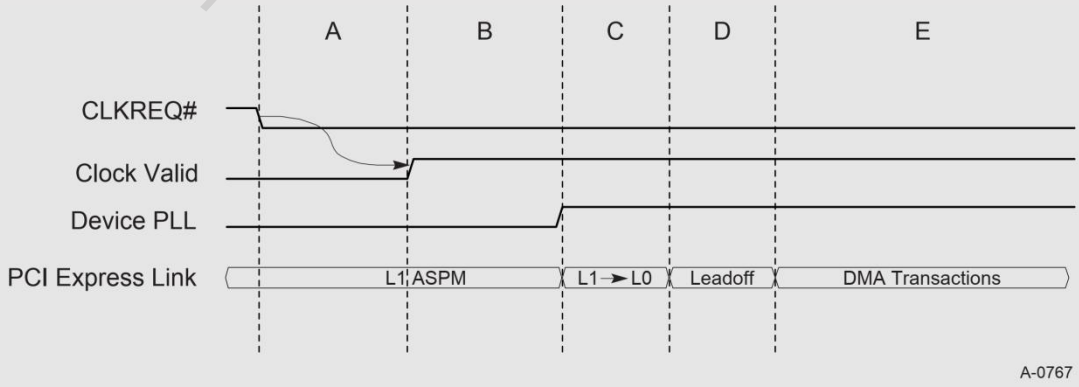

Figure 6-16: CLKREQ# and Clock Power Management

Time A 是从器件置起 CLKREQ＃到器件收到有效时钟信号之间的与平台实现相关的时间。该值不会超过实际的延迟。

公众号·芯片业者相承双

Time B 是与设备实现相关的时间段，介于设备具有有效时钟和它可以启动重训练序列以从 L1 ASPM 过渡到 L0 之间的时间。

Time C 是从 L1 ASPM 过渡到 L0 的时间段。

Time D 对于读事务来说，是从 Request TLP 的 END 符号发送到该请求的 Completion TLP 的 STP 符号接收之间的时间。Time D 对于写事务来说，是从发送耗尽 FC 信用的 TLP 的 END 符号到在 DLLP 中为该请求返回更多信用的 SDP 符号接收之间的时间。该值不会超过实际的延迟。

Time E 是从 Endpoint 到系统内存的数据路径打开且数据事务不受引入延迟影响的时间段。

LTR 延迟语义反映了由以下一项或两项衡量的设备看到的可忍受的延迟：

Case 1：设备可能支持或可能不支持 Clock PM，但尚未取消置位其 CLKREQ＃信号——在 Figure 6-16 中，设备观察到的延迟表示为 Time C 和 Time D 之和。

Case 2：设备支持 Clock PM 并清掉 CLKREQ＃——设备观察到的延迟表示为 Time A，Time C 和 Time D 的总和。

为了有效地将 LTR 机制与 Clock PM 结合使用，该设备将知道或能够测量 Time B 和 Time C，以便它知道何时置起 CLKREQ＃。Time A、Time C 和 Time D 的实际值可能会动态变化，并且平台有责任确保总和不会超过延迟。

在一个非常简单的模型中，Endpoint 可以选择实现 LTR，如 Figure 6-17 所示。当 Endpoint 确定其处于空闲状态时，它将发送 LTR 消息，其中配置了软件的最大延迟或该 Endpoint 可以支持的最大延迟。

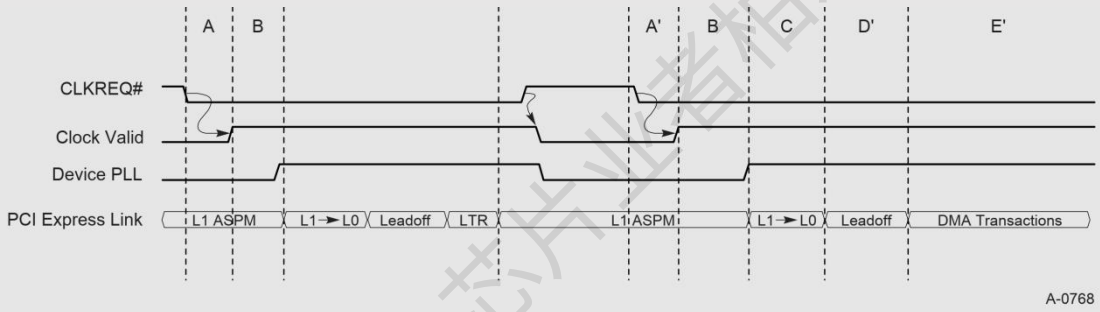

Figure 6-17: Use of LTR and Clock Power Management

当 Endpoint 确定需要与 RC 保持持续的数据传输时，Endpoint 将发送一个新的 LTR 消息，其时延较短（在 Time E'）。在下一次数据刷新之前发送此 LTR 消息的时间等于之前发送的最大延迟时间（Time E'和 Time D'之间的时间）。在 Time E'和 Time A'之间，Endpoint 可以返回低功耗状态，而平台转换为可以在设备下次需要传输数据时提供较短延迟的状态。

请注意，如果 RC 满足设备的服务要求，则它可能会延迟对设备请求 TLP 的处理。例如，如果连接到 Root Port 1 的 Endpoint 报告的延迟容限为 100 µs，而 Root Port 2 上的 Endpoint 报告的延迟值为 30 µs，则 RC 可能会实施以下策略：从 Root Port1 持续 70 µs，然后以 30 µs 的延迟为请求提供服务，从而为 100 µs 的第一个 Endpoint 提供了可感知的服务延迟。 这种 RC 行为使 RC 能够将请求批量组合在一起，以进行更有效的服务。

建议 Endpoint 尽可能多地缓冲请求，然后在突发可以实际支持的范围内，在突发中使用完整的链路带宽，因为这通常会导致最佳的整体平台电源效率。

请注意，在并非所有 Endpoint 都支持 LTR 的环境中就可以启用 LTR，并且在这种环境中，不支持 LTR 的 Endpoint 可能会遇到次佳的服务。

6.19 Optimized Buffer Flush/Fill (OBFF) Mechanism

该功能是用来省系统 power 的， 当启用 OBFF 时，主机软件能够将系统状态传输至具有主线总控能力的下游装置，使用它们来优化传输排程，并使系统在节能模式下工作的时间更长。

当 Endpoint 设备的总线主控和/或中断事务使得平台功耗的增量相对较低时，通过 Optimized Buffer Flush/Fill (OBFF)机制，RC 可以向端点报告（遍及整个层次结构）一个时间窗口。通常，这个时间窗口对应于与 RC 关联的主机 CPU，内存和其他中央资源处于活动状态以服务某些其他活动（例如操作系统计时器滴答）的时间。这种窗口的性质和时间的确定是特定于实现的。

OBFF 是一种 hint-只要启用此功能，就仍然允许 Function 发起总线主控和/或中断事务，尽管这对于平台电源而言不是最佳选择，应尽可能避免。

使用 WAKE # 信号或消息来发出 OBFF（请参阅 Section 2.2.8.9）。消息仅在 WAKE # 信号不可用的互连结构上使用。OBFF 或 CPU Active 的 WAKE # 信令仅在系统处于运行状态时才由根端口发出，该状态在符合 ACPI 的系统中对应于 S0 状态。处于 non-D0 状态的 Function 不得响应 OBFF 或 CPU Active 信令。

对于点对点，OBFF 消息路由定义为 100b，并且仅允许往下游方向上发送。区分了多个 OBFF 事件。使用 OBFF 消息时，OBFF 编码字段用于区分不同的 OBFF 场景：

1111b “CPU Active” - 系统对所有设备动作（包括总线主控和中断）完全处于活动状态。

0001b “OBFF” - 系统内存路径可用于设备内存读/写的总线主控活动。

0000b “Idle” - 系统处于空闲、低功耗状态。

所有其他值保留。

当使用 WAKE # 信令时，这些编码对应于 WAKE # 的各种时序模式，如 Figure 6-18 所示。用信号通知 OBFF 时有一个拉低过程，每次用信号通知 CPU Active 时有两个拉低过程。使用 WAKE # 时所需的电气参数在 *PCI Express CEM Specification, Revision 2.0* (or later) 的 WAKE # 信令部分中定义。

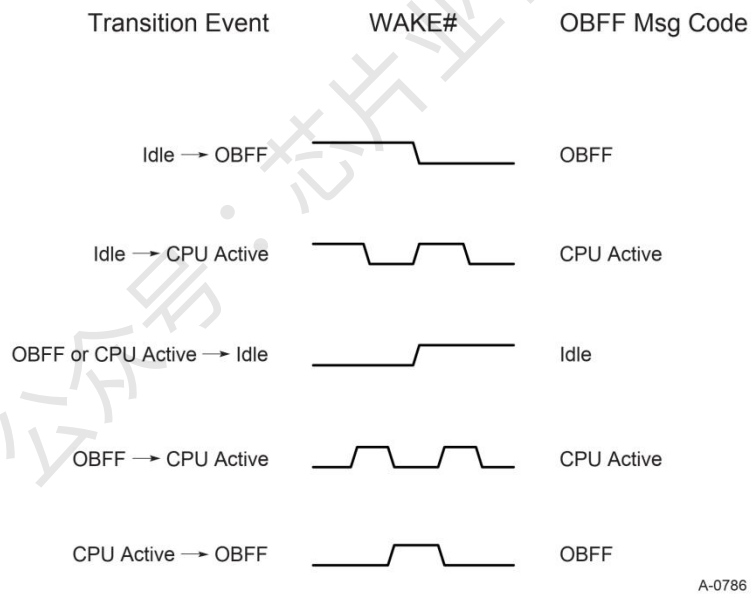

Figure 6-18: Codes and Equivalent WAKE# Patterns

当接收到 OBFF 消息的编码为保留值时，如果启用了 OBFF，则接收方必须将该指示视为“ CPU Active”指示。

在未实现 OBFF 或未启用 OBFF 的端口上收到的 OBFF 消息必须作为 Unsupported Request (UR)处理。

OBFF 指示反映了中央资源电源管理状态转换，并且在平台拓扑支持时使用 WAKE # 进行信号指示，或者在 WAKE # 不可用时通过消息进行信号传递。可通过 Chapter 7 中介绍的报告和控制寄存器来发现并启用 OBFF 支持。除非平台支持将 OBFF 指示传递到该 Endpoint，否则软件不得在 Endpoint 中启用 OBFF（就是说 OBFF 完全由主机软件控制，设备不能自己开启该功能）。

当平台指示 CPU Active 或 OBFF 窗口的开始时，建议平台在少于 10µs 的时间内不返回空闲状态。允许在实际进入平台空闲之前

公众号·芯片业者的承双

指示返回到空闲状态，但是强烈建议这样做只是为了防止后期的 Endpoint 活动导致立即退出空闲状态，并且提前时间应尽可能短。

建议 Endpoint 不要假定 CPU Active 窗口或 OBFF 窗口在任何特定时间段内保持打开状态。

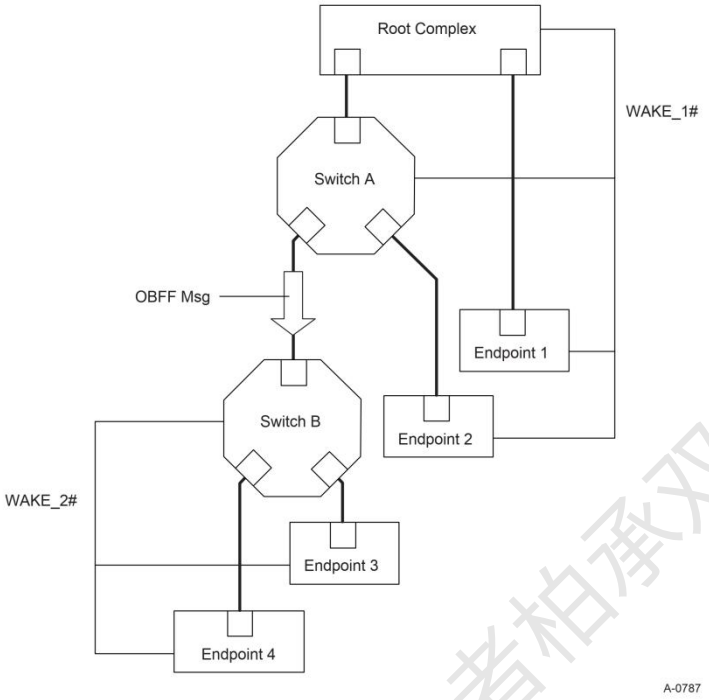

Figure 6-19: Example Platform Topology Showing a Link Where OBFF is Carried by Messages

Figure 6-19 显示了一个示例系统，其中 Switch A 有必要将使用 WAKE # 接收的 OBFF 指示转换为 OBFF 消息，在这种情况下，该消息将被另一个 Switch B 接收并转换回使用 WAKE # 信令 。HwInit 配置机制（由硬件或固件设置）用于识别这种情况（如本示例中所示，其中 Switch A 和 Switch B 之间的链路需要使用 OBFF 消息），并且系统固件/软件必须相应地配置 OBFF。

如果将 Switch 配置为在其上游端口使用 OBFF 消息，并在一个或多个下游端口使用 WAKE # ，反之亦然，当启用 OBFF 时，要求 Switch 将在上游端口收到的所有 OBFF 指示转换为下游端口的适当的形式。

使用 WAKE # 时，除非存在多个 WAKE # 信令，否则对任何特定根端口的启用都会启一个全局的 WAKE # ，在这种情况下，仅会影响关联的 WAKE # 信令。当使用 OBFF 消息时，对特定根端口的启用仅允许从该根端口传输 OBFF 消息。为了确保在平台中完全启用 OBFF，必须为 OBFF 启用所有指示 OBFF 支持的根端口。允许系统固件/软件选择性地启用 OBFF，但是这种启用超出了本规范的范围。

为了最大程度地降低功耗，强烈建议系统固件/软件仅在 WAKE # 信令不适用于给定链路时才启用 OBFF 消息的机制。

如果使用 WAKE # 的 OBFF 信令是共享的 WAKE # 信令，则仅应报告为连接到 Switch 的所有组件都支持。在这些拓扑中，即使 Switch 本身不支持 OBFF，也允许软件为连接到 Switch 的组件启用 OBFF。

允许（尽管不鼓励）连续多次指示同一 OBFF 事件。

当 Switch 向下游传播 OBFF 指示时，强烈建议传播所有 OBFF 指示。但是，尤其是在使用消息时，Switch 可能有必要丢弃或折叠 OBFF 指示。当接收到相同或不同类型的指示时，允许丢弃并替换给定类型的较早指示。

下游端口可以配置为以两种方式传输 OBFF 消息，称为 Variation A 和 Variation B。对于 Variation A，如果链路处于 L0 状态，则端口必须传输 OBFF 消息，但是当链路处于 Tx_L0s 或 L1 状态时，必须丢弃该消息。当下游端口导致设备对通信的要求不是时间紧迫的，并且期望设备通过将链路状态返回到 L0 发出紧急信号来引起注意时，则最好采用这种变化形式。对于 Variation B，如果链路处于 L0 状态，则端口必须发送 OBFF 消息，或者，如果链路处于 Tx_L0s 或 L1 状态，则端口必须将链路定向到 L0 状态，然

后发送 OBFF 消息。当下游端口连接到可以从及时通知平台状态中受益的设备时，首选这种变化形式。

最初配置为 OBFF 操作时，无论 WAKE # 信令的逻辑值如何，初始假定指示必须为 CPU Active 状态，直到观察到第一个转换为止。

为 OBFF 启用端口时，建议在启用“下游端口”之前先启用所有“上游端口”，并且在启用所有其他端口之后必须启用“根端口”。 对于热插拔端口，通常将无法执行此序列，并且允许使用 WAKE # 将 OBFF 启用到未连接的热插拔下游端口。建议不要将未连接的热插拔下行端口用于 OBFF 消息传输。

OBFF Considerations for Endpoints

在正常情况下，可能会合法地发生一些事件，这些事件可能导致 Endpoint 误解从空闲窗口到 CPU Active 窗口或 OBFF 窗口的转换。例如，非 OBFF 的 Endpoint 可以将 WAKE # 声明为一种唤醒机制，从而掩盖了系统的信号转换。这可能导致 Endpoint 出于电源或性能方面的考虑而无法达到最佳状态，但是对于 Endpoint 或主机系统而言，不应恢复。

为了使 Endpoint 能够维护主机状态的最准确状态，建议 Endpoint 在收到确定由主机发起的请求时将其内部状态跟踪逻辑置于 CPU Active 状态，并且在 Endpoint 具有由主机软件服务的挂起中断的任何时刻。

6.20 PASID TLP Layout

PASID TLP Prefix 是一种 Section 2.2.1 中定义的 End-End TLP Prefix。PASID TLP Prefix 的布局如 Figure 6-20 和 Table 6-14 所示。

当存在 PASID TLP Prefix 时，此 Prefix 中的 PASID 值与 Requester ID 一起标识与此请求相关的 Process Address Space ID。每个 Function 都有一组不同的 PASID 值。一个 Function 使用的 PASID 值与其他 Function 使用的 PASID 值无关。

在以下情况下允许使用 PASID TLP Prefix：

- 带有 Untranslated Addresses 的 Memory Request（包括 AtomicOp Request）（请参见 2.2.4.1 节）。
- Address Translation Request、ATS Invalidation Message、Page Request Messages 和 PRG Response Messages (see Section 10.1.3) 。

在任何其他 TLP 上均不允许使用 PASID TLP Prefix。

设备的 DMA 重定向硬件将来自于 I/O 子系统的内存访问请求分为两类：

1. 不带地址空间 ID 的请求（Request without Process Address Space Identifier，即 Request-without_PASID），相当于 GPA（Guest Physical Address），这是一般 Endpoint 设备发出的内存访问请求，这类请求通常会表明该请求的类型（读、写或原子操作），DMA 目标的地址、大小和发起请求的源设备的 ID。
2. 带有地址空间 ID 的请求（Request with Process Address Space Identifier，即 Request-with-PASID），相当于 GVA（Guest Virtual Address），能够发出这类请求的源 PCI 设备需要拥有 virtual address capability，该请求带有额外的信息用于定位目标地址空间和一些其他信息。

不同虚拟机之间的隔离是通过防止分配到其他虚拟机的资源（CPU、I/O 设备）访问到本虚拟机的物理地址。每个虚拟机都会有自己独立的物理地址空间，即 GPA（Guest Physical Address）空间，该空间不同于主机物理地址空间，即 HPA（Host Physical Address）空间。DMA 重定向硬件将从 I/O 设备发过来的访问请求中包含的地址看做是 DMA 地址，根据不同的使用配置，该 DMA 地址可能是 GPA；可能是跟 PASID（Process Address Space ID）相关的 VA（application Virtual Address）；也可能是由软件定义的 I/O 虚拟地址（IOVA）。不管怎样，DMA 重定向硬件将 DMA 地址最终转化为 HPA（Host Physical Address）实现最终主机物理地址的访问。

6.20.1 Managing PASID TLP Prefix Usage

使用 PASID TLP Prefix 之前必须要使能。除非使能了 PASID TLP Prefix 功能，否则不允许组件传输 PASID TLP Prefix。

对于 Endpoint Function（包括 RC Integrated Device），要适用以下规则：

- 除非 PASID Enable（参见 Section 7.8.8.3）设置为 1，否则不允许 Function 发送和接收带有 PASID TLP Prefix 的 TLP。
- Function 必须具有一种机制，可以动态地将 PASID 的使用与特定的 Function 的上下文相关联。此机制是特定于设备的。
- Function 必须具有一种机制来请求其正常停止使用特定的 PASID。该机制是特定于设备的，但必须满足以下规则：
  - Function 可能支持有限数量的同时发起 PASID 的停止请求。软件应推迟发出新的停止请求，直到较旧的停止请求完成为止。
  - 一个 Function 中的 PASID 的停止请求不得影响任何其他 Function 的操作。
  - PASID 的停止请求不得影响 Function 内任何其他 PASID 的操作。
  - PASID 的停止请求不得影响与 PASID 不相关的事务的操作。
  - 当 PASID 的停止请求机制指示此请求完成时，该 Function 应该：
    - 停止为此 PASID 排队新请求。
    - 完成了与此 PASID 关联的所有 Non-Posted Request。
    - 将所有 Posted Request 发送到主机，以解决 PASID 使用的所有 TC 的主机内存。 用于此目的的机制是特定于设备的（例如：对主机存储器进行的 non-relaxed Posted Write 或处理器对 Function 的读取可以刷新 TC0；对主机存储器进行零长度读可以刷新 non-zero TC）。
    - 可选的将所有 Peer-to-Peer Posted Request 刷新到其目的地。用于此的机制是特定于设备的。
    - 如果用此 PASID 发出了 Address Translation 或 Page Request，则要符合 Address Translation Services (Chapter 10)中描述的其他规则。

对于 RC，适用以下规则：

- RC 必须具有特定于设备的机制，以指示对 PASID TLP Prefix 的支持。
- 支持 PASID TLP Prefix 的 RC 必须具有特定于设备的机制才能启用它们。默认情况下，禁用 PASID TLP Prefix 的使用。
- 支持 PASID TLP Prefix 的 RC 可以可选地具有特定于设备的机制，用于以比整个 RC 更细的粒度启用它们（例如，针对特定 Root Port、Requester ID、Bus Number、Requester ID 或 Requester ID/PASID 组合的启用）。

6.20.2 PASID TLP Layout

一个 TLP 最多可以包含一个 PASID TLP Prefix。

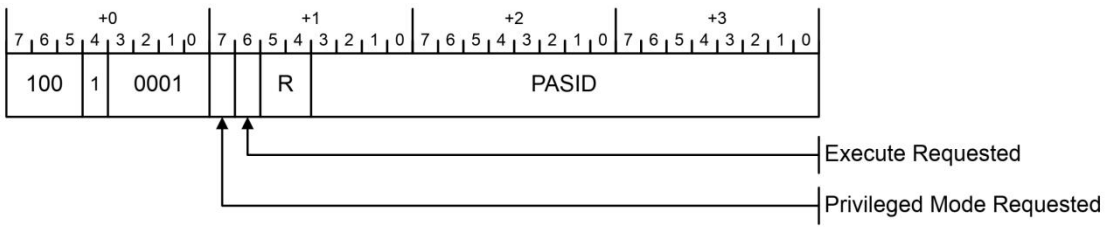

Figure 6-20 PASID TLP Prefix

公众号：芯片业者相硬双

Table 6-14 PASID TLP Prefix

Bits	Description
Byte 0 bits 7:5	100b - indicating TLP Prefix
Byte 0 bit 4	1b - indicating End-End TLP Prefix
Byte 0 bits 3:0	0001b - indicating PASID TLP Prefix
Byte 1 bit 7	***Privileged Mode Requested*** - If Set indicates a Privileged Mode entity in the Endpoint is issuing the Request, If Clear, indicates a Non-Privileged Mode entity in the Endpoint is issuing the Request. Usage of this bit is specified in Section 6.20.2.3 .
Byte 1 bit 6	***Execute Requested*** - If Set indicates the Endpoint is requesting Execute Permission. If Clear, indicates the Endpoint is not requesting Execute Permission. Usage of this bit is specified in Section 6.20.2.2 .
Byte 1 bit 3: Byte3 bit 0	**Process Address Space ID (PASID)** - This field contains the PASID value associated with the TLP. Usage of this field is defined in Section 6.20.2.1 .

6.20.2.1 PASID field

PASID 字段标识与请求对应的用户进程。所有 PASID TLP Prefix 中都存在此字段。

PASID 字段为 20 位宽。Endpoint 和 RC 不需要支持整个字段。对于 Endpoint 点，Max PASID Width 字段指示支持的 PASID 值范围（Section 7.28.2）。对于 RC，使用特定于实现的机制来提供此信息。

除非 PASID Enable 字段（Section 7.28.3）设置为 1，否则不允许端 Endpoint 发送带有 PASID TLP Prefix 的 TLP。支持 PASID TLP Prefix 的 Endpoint 在收到带有 PASID TLP Prefix 的 TLP 并且 PASID Enable 字段为 0 时，必须发出 Unsupported Request (UR) 。

RC 可以选择支持带有 PASID TLP Prefix 的 TLP。用于检测 RC 是否支持 PASID TLP Prefix 的机制是特定于实现的。

对于 Endpoint，以下规则要应用：

- 不允许 Endpoint 发送 PASID 值大于或等于 $2^{\text{Max PASID Width}}$ 的 TLP。
- 当 Endpoint 接收到 PASID 值大于或等于 $2^{\text{Max PASID Width}}$ 的请求时，可以选择让 Endpoint 发出错误信令。这是与接收端口相关的 Unsupported Request 错误（Section 6.2）。

对于 RC，以下规则要应用：

- RC 不允许发送 PASID 值大于其支持的 TLP。
- 当 RC 收到 PASID 值大于其支持的请求时，可以选择允许 RC 发出错误信令。这是与接收端口相关的 Unsupported Request 错误（Section 6.2）。

对于 Completer，以下规则要应用：

- 对于 Untranslated Memory Request，PASID 值和 Untranslated Address 都用于判定满足请求所使用的 Translated Address。对于与地址转换相关的 TLP，在 Address Translation Services (Chapter 10)中定义了此字段的用法。

PASID Width Homogeneity

每个 Function 的 PASID 值是唯一的，因此最初的意图是该 Function 支持的 PASID 值的宽度可以基于该 Function 的需求。但是，当前的系统软件通常不遵循该模型，而是在访问特定地址空间的所有 Function 中使用相同的 PASID 值。为实现此目的，系统软件

公众号·芯片业者相承双

通常为 RC 和 persistent translation agent 确保一个公共的系统的 PASID 宽度。此类系统软件通常会在报告支持 PASID 宽度（小于公共的系统的 PASID 宽度）的任何热插拔 Endpoint Function 或 translation agent 上禁用 ATS。

RC、Endpoint 和 translation agent 通常独立于系统软件实现，因此强烈建议硬件四线最大 20 位宽度，以确保与系统软件的互操作性。

Endpoint 可以以特定于实现的方式将 20 位系统 PASID 映射到承载较小宽度的内部表示形式。如果做到这一点，那么至关重要的是，Endpoint 必须做到这一点，而又不影响系统软件，该系统软件没有机制可以将这种实现与本机实现全部 20 位宽度的实现区分开。

6.20.2.2 Execute Requested

如果 Execute Requested 字段被设置为 1，则端 Endpoint 正在请求允许该 Endpoint 点在与该请求相关联的内存范围内执行指令。执行权限的含义不在本规范的范围内。

Endpoint 点不允许发送 Execute Requested 字段设置为 1 的 TLP，除非同时设置了 Execute Permission Supported (Section 7.8.8.2)和 Execute Permission Enable(Section 7.8.8.3)字段。

对于 RC，以下规则要应用：

- RC 对 Execute Requested 支持是可选的。用于确定 RC 是否支持 Execute Requested 的机制是特定于实现的。
- 支持 Execute Requested 字段的 RC 应具有特定于实现的机制，以使其能够使用该位。
- 支持 Execute Requested 字段的 RC 应具有特定于实现的机制，以允许以更细的粒度使用该位（例如，对于特定的 Root Port、Bus Number、 Requester ID 或 Requester ID/PASID 的组合），并且其默认值是实现特定的。

对于 Completer，以下规则要应用：

- Completer 具有有效位的概念。对于给定的请求，如果支持 Execute Requested 字段并且已使能该请求的使用，则该字段的有效值为请求中的值；否则，该位的有效值为 0b。
- 对于 Untranslated Memory Read Request，Completer 使用该位的有效值作为保护检查的一部分。如果此保护检查失败，则 Completer 将请求视为未映射到内存。
- 对于 Untranslated Memory Read Request，除了 Untranslated Memory Read Request，该位保留。对于与地址转换相关的 TLP，此位的用法在 Address Translation Services (Chapter 10)中定义。

6.20.2.3 Privileged Mode Requested

如果 Privileged Mode Requested 设置为 1，则 Endpoint 发出针对与 Privileged Mode 相关的内存的请求。如果 Privileged Mode Requested 清除为 0，则 Endpoint 发出针对与 Non-Privileged Mode 相关的内存的请求。

Privileged Mode 和 Non-Privileged Mode 的含义以及 Endpoint 以特权或非特权模式运行的含义取决于系统的保护模型，并且不在本规范范围之内。

除非 Privileged Mode Supported(Section 7.8.8.2) 和 Privileged Mode Enable(Section 7.8.8.3) 都设置为 1，否则不允许 Endpoint 发送设置了 Privileged Mode Requested 字段的 TLP。

对于 RC，以下规则要应用：

- RC 对 Privileged Mode Requested 支持是可选的。用于确定 RC 是否支持 Privileged Mode Requested 的机制是特定于实现的。

公众号：芯片业者相承双

- 支持 Privileged Mode Requested 字段的 RC 应具有特定于实现的机制，以使其能够使用该位。
- 支持 Privileged Mode Requested 字段的 RC 应具有特定于实现的机制，以允许以更细的粒度使用该位（例如，对于特定的 Root Port、Bus Number、Requester ID 或 Requester ID/PASID 的组合），并且其默认值是实现特定的。

对于 Completer，以下规则要应用：

- Completer 具有有效位的概念。对于给定的请求，如果支持 Privileged Mode Requested 字段并且已使能该请求的使用，则该字段的有效值为请求中的值；否则，该位的有效值为 0b。
- 对于 Untranslated Memory Read Request，Completer 使用该位的有效值作为保护检查的一部分。如果此保护检查失败，则 Completer 将请求视为未映射到内存。
- 对于与地址转换相关的 TLP，此位的用法在 Address Translation Services (Chapter 10)中定义。

6.21 Lightweight Notification (LN) Protocol

Lightweight Notification（LN）协议允许 Endpoint 注册在主机内存中的感兴趣的特定 cacheline，并在他们被更新时通过硬件机制进行通知。LN Requester（LNR）是 Endpoint 中的客户机子系统，它发送 LN Read/Write Request 并接收 LN Message。LN Completer（LNC）是主机中的一个服务子系统，它接收 LN Read/Write，并在更新注册的 cacheline 时发送 LN Message。

LN 协议为感兴趣的 cacheline 何时更新提供通知服务。最常见的是，LNR 将 LN Read 发送到相应的 LNC 的存储空间范围，以请求复制 cacheline（在本节中简称为“line”）。LNC 将请求的 line 返回给 LNR，并记录该 line 更新时 LNR 已发出通知请求；也就是说，LNC“register”了该 line。稍后，当某些实体更新 line 时，LNC 会通过 LN Message 通知 LNR，因此 LNR 可以采取适当的措施。对于 LN 写入，存在类似的通知服务，其中在更新 line 时，稍后可以通知写入 line 的 LNR。

LN 协议允许多个 LNR 并发注册同一条 line。无论 LNR 是否已注册该 line，实现了 LNR 的 Endpoint 都可以随时写入该 line。

一个典型的系统由主机处理器，主机内存，主机内部结构，根端口 Switch 和 Endpoint 组成。Figure 6-21 展示了这些元件以及它们如何互连以提供描述 LN 协议如何工作的上下文的简化视图。

公众号：芯片业者柏承汉

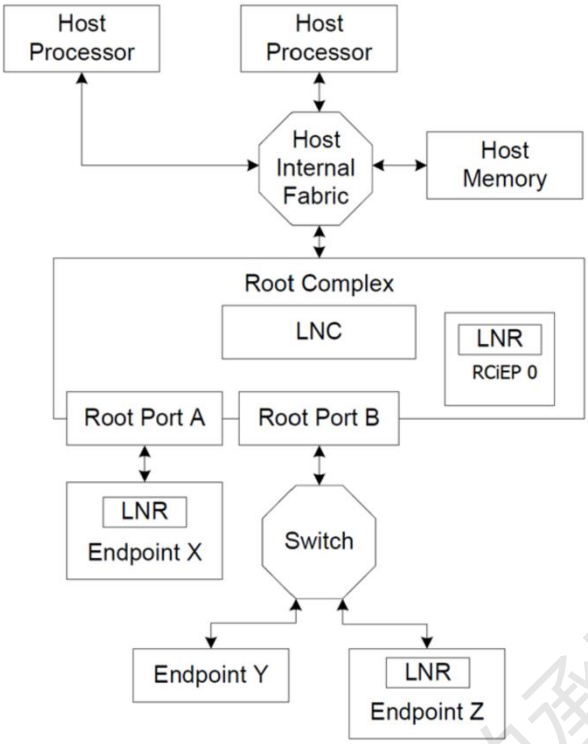

Figure 6-21 Sample LN System Block Diagram

在上图中，Endpoint X，Endpoint Z 和 RCiEP 0 各自包含一个 LNR。RC 包含一个 LNC。

6.21.1 LN Protocol Operation

LN 是一个简单的协议，它以最小的复杂性和成本满足了几种关键的使用模型。

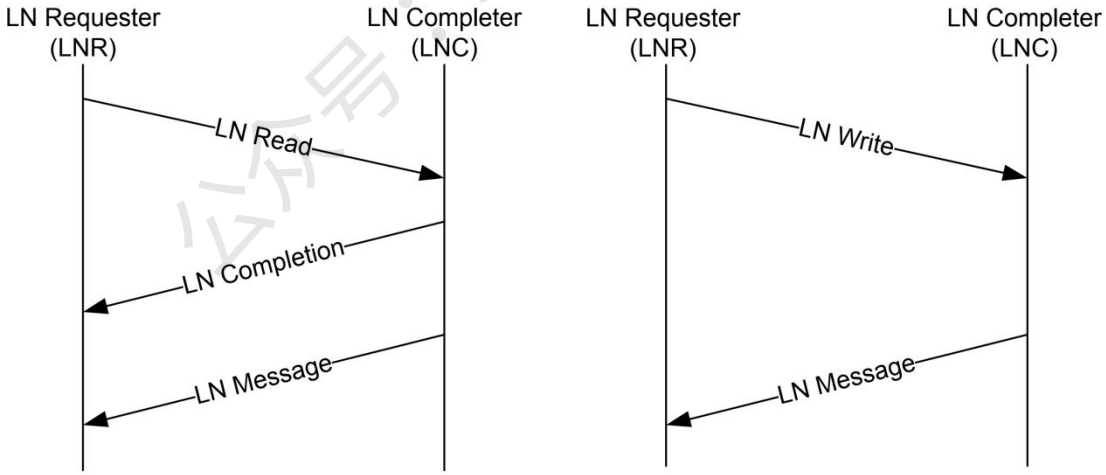

Figure 6-22 LN Protocol Basic Operation

上图显示了具有单个 LNR 的 LN 协议操作。对于 Read，如左图所示：（1）LNR 使用 LN Read 从主机存储器中请求 line 的副本；（2）LNC 在 LN Completion 中返回该 line 并记录 LNR 已注册该 line；（3）LNC 稍后在注册的 line 被更新时使用 LN Message 通知 LNR。对于 Write，如右图所示：（1）LNR 使用 LN Write 将 line 写到主机存储器中；（2）LNC 记录 LNR 已经注册了该 line；（3）LNC 稍后在注册的 line 被更新时使用 LN Message 通知 LNR。

公众号·芯片业者相承双

如果某个实体（例如 CPU 或设备）更新了已注册的 line，或者 LNC 不再打算跟踪该 line 的状态，则 LNC 必须发送 LN Message。后一种情况称为驱逐（或回收），并由 LN Message 中的 Notification Reason（NR）字段指示。

如果 LN Requester 对其已注册的 line 进行 LN Read 或 LN Write，则 LN Requester 通常无法确定其为该 line 接收到的后续 LN Message 是否用于最近的 LN Read/Write Request。

LN 协议允许多个 LNR 同时注册同一 line。在这种情况下，LNC 通过向每个 LNR 发送定向 LN Message 或向每个受影响的层次结构域发送广播 LN Message 来通知多个 LNR。

一旦 LNC 更新或驱逐给定的 line 并发送一个或多个 LN Message 以通知所有相关的 LNR，LNC 将不会为该特定 line 发送其他 LN Message，除非它收到对该 line 的新的 LN Read 或 LN Write 操作。

LNC 可以针对每个给定 line 独立跟踪多少个 LNR 具有特定于实现的限制。达到或低于此限制时，LNC 通常使用定向 LN Message 进行通知。超过此限制，LNC 将使用广播的 LN Message。允许实现的限制为零，并且只能使用广播 LN Message。

具有特定 NR 字段值的单个 LN Message 可以指示已注册到该 LNR 的所有 line。 允许使用此“所有驱逐”的 LN Message 的定向版本和广播版本。

允许 LNC 对其可以同时注册的 line 的数目或 line 的集合具有特定于实现的限制。如果 LNC 接收到注册新 line 的请求，但 LNC 没有足够的资源来注册新 line，则允许 LNC 驱逐一条或多条旧的 line，以释放必要的资源，或发送一条 LN Message 来指示新的 line 被逐出。

Excessive Use of Broadcast LN Messages

为了避免性能问题，应该实现使用 Broadcast LN Message 的 LNC，以最小化每个 Broadcast LN Message 发送到的层次结构域的数量，并使 Broadcast LN Message 的速率保持在合理范围内。每个 Broadcast LN Message 都会消耗链路带宽，并且某些 Endpoint 可能以相对较低的速率处理 Broadcast LN Message。

特别是，不支持 LN Requester capability 的 Endpoint 可以使用低性能机制来处理接收到的 Broadcast LN Message，作为例外情况，例如，使用设备固件而不是硬件来解码消息。可以想象，每个消息可能要花费几微秒的时间 [129]，而且它们的传输速率过高可能会轻易导致架构中发送请求的反压，从而导致严重的性能问题。

尽管不支持 LN Requester capability 的 Endpoint 也可以使用低性能机制来处理 directed LN Message（作为例外情况），但这不会引起性能问题。使用 LN 协议，LNC 仅将 directed LN Message 发送到支持 LN Requester capability 的 Endpoint，并且大概这些 Endpoint 将能够以避免性能问题的速率处理 directed LN Message 和 Broadcast LN Message。

注 129：Posted Request Acceptance Limit 允许 Endpoint 最多花费 10μs 来处理每个收到的 Posted Request。请参阅 Section 2.3.1。

6.21.2 LN Registration Management

由于 LNC 的资源不足，无法注册 cacheline，因此每个 LNR 中的 LLNR Capability structure 为软件提供了限制 LNR outstanding 的注册的最大数量的机制。软件读取 LNR Registration Max 字段以确定 LNR 能够拥有的最大数量。如有必要，软件可以将 LNR Registration Limit 字段设置为实现指定的限制。

LNC 注册资源限制可能是高度实现特定的，可能包括诸如集合关联性，集合的最大数量以及用于主机内存不同区域的不同资源集之类的因素。软件如何发现这些资源限制超出了本规范的范围。

为了管理其 outstanding 注册的数量，LNR 可以通过向要注销的每一 line 发送一个零长度的 LN Write 来注销 outstanding 的注册。

LNC 不需要接受主机存储器中所有位置的注册。但是，要求接受注册的内存区域的粒度不比对齐的 4KB 区域细。为了确定给定的对齐 4KB 区域是否接受注册，LNR 可以对区域内的任何 cacheline 执行 LN Read，并查看是否返回了 LN Completion。 如果 LNR 希望“探测”一个具有注册功能的区域而不实际创建注册，则 LNR 可以使用零长度 LN Read，这是提供此语义所必需的。

6.21.3 LN Ordering Considerations

LN Read 具有与其他 Memory Read Request 相同的排序要求，而 LN Write 具有与其他 Memory Write Request 相同的排序要求。LN Completion 与其他内存读取的 Completion 具有相同的排序要求。LN Message 与其他 Posted Request 具有相同的排序要求。

对于给定的 cacheline，当 LN Completer 接收到 LN Read 并随后对该 cacheline 进行更新时，允许 LN Completer 以任一顺序发送 LN Completion 和 LN Message。

对于给定的 cacheline，当 LN Completer 收到触发 eviction LN Message 的 LN Read 时，LN Completer 被允许以任一顺序发送 LN Completion 和 LN Message。

对于给定的 cacheline，当 LN Completer 收到触发 eviction LN Message 的 LN Write 时，然后对该 cacheline 进行更新时，允许 LN Completer 以任一顺序发送两个 LN Message。

对于不同的 cacheline，允许 LN Completer 以任何顺序发送 LN Message。

6.21.4 LN Software Configuration

LN 协议支持 2 种 cache line size（CLS）：64 字节和 128 字节。对于 RC 和 Endpoint，每个 CLS 的支持都是可选的。对系统有效的 CLS 由主机确定，并由适用的根端口和 RCRB 中的 LN System CLS 字段指示。 支持 LN Completer 的所有根端口和 RCRB 必须指示相同的 CLS。否则，结果不确定。当任何操作系统正在运行时，主机不得更改系统 CLS。否则，结果不确定。

支持 LN 协议的 Endpoint 必须支持一个或两个 CLS，并通过 LNR-64 Supported 和 LNR-128 Supported 字段进行指示。使能 LN Requester 时，软件必须确保将相关的 LNR CLS 控制字段配置为与系统 CLS 相匹配；否则，结果不确定。仅支持一个 CLS 的 LN Requester 可以将其 LNR CLS 控制字段硬连线到相应的值。

除非 LNR Enable 控制字段已被清 0 或正在使用相同的配置写入进行清 0，否则软件不得更改 LNR CLS 控制字段或 LNR Registration Limit 字段的值。否则，结果不确定。如果 LNR Enable 字段已经被清 0，则允许软件使用单个配置写入操作在设置 LNR Enable 字段的同时更改 LNR CLS 和 LNR Registration Limit 字段的值。

允许软件随时清除 LNR Enable 字段。当 LNR Enable 字段为 0 时，LNR 必须清除其所有内部注册状态。

6.21.5 LN Protocol Summary

LN 协议的详细规则和要求在本规范的其余部分中进行了描述，这里是总的摘要以及一些独特的要求。

- LN Read 是一个 Memory Read Request，其 TLP Header 中将 LN 字段置 1。LN Completion 是 TLP Header 中 LN 字段设置为 1 的 Completion。LN Write 是一个 Memory Write Request，其 TLP Header 中的 LN 字段置 1。
- 除非另有明确说明，否则所有有关 Memory Read Request 的要求都适用于 LN Read。
  - LN Read 最多只能访问由系统 CLS 确定的单个 cacheline。LN Completer 必须以 Completer Abort 来处理违反此规则的情况，除非完成者检测到更高的优先级错误。允许 Partial cacheline LN Read，包括 zero-length LN Read。
  - 如果 LN Completer 将一个 LN Read 视为 Uncorrectable Error 或 Advisory Non-Fatal Error，则 LN Completer 不得为该请求注册通知服务。
  - LN Completer 必须将 zero-length LN Read 作为目标存储区域的 Probe 来处理，以实现注册功能。 参见 Section 6.20.2。如果 Completion Status 为 Successful Completion，则 TLP Header 中的 LN 字段必须指示该区域是否支持注册功能，但 LNC 不得为此情况创建注册。LN Completer 必须支持注册功能，其粒度不得比对齐的 4KB 区域更细。

公众号·芯片业者相承双

- LN Read 的顺序和流控制规则与内存读取请求的顺序和流控制规则相同。

- 除非另有明确说明，否则 Memory Read Completion 的所有要求均适用于 LN Completion。
  - 如果完成者是 LN Completer，目标存储区接受注册且 Completion Status 为 Successful Completion，则 LN Read 的 Completion 必须是 LN Completion（即，在其 TLP Header 中设置了 LN 位）；否则，Completion 必须在其 TLP Header 中将 LN 位清 0。请注意，中毒的 Completion 将具有 Successful Completion 的 Completion Status。参见 Section 2.7.2.2。
  - LN Completion 的排序和流控制规则与 TLP Header 具有 LN 位清 0 的 Completion 相同。

- 除非另有明确说明，否则所有关于 Memory Write Request 的要求都适用于 LN Write。
  - 根据系统 CLS 值，LN Write 最多只能访问单个 cacheline。LN Completer 必须以 Completer Abort 来处理违反此规则的情况，除非 Completer 检测到更高的优先级错误。
  - 如果 LN Completer 将 LN Write 视为不可纠正的错误，则 LN Completer 不得为该请求注册通知服务。请注意，根据其配置，LN Completer 可能会将中毒的 LN Write 处理为 Uncorrectable Error、Advisory Non-Fatal Error 或者 masked error。
  - LN Completer 必须能处理 zero-length LN Write，以注销现有注册的请求。参见 Section 6.20.2。如果 zero-length LN Write 的目标的 cacheline 先前未注册，则必须保持未注册状态。
  - LN Write 的排序，流控制和数据中毒规则与内存写入请求的规则相同。
  - 请求者不得为 MSI 或 MSI-X 中断生成 LN Write 操作。LN Completer 必须将针对中断地址范围的 LN Write 处理为 Completer Abort 错误，除非完成者检测到更高的优先级错误。

- 对于 LN Read 和 LN Write，地址必须是正确的类型，如地址类型（AT）字段所示。参见 Section 2.2.4.1 节。正确的类型取决于是否正在使用转换代理（TA）。参见 Chapter 10。
  - 如果使用了 TA，则该地址必须是转换后的地址。LN Requester 必须支持 ATS 才能获取和使用转换后的地址。
  - 如果未使用 TA，则该地址必须是默认/未转换的地址。
  - 检测到违反上述规则的 LN Completer 必须将请求作为 Completer Abort 错误处理，除非检测到更高的优先级错误。

6.22 Precision Time Measurement (PTM) Mechanism

6.22.1 Introduction

精密时间测量（PTM，Precision Time Measurement）能够通过独立的本地时钟在多个组件之间精确地协调事件。通常情况下，由于各个时钟对时间的值和变化率有不同的概念，这种精确的协调将是困难的。为了解决这个限制，PTM 允许组件计算其本地时间和共享的 PTM Master Time 之间的关系：与 PTM Root 相关联的独立时域。

增强的精确时间管理（ePTM）对 PTM 设备提出了额外的要求。ePTM Capable 字段用于表示支持 ePTM。

PTM 定义如下：

- PTM Requester —— 能将 PTM 用作 Endpoint 或 Upstream Port 的 consumer 的 Function。
- PTM Responder —— 能使用 PTM 为 Port 或 RCRB 提供 PTM Master Time 的 Function。
- Time Source —— 与 PTM Responder 相关联的本地时钟。
- PTM Root —— PTM 层次结构的 PTM Master Time 的来源。PTM Root 还必须是时间源，并且通常也是 PTM Responder。

公众号：法片业者相承双

每个 PTM Root 向所有 PTM 层次结构提供一个 PTM Master Time：一组与单个 PTM Root 关联的 PTM Requester。

Figure 6-23 说明了一些使用 PTM 的系统拓扑示例。这些只是说明性的例子，并不意味着任何限制或要求。

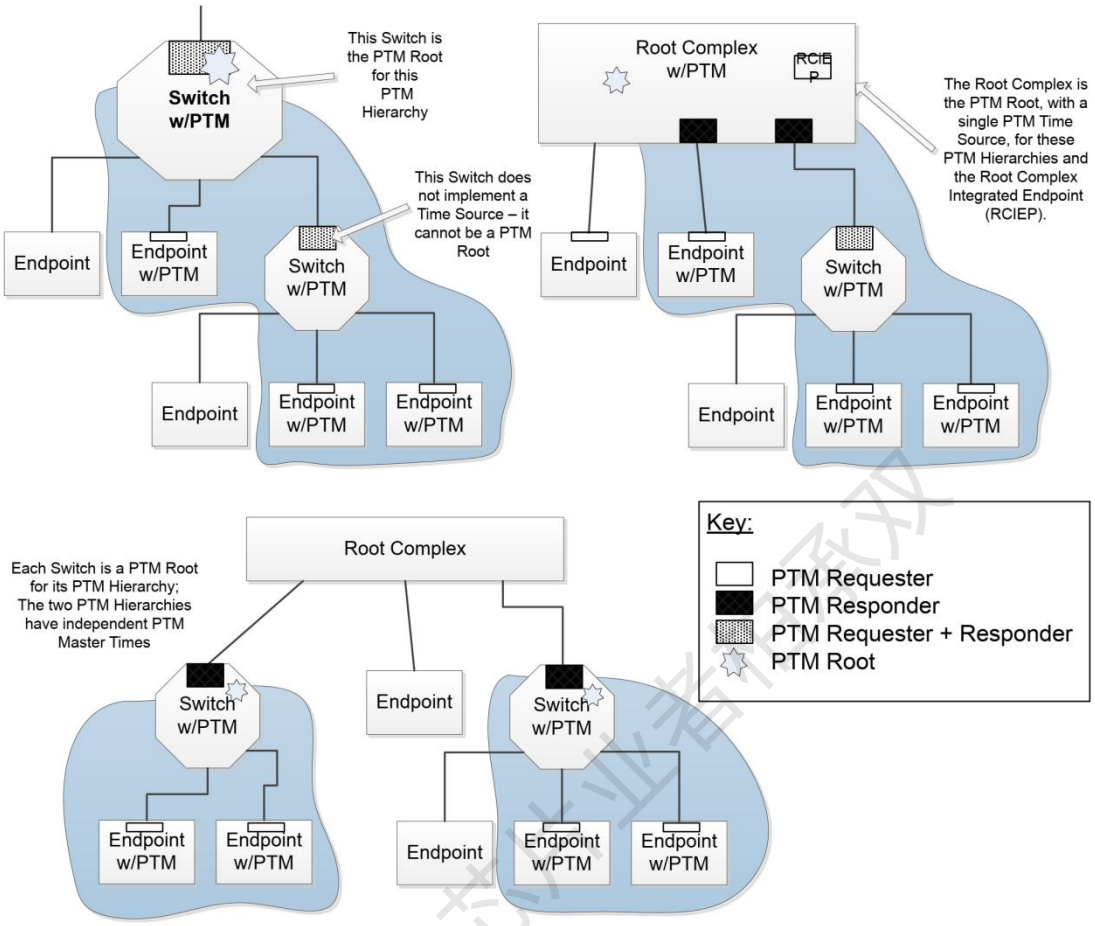

Figure 6-23 Example System Topologies using PTM

PTM and Retimers

PCIe Retimer 可以通过引入非对称链路延迟来影响 PTM 准确性。设计用于维持对称链路延迟的 Retimer 将实现最佳 PTM 精度。不对称性越大且变量越大，对 PTM 的影响就越大。请查阅制造商的文档，以确定适用于 PTM 的 Retimer 实现的适用性。

6.22.2 PTM Link Protocol

当在链路上的两个组件之间使用 PTM 时，代表 PTM Requester 的 Upstream Port 将 PTM Request 发送到同一链路上代表 PTM Responder 的 Downstream Port。

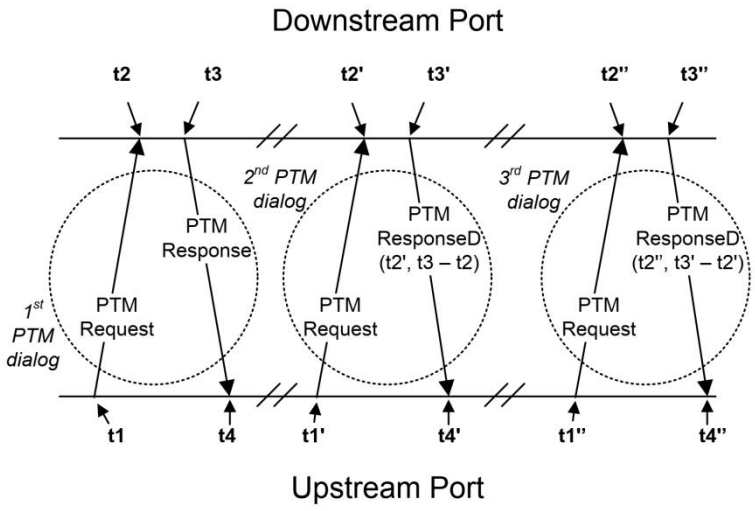

Figure 6-24 Precision Time Measurement Link Protocol

Figure 6-24 说明了 PTM 链路协议。上图中的点 t1、t2、t3 和 t4 表示每个端口在发送和接收 PTM Message 时本地捕获的时间戳。与每个端口相关联的组件将来自第一个 PTM 会话的这些时间戳存储在内部寄存器中，以便在第二个 PTM 会话中使用，依此类推，用于后续的 PTM 会话。

Upstream Port 代表 PTM Requester，通过发送 PTM Request Message 来发起 PTM 对话。

Downstream Port 代表 PTM Responder，拥有可以直接或间接访问 PTM Master Time 的能力。

在每个对话中，Downstream Port 根据在先前的 PTM 对话中存储的时间戳填充 PTM ResponseD Message，如 Section 6.22.3.2 所定义。

一旦每个组件具有来自前面对话框的历史时间戳，与 Upstream Port 相关联的组件就可以将其时间戳与 PTM Response MessageD 中传递的时间戳结合起来，使用以下公式计算 PTM Master Time：

$$\text{PMT Master Time at } t1' = \frac{\left((t4 - t1) - (t3 - t2)\right)}{2}$$

Equation 6-2 PTM Master Time

值 t1、t2、t3、t4 和 t2'表示在 PTM 对话期间捕获的时间戳，如 Figure 6-24 所示。

支持 PTM 的组件通常会记录这些时间戳计算的结果，并且可以通过特定于实现的方式使它们可供软件使用。在此，本文档将该结果定时信息称为组件的“PTM context”。

对于实现 PTM 的 Switch，Switch 内部的时间同步机制是特定于实现的。

PTM Theory and Operation

通过在 PTM 对话中捕获的时间戳，可以计算 PTM Requester 和 PTM Responder 之间的时序关系。值（t3-t2）衡量给定 PTM 对话的 PTM Responder 消耗的时间。 时间（t4-t1）是从请求到响应的时间。 因此，（（t4-t1）-（t3-t2））有效地给出了两个组件之间的往返消息传输时间，该数量除以 2 近似于链路延迟，即 t1 与 t2 之间的时间差。假设从 PTM Requester 到 PTM Responder 再到返回的链路传输时间对称，这通常是一个很好的假设（参阅 Implementation Note on PTM Timestamp Capture Mechanisms）。

公众号·芯片业者拍摄

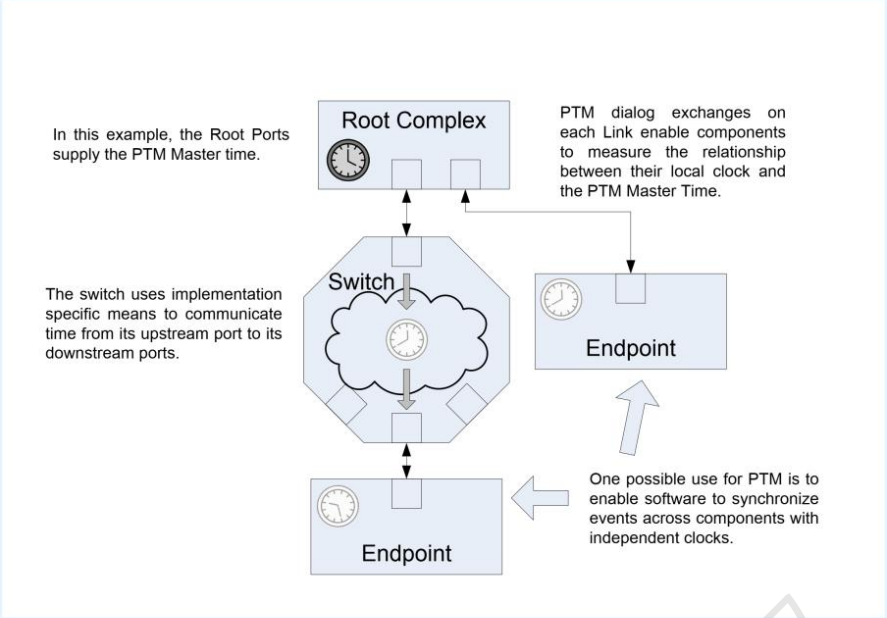

Figure 6-25 Precision Time Measurement Example

Figure 6-25 说明了使用 PTM 的简单设备层次结构。每个 Upstream Port 都会启动 PTM 对话，以建立其本地时间与 Root Port 提供的 PTM Master Time 之间的关系。

在此示例中，Switch 在其 Upstream Port 上启动 PTM 对话，以获取 PTM Master Time，以用于填充在其 Downstream Port 处接收到的 PTM Request Message。该 Switch 采用特定于实现的方式来将 PTM Master Time 从其 Upstream Port 传递到其 Downstream Port。

实现 PTM 功能的组件可以提供它们的 PTM context 供软件检查，从而使软件能够在本地时间和 PTM Master Time 之间转换计时信息。反过来，此功能使软件可以非常精确地协调多个组件之间的事件。

同样，强烈建议实现 PTM 的平台也将 PTM Master Time 提供给软件。

6.22.3 Configuration and Operational Requirements

除非链路上 Downstream Port 已经在其 PTM Control 寄存器中把 PTM Enable 字段设置为 1，否则软件不得在与其关联的 Upstream Port 的 Function 上把 PTM Control 寄存器中的 PTM Enable 字段设置为 1。

PTM Extended Capability structure 用以表示 Function 对 PTM 的支持。不需要层次结构中的所有 Endpoint 都支持 PTM，并且不需要软件在支持 PTM 的所有 Endpoint 中启用 PTM。

如果不支持 PTM 的端口接收到 PTM Message，或者当 PTM Enable 字段清 0 时 Downstream Port 接收到 PTM Message，则必须将该消息视为 Unsupported Request。这是报告的与接收端口相关的错误（see Section 6.2）。 支持 PTM 的 Upstream Port 接收到的正确形成的 PTM Response，但其 PTM Enable 字段被清 0，则必须静默丢弃此 PTM Response。

通过监测 PTM，PTM Master Time 必须满足以下行为要求：

- Time 值必须是单调的，并且必须严格增加。
- 感知的粒度必须不大于 PTM Capability 寄存器的 Local Clock Granularity 字段中报告的值。
- 感知时间必须不迟于 PTM Root 处理其第一条 PTM Request Message 的开始时间。

参考 Figure 6-24，以下规则定义了时间戳的捕获：

- 即使此发送是重播，PTM Requester 在发送 PTM Request Message 时也必须更新其存储的 t1 时间戳。

- 即使收到的 TLP 是重复的，当收到 PTM Request Message 时，PTM Responder 也必须更新其存储的 t2 时间戳。
- 传输 PTM Response 或 PTM ResponseD Message 时，PTM Responder 必须更新其存储的 t3 时间戳，即使该传输是重播。
- 即使收到的 TLP 是重复的，当收到 PTM Response MessageD 时，PTM Requester 也必须更新其存储的 t4 时间戳。
  - 时间戳必须基于构成 TLP 的 STP Symbol 或 Token，就像在端口引脚上观察该 Symbol 或 Token 的第一比特一样。通常，这将需要对实现进行特定的调整，以补偿无法直接在实际引脚上测量时间的情况，因为该时间通常是在 Rx 或 Tx 路径的某个内部点进行测量的。此测量的准确性和一致性不受本规范的限制，但强烈建议您达到最高的实用水平的准确性和一致性。

6.22.3.1 PTM Requester Role

- 通过将 PTM Capability 寄存器中的 PTM Requester Capable 字段置 1 来表示对 PTM Requester 角色的支持。
- 仅当启用 PTM 时，才允许 PTM Requester 请求 PTM Master Time。触发 PTM Requester 发出这样的请求的机制是特定于实现的。
  - Upstream Port 通过 PTM 对话获取 PTM Master Time，如 Section 2.2.8.10 所述。
  - RCiEP 请求 PTM Master Time 的机制是特定于实现的。
- 发出 PTM Request Message 后，Upstream Port 在接收到 PTM Response Message、PTM ResponseD Message、Reset 或在 100µs 之前 Downstream Port 没有相应的 PTM Message，则不得发送其他 PTM Request Message。
- 收到 PTM Response 后，Upstream Port 必须等待至少 1µs 才能发出另一个 PTM Request Message。
- 对于包含多个 PTM Requester 的 Multi-Function Device（MFD），与该 MFD 相关联的 Upstream Port 必须在每次 PTM context 刷新期间发出单个 PTM 对话。 MFD 中的 PTM Requester 使用一个设备范围的 PTM 对话维护各自的 PTM context。从一个 PTM 对话刷新多个 PTM context 的机制是特定于实现的。
- 强烈建议在发生以下任何情况时，Upstream Port 使其内部 PTM context 无效。如果支持 ePTM，则在发生以下任何情况时，Upstream Port 必须使其内部 PTM context 无效：
  - 重发 PTM Request。
  - 收到重复的 PTM ResponseD TLP。
  - PTM Master Time 与 Upstream Port 的本地时间之间的关系由特定的实现标准确定。例如，这可能由于过渡到 non-D0 状态或由于累积的 PPM 漂移而发生。

  这些事件在 Figure 6-26 中标记为“Local Time Invalidation Event”。
- 如果支持 ePTM，则在重发 PTM TLP 时，Upstream Port 必须使它的 PTM context 无效，直到成功连续完成两个连续的 PTM 对话且没有重发。

公众号·芯片业者拍承双

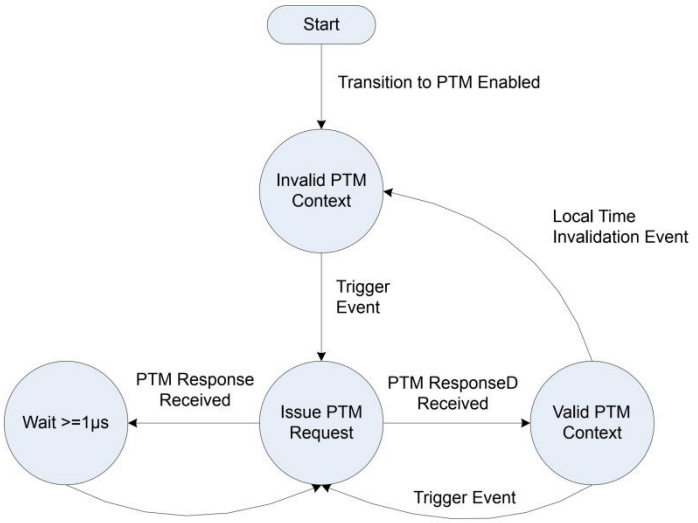

Figure 6-26 PTM Requester Operation

PTM Invalidation on the Reception of duplicate TLPs

在数据链路层中检测到重复的 TLP 并将其丢弃，而在事务层中则标识了 PTM Message。在某些实现中，将重复的 PTM TLP 与其他重复的 TLP 进行区分可能很困难或过于复杂。

因为允许 Upstream Port 根据特定于实现的标准使它们的内部 PTM context 无效，所以除接收到任何重复的 PTM TLP 之外，还允许 PTM Requester 在接收到任何重复的 TLP 时使其内部 PTM context 无效。类似地，如果支持 ePTM，则在接收到任何重复的 TLP 时，允许 PTM Responder 使其历史时间戳（t2-t3）无效。

6.22.3.2 PTM Responder Role

- 通过将 PTM Capability 寄存器中的 PTM Responder Capable 字段置 1 来表示对 PTM Responder 角色的支持。
- 允许 Switch 和 RC 实现 PTM Responder 角色。
  - 具有 PTM 能力的 Switch 将 Switch Upstream Port 的 PTM Control 寄存器中的 PTM Enable 字段置 1 来启用 PTM 后，必须响应在其任何 Downstream Port 收到的所有 PTM Request Message。
  - RC 将 PTM Master Time 传达给 RCiEP 的机制是特定于实现的。
- PTM Responder 必须按以下方式填充 PTM ResponseD Message（请参阅 Figure 6-24 和附着的实现说明）：
  - PTM Master Time 字段是一个 64 位值，存储收到当前 PTM 对话的 PTM Request Message 时的 PTM Master Time 值。在 Figure 6-24 中，对于第二个 PTM 对话，这指的是时间 t2'的 PTM Master Time。
  - Propagation Delay 字段是一个 32 位的值，其中存储在接收 PTM Request Message 和发送针对先前 PTM 对话的 PTM Response Message 之间的时间间隔。在 Figure 6-24 中，对于第二个 PTM 对话，这是指在第一个 PTM 对话期间捕获的 t2 和 t3 之间的时间间隔。
  - 两个字段的度量单位均为 1 ns。
  - 具有多个 Downstream Port 的 PTM Responder 必须在其所有 PTM 端口的 Downstream Port 上填充来自单个 PTM Root 的值的所有 PTM ResponseD Message。
- 充当 PTM Responder 的 Switch Downstream Port 和 Root Port 必须根据以下规则，使用 PTM Response 或 PTM ResponseD 响应

公众号·芯片业者相承双

在其 Downstream Port 收到的每个 PTM Request Message：

- 在未先收到 PTM Request Message 的情况下，PTM Responder 不得发送 PTM Response 或 PTM ResponseD 消息。
- 收到 PTM Request Message 后，PTM Responder 必须尝试在 10µs 内发出 PTM Response 或 PTM ResponseD 消息。
- 当 Downstream Port 没有有效的历史时间戳记（t3-t2）来满足 PTM Request Message 时，PTM Responder 必须发出 PTM Response Message。
- 如果支持 ePTM，则 PTM Responder 必须在重发任何 PTM Response 或 PTM ResponseD 消息时立即使其历史时间戳（t3-t2）无效。在收到任何重复的 PTM Request 之后，PTM Responder 必须使其历史时间戳（t3-t2）无效。

- 当 PTM Responder 存储了填充 PTM ResponseD Message 所需的值的副本时，它必须发出 PTM ResponseD Message：历史时间戳记（t3-t2）和收到最新 PTM Request Message 时的 PTM Master Time（t2'）。
- 当 PTM Responder 存储了历史时间戳记（t3-t2）的副本，但必须从其他地方请求 PTM Master Time 时，才可以发出 PTM Response Message。在这种情况下，如果预期检索花费的时间超过 10µs，则可以在发出 PTM Master Time 的同时发出响应 PTM Request Message 的 PTM Response Message。
- PTM Responder 发送的历史时间戳和 PTM Master Time 值的感知粒度不得超过 PTM Capability 寄存器的 Local Clock Granularity 字段中报告的粒度。

6.22.3.3 PTM Time Source Role - Rules Specific to Switches

Switch 除了上面列出的有关 PTM Requester 和 PTM Responder 角色的要求之外，还必须遵守以下要求：

- 当 Upstream Port 与 Multi-Function Device 关联时，仅允许与该 Upstream Port 关联的单个 Function 实现 PTM Extended Capability structure。对于 Switch，必须通过该结构控制与 Switch 关联的所有 PTM 功能。不需要将实现 PTM Extended Capability structure 的 Function 实现为 Switch Upstream Port Function。
- Switch 的 PTM Extended Capability structure 必须表明同时支持 PTM Requester 和 PTM Responde 角色。
- 允许 Switch 充当 PTM Root，或在其 Upstream Port 上发出 PTM Request 以获得 PTM Master Time，以用于填充在其 Downstream Port 处接收到的 PTM Request。在后一种情况下，Switch 必须考虑到 Switch 内部的任何内部延迟。
- 允许 Switch 维护本地 PTM context，以用于填充在其 Downstream Port 上收到的 PTM Request。
- 从 Upstream Port 上的最后一个 PTM 对话起，不充当 PTM Root 的 Switch 必须使其本地上下文无效不超过 10 ms。然后，Switch 必须刷新其本地 PTM 上下文，然后才能在其 Downstream Port 上发送更多的 PTM ResponseD Message。如何通过特定于实现的方式保证定期刷新的要求是可选的，这意味着 Switch 的本地时钟与 PTM Master Time 相位锁定。
- 为了维护本地 PTM 上下文而实现本地时钟的任何 Switch 必须报告该时钟的粒度，如 PTM Capability structure（Section 7.9.16）中所定义。

PTM Timestamp Capture Mechanisms

PTM 使用来自数据链路层和事务层的服务。精度要求时间测量尽可能靠近物理层。相反，消息传递协议本身属于事务层。PTM 消息协议适用于单个链路，其中 Upstream Port 是请求者，而 Downstream Port 是响应者。

公众号：芯片业者相承双

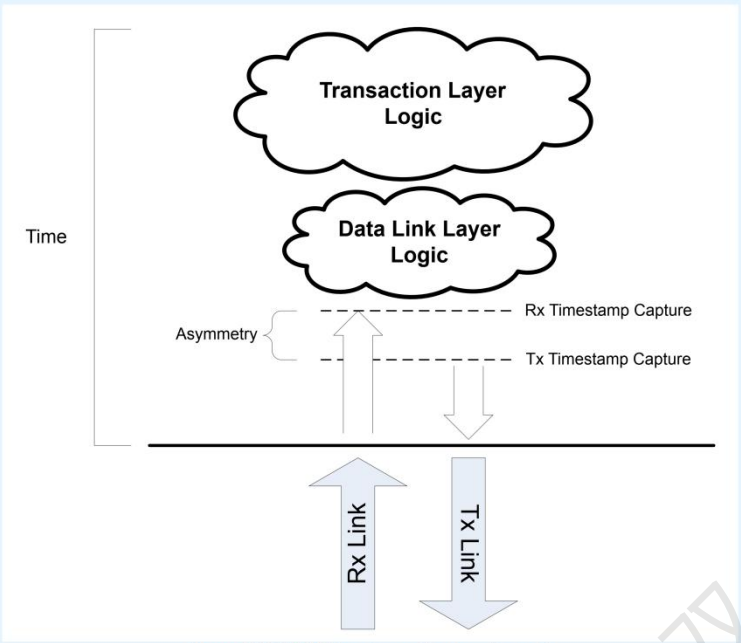

Figure 6-27 PTM Timestamp Capture Example

Figure 6-27 说明了如何选择合适的时间戳捕获点。对于某些实现，事务层和数据链路层中的逻辑是不确定的。实现细节和当前条件对特定数据包何时可能遇到任何特定处理步骤的确切时间具有相当大的影响。如果在较高层中捕获了时间戳，那么实际上就不可能捕获任何可以准确记录特定物理事件时间的时间戳。

6.23 Readiness Notifications (RN)

就绪通知（RN，Readiness Notifications）旨在减少在 DRS 事件或 FRS 事件之后向设备或 Function 发出配置请求之前，软件需要等待的时间。RN 包括设备就绪状态（DRS，Device Readiness Status）和功能就绪状态（FRS，Function Readiness Status）机制。这些机制直接指示 Configuration-Readiness（请参阅“Configuration-Ready”的术语和缩写词条目）。使用 DRS 和 FRS 时，它的行为优于 CRS 机制，并且在复位后消除了长达 1 秒的相关周期性轮询时间。

允许系统软件/固件提供取代 FRS 和/或 DRS 机制的机制，但是此类软件/固件机制不在本规范的范围之内。

Optimizing Configuration Readiness

强烈建议系统固件/软件的实现者尽可能避免不必要的延迟。强烈建议将硬件设计为消除或最小化所需的延迟，并充分利用本规范和相关规范中提供的机制来传达所需的延迟。硬件实现者应适当记录实现行为，以使系统固件/软件能够实现最佳行为。

即使有良好的文档记录，某些情况起初也可能会出现问题-例如，当需要从根端口配置空间中进行读取时，系统固件如何从设备就绪状态（DRS）机制中受益？在这种情况下，需要特定于平台的知识，即根端口要支持 Immediate Readiness。

6.23.1 Device Readiness Status (DRS)

实现 DRS 后，必须使用 DRS 来指示出现以下任何设备级别事件之后设备何时准备就绪，这些事件在之后描述中被称为“ DRS Event”：

- Exit from Cold Reset
- Exit from Warm Reset, Hot Reset, Loopback, or Disabled

- Exit from L2/L3 Ready
- Any other scenario where the Port transitions from DL_Down to DL_Up status.

DRS Message 协议要求包括以下内容：

- DRS 没有启用或禁用机制。对于支持 DRS 的 Downstream Port，必须将 Link Capabilities 2 寄存器中的 DRS Supported 位置 1。对于支持 DRS 的 Upstream Port，强烈建议将 Link Capabilities 2 寄存器中的 DRS Supported 位置 1。即使 DRS Supported 位为 0，也允许 Upstream Port 发送 DRS Message。
- 当与该 Upstream Port 关联的逻辑总线上的所有 non-VF Function 都准备就绪时，在每次从 DL_Down 到 DL_Up 转换之后，必须通过具有 DRS 功能的 Upstream Port 发送 DRS Message。
  - 当 Configuration-Ready 时，Type 0 Function 准备就绪。
  - 对于 Switch，当 Configuration-Ready 且其二级总线上的所有 Function 都准备就绪时，作为 Switch Upstream Port 的 Type 1 Function 准备就绪。
  - 当 Function 本身为 Configuration-Ready 状态时，非 Switch Upstream Port 的 Type 1 Function 准备就绪。
- 设备发送 DRS Message 后，除非发生后续 DRS Event，否则该 DRS Message 指示为配置就绪的 non-VF Function 不得返回带有 CRS 的完成。

与实现 DRS 的 Switch 有关的其他要求包括：

- 必须在所有端口中支持 DRS。
- 在 DRS Signaling Control 字段的每个 Downstream Port 实现。
- 对于出现在 Switch Downstream Port 下方的任何物理集成设备，Switch 发送的 DRS 并不表示该设备的配置就绪状态。
  - 对于这样的设备，DRS 的实现和使用独立于 Switch。

Root Port 和 Switch Downstream Port 的其他要求包括：

- DRS Message Received 字段指示接收到 DRS Message。

DRS Messages and ACS Source Validation

在为 Function 分配 Bus Number 之前，允许 Function 发送 DRS Message。这样的消息将具有 Bus Number 为 00h 的 Requester ID。如果 Downstream Port 启用了 ACS Source Validation，则可能会将这些消息（请参阅 Section 6.12.1.1）检测为 ACS 违规错误。

6.23.2 Function Readiness Status (FRS)

实现 FRS 后，必须使用 FRS 在以下任何 Function 事件出现之后将特定 Function 指示为 Configuration-Ready，这些 Function 事件在之后的描述中被称为“ FRS Event”：

- Function Level Reset (FLR)
- Completion of $D3_{Hot}$ to D0 transition
- Setting or Clearing of VF Enable in a PF (SR-IOV)

FRS Message 协议要求包括以下内容：

- FRS Message 的 Requester ID 必须指示 Function 已更改就绪状态（请参见 Section 2.2.8.6.4）。

公众号·芯片业者柏承汉

- FRS Message 中的 FRS Reason 字段必须指出该 Function 更改了就绪状态的原因。
- Function 发送 FRS Message 后，除非随后的 DRS Event 或 FRS Event 发生，否则指示的 Function 不得返回带有 CRS 的完成。

实现 FRS 的 Switch 的其他要求包括：

- 必须在 Upstream Port 和所有 Downstream Port 中支持 FRS。
- 当 FRS 协议要求时，能够向上游传输 FRS Message。

Physical Function（PF）的其他要求包括：

- 当 VF Enable 或 VF Disable 过程完成时，能够向上游传输 FRS Message。

实现 FRS 的 Root Port 和 Root Complex Event Collector 的其他要求包括：

- 必须实现 FRS Queuing Extended Capability（请参阅 Section 7.8.9）。

6.23.3 FRS Queuing

支持 FRS 的 Root Port 和 Root Complex Event Collector 必须实现 FRS Queuing Extended Capability（请参阅 Section 7.8.9）。

对于 Root Port，FRS Message Queue 包含由根端口接收或由根端口生成的 FRS Message 。

对于 Root Complex Event Collector，FRS Message Queue 包含由与 Root Complex Event Collector 相关联的 RCiEP 生成的 FRS Message（请参阅 Section 7.9.10）或由 Root Complex Event Collector 自身生成的 FRS Message 。

FRS Message Queue 必须满足以下要求：

- 复位后，FRS 消息队列必须为空。
- 对于根端口，当链路转到 DL_Down 时，必须清空 FRS Message Queue。
- FRS Message 必须按收到的顺序排队。
- 如果收到或内部生成 FRS Message 时 FRS Message Queue 未满，则必须在队列中加入该 FRS Message，并且 FRS Message Received 字段必须设置为 1b。
- 如果收到或内部生成 FRS Message 时 FRS Message Queue 已满，则必须丢弃该 FRS Message，并且 FRS Message Overflow 字段必须设置为 1b。必须保留先前存在的 FRS Message Queue。
- 最旧的 FRS Message 必须在 FRS Message Queue 寄存器中可见（请参见 Section 7.8.9.4）。
- 写入 FRS Message Queue 寄存器必须从队列中删除最早的元素。
- 当 FRS Message Received 或 FRS Message Overflow 从 0b 变为 1b 时，如果使能了中断，则必须产生中断。

6.24 Enhanced Allocation

Enhanced Allocation Capability（EA）是一项可选的 Capability，它允许以 Type 0 和 Type 1 配置头中的 BAR 和 Base/Limit 机制无法实现的方式分配 I/O，Memory 和 Bus Number 资源。

仅允许基于 PCI 配置空间中所见 Function 的层次结构，以及基于平台环境中 Function 存在方式的某些方面，将 EA 应用于某些 Function（参见 Figure 6-28）。

公众号：芯片业者相承双

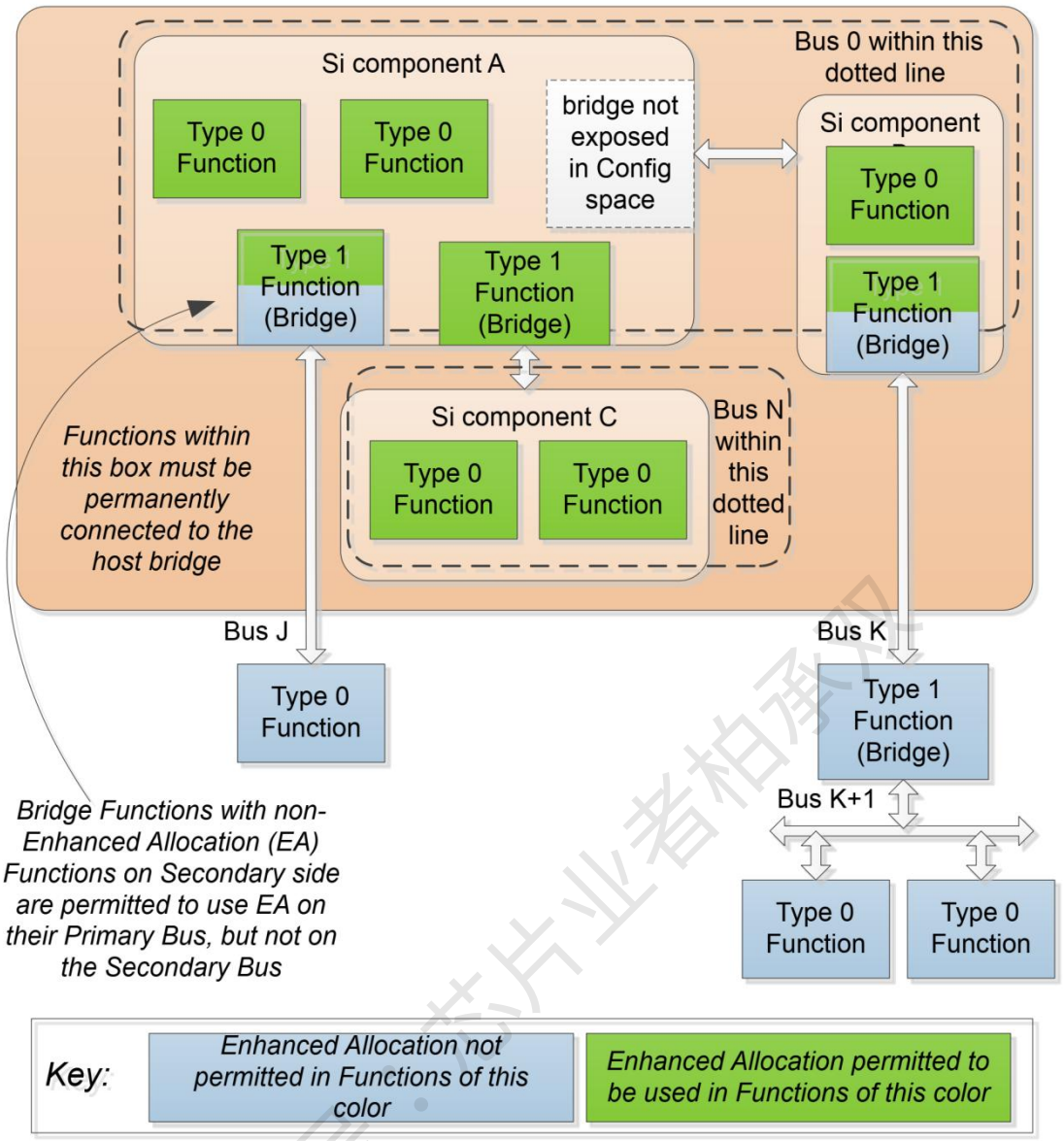

Figure 6-28 Example Illustrating Application of Enhanced Allocation

只有永久连接到主机桥的 Function 才允许使用 EA。当且仅网桥后面的 Function 也永久连接（在一个或多个网桥下方）到主机桥时，一个 Bridge Function（即具有 Type 1 配置头的任何 Function）才允许对其 Primary Side 和 Secondary Side 使用 EA，如 Figure 6-28 中的"Si component C"所示。

如果桥后面的 Function 未永久连接到桥，则 Bridge Function 只能在其 Primary Side 使用 EA，如 Figure 6-28 中的 Bus J 和 Bus K 上方的桥一样，在这种情况下总线编号，MMIO 范围和 I/O 范围的 Type 1 头中的非 EA 资源分配机制用于桥的 Secondary Side。系统软件必须确保分配的总线号在 EA 的 Fixed Secondary Bus Number 和 Fixed Subordinate Bus Number 寄存器中指定的范围内。系统软件必须确保分配的 MMIO 和 I/O 范围在 EA 的相应属性指示的范围内，以便在桥后分配资源。对于桥后面的总线号，MMIO 和 I/O 范围，允许硬件在多个 Bridge Function 中指示重叠范围，但是，在这种情况下，系统软件必须确保实际分配的范围不重叠。

仅依赖 EA 进行 I/O 和内存地址分配的 Function 必须将 PCI 头中所有 BAR 硬连线为 0。这些 Function 必须清楚地记录为依赖 EA 进行正确操作，平台集成商必须确保只有支持 EA 的固件/软件与这些 Function 一起使用。

当 Function 使用 EA 分配资源并指示资源范围与等效 BAR 编号相关联时，该 Function 不得通过等效 BAR 请求资源，而必须通过将等效 BAR 的所有位硬接线为 0 来指示资源。

对于根据上述规则被允许实现 EA 的桥，允许但不要求桥使用 EA 机制来指示位于桥后面的资源范围。在 Figure 6-28 所示的示例中，允许总线 N 上方的桥使用 EA 机制指示“Si component C”中两个 Function 所使用的资源，或者允许该桥不指示由“Si component C”中两个 Function 所使用的资源。系统固件/软件必须理解，不需要这样的桥来包含地表示桥之后的所有资源，因此系统固件/软件必须对桥后的所有 Function 进行完整搜索以理解这些 Function 所使用的资源。

带有 Expansion ROM 的 Function 允许使用现有机制或 EA 机制，但不允许两者都支持。如果某个 Function 使用 EA 机制（BEI 为 8 的 EA entry），则 Expansion ROM Base Address 和 Expansion ROM Enable 字段必须硬连线为 0（请参见 Section 7.5.1.2.4）。EA entry 的使能字段等效于 Expansion ROM Enable 字段。如果 Function 使用 Expansion ROM Base Address 寄存器机制，则不允许 BEI 为 8 的 EA entry。在这两种情况下，Expansion ROM 检验结果都使用 Expansion ROM Validation Status 和 Expansion ROM Validation Details 字段（请参阅 Section 7.5.1.2.4）表示。

EA 对启用和/或禁用 I/O 和/或存储器范围的解码的要求未更改，包括但不限于 Command 寄存器中的 Memory Space Enable 和 I/O Space Enable 字段。

使用 EA 分配的任何资源都不得与使用 EA 分配的任何其他资源重叠，除非上面为标识桥后的资源的允许地址范围所允许。

6.25 Emergency Power Reduction State

紧急功率降低状态（Emergency Power Reduction State）是一种可选实现的机制，用于请求 Function 快速降低其功耗。紧急功率降低是一种故障安全机制，旨在用于防止系统损坏，不用于提供常规的动态电源管理。

如果某个 Function 实现了 Emergency Power Reduction State，则它还必须实现 Power Budgeting Extended Capability，并且必须报告该状态的功率预算值（请参阅 Section 7.8.1）。集成在系统板上的设备不需要实现 Power Budgeting Extended Capability，但是如果实现了，则必须满足上述要求。

Function 根据自主或通过外部请求进入和退出此状态。外部请求可以遵循在适用的外形规格规范中定义的信令协议，也可以遵循特定于供应商的方法。Table 6-15 定义了 Emergency Power Reduction Supported 和 Emergency Power Reduction Initialization Required 字段如何判定允许触发进入和退出此状态的机制（请参阅 Section 7.5.3.15）。

Table 6-15 Emergency Power Reduction Supported Values

Emergency Power Reduction Supported	Emergency Power Reduction Initialization Required	Entry/Exit Permitted by		
		Form Factor Mechanism	Vendor Specific Mechanism(s)	Autonomous Mechanisms
00b	0	No	Yes	Yes
	1	No	No	No
01b	Any	No	Yes	Yes
10b	Any	Yes	Yes	Yes
11b	Reserved			

Function 可能指示它们从该状态退出时需要重新初始化：

- 如果 Emergency Power Reduction Initialization Required 字段为 0（Section 7.5.3.15）：
  - 进入此状态后，Function 要么正常运行（可能会降低性能），要么进入设备特定的“power reduction dormant state”。

设备的上游端口保持运行。由 Function 发起或定向到此 Function 的 Outstanding request 必须正常完成。

- 从该状态退出后，Function 将正常运行（也许会恢复正常性能）。进入“power reduction dormant state”的 Function 将退出该状态。无论哪种情况，都不需要软件干预。

- 如果 Emergency Power Reduction Initialization Required 字段为 1（Section 7.5.3.15）：
  - 进入此状态后，Function 停止正常运行。允许关联设备的上游端口进入 DL_Down。
    - 如果上游端口停留在 DL_Up 中，则指向 Function 或由 Function 发起的 outstanding request 必须正常完成。
    - 如果上游端口进入 DL_Down，则 Section 2.9.1 节中定义了 outstanding request 行为。此切换可能会导致 Surprise Down 错误。
    - 在此状态下必须保留粘性位。
  - 从该状态退出时，需要软件干预才能恢复正常操作。在需要时通知软件的机制超出了本规范的范围（例如，设备特定的中断）。如果上游端口进入 DL_Down，则将复位设备的所有 Function，并且需要完全重新配置（请参见 Section 2.9.2）。

以下规则适用于 Emergency Power Reduction State：

- 如果上游端口中至少有一个 Function 支持 Emergency Power Reduction State，则表示此设备支持 Emergency Power Reduction State（即，支持的紧急功率降低为非零）。
- 紧急功率降低状态与设备相关。设备中支持它的所有 Function 会同时进入和退出此状态。
- 只要上游端口保持在 DL_Up 中，Emergency Power Reduction Supported 字段为 00b 的 Function 就不受设备的紧急功率降低状态的影响。此时检测到的 Emergency Power Reduction Detected 字段是 RsvdZ。
- 当 Function 的 Emergency Power Reduction Supported 字段为 01b 或 10b：
  - 当设备进入紧急功率降低状态时，将 Emergency Power Reduction Detected 字段置 1。
  - 如果设备已退出紧急功率降低状态，则 Emergency Power Reduction Detected 字段清 0。
- 对于 Switch，Downstream Switch Port 与关联的 Upstream Switch Port 同时进入和退出紧急功率降低状态。Downstream Switch Port 配置空间中的相应字段为保留值。
- 对于 SR-IOV 设备，VF 与 PF 同时进入和退出紧急功率降低状态。VF 配置空间中的相应字段为保留值。
- 除非相关的外形规格规范定义了用于请求紧急功率降低的机制，否则不得使用编码 10b。
- 强烈建议在初始设备枚举之前，通过 Function 内的硬件或固件初始化 Emergency Power Reduction Supported 字段。当不可行时（例如，没有固件 ROM 时），可以将此初始化推迟到设备驱动程序加载中。

Diagnostic Checking of Emergency Power Reduction Detected

Emergency Power Reduction Detected 字段允许系统软件检测到进入紧急功率降低状态，即使是瞬间检测到。Emergency Power Reduction Request 字段可以由软件用于请求进入。通常，软件将使用特定于系统的方法通过外部机制进入紧急功率降低状态。

Emergency Power Reduction State: Example Add-in Card

Figure 6-29 显示了一个示例多设备附加卡，该卡支持 Emergency Power Reduction。请注意，设备 C 不支持紧急功率降低状态。设备 C 可能是扇出到设备 A 和 B 的 Switch。

公众号：芯片业者怕承双

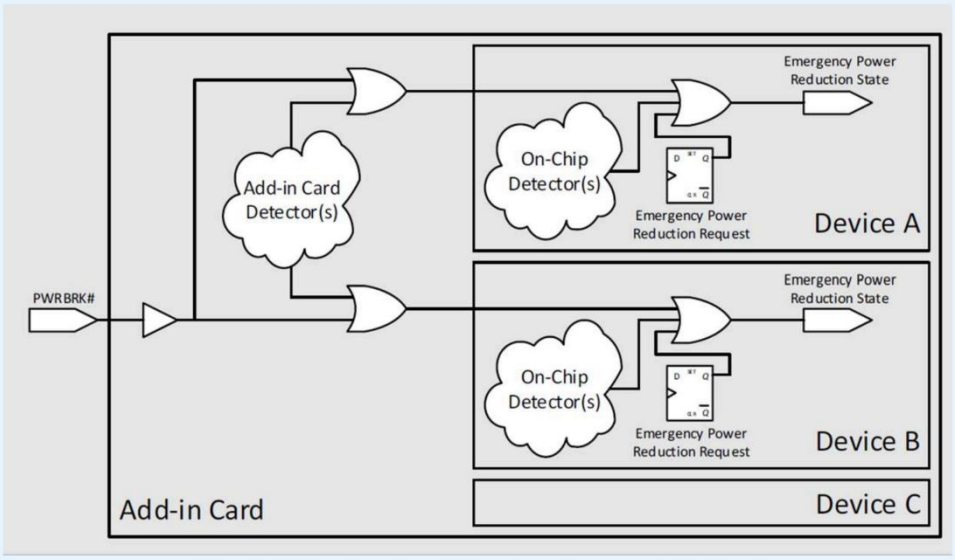

Figure 6-29 Emergency Power Reduction State: Example Add-in Card

6.26 Hierarchy ID Message

当软件初始化 PCI 层次结构时，它将为每个组件分配唯一的总线和设备号，以便层次结构中的每个 Function 在该层次结构中都具有唯一的 Routing ID。为确保 Routing ID 在包含多个层次结构的大型系统中和在包含多个层次结构的集群系统中是唯一的，需要附加信息来扩充 Routing ID 以产生唯一编号。可以通过以下组合来唯一标识 Function：

- Unique Identifier for the System (or RC)
- Unique Identifier for the Hierarchy within that RC
- Routing ID within that Hierarchy

Hierarchy ID Message（请参阅 Section 2.2.8.6.5）用于为 Function 提供在多层次结构平台中唯一标识其自身所需的附加信息。

Hierarchy ID Message 由 Downstream Port 根据软件请求生成。Upstream Port 收到此消息后在 Hierarchy ID Extended Capability 进行报告（参阅 Section 7.9.18）。

Hierarchy ID Message 是 PCI-SIG 定义的 Type 1 VDM。Hierarchy ID Message 可以随时安全地发送，不认识它们的组件将静默忽略它们。

Hierarchy ID Message 通常是从层次结构顶部的 Downstream Port（例如 Root Port）发送的。在根端口不支持 Hierarchy ID Message 的系统中，可以从 Switch Downstream Port 发送 Hierarchy ID Message。

Hierarchy ID Message 旨在供软件，固件和/或硬件使用。使用 Hierarchy ID Message 时，必须比较 Hierarchy ID、System GUID 和 System GUID Authority ID 字段的所有位，而不考虑任何内部结构。如何使用此信息超出了本规范的范围。

Hierarchy ID Message 的布局如 Figure 2-33 所示。Hierarchy ID Message 中的字段如下：

Hierarchy ID 包含与此层次结构关联的 Segment Group Number（由 PCI Firmware Specification 定义）。该字段可以与 Routing ID 结合使用，以唯一地标识系统中的 Function。值 0000h 表示 RC 的默认（或唯一）层次结构。非零值表示其他层次结构。

System GUID[143:0]与 System GUID Authority ID 一起为系统提供全局唯一标识。

System GUID[143:136]对应 Hierarchy ID Message 中的 byte 14 。

System GUID[135:128]对应 Hierarchy ID Message 中的 byte 15 。

公众号·芯片业者也承双

...

System GUID[7:0]对应 Hierarchy ID Message 中的 byte 31 。

System GUID Authority ID 标识用于确保系统 GUID 全局唯一的机制。选择给定系统使用哪个 Authority ID 的机制是特定于实现的。定义的值在 Table 6-16 中显示。

Table 6-16 System GUID Authority ID Encoding

Authority ID	Description
00h	**None** - System GUID[143:0] is not meaningful.System GUID[143:0] must be 0.
01h	**Timestamp** - System GUID [63:0]包含与特定系统关联的时间戳。 编码是 Unix 64 位时间（自 1970 年 1 月 1 日午夜 UTC 以来的秒数）。 选择时间戳代表系统的机制是特定于实现的。 System GUID [143:64]必须为 0。
02h	**IEEE EUI-48** - System GUID [47:0]包含与特定系统相关联的 48 位扩展唯一标识符（EUI-48）。 编码由 IEEE 定义。有关详细信息，请参见[EUI-48]。EUI-48 值通常用作网络接口 MAC 地址。 选择 EUI-48 值代表系统的机制是特定于实现的。 System GUID [143:48]必须为 0。
03h	**IEEE EUI-64** - System GUID [63:0]包含与特定系统相关联的 64 位扩展唯一标识符（EUI-64）。 编码由 IEEE 定义。有关详细信息，请参见[EUI-64]。 选择 EUI-64 值代表系统的机制是特定于实现的。 System GUID [143:64]必须为 0。
04h	**FC-4122 UUID** - System GUID [127:0]包含由[RFC-4122]中的 IETF 定义的 UUID。该定义在技术上等同于[ITU-T-Rec.-X.667]或[ISO-IEC-9834-8]。 选择 UUID 值代表系统的机制是特定于实现的。 System GUID [143:128]必须为 0。
05h	**IPv6 Address** - System GUID [127:0]包含系统网络接口之一的唯一 IPv6 地址。 选择 IPv6 值代表系统的机制是特定于实现的。 System GUID [143:128]必须为 0。
06h-7Fh	**Reserved** - System GUID[143:0]包含唯一值。确保唯一性的机制超出了本规范的范围。
80h-FFh	**PCI-SIG Vendor Specific** - System GUID Authority ID 值 80h 至 FFh 保留用于 PCI-SIG 供应商特定用途。 System GUID [143:128]包含 PCI-SIG 分配的 Vendor ID。

公众号·芯片业者相承双

System GUID [127:0]包含该供应商分配的唯一编号。用于分配数字的机制是特定于实现的。 一种可能的机制是使用分配给系统的序列号。

在这些系统 GUID 授权 ID 之间进行选择的机制是特定于实现的。 一种用法是允许供应商定义多达 128 个不同的 128 位系统 GUID 方案。

System GUID Consistency and Stability

为了支持 System GUID 的目的，软件应确保单个系统在任何地方都使用相同的 System GUID 和 System GUID Authority ID 值。

实现者应仔细考虑其对 System GUID 值的稳定性要求。例如，某些用例可能要求重新启动系统时该值保持不变。在那些情况下，如果结果由于硬件故障，系统重新配置或发现算法中的变化/并行度而发生变化，则选择与发现的第一个以太网 MAC 地址关联的 EUI-48 值的机制可能会出现问题。

Hierarchy ID vs. Device Serial Number

Device Serial Number 机制还可以用于唯一标识组件（请参见 Section 7.9.3）。如果 Device Serial Number 涉及与每个组件关联的 ROM，则可能是解决此问题的更昂贵的方法。

Virtual Functions and Hierarchy ID

Hierarchy ID capability 可以由 Virtualization Intermediary（VI）进行仿真。这样做使 VF 软件可以访问此 Hierarchy ID 的信息。

当 VF 硬件需要访问此信息时，VF 应该实现 Hierarchy ID capability。这样就可以访问 VF 软件和硬件。

在某些情况下，VF 应该获得与 PF 相同的信息。在其他情况下，尤其是那些涉及虚拟机迁移的情况，可能需要为 VF 提供不同于关联 PF 和与该 PF 关联的其他 VF 的 Hierarchy ID 信息。

支持以下机制：

	VF Hierarchy ID Capability	Hierarchy ID VF Configurable	Hierarchy ID Writeable	VF Software has access	VF Hardware has access	VF Hierarchy Data / GUID
1	Not Present	n/a	n/a	No	No	Not Emulated
2				Yes	No	Emulated
3	Present	0b	0b	Yes	Yes	Same as PF
4		1b	0b	Yes	Yes	Same as PF
5		1b	1b	Yes	Yes	Configured by VI

在 mechanism 1 中，Virtualization Intermediary 不模拟该 Function。VF 软件和硬件无法访问。

在 mechanism 2 中，Virtualization Intermediary 模拟 Function 并返回所需的任何 Hierarchy ID 信息。VF 软件具有访问权限。VF 硬件无权访问。

在 mechanism 3 和 4 中，VF 信息与 PF 相同，并从收到的 Hierarchy ID Message 中自动填充。VF 硬件和软件都可以访问。

在 mechanism 5 中，VF 信息由软件（可能是 VI）配置。VF 硬件和软件都可以访问。

公众号·志先业者拒绝双

6.27 Flattening Portal Bridge (FPB)

6.27.1 Introduction

Flattening Portal Bridge（FPB）是一种可选实现的机制，可用于提高 Routing ID 和内存空间资源的可伸缩性和运行时重新分配。

对于与上游端口关联的 non-ARI Function，Routing ID 包括一个 3-bit 的 Function Number（由上游端口硬件的结构确定）和一个 13-bit 的 Bus Number 和 Device Number（由以下部分确定）： 上游端口上方的下游端口。

对于与上游端口关联的 ARI Function，Routing ID 由 8-bit 的 Function Number 和由上游端口上方的下游端口确定 8-bit 的 Bus Number 构成。

实现 FPB Capability 的桥本身也可以称为 FPB。FPB Capability 可以应用于任何逻辑桥，如 Figure 6-30 所示。

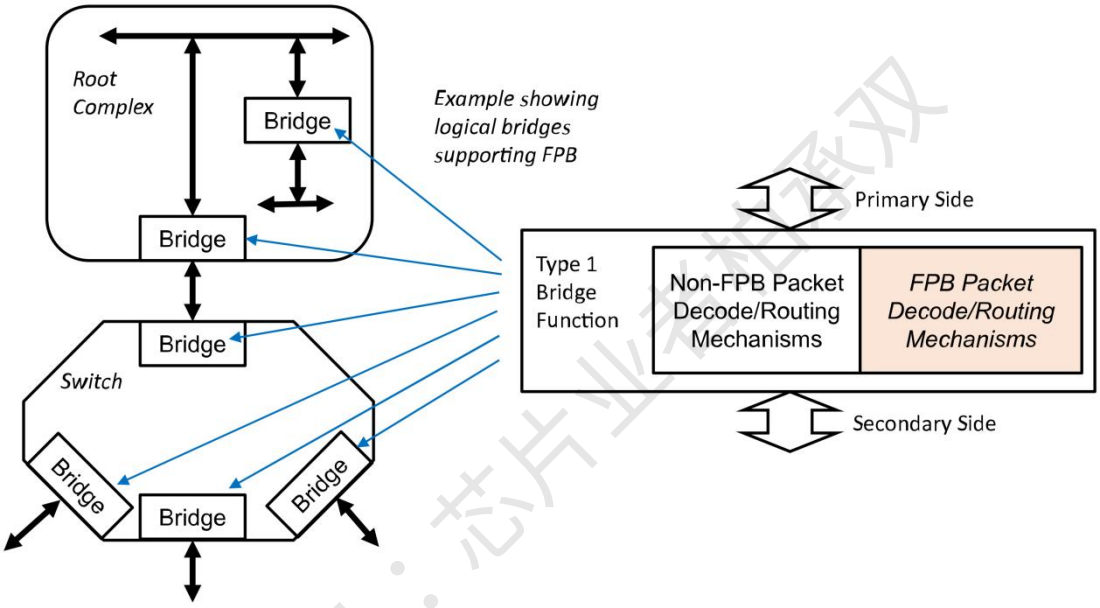

Figure 6-30 FPB High Level Diagram and Example Topology

FPB 通过“扁平化”Switch 内部和下游端口使用 Bus Number 的方式（见 Figure 6-31），改变 Switch 使用 Bus Number 的方式以减少浪费。

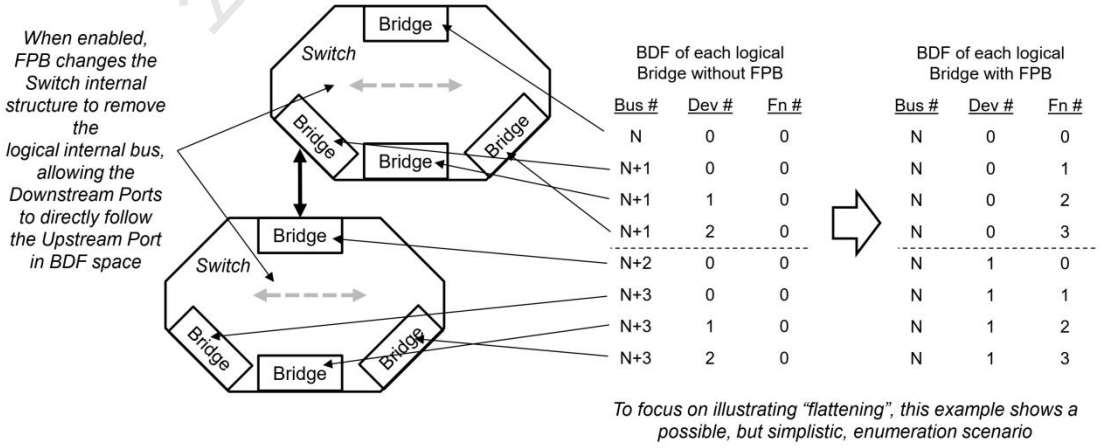

Figure 6-31: Example Illustrating “Flattening” of a Switch

FPB 定义了一种机制，使得系统软件可以在非连续范围内分配 Routing ID 和内存空间资源，从而使系统软件能够分配这些资源池，从而可以从中分配“ bin”给 FPB 下方的 Function。这是通过使用位向量完成的，其中，在对应位置 1 时，每个位都将相应范围的资源分配给桥的 Secondary Side（见 Figure 6-32）。

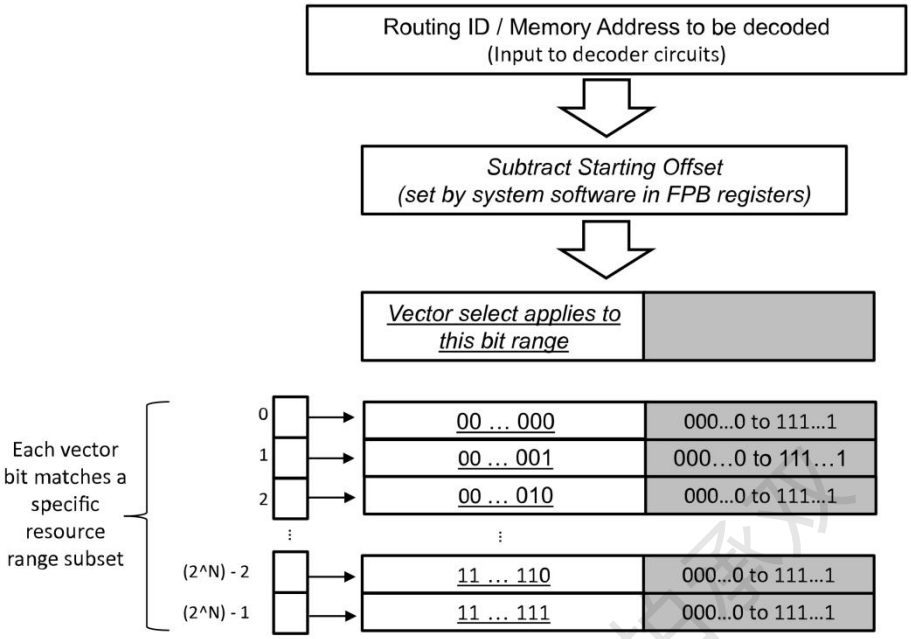

Figure 6-32 Vector Mechanism for Address Range Decoding

这允许系统软件分配设备热添加所需的 Routing ID 和/或内存空间资源，而不必重新平衡其他已经分配的资源范围，并返回释放的资源池，例如通过热移除事件。

FPB 被定义为允许 non-FPB 机制和 FPB 机制同时运行，例如，系统固件/软件可以实现一种策略，其中在系统中不需要 FPB 机制的地方继续使用 non-FPB 机制（请参阅 Figure 6-33）。在此图中，假设将给定的 TLP 解码为与桥的 Secondary Side 相关联时，解码逻辑将输出“1”。non-FPB 解码机制适用于没有 FPB 的情况，因此，例如，当评估 ID 路由的 TLP 时，non-FPB 解码逻辑仅测试 Routing ID 的总线号部分（Bit 15:8）。

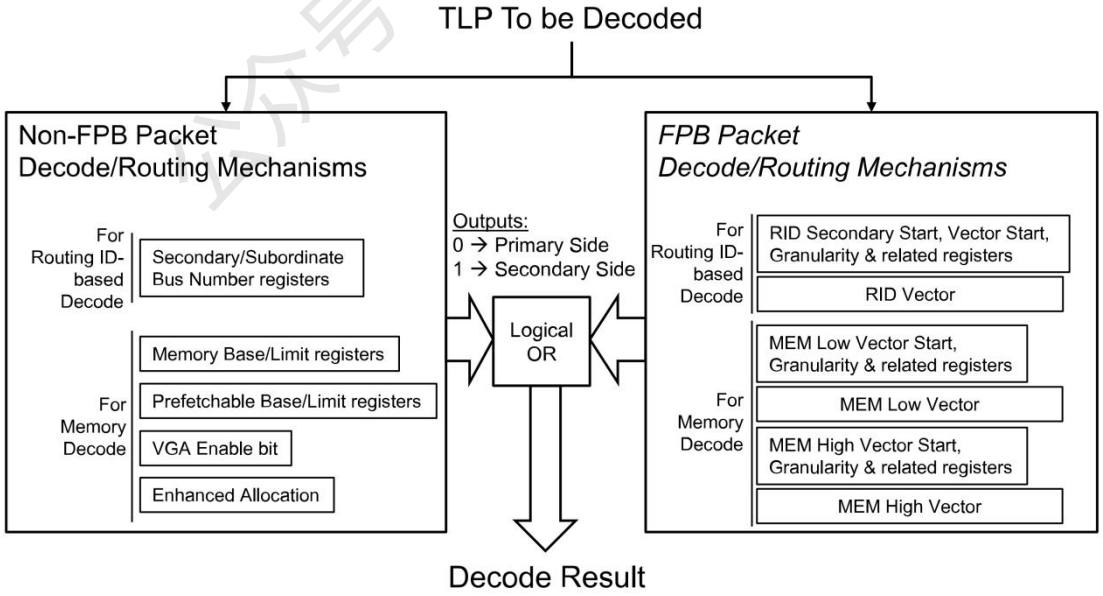

Figure 6-33 Relationship between FPB and non-FPB Decode Mechanisms

重要的是要认识到，尽管 FPB 为特定桥增加了解码特定 TLP 的其他方式，但 FPB 并未改变桥在 Switch 和 RC 架构中运行的基本方

式。FPB 使用相同的体系结构概念为三种不同的资源类型提供管理机制：

1. Routing IDs
2. Memory below 4 GB (“MEM Low”)
3. Memory above 4 GB (“MEM High”)

FPB 的硬件实现允许支持这三种机制的任意组合。对于每个机制，FPB 使用位向量来指示，对于所选资源类型的特定子集范围，该范围内的资源是否与 FPB 的 Primary Side 或 Secondary Side 相关联。允许硬件实现为这些向量实现小范围的大小，并且系统固件/软件能够通过选择应用向量的初始偏移量和向量内的单个比特的粒度来指示资源的大小，从而最有效地利用可用向量给定向量中的位所适用的范围。

6.27.2 Hardware and Software Requirements

当使用任何 FPB 机制时，以下规则适用：

- 如果系统软件违反了有关 FPB 的任何规则，则硬件行为是未定义的。
- 允许在任何 PCI 桥（Type 1）Function 中实现 FPB，每个实现 FPB 的 Function 必须实现 FPB Capability（见 Section 7.8.10）。
- 如果 Switch 实现 FPB，那么 Switch 的上游端口和所有下游端口都必须实现 FPB。
- 允许软件在某些 Switch 端口启用 FPB，而在其他端口不启用 FPB。
- 允许 RC 在某些 Root Port 上实现 FPB，但在其他 Root Port 上不实现 FPB。
- 允许使用 Type 1 Function 来实现适用于以下任何一种，两种或三种基本机制的 FPB 机制：
  - Routing IDs
  - Memory below 4 GB (“MEM Low”)
  - Memory above 4 GB (“MEM High”)
- 允许系统软件启用特定 FPB 支持的任何基本机制的组合（包括全部或不包含）。
- 错误处理和报告机制，除非在本节中进行了明确修改，否则不受 FPB 的影响。
- FPB Function 复位后，FPB 硬件必须清除所有已实现向量中的所有位。
- 启用后（通过 FPB RID Decode Mechanism Enable、FPB MEM Low Decode Mechanism Enable 和/或 FPB MEM High Decode Mechanism Enable 字段），如果系统软件随后禁用 FPB 机制，则关联向量中条目的值未定义 ，并且如果系统软件随后重新启用了 FPB 机制，则 FPB 硬件必须清除关联向量中的所有位。
- 如果使用 No_Soft_Reset 位清 0 来实现 FPB，则当该 FPB 通过 D0→$D3_{Hot}$→D0 循环时，必须禁用所有 FPB 机制，并且 FPB 必须清除所有已实现向量中的所有位。
- 如果使用 No_Soft_Reset 位置 1 来实现 FPB，则当该 FPB 通过 D0→$D3_{Hot}$→D0 循环时，所有 FPB 配置状态都不得更改，并且 FPB 向量中的条目必须由硬件保留。
- 不需要硬件对 FPB 计算执行任何类型的边界检查，并且系统软件必须确保正确编程了 FPB 参数。
  - 明确允许系统软件对 Vector Start 值进行编程，以使相应向量的高阶位超过与给定 FPB 相关的资源范围，但是在这些情况下，系统软件必须确保向量的那些高阶位是 0。

公众号·芯片业者相承双

- 系统软件必须避免的错误示例包括资源分配重复，起始偏移量与设置的矢量位的组合，这些组合位可能会产生“环绕”或边界错误。

以下规则适用于 FPB Routing ID（RID）机制：

- 如果 Bus Number 部分落入由 Secondary 和 Subordinate Bus Number 寄存器中编程的值指示的 Bus Number 范围内，则 FPB 硬件必须考虑将特定范围的 RID 与 FPB 的 Secondary Side 相关联，或与编程到 FPB RID Vector 中的相应的值相关联。
- 如果打算仅使用 FPB RID 机制进行 BDF 解码，则系统软件必须确保 Secondary 和 Subordinate Bus Number 寄存器均为 0。
- 系统软件必须确保对 FPB 路由机制进行配置，以便 FPB 将针对 Function 的配置请求从 FPB 的 Primary Side 路由到 Secondary Side。

当未启用 ARI 时，可以使用不同的粒度应用 FPB RID 机制，系统软件可以通过 FPB RID Vector Control 1 寄存器中的 FPB RID Vector Granularity 字段对 FPB RID 机制进行编程。Figure 6-34 说明了 RID 的布局与支持的粒度之间的关系。在考虑以下定义的要求以及在 Flattening Portal Bridge (FPB) Capability 的定义（请参阅 Section 7.8.6）中，读者可能会发现参考该图很有帮助。

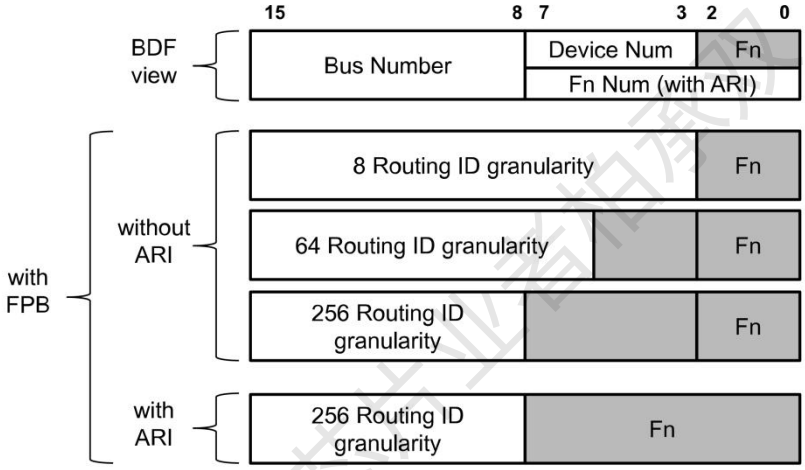

Figure 6-34 Routing IDs (RIDs) and Supported Granularities

- 系统软件必须按照 FPB RID Vector Control 1 寄存器中描述的约束，对 FPB RID Vector Granularity 和 FPB RID Vector Start 字段进行编程。
- 对于所有 FPB（与 Switch 上游端口相关的 FPB 除外）：
  - 当不支持 ARI Forwarding 时，或者 Device Control 2 寄存器中的 ARI Forwarding Enable 字段清零时，FPB 硬件必须在以下情况下将 FPB 的 Primary Side 接收到的内容转换为 FPB 的 Secondary Side 的 Type 0 类型配置请求： Type 1 配置请求的 Routing ID 的位 15:3 与 FPB RID Vector Control 2 寄存器的 RID Secondary Start 字段中的值匹配，并且系统软件必须相应地配置 FPB。
  - 当 Device Control 2 寄存器中的 ARI Forwarding Enable 置 1 时，当总线号部分出现时，FPB 硬件必须将在 FPB 的 Primary Side 收到的 Type 1 配置请求转换为在 FPB 的 Secondary Side 收到的 Type 0 配置请求。Type 1 配置请求的 Routing ID 的 FOB 与 FPB RID Vector Control 2 寄存器的 Secondary Start 字段的总线号地址（仅位 15:8）中的值匹配，并且系统软件必须相应地配置 FPB。
- 仅对于与 Switch 的上游端口相关联的 FPB，当 FPB RID Decode Mechanism Enable 设置为 1 时，FPB 硬件必须使用 FPB Capability 寄存器的 FPB Num Sec Dev 字段来指示与设备的 Secondary Side 相关联的设备号的数量。上游端口桥，除了 FPB RID Vector Control 2 寄存器中的 RID Secondary Start 字段外，FPB 还必须使用它来确定 FPB 的 Primary Side 收到的配置请求何时以 Switch 的下游端口之一为目标 ，确定何时必须将此类请求从 Type 1 配置请求转换为 Type 0 配置请求，并且系统软件必须适当地配置 FPB。

- 配置 FPB 的系统软件必须理解，Switch 的逻辑内部结构将根据 Switch 上游端口中的 FPB RID Decode Mechanism Enable 字段的值而改变。
- 下游端口必须使用其相应的 RID 值以及其 Requester ID 和 Completer ID，这些值由上游端口的 FPB Num Sec Dev 和 RID Secondary Start 值确定。
- 在由 Switch Upstream Port Function 的 RID Secondary Start 和 FPB Num Sec Dev 确定的范围内，所有已实现的 Function 必须是与该 Switch Upstream Port 相关联的 Switch Downstream Port； 需要系统软件来扫描该范围内的所有 Function，以判定已实现的 Function。
- 强烈建议系统软件分配 RID Secondary Start，以使总线号和设备号与 Switch Upstream Port 的不同。 否则，导致的硬件行为是不确定的。

- FPB 必须为 INTx 虚拟线实现桥映射（请参阅 Section 2.2.8.1）。
- 硬件和软件必须应用此算法（或等效的逻辑方法）来确定 FPB RID Vector 中的哪个条目适用于给定的 Routing ID（RID）地址：
  - 如果 RID 低于 FPB RID Vector Start 值，则 RID 超出范围（在起始值以下），因此无法与桥的 Secondary Side 相关联，否则
  - 首先减去 FPB RID Vector Start 的值，然后根据 FPB RID Vector Granularity 的值除以确定矢量中的位索引，从而计算矢量中的偏移。
  - 如果位索引值大于 FPB RID Vector Size Supported 指示的长度，则 RID 超出范围（超出向量覆盖的范围的顶部），因此不能与桥的 Secondary Side 相关联，否则
  - 如果在计算出的位索引位置处向量内的位值为 1b，则 RID 地址与桥的 Secondary Side 相关，否则，RID 地址与桥的 Primary Side 相关。

以下规则适用于 FPB MEM Low 机制：

FPB MEM Low 机制可以应用不同的粒度，可以通过 FPB MEM Low Vector Control 寄存器中的 FPB MEM Low Vector Granularity 字段由系统软件进行编程。Figure 6-35 说明了应用 FPB MEM Low 机制的 4 GB 以下内存地址空间中地址布局之间的关系。在考虑以下定义的要求以及在 Flattening Portal Bridge (FPB) Capability (see Section 7.8.10)的定义中，读者可能会发现参考该图很有帮助。

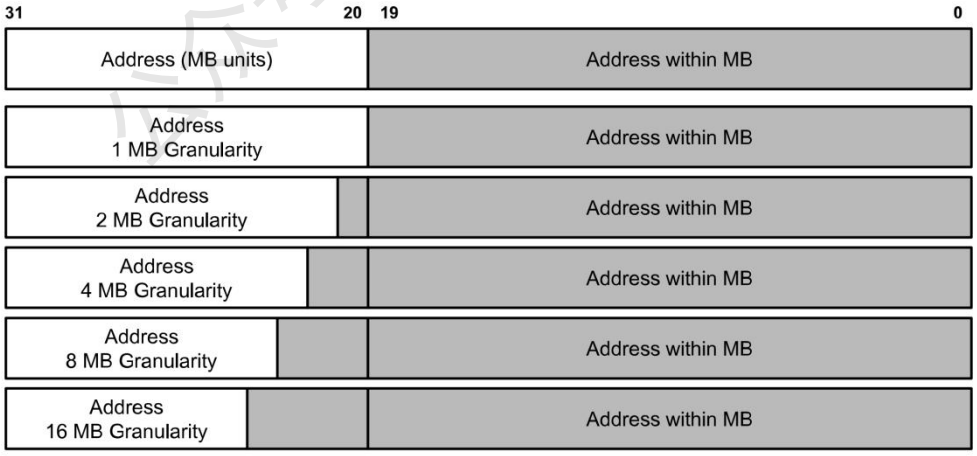

Figure 6-35 Addresses in Memory Below 4 GB and Effect of Granularity

- 系统软件必须根据 FPB MEM Low Vector Control 寄存器中描述的约束，对 FPB MEM Low Vector Control 寄存器中的 FPB MEM Low Vector Granularity 和 FPB MEM Low Vector Start 字段进行编程。
- FPB 硬件必须考虑与 FPB 的 Secondary Side 相关联的特定内存地址，前提是该内存地址在逻辑上或与 FPB MEM Low 中相应

公众号·芯片业者怕承双

条目中编程的值一起在其他桥内存解码寄存器（如下所述）中编程的值所指示的任何范围内矢量。其他桥存储器解码寄存器包括：

- Memory Base/Limit registers
- Prefetchable Base/Limit registers
- VGA Enable bit in the Bridge Control register
- Enhanced Allocation (EA) Capability (if supported)
- FPB MEM High mechanism (if supported and enabled)

- 硬件和软件必须应用此算法（或等效的逻辑方法）来确定 FPB MEM Low Vector 中的哪个条目适用于给定的存储器地址：
  - 如果 Memory address 低于 FPB MEM Low Vector Start 值，则 Memory address 超出范围（在起始值以下），因此无法与桥的 Secondary Side 相关联，否则
  - 首先减去 FPB MEM Low Vector Start 的值，然后根据 FPB MEM Low Vector Granularity 的值除以确定矢量中的位索引，从而计算矢量中的偏移。
  - 如果位索引值大于 FPB MEM Low Vector Size Supported 指示的长度，则 Memory address 超出范围，因此不能与桥的 Secondary Side 相关联，否则
  - 如果在计算出的位索引位置处向量内的位值为 1b，则 Memory address 与桥的 Secondary Side 相关，否则，Memory address 与桥的 Primary Side 相关。

以下规则适用于 FPB MEM High 机制：

- 系统软件必须根据 FPB MEM High Vector Control 1 寄存器中描述的约束，对 FPB MEM High Vector Control 1 寄存器中的 FPB MEM High Vector Granularity 和 FPB MEM High Vector Start Lower 字段进行编程。
- FPB 硬件必须考虑与 FPB 的 Secondary Side 相关联的特定内存地址，前提是该内存地址在逻辑上或与 FPB MEM High Vector 中相应条目中编程的值一起在其他桥内存解码寄存器（如下所述）中编程的值所指示的任何范围内矢量。其他桥存储器解码寄存器包括：
  - Memory Base/Limit registers
  - Prefetchable Base/Limit registers
  - VGA Enable bit in the Bridge Control register
  - Enhanced Allocation (EA) Capability (if supported)
  - FPB MEM Low mechanism (if supported and enabled)
- 硬件和软件必须应用此算法（或等效的逻辑方法）来确定 FPB MEM High Vector 中的哪个条目适用于给定的存储器地址：
  - 如果 Memory address 低于 FPB MEM High Vector Start Upper/FPB MEM High Vector Start Lower 值，则 Memory address 超出范围（在起始值以下），因此无法与桥的 Secondary Side 相关联，否则
  - 首先减去 FPB MEM High Vector Start Upper/FPB MEM High Vector Start Lower 的值，然后根据 FPB MEM High Vector Granularity 的值除以确定矢量中的位索引，从而计算矢量中的偏移。
  - 如果位索引值大于 FPB MEM High Vector Size Supported 指示的长度，则 Memory address 超出范围，因此不能与桥的 Secondary Side 相关联，否则

公众号：芯片业者柏承双

- 如果在计算出的位索引位置处向量内的位值为 1b，则 Memory address 与桥的 Secondary Side 相关，否则，Memory address 与桥的 Primary Side 相关。

FPB Address Decoding

FPB 使用位向量机制来解码 Routing ID 的范围以及 4 GB 以上的内存地址。支持 FPB 的桥在支持使用 FPB 的每种资源类型/范围内包含以下内容：

- A Bit vector
- A Start Address
- A Granularity

桥使用它们来确定给定地址是否属于 FPB 解码的范围的一部分，该范围与桥的 Secondary Side 相关联。使用 non-FPB 解码机制和 FPB 解码机制中的任何一个或两个确定不与桥的 Secondary Side 相关联的地址（默认情况下）与桥的 Primary Side 相关联。在这里，当我们使用“关联的”一词时，是指，例如，桥将对 TLP 进行以下处理：

- Associated with Primary, Received at Primary → Unsupported Request (UR)
- Associated with Primary, Received at Secondary → Forward upstream
- Associated with Secondary, Received at Primary → Forward downstream
- Associated with Secondary, Received at Secondary → Unsupported Request (UR)

在 FPB 中，向量中的每个位代表一个资源范围，其中该范围的大小由所选的粒度确定。如果向量中的某个位置 1，则表示寻址到相应范围内地址的 TLP 将与桥的 Secondary Side 相关联。每个位代表的资源的特定范围取决于该位的索引以及 Start Address & Granularity 中的值。Start Address 指示由位向量描述的最低地址。 Granularity 表示由每个位表示的区域的大小。向量中的每个连续位适用于后续范围，并根据粒度随每个位增加。

例如，考虑使用 FPB 来描述 MEM Low 范围的桥。FPB MEM Low Vector Start 设置为 FC0h，指示位向量描述的范围始于地址 FC000000。FPB MEM Low Vector Granularity 设置为 0000b，指示每一位代表 1 MB 范围。

从这些值中，我们可以确定向量的 Bit-0 代表从 FC000 0000（FC00 0000-FC0F FFFF）开始的 1MB 范围，Bit-1 代表 FC10 0000-FC1F FFFF 等。

向量中设置为 0 的比特表示该范围不包括在 FPB 描述的范围内。在上面的示例中，如果 Bit-0 为 0，则由于 FPB，寻址到 FC00 0000 和 FC0F FFFF 之间的任意位置的数据包不应路由到桥的 Secondary Bus。

Hardware and Software Considerations for FPB

FPB 旨在解决 PCI/PCIe 体系结构中与资源分配效率低下有关的一类问题。 这些问题可以分为“静态”或“动态”用例场景，其中静态用例是指在系统启动时分配资源，然后通常不会再次更改的场景，而动态用例是指运行时分配资源的场景。由于热插入/移除或其他需求，需要重新平衡（例如，分配新资源，释放不再需要的资源）。

在静态情况下，由于使用额外的总线号，并且由于 Switch 和下游端口的 PCI/PCIe 体系结构定义而导致缺少设备号，因此层次结构的大小和 Endpoint 数受到限制。FPB 通过“扁平化”Routing ID（RID）来解决此类问题，从而使 Switch 和下游端口能够更有效地利用可用 RID。

对于没有 FPB 的动态情况，避免重新平衡的“最佳方法”是在相关端口或 Endpoint 点上方的桥中保留大范围的总线号和内存空间，以便希望可以在预定义时间内满足任何将来的需求。这会导致潜在的未使用分配，这使 Routing ID 问题变得更糟，并且在资源受限的平台中，即使在相对简单的情况下（例如，可能有一个实现单个扩展卡的附加卡）的情况下，这种方法也难以实现，Endpoint

公众号：芯片业者相承双

点被另一个具有 Switch 和两个 Endpoint 的附加卡代替，因此尽管仅分配一个总线的初始分配就足够了，但是新分配卡会立即中断初始分配。

对于内存空间，当热插拔 Endpoint 可能需要分配小于 4 GB 的内存空间时，预分配方法会出现问题，这从本质上来说是一种有限的资源，即使分配的内存量相对较小，也会很快用完， 由于多个系统元素对系统地址空间的分配要求低于 4 GB，因此对于哪些预分配没有吸引力。

FPB 包括为 Requester ID 和内存空间启用不连续资源范围分配/重新分配的机制。目的是使系统软件能够维护可在运行时分配（并释放回）的资源“池”，而不会中断重新平衡所需的其他正在进行的操作。

为支持系统软件在运行时使用 FPB，FPB 硬件实现应避免在运行中引入暂停或其他类型的事务中断，包括在系统软件修改 FPB 硬件状态期间。然而，预计硬件不会试图识别系统软件错误地修改 FPB 配置，从而影响传输中的事务的情况。与非 FPB 机制一样，系统软件有责任确保系统操作不会因重新配置操作而损坏。

不明确要求系统固件/软件以特定顺序执行 FPB 机制的启用和/或禁用，但是应注意在层次结构中实现资源分配操作，以便系统的硬件和软件元素不会损坏或导致故障。

6.28 Vital Product Data (VPD)

重要产品数据（VPD，Vital Product Data）是为系统的硬件、软件和微码元素等信息唯一定义的。VPD 为系统提供有关各种 FRU（Field Replaceable Unit，现场可替换单元）的信息，包括 Part Number、Serial Number 和其他详细信息。VPD 还提供了一种机制，用于在受监视的设备上存储诸如性能和故障数据之类的信息。 从系统的角度来看，目标是通过从硬件，软件和微码元素中读取信息来收集此信息。

附加卡中支持 VPD 是可选的，具体取决于制造商。尽管支持 VPD 是可选的，但由于其对附加卡、系统制造商和即插即用的固有好处，因此鼓励附加卡制造商提供 VPD。

Section 7.9.19 节记录了访问 VPD 的机制。

PCI Express 的 VPD 与[PCI-3.0]中的定义保持不变。反过来，这个定义是基于[PCI]和[PLUG-PLAY-ISA-1.0a]的早期版本。

重要产品数据（VPD）由 Small Resource Data Type 和 Large Resource Data Type 组成。

Table 6-17 Small Resource Data Type Tag Bit Definitions

Offset	Field Name		
Byte 0	Value = 0xxx xyyyb		
	Bit 7	Small Resource Type	0b
	Bits 6:3	Small Item Name	xxxx
	Bits 2:0	Length in bytes	yy
Bytes 1 to n	Actual information		

Table 6-18 Large Resource Data Type Tag Bit Definitions

Offset	Field Name		
Byte 0	Value = 1xxx xxxxb		
	Bit 7	Large Resource Type	1b
	Bits 6:0	Large Item Name	xxxxxxx
Byte 1	Length in bytes of data items bits[7:0] (lsb)		
Byte 2	Length in bytes of data items bits[15:8] (msb)		
Bytes 3 to n	Actual data items		

第一个 VPD 标签是 Identifier String（02h），提供设备的产品名称。

一个 VPD-R（10h）标签用作只读关键字的 Header。VPD-R 列表（包括标签和长度）的校验和必须为零。写入只读数据的尝试将作为禁止操作执行。

一个 VPD-W（11h）标签用作读写关键字的 Header。包含读/写数据的存储组件是一个非易失性设备，断电时将保留数据。

最后一个标记必须是 End Tag（0Fh）。

Table 6-19 给出了一个典型 VPD 中使用的资源数据类型标签的小例子。

Table 6-19 Resource Data Type Flags for a Typical VPD

TAG 02h	Identifier String	Large Resource Data Type
TAG 10h	VPD-R list containing one or more VPD keywords	Large Resource Data Type
TAG 11h	VPD-W list containing one or more VPD keywords	Large Resource Data Type
TAG 0Fh	End Tag	Small Resource Data Type

6.28.1 VPD Format

VPD 资源类型中的信息字段由一个三字节的报头和一些数据组成（见 Figure 6-36）。三字节报头包含一个两字节的关键字和一个字节的长度。关键字是两个字符（ASCII）的助记符，它唯一地标识字段中的信息。报头的最后一个字节是二进制的，表示后面数据的长度值（以字节为单位）。

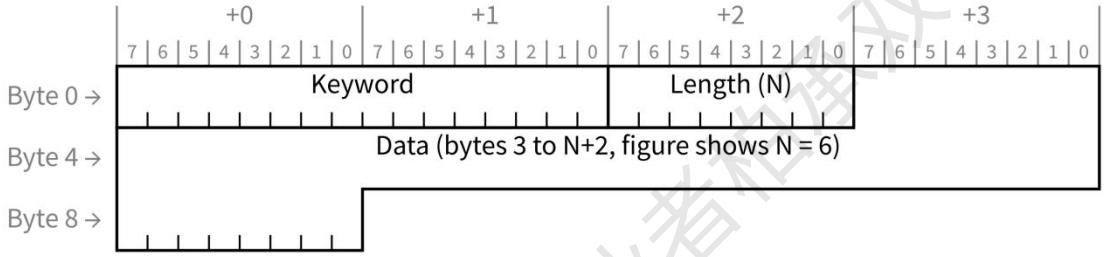

Figure 6-36 VPD Format

VPD 关键字分为两类：只读字段和读/写字段。除非另有说明，否则关键字数据字段以 ASCII 字符形式提供。使用 ASCII 可以在不同的企业计算机系统之间移动关键字数据，而不存在翻译困难。

“add-in card serial number”的 VPD 项的示例如下：

Table 6-20 Example of Add-in Serial Card Number

Byte 0	53h “S”	Keyword: SN
Byte 1	4Eh “N”	
Byte 3	08h	Length: 8
Byte 4	30h “0”	Data: “00000194”
Byte 5	30h “0”	
Byte 6	30h “0”	
Byte 7	30h “0”	
Byte 8	30h “0”	
Byte 9	31h “1”	
Byte 10	39h “9”	
Byte 11	34h “4”	

公众号·芯片业者相建双

6.28.2 VPD Definitions

本节描述了当前的 VPD 大小资源数据标签以及 VPD 关键字。此列表可以随时增强。希望定义新关键字的公司应与 PCISIG 联系。所有未指定的值均保留用于 SIG 分配。

6.28.2.1 VPD Large and Small Resource Data Tags

VPD 包含在四种类型的 Large 和 Small Resource Data Tag 中。PCI 设备中可能会提供以下标签和 VPD 关键字字段。

Table 6-21 VPD Large and Small Resource Data Tags

Tag	Description
Large resource type Identifier String Tag (02h)	This tag is the first item in the VPD storage component. It contains the name of the add-in card in alphanumeric characters.
Large resource type VPD-R Tag (10h)	This tag contains the read only VPD keywords for an add-in card.
Large resource type VPD-W Tag (11h)	This tag contains the read/write VPD keywords for an add-in card.
Small resource type End Tag (0Fh)	This tag identifies the end of VPD in the storage component.

6.28.2.2 Read-Only Fields

Table 6-22 VPD Read-Only Fields

Keyword	Name	Description
PN	Add-in Card Part Number	This keyword is provided as an extension to the Device ID (or Subsystem ID) in the Configuration Space header in Figure 6-36 .
EC	Engineering Change Level of the Add-in Card	The characters are alphanumeric and represent the engineering change level for this add-in card.
FG	Fabric Geography	Reserved for legacy use by [PICMG] specifications.
LC	Location	Reserved for legacy use by [PICMG] specifications.
MN	Manufacture ID	This keyword is provided as an extension to the Vendor ID (or Subsystem Vendor ID) in the Configuration Space header in Figure 6-36 . This allows vendors the flexibility to identify an additional level of detail pertaining to the sourcing of this device.
PG	PCI Geography	Reserved for legacy use by [PICMG] specifications.
SN	Serial Number	The characters are alphanumeric and represent the unique add-in card Serial Number.

公众号 · 芯片业者相承双

Keyword	Name	Description
TR	Thermal Reporting	This keyword provides a standard interface for reporting four fields: ***AFI Level***, ***MaxTherm***, ***DTherm***, and ***MaxAmbient***. The data area for this field is four bytes long. This data is encoded as a 4-byte binary value in little endian order (byte 0 contains bits 7:0). This value contains the four fields as follows: AFI Level bits [3:0], MaxTherm bits [7:4], DTherm bits [11:8], and MaxAmbient bits [19:12] are placed in bits 19:0. Bits 31:20 are Reserved and must be set to 000h. Field description is provided within the [ECN-CEM-THERMAL] to the [CEM-3.0]. This keyword is intended to be used only in designs based on that form factor specification. Note that due to the character nature of the VPD encoding mechanism, this binary value is permitted to start on any byte boundary within the VPD.
Vx	Vendor Specific	This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or A through Z.
CP	Extended Capability	This field allows a new capability to be identified in the VPD area. Since dynamic control/status cannot be placed in VPD, the data for this field identifies where, in the device's memory or I/O address space, the control/status registers for the capability can be found. Location of the control/status registers is identified by providing the index (a value between 0 and 5) of the Base Address register that defines the address range that contains the registers, and the offset within that Base Address register range where the control/status registers reside. The data area for this field is four bytes long. The first byte contains the ID of the extended capability. The second byte contains the index (zero based) of the Base Address register used. The next two bytes contain the offset (in little-endian order) within that address range where the control/status registers defined for that capability reside.
RV	Checksum and Reserved	The first byte of this item is a checksum byte. The checksum is correct if the sum of all bytes in VPD (from VPD address 0 up to and including this byte) is zero. The remainder of this item is reserved space (as needed) to identify the last byte of read-only space. The read-write area does not have a checksum. This field is required.

6.28.2.3 Read/Write Fields

Table 6-23 VPD Read/Write Fields

Keyword	Name	Description
Vx	Vendor Specific	This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or A through Z.
Yx	System Specific	This is a system specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 or B through Z.
YA	Asset Tag Identifier	This is a system specific item and the characters are alphanumeric. This keyword contains the system asset identifier provided by the system owner.
RW	Remaining Read/ Write Area	This descriptor is used to identify the unused portion of the read/write space. The product vendor initializes this parameter based on the size of the read/write space or the space remaining following the Vx VPD items. One or more of the Vx, Yx, and RW items are required.

6.28.2.4 VPD Example

以下是典型 VPD 的示例。

Table 6-24 VPD Example

Offset	Item Value
0	Large Resource Type ID String Tag (02h) 82h “Product Name”
1	Length 0021h
3	Data “ABCD Super-Fast Widget Controller”
36	Large Resource Type VPD-R Tag (10h) 90h
37	Length 0059h
39	VPD Keyword “PN”
41	Length 08h
42	Data “6181682A”
50	VPD Keyword “EC”
52	Length 0Ah
53	Data “4950262536”
63	VPD Keyword “SN”
65	Length 08h
66	Data “00000194”
74	VPD Keyword “MN”
76	Length 04h
77	Data “1037”
81	VPD Keyword “RV”
83	Length 2Ch
84	Data Checksum
85	Data Reserved (00h)
128	Large Resource Type VPD-W Tag (11h) 91h
129	Length 007Ch
131	VPD Keyword “V1”
133	Length 05h
134	Data “65A01”
139	VPD Keyword “Y1”
141	Length 0Dh
142	Data “Error Code 26”
155	VPD Keyword “RW”

公众号：芯片业者相承双

Offset	Item Value
157	Length 61h
158	Data Reserved (00h)
255	Small Resource Type End Tag (0Fh) 78h

6.29 Native PCIe Enclosure Management（NPEM）

NPEM 是可选实现的 PCIe Extended Capability，可提供用于附件管理的机制。此机制旨在为包含 PCIe SSD 的附件提供管理，该管理与存储生态系统中已建立的功能一致。

本节定义了该机制的体系结构方面。Section 7.9.20 中定义了 NPEM Extended Capability。

附件是包含一个或多个 PCIe SSD 的任何平台，盒子，机架或盒子集合。NPEM Capability 可为 PCIe SSD 提供与存储相关的附件控制（例如，LED 状态控制）。NPEM Capability 可以驻留在下游端口或 Endpoint（即 PCIe SSD）中。Figure 6-37 显示了具有单个下游端口的示例配置，该下游端口包含 NPEM Capability 和供应商特定的逻辑来控制关联的 LED。

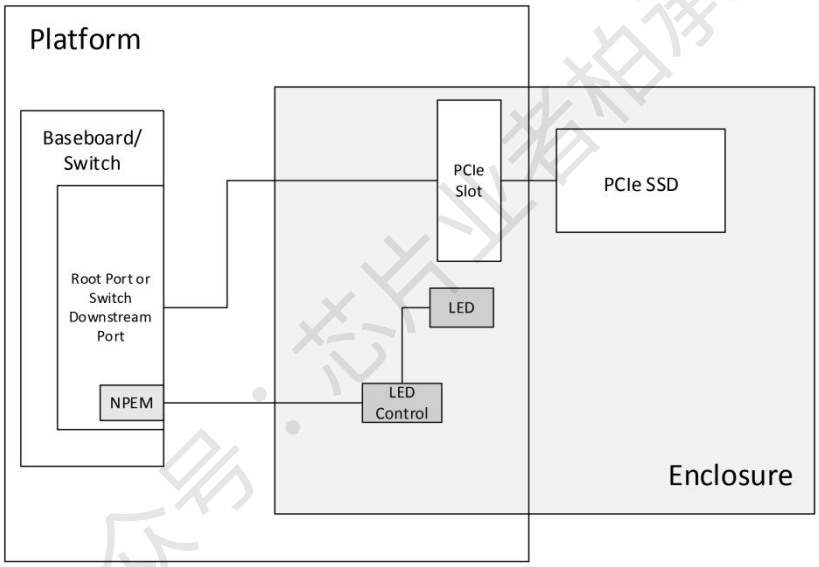

Figure 6-37: Example NPEM Configuration using a Downstream Port

Figure 6-38 显示了一个示例配置，其中 NPEM Capability 位于上游端口（在本例中为 SSD 功能）。

公众号·芯片业者相承双

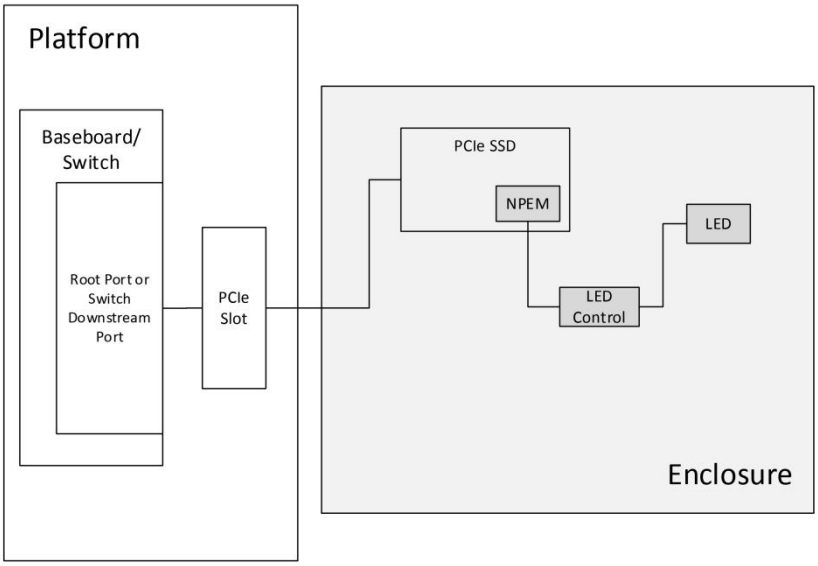

Figure 6-38: Example NPEM Configuration using an Upstream Port

软件通过写入 NPEM Control 寄存器来发出 NPEM Command，以更改与 SSD 相关的指示。NPEM Command 是对 NPEM Control 寄存器的单次写入，可更改零个或多个位的状态。 NPEM 使用命令完成机制通知软件已成功完成。Figure 6-39 显示了整个流程。

该规范定义了 NPEM Capability 提供的软件接口。Port 到附件的接口、附件、附件到 LED 的接口、每个 SSD 的 LED 数量以及相关的 LED 闪烁模式均不在本规范范围内。

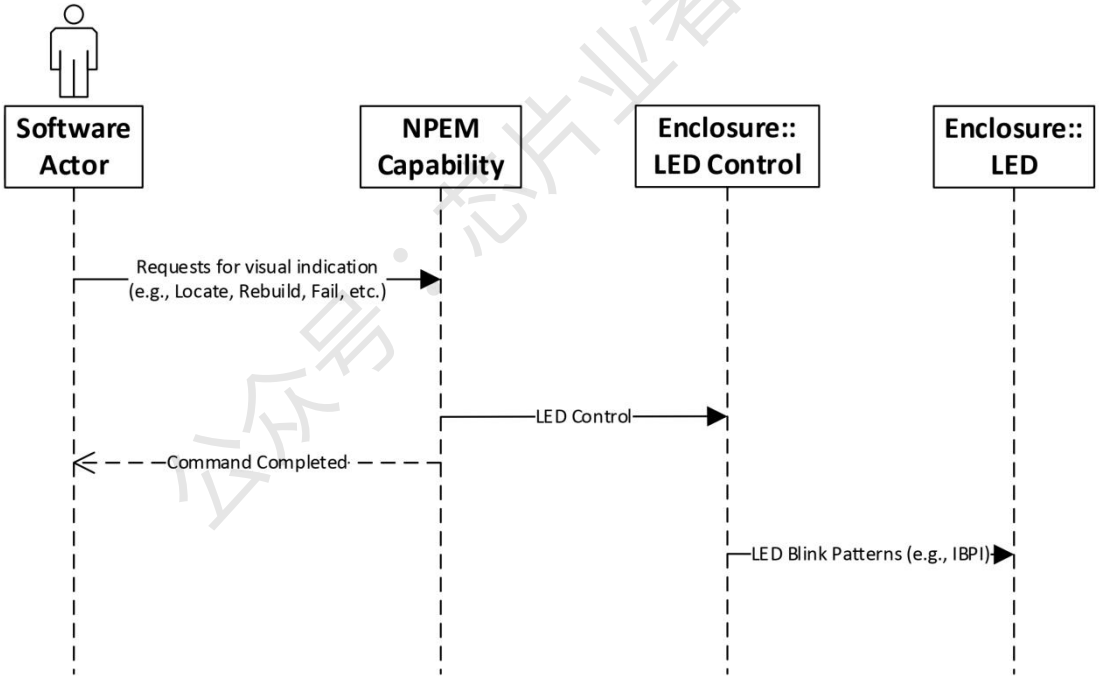

Figure 6-39: NPEM Command Flow

NPEM 为系统软件提供了一种机制，可以通过独立于 PCIe 链路本身的 NPEM 复位机制向附件内的 LED 控制元件发出复位。NPEM 命令完成机制也适用于 NPEM 复位。

存储系统管理员或软件通过 NPEM Capability 控制各种设备状态的指示。

NPEM States

Table 6-25 显示了 NPEM 状态的示例以及某些附近可能分配给已架构 NPEM 状态的可能含义。

公众号·芯片业者相关改

Table 6-25 NPEM States

NPEM State	Actor	Definition
OK	System Admin or Storage Software	OK state may mean the drive is functioning normally. This state may implicitly mean that an SSD is present, powered on, and working normally as seen by the software. A more granular indication of drive not physically present or present but not powered up are both outside the scope of this specification.
Locate	System Admin	Locate state may mean the specific drive is being identified by an admin.
Fail	Storage Software	Fail state may mean the drive is not functioning properly
Rebuild	Storage Software	Rebuild state may mean this drive is part of a multi-drive storage volume/array that is rebuilding or reconstructing data from redundancy on to this specific drive.
PFA	Storage Software	PFA stands for Predicted Failure Analysis. This state may mean the drive is still functioning normally but predicted to fail soon.
Hot Spare	Storage Software	Hot Spare state may mean this drive is marked to be automatically used as a replacement for a failed drive and contents of the failed drive may be rebuilt on this drive.
In A Critical Array	Storage Software	In A Critical Array state may mean the drive is part of a multi-drive storage array and that array is degraded.
In A Failed Array	Storage Software	NPEM In A Failed Array state may mean the drive is part of a multi-drive storage array and that array is failed.
Invalid Device Type	Storage Software	Invalid Device Type state may mean the drive is not the right type for the connector (e.g., An enclosure supports SAS and NVMe drives and this drive state indicates that a SAS drive is plugged into an NVMe slot).
Disabled	Storage Software	Disabled state may mean the drive in this slot is disabled. A removal of this drive from the slot may be safe. The power from this slot may be removed.

Software Polling of NPEM Command Completed

不同的 NPEM 实现可能需要很长时间才能完成 NPEM Command，从瞬时到数十毫秒。为避免或最小化软件轮询开销，建议软件实现以下一项或两项优化。

与软件发出写命令，然后立即轮询是否完成不同，建议软件反转此顺序。准备发出写新命令时，软件首先轮询上一个命令的完成情况，然后再发出写的新命令。这样可以实现重叠操作，通常完全隐藏硬件执行 NPEM 命令所需的时间。要启用此轮询模型，软件必须在复位后通过写入 no op 命令初始化硬件，以便硬件生成第一个 NPEM 命令完成。

对于软件轮询该比特位时 NPEM 命令尚未完成的情况，建议软件不要在轮询时连续“旋转”，而是在中断下以较低的速率轮询；例如以 10 毫秒的间隔。

6.30 Conventional PCI Advanced Features Operation

对于集成到 RC 中的常规 PCI 设备，Conventional PCI Advanced Features Capability（AF）提供了使用最初为 PCI Express 开发的高级功能的机制。

- Function Level Reset（INITIATE_FLR）机制使软件能够以 Function-level 的粒度静默并复位硬件。
- Transactions Pending（TP）机制用于指示该 Function 已发出一个或多个尚未完成的 Non-Posted Transaction（包括延迟事务）。

此处定义的 FLR 和 TP 机制仅适用于集成到 RC 中的常规 PCI 设备，其中实现允许给定常规 PCI Function 完成 Non-Posted Transaction，

即使其 Command 寄存器中的 Bus Master Enable 的值为 0b 。不满足此要求的实现不得实现 FLR 和 TP 机制。

FLR 按以下方式修改 Function 状态：

必须将 Function 的寄存器和特定于 Function 的状态机设置为本文档中指定的初始化值，但以下比特位除外，不得修改：Fast Back-to-Back Transactions Enable、Cache Line Size、Latency Timer、Interrupt Line、PME_En 和 PME_Status。

请注意，将清除使 Function 能够启用总线事务的控件，包括 Command 寄存器中的 Bus Master Enable， MSI Capability Structure 中的 MSI Enable 等，从而有效地使 Function 变为静默。

启动 FLR 后，Function 必须在 100 毫秒内完成 FLR。如果在 Transactions Pending 为 1b 时软件启动了 FLR，则软件必须等到有足够的时间来合理确定所有 outstanding transaction 已完成之前，才能初始化 Function。FLR 完成后，必须把 Transactions Pending 清 0。

FLR 修改了本规范未描述的 Function 状态（除了本规范所描述的状态之外），因此必须使用特定于 Function 的知识来应用以下条件来评估 Function 对 FLR 的响应：

- 该 Function 不得在该 Function 控制的任何外部接口上显示带有活动主机的初始化适配器。终止外部接口上的活动所需的步骤超出了本规范的范围。
    - 例如，网络适配器不得响应要求主机系统进行适配器初始化或与活动主机系统进行交互的查询，但可以执行设计为执行的操作而无需主机初始化或进行交互。如果网络适配器包括在同一外部网络接口上运行的多个 Function，则此规则仅影响与由 FLR 复位的特定 Function 相关的那些方面。
- Function 不得在其内部保留可能包含与该 Function 的任何先前使用相关的秘密信息的软件可读状态。 分配给 Function 的主机内存不得由 Function 修改。
    - 例如，具有内部存储器可被主机软件直接或间接读取的 Function 必须清除或随机分配该存储器。
- Function 必须返回一种状态，以使 Function 的 PCI 接口的正常配置将使该 Function 可由通常与该 Function 相关联的驱动程序使用。

启动 FLR 时，目标 Function 必须具有以下行为：

- Function 必须正常完成启动 FLR 操作的配置写操作，然后启动 FLR。
- 在处于 FLR 过程中时：
    - Function 不得响应总线上的任何请求（即以 Function 为目标的请求将中止）。

Transactions Pending（TP）字段指示该 Function 已发出一个或多个尚未完成的 Non-Posted Transaction。软件可以使用该字段来确定 Function 何时变为静默。

Avoiding Issues with Pending Transactions

FLR 导致 Function 失去对任何挂起（outstanding non-posted）事务的跟踪。根据 RC-integrated PCI Function 的具体实现，如果软件在存在挂起事务的情况下发出 FLR，则有可能发生数据损坏，如"Avoiding Data Corruption From Stale Completions"中所述。

为避免 RC 实现中可能出现 Stale Completion 或 Discard Timer 的潜在问题，建议软件使用类似于以下算法：

1. 执行 FLR 的软件与可能直接访问 Function 的其他软件进行同步，并确保在此算法期间不会发生此类访问。
2. 软件会清除整个 Command 寄存器，从而使 Function 无法控制任何新事务。
3. 软件会轮询 AF Status 寄存器中的 Transactions Pending 字段，直到清除该数据或足够长的时间以使所有剩余未完成的事务都永远不会完成的合理确定性为止。在许多系统上，Transactions Pending 字段通常会在几毫秒内清除，因此软件可能会选择

公众号·芯片业者怕承改

在此初始期间使用紧密的软件循环进行轮询。在极少数情况下，此时尚未清除 Transactions Pending 字段，软件将需要轮询更长的系统特定时间段（可能是几秒钟），因此软件可能选择使用基于计时器的中断轮询机制来执行此轮询。

4. 软件发起 FLR。
5. 软件等待 100 ms。
6. 软件会重新配置 Function 并使其正常运行。

公众号·芯片业者柏承双

7 Software Initialization and Configuration

PCI Express 提供两种配置空间（Configuration Space）访问机制：

- PCI 兼容配置访问机制（CAM， PCI compatible Configuration Access Mechanism）。
- PCI Express 增强配置访问机制（ECAM，PCI Express Enhanced Configuration Access Mechanism）。

PCI 兼容机制支持与 PCI 3.0 或更高版本的操作系统及其对应的总线枚举和配置软件 100％二进制兼容。

PCI Express 增强配置访问机制增加了可用配置空间的大小并优化了访问机制。

X86 的 CPU 可以直接访问 memory 空间和 IO 空间，但是不能直接访问 PCIe 配置空间（原因很简单，X86 的 CPU 只有 memory 指令和 IO 指令，没有配置指令）。因此，需要把 PCIe 配置空间映射到 memory 空间或者 IO 空间（一般不推荐映射到 IO 空间）。或者说 CPU 访问 PCIe 配置空间需要一个翻译官（RC）。这个翻译官把 CPU 的 memory 访问或者 IO 访问转换成 PCIe 域的配置请求。

在 X86 系统中，IO 方式的翻译官就是 CONFIG_DATA 和 CONFIG_ADDRESS，这个两个位于 IO 空间的端口。这就是所谓的 PCI 的 CAM 方式，只能访问 PCI 兼容的配置空间（前面 256Byte）。

在 X86 系统中，Memroy 方式的翻译官就是 MCFG（Memory mapped configuration space base address description table，可以通过 cat /proc/iomem 查看 PCI MMCONFIG 得到在 memory 空间的映射），其实就是 Bus 0、 Dev 0 和 Function 0 的基地址。这就是所谓的 PCIe 的 ECAM 方式，可以访问 PCIe 全部 4K 配置空间。

PCIe 总线最多支持 256 个子总线，每个子总线最多支持 32 个设备， 每个设备最多支持 8 个功能， 每个配置空间是 4KB，所以需要映射 256 buses × 32 devices × 8 functions × 4 KB = 256 MB 空间的大小。

7.1 Configuration Topology

为了保持与 PCI 软件配置机制的兼容性，所有类型的 PCI Express 设备都要实现 PCI 兼容的配置空间。每个 PCI Express 链路都可以映射为一个逻辑 PCI-PCI Bridge，并作为该桥的二级总线映射到配置空间中。根端口是一种 PCI-PCI Bridge 结构，它从 PCI ExpressRC（Figure 7-1）中创建 PCI Express 链路。

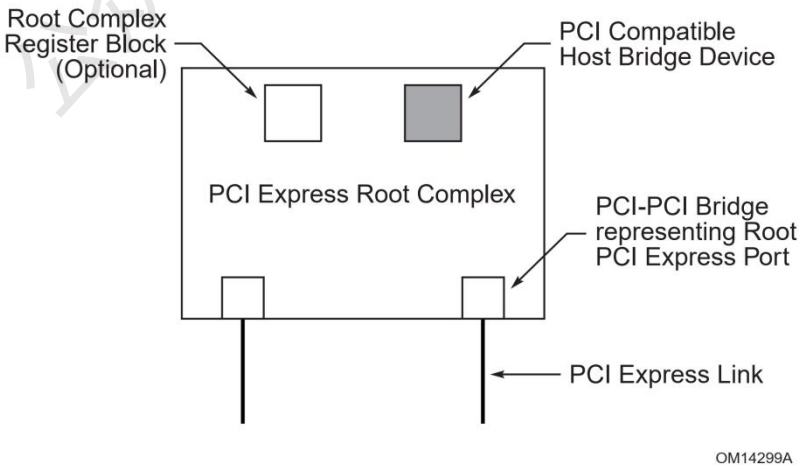

Figure 7-1: PCI Express Root Complex Device Mapping

一个 PCI Express Switch 由将 PCI Express 链路连接到内部逻辑 PCI 总线的多个 PCI-PCI Bridge 结构表示（Figure 7-2）。Switch 的 Upstream Port 是 PCI-PCI Bridge；该桥的二级总线代表 Switch 的内部路由逻辑。Switch 的 Downstream Port 是从内部总线桥接到

公众号：芯片业者相承双

表示来自 PCI Expres Switch 的下游 PCI Express 链路的总线的 PCI-PCI Bridge。内部总线上只能出现代表 Switch 的 Downstream Port 的 PCI-PCI Bridge。Endpoint（由 Type 0 Configuration Space header 表示）不允许出现在内部总线上。

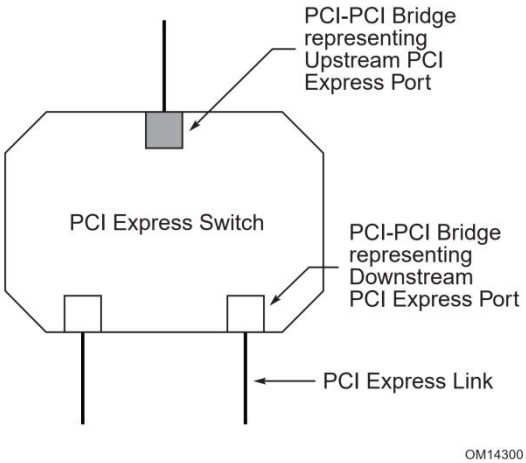

Figure 7-2: PCI Express Switch Device Mapping [91]

PCI Express Endpoint 作为设备中的单个 Function 映射到配置空间，该设备可能包含多个 Function 或仅包含一个 Function。PCI Express Endpoint 和 Legacy Endpoint 必须出现在由 RC 发起的一个层次结构域中，这意味着它们必须出现在以根端口为头的树中的配置空间中。RC Integrated Endpoint 和 Root Complex Event Collector 未出现在该层次结构域中。这些作为根端口的对等方出现在配置空间中。

除非另有说明，否则设备的配置空间定义中的要求适用于 single-Function 设备以及 multi-Function 设备的每个 Function。

7.2 PCI Express Configuration Mechanisms

PCI Local Bus Specification 定义了 256 byte 的配置空间（FFh，64 个 DW），PCI Express 把配置空间扩展到 4096 byte（FFFh，1024 个 DW）。PCI Express 配置空间分为 PCI 3.0 兼容区域和 PCI Express 扩展配置空间，PCI 3.0 兼容区域由前 256 个字节组成，而 PCI Express 扩展配置空间由其余部分构成（see Figure 7-3）。PCI 3.0 兼容的配置空间即可以使用 *PCI Local Bus Specification* 中定义的机制访问，也可以使用 ECAM 来访问。PCI Express 扩展配置空间只能使用 ECAM 来访问。

ECAM 的操作独立于 *PCI Local Bus Specification* 中定义的用于生成配置事务的机制。两者之间没有隐含的顺序。

公众号：芯片业者相会议

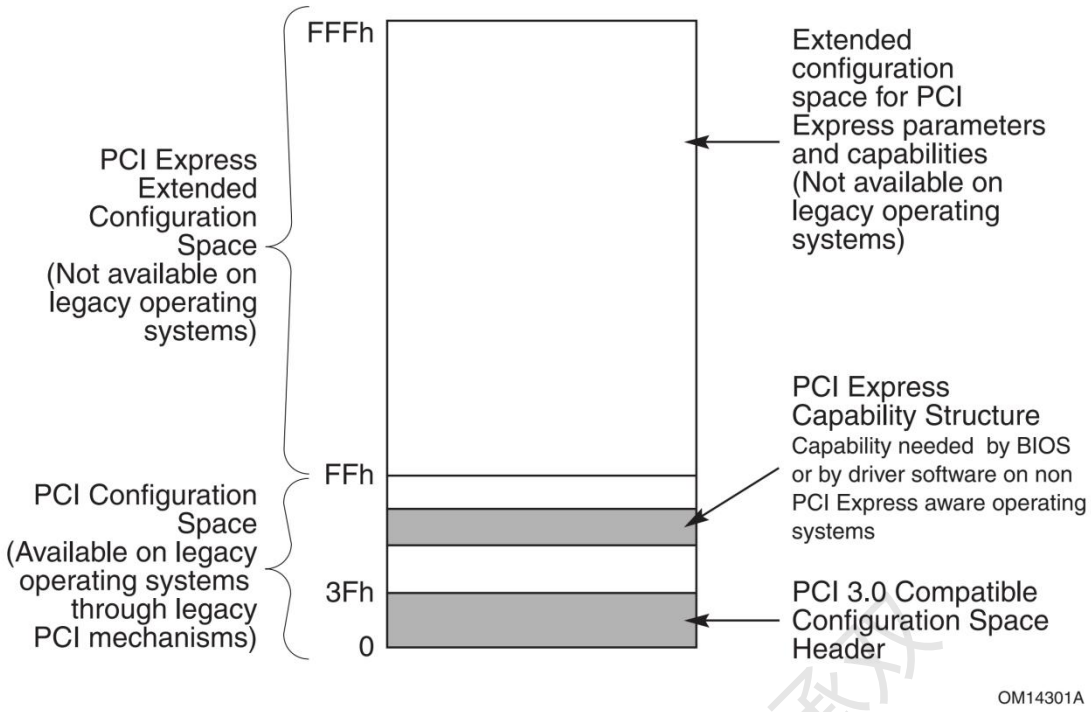

Figure 7-3: PCI Express Configuration Space Layout

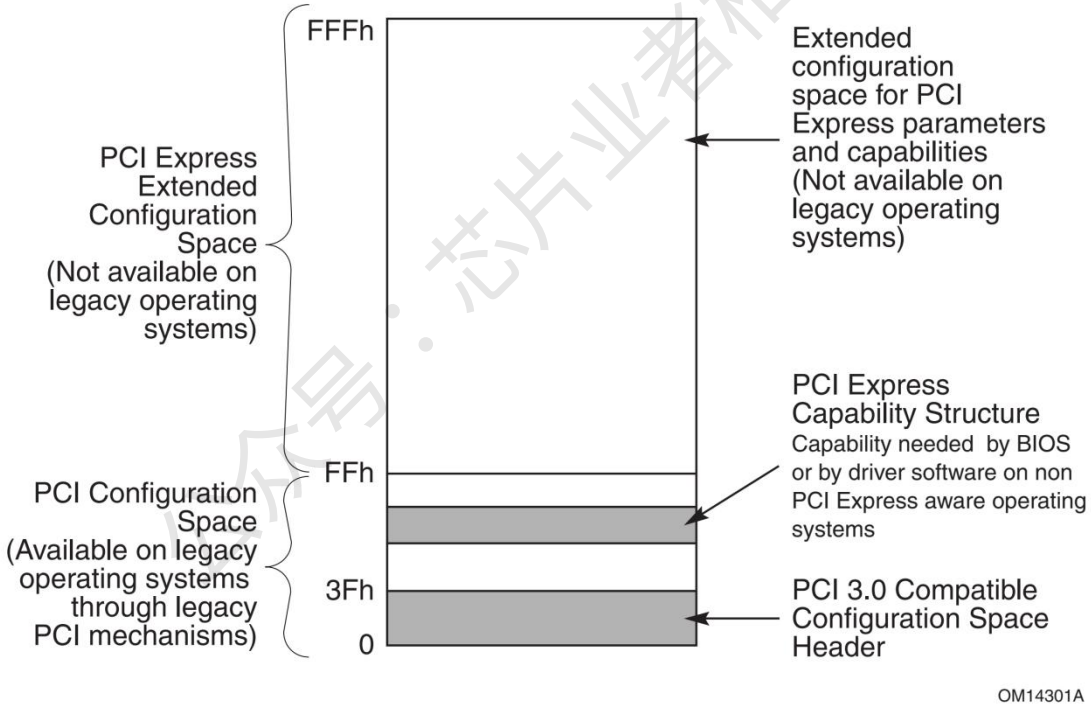

Figure 7-3: PCI Express Configuration Space Layout

这张图的理解：

1. PCI 规范本身定义的配置空间范围为 0~FFh，PCI Express 规范扩大了这个范围为 0~FFFh。
2. PCI Express 规范在 PCI 配置空间中实现了 PCI Express Capability Structure 用于实现最基本的 PCIe 设备、链路和插槽的管理，所有 PCIe 设备都要实现这个 Structure，其 Capability_ID 为 10h。枚举软件就是通过这个值知道设备是一个 PCIe 设备的。
3. PCI Power Management Capability Structure 也是必须实现的 Structure，其 Capability_ID 为 01h。

公号·芯片业者相承双

7.2.1 PCI 3.0 Compatible Configuration Mechanism

兼容 PCI 3.0 的 PCI Express 配置机制支持 *PCI Local Bus Specification* 中定义的 PCI 配置空间编程模型。通过遵守该模型，带 PCI Express 接口的系统仍符合常规的 PCI 总线枚举和配置软件。

与 PCI 3.0 设备的 Function 相同，需要 PCI Express 设备 Function 为软件驱动的初始化和配置提供配置空间。 除了本章中描述的差异外，PCI Express 配置空间头的组织方式与 PCI Local Bus Specification（Section 6.1）中定义的格式和行为相对应。

兼容 PCI 3.0 的配置访问机制使用与 ECAM 有相同的请求格式。对于与 PCI 兼容的配置请求，扩展寄存器地址字段必须全为零。

7.2.2 PCI Express Enhanced Configuration Access Mechanism (ECAM)

对于与 PC 兼容的系统，或没有实现处理器架构特定的固件接口标准的系统，就需要使用 ECAM。

对于实现了处理器架构特定的固件接口标准的系统，操作系统使用标准固件接口，并且不需要本节中定义的硬件访问机制。例如，对于符合 *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64), Version 2.1* 的系统，操作系统使用 SAL 固件服务程序来访问配置空间。

在所有系统中，都鼓励设备驱动程序使用操作系统提供的应用程序编程接口（API）来访问其设备的配置空间，而不是直接使用硬件机制。

ECAM 利用内存映射地址空间来访问设备配置寄存器。在这种情况下，存储器地址确定了要访问的配置寄存器，并且存储器数据更新（用于写入）或返回（用于读取）寻址寄存器的内容。Table 7-1 中定义了从内存地址空间到 PCI Express 配置空间地址的映射。

映射到配置空间的内存地址范围的大小和基址由主机桥和固件的设计确定。它们由固件以特定于实现的方式报告给操作系统。范围的大小取决于主机桥映射到配置地址中 Bus Number 字段的位数。在 Table 7-1 中，此位数表示为 n，其中 $1 \le n \le 8$。将 n 个内存地址位映射到 Bus Number 字段的主机桥支持从 0 到 $2^n - 1$（含）的 Bus Number。该范围的基址与 $2^{(n+20)}$ 字节的存储器地址边界对齐。Bus Number 字段中未从内存地址位映射的任何位必须清零。

例如，如果系统将 3 bit 的内存地址位映射到 Bus Number 字段，则以下条件均成立：

- n = 3 。
- 地址位 A [63:23]用于基址，该基址与 2^{23} 字节（8 MB）的边界对齐。
- 地址位 A [22:20]映射到 Bus Number 字段中的 bit[2:0]。
- Bus Number 字段的 bit[7:3]清 0。
- 该系统能够处理 0 到 7 之间的 Bus Number（包括 0 和 7）。

至少有 1 bit 内存地址位（n = 1）映射到 Bus Number 字段。鼓励系统根据需要将其他内存地址位映射到 Bus Number 字段，以支持更多数量的总线。鼓励支持 4 GB 以上内存地址的系统将 8-bit 内存地址（n = 8）映射到 Bus Number 字段。请注意，在包含多个主机桥且分配给每个主机桥的总线号范围不同的系统中，系统的最高总线号受分配了最高总线号的主机桥映射的位数限制。在这样的系统中，在大多数情况下，分配给特定主机桥的最高总线号将大于分配给该主机桥的总线数。换句话说，对于每个主机桥，映射到 Bus Number 字段的位数必须足够大，以使分配给每个特定桥的最高总线号必须小于或等于该桥的 $2^n - 1$。

在某些处理器体系结构中，可能会生成无法在单个配置请求中表示的内存访问，例如由于跨越 DW 对齐边界或由于使用了锁定访问而无法表达。不需要 RC 将此类访问转换为配置请求。

公众号·志存业者相承双

Table 7-1: Enhanced Configuration Address Mapping

Memory Address[94]	PCI Express Configuration Space
A[(20 + *n* – 1):20]	Bus Number 1 ≤ *n* ≤ 8
A[19:15]	Device Number
A[14:12]	Function Number
A[11:8]	Extended Register Number
A[7:2]	Register Number
A[1:0]	Along with size of the access, used to generate Byte Enables

Note：对于针对 ARI 设备中扩展功能的请求，A [19:12]表示 8-bit Function Number，它代替了上面的 5-bit Device Number 和 3-bit Function Number 字段。

系统硬件必须为系统软件提供一种方法，以确保在继续执行系统软件之前，由完成者完成使用 ECAM 的写事务。

Ordering Considerations for the Enhanced Configuration Access Mechanism

ECAM 将来自主机 CPU 的内存事务转换为 PCI Express 结构上的配置请求。 这种转换可能会给软件带来排序问题，因为对内存地址的写入通常是 Posted 事务，而对配置空间的写入在 PCI Express 拓扑上是一个 Non-Posted 事务。

通常，软件不知道完成者何时完成 Posted 事务。而在某些情况下，软件必须知道完成者完成了 Posted 事务，软件通常使用的一种技术是读取刚写入的位置。对于始终遵循 PCI 排序规则的系统，只有完成 Posted Write 操作后，读取事务才会完成。 但是，由于 PCI 排序规则允许相对于彼此重新排序未发布的写入和读取事务，因此 CPU 必须等待 PCI Express 上的 Non-Posted Write 完成，以确保完成该事务。

例如，软件可能希望通过使用 ECAM 写入设备来配置设备某 Function 的 Base Address 寄存器，然后读取该 Base Address 寄存器描述的存储器映射范围内的位置。如果软件在完成 ECAM 写入之前发出内存映射的读取，则可能会重新排序该内存映射的读，并在配置写入请求之前到达设备，从而导致不可预测的结果。

为避免此问题，处理器和主机桥的实现必须确保存在一种方法，供软件确定何时使用 ECAM 完成写入程序。

此方法可能只是处理器本身将专用于映射 ECAM 访问的内存范围识别为唯一的，并以与对待其他访问在 PCI Express 矩阵上生成 Non-Posted Write 的访问相同的方式来对待对该范围的访问，即从处理者的角度来看，该事务是 Non-Posted 事务。另一种机制是让主机桥（而不是处理器）识别内存映射的配置空间访问，并在 PCI Express 架构上完成 Non-Posted 的配置事务之前，不向处理器指示此写入已被接受 。第三种选择是让处理器和主机桥将内存映射的写操作发布到 ECAM，并让主机桥提供一个单独的寄存器，软件可以读取该寄存器，以确定何时在 PCI Express 架构上完成配置写请求。其他替代方案也是可能的。

Generating Configuration Requests

由于 RC 不需要支持跨 DW 边界或使用锁定语义的配置请求的生成，因此在使用内存映射的 ECAM 时，软件应注意不要引起此类访问的生成，除非知道正在使用 RC 的实现支持确实支持该转换。

7.2.2.1 Host Bridge Requirements

对于那些实现了 ECAM 的系统，要求 PCI Express Host Bridge 将内存映射的 PCI Express 配置空间访问从主机处理器转换为 PCI Express 配置事务。保留主机桥 PCI 的 Class Code 的使用是为了向后兼容。主机桥配置空间对于标准 PCI Express 软件是不透明的，并且可以以与 PCI Host Bridge Type 0 Configuration Space 兼容的特定于实现的方式来实现。不需要 PCI Express Host Bridge 通过 Root Complex Event Collector 发出错误信号。对于 PCI Express Host Bridge，此支持是可选的。

7.2.2.2 PCI Express Device Requirements

设备必须支持额外的 4bit 来解码配置寄存器访问，即它们必须能解码配置请求头的 Extended Register Address[3:0]字段。

7.2.3 RC Register Block

Root Port 或 RC Integrated Endpoint 可以与称为映射 RC Register Block (RCRB)的内存映射寄存器的可选 4096 字节块相关联。这些寄存器的使用方式类似于配置空间，并且可以包括 PCI Express Extended Capabilities 以及适用于 RC 的其他特定于实现的寄存器。RCRB 的结构在 Section 7.9.2 中描述。

多个根端口或内部设备允许与同一 RCRB 关联。 RCRB 内存映射的寄存器不得与内存映射的配置空间或内存空间位于同一地址空间。

不需要 RC 实现来支持对跨越 DWORD 对齐边界的 RCRB 的访问或使用锁定语义的访问。

Accessing RC Register Block

由于不需要 RC 实现来支持对跨越 DW 边界或使用锁定语义的 RCRB 的访问，因此软件应注意在访问 RCRB 时不要引起此类访问的产生，除非 RC 支持此访问。

7.3 Configuration Transaction Rules

7.4 Configuration Register Types

配置寄存器的各个字段可能拥有 Table 7-2 描述的属性之一。除 RC 和系统集成设备外，所有 PCI Express 组件都将寄存器字段初始化为指定的默认值。 RC 和系统集成的设备会根据固件为特定系统实施所需的初始化寄存器字段。

Table 7-2: Register and Register Bit-Field Types

Register Attribute	Description
HWInit	**Hardware Initialized** —— 这种寄存器字段要通过固件或硬件机制初始化，这些只读在初始化后是只读的，并且只能通过基本复位（参见 Section 6.6.1）进行复位。HwInit 寄存器字段不会被 FLR 修改。
RO	**Read-only** —— 软件只能读不能修改。
RW	**Read-Write** —— 软件即可度也可写。
RW1C	**Write-1-to-clear status** —— 软件向该字段写 1 会把该字段清 0，写 0 无效。
ROS	**Sticky - Read-only** —— 这种寄存器字段是只读的，不能由软件更改。通过 Hot Reset 或 FLR 既不初始化也不修改该字段。 这种寄存器字段使用辅助电源（通过 Aux Power PM Enable 或 PME_En），消耗 Aux 功率的设备必须保留粘性寄存器值。在这些情况下，寄存器既不会通过热复位，热复位或冷复位进行初始化或修改（请参见 Section 6.6）。

公众号·志告业者柏承双

RWS	**Sticky - Read-Write** —— 寄存器字段是可读写的，并由软件置 1 或清除为所需状态。 通过 Hot Reset 或 FLR 既不初始化也不修改该字段。 这种寄存器字段使用辅助电源（通过 Aux Power PM Enable 或 PME_En），消耗 Aux 功率的设备必须保留粘性寄存器值。在这些情况下，寄存器既不会通过热复位，热复位或冷复位进行初始化或修改（请参见 Section 6.6）。
RW1CS	**Sticky - Write-1-to-clear status** —— 这些寄存器指示读取的一些状态，指示状态事件，该状态事件可通过写入 1b 清除。向 RW1CS 位写入 0b 无效。通过 Hot Reset 或 FLR 既不初始化也不修改该字段。 这种寄存器字段使用辅助电源（通过 Aux Power PM Enable 或 PME_En），消耗 Aux 功率的设备必须保留粘性寄存器值。在这些情况下，寄存器既不会通过热复位，热复位或冷复位进行初始化或修改（请参见 Section 6.6）。
RsvdP	**Reserved and Preserved** —— 为将来的 RW 实施保留。寄存器是只读的，读取时必须返回零。
RsvdZ	**Reserved and Zero** —— 保留用于将来的 RW1C 实现。寄存器是只读的，读取时必须返回零。软件必须使用 0b 写入该字段。

7.5 PCI and PCIe Capabilities Required by the Base Spec for all Ports

所有遵循 PCIe 规范的 Port 必须要实现本节描述的寄存器。这些是 PCIe 能正常工作的基础。

7.5.1 PCI-Compatible Configuration Registers

PCI Express Configuration Space 的头 256 个字节的来源是 PCI3.0 兼容区域。这个区域又叫 Function 的 PCI 3.0 Configuration Space。Legacy PCI 设备也可以通过 ECAM 进行访问，而无需对设备硬件或设备驱动程序进行任何修改。本节建立了 PCI 3.0 和 PCI Express 之间的映射，以描述 PCI 3.0 兼容寄存器的格式和行为。

本节中未描述的所有寄存器和字段都与 PCI Local Bus Specification 有相同的定义。配置空间的布局和各个配置寄存器的格式均遵循 PCI Local Bus Specification 中使用的 little-endian 的约定。

7.5.1.1 Type 0/1 Common Configuration Space

Figure 7-4 详细介绍了 PCI Express device Function 的 PCI 3.0 Type 0 和 Type 1 配置空间头的通用寄存器字段的分配。

公众号·芯片业者相承及

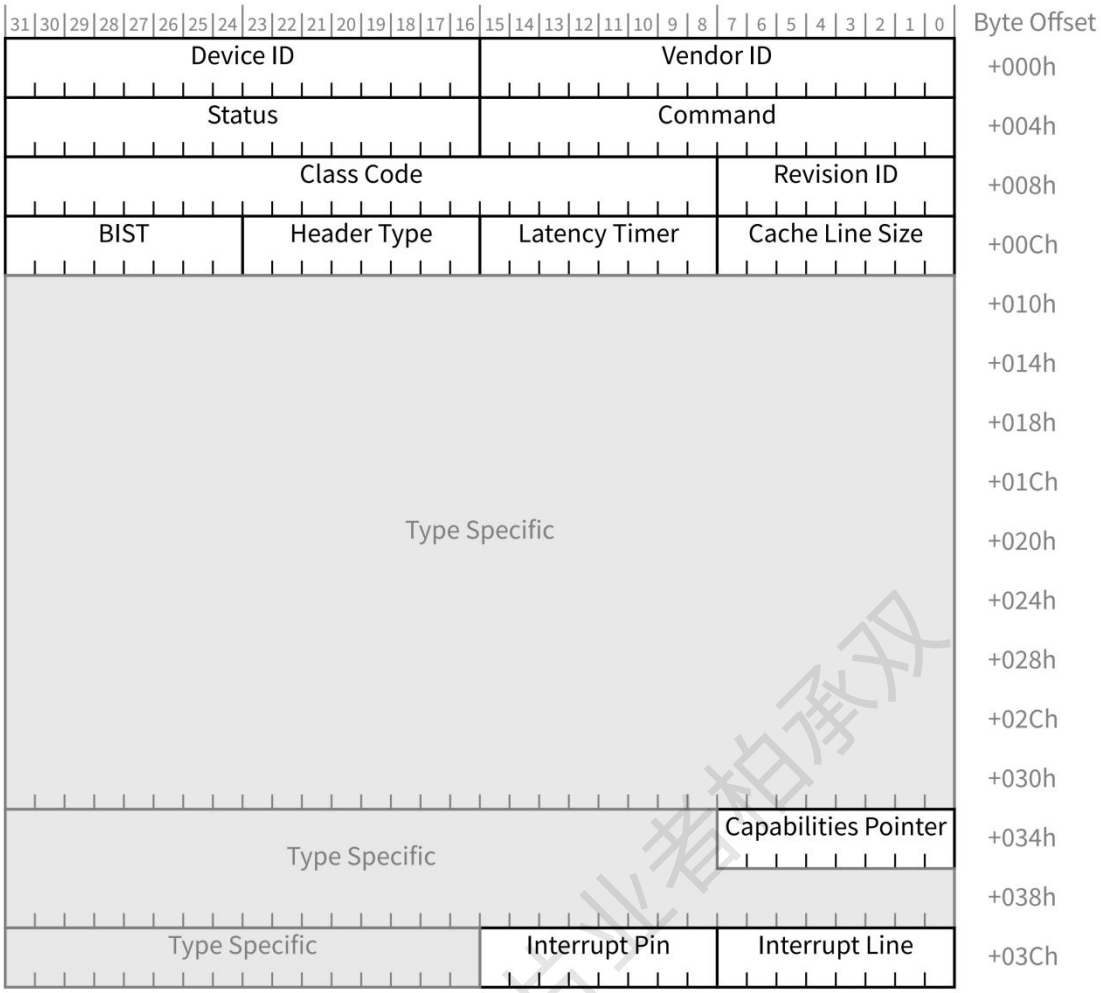

Figure 7-4 Common Configuration Space Header

这些寄存器是为 Type 0 和 Type 1 配置空间头定义的。这些寄存器的 PCI Express 的解释在本节中给出。

7.5.1.1.1 Vendor ID Register (Offset 00h)

Vendor ID 寄存器为 HwInit，并且该寄存器中的值用于标识 Function 的制造商。PCI-SIG 必须分配有效的供应商标识符，以确保唯一性。每个供应商必须至少具有一个 Vendor ID。建议软件读取 Vendor ID 寄存器以确定是否存在此 Function，其中 FFFFh 的值表示不存在该 Function。

7.5.1.1.2 Device ID Register (Offset 02h)

Device ID 寄存器为 HwInit，该寄存器中的值用于标识特定的 Function。Device ID 必须由供应商分配。Device ID 与 Vendor ID 和 Revision ID 一起用作软件确定应加载哪个驱动程序的一种机制。供应商必须确保所选的值不会导致使用不兼容的设备驱动程序。

7.5.1.1.3 Command Register (Offset 04h)

Table 7-3 定义了 Command 寄存器，寄存器的布局如 Figure 7-5 所示。Command 寄存器中的各个字段可能会实现，也可能不会实现，具体取决于 Function 支持的 Feature 集。对于 PCI Express to PCI/PCI-X Bridge，请参阅 *PCI Express to PCI/PCI-X Bridge Specification* 以了解此寄存器的要求。

公众号：芯片业者怕承双

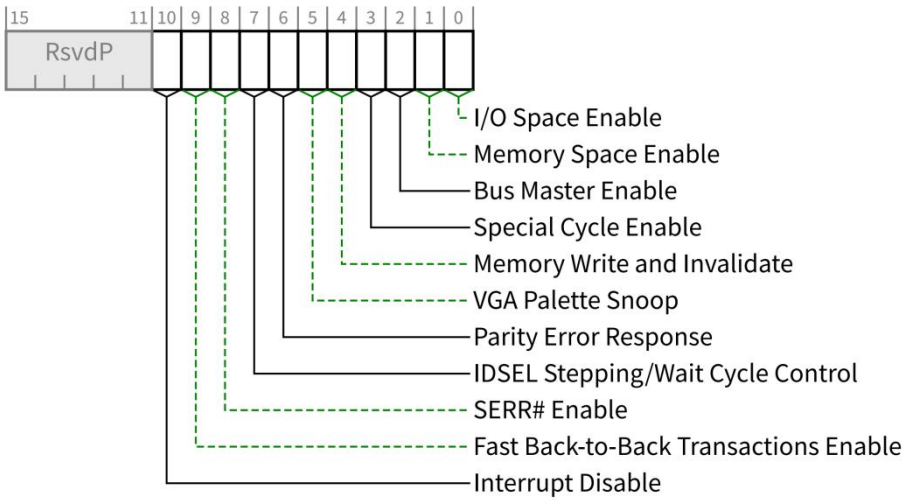

Figure 7-5 Command Register

Bit Location	Register Description	Attributes
0	**I/O Space Enable** - 控制 Function 对 I/O 空间访问的响应。为 0 时，所有接收到的 I/O 访问都被视为 Unsupported Request。当为 1 时，该 Function 才允许解码地址并进一步处理对 I/O 空间的访问。对于 Type 1 配置空间头的 Function，此位控制对在其 Primary Side 收到的 I/O 空间访问的响应。 默认值为 0b。 如果某个 Function 不支持 I/O 空间访问，则允许将该位硬连线为 0b。	RW
1	**Memory Space Enable** - 控制 Function 对 Memory 空间访问的响应。为 0 时，所有接收到的 Memory 访问都被视为 Unsupported Request。当为 1 时，该 Function 才允许解码地址并进一步处理对 Memory 空间的访问。对于 Type 1 配置空间头的 Function，此位控制对在其 Primary Side 收到的 Memory 空间访问的响应。 默认值为 0b。 如果某个 Function 不支持 Memory 空间访问，则允许将该位硬连线为 0b。	RW
2	**Bus Master Enable** - 控制 Function 发出 Memory 读写请求和 I/O 读写请求的能力，以及端口往上游方向转发 Memory 读写请求和 I/O 读写请求的能力。BME 对实现 Type 0 配置空间头的 Function： 当该位置 1，允许该 Function 发出 Memory 请求或 I/O 请求。 当该位置 0，不允许该 Function 发出 Memory 请求或 I/O 请求。 请注意，由于 MSI/MSI-X 中断消息是带内 Memory Write 操作，因此将该位设置为 0b 也会禁用 MSI/MSI-X 中断消息。 除 Memory 请求或 I/O 请求外的其他请求均不受此位的控制（配置访问可以正常进行、消息也可以正常进行）。 默认值为 0b。	RW

如果某个 Function 不支持发出 Memory 请求或 I/O 请求，则允许将该位硬连线为 0b。

对实现 Type 1 配置空间头的 Function：

该位控制端口在上行方向上对 Memory 请求或 I/O 请求的转发。当此位为 0b 时，必须将根端口或 Switch 端口的下游侧收到的 Memory 请求或 I/O 请求作为 Unsupported Request 处理，其中对于 Non-Posted Request，必须返回具有 UR 完成状态的 Completion。该位不影响 Completion 在上游或下游方向上的转发。

除了 Memory 请求或 I/O 请求之外，其他请求的转发均不受此位的控制。

默认值为 0b。

3	**Special Cycle Enable** - 该位在 PCI 规范中进行了描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
4	**Memory Write and Invalidate** - 该位在 PCI 和 PCI-to-PCI-Bridge 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。 对于 PCI Express to PCI/PCI-X Bridge，请参阅 PCIe-to-PCI-PCI-X-Bridge 规范了解此寄存器的要求。	RO
5	**VGA Palette Snoop** - 该位在 PCI 和 PCI-to-PCI-Bridge 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
6	**Parity Error Response** - see Section 7.5.1.1.14。 该位控制 Status Register 中的 Master Data Parity Error 字段关于中毒 TLP 的记录。与 Root Complex Event Collector 无关的 RCiEP 允许将该位硬连线为 0b。 默认值为 0b。	RW
7	**IDSEL Stepping/Wait Cycle Control** - 该位在 PCI 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
8	**SERR# Enable** - see Section 7.5.1.1.14。 为 1 时，该位使能向上游报告此 Function 检测到的非致命和致命错误。请注意，如果通过该位或通过 Device Control Register（see Section 7.5.3.4）中的 PCI Express 特定字段使能了错误报告功能，就会报告错误。 另外，对于具有 Type 1 配置空间头的 Function，此位控制由二级接口转发到主接口的 ERR_NONFATAL 和 ERR_FATAL 错误消息的传送。该位不影响转发的 ERR_COR 消息的传输。 与 Root Complex Event Collector 无关的 RCiEP 允许将该位硬连线为 0b。 默认值为 0b。	RW
9	**Fast Back-to-Back Transactions Enable** - 该位在 PCI 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
10	**Interrupt Disable** - 控制 Function 生成 INTx 中断的能力。为 1 时，将阻止 Function 发出 INTx 中断。	RW

公众号·芯片业者相承双

当该位置 1 时，必须 deasserted 掉 Function 已经 asserted 的任何 INTx 中断。

如 Section 2.2.8.1 所述，INTx 中断使用虚拟线，当该位置 1 时，如果发出 asserted，则必须使用适当的 Deassert_INTx 消息将其置为无效。

仅此位置 1 的 Function 的 INTx 虚拟线中断受到影响。

对于具有生成 INTx 中断的 Type 0 配置空间头的 Function，此位是必需的。对于不产生 INTx 中断的 Type 0 配置空间头的 Function，此位是可选的。如果未实现，则该位必须硬连线为 0b。

对于自己生成 INTx 中断的 Type 1 配置空间头的 Function，则需要此位。该位对从二级侧转发的中断没有影响。

对于自己不生成 INTx 中断的 Type 1 配置空间头的 Function，此位是可选的。如果未实现，则该位必须硬连线至 0b。

默认值为 0b。

7.5.1.1.4 Status Register (Offset 06h)

Table 7-4 定义了 Status Register，该寄存器的布局如 Figure 7-6 所示。Function 可能不需要实现所有位，具体取决于 Function 支持的 Feature 集。对于 PCI Express to PCI/PCI-X Bridge，请参阅 *PCI Express to PCI/PCI-X Bridge Specification* 以了解此寄存器的要求。

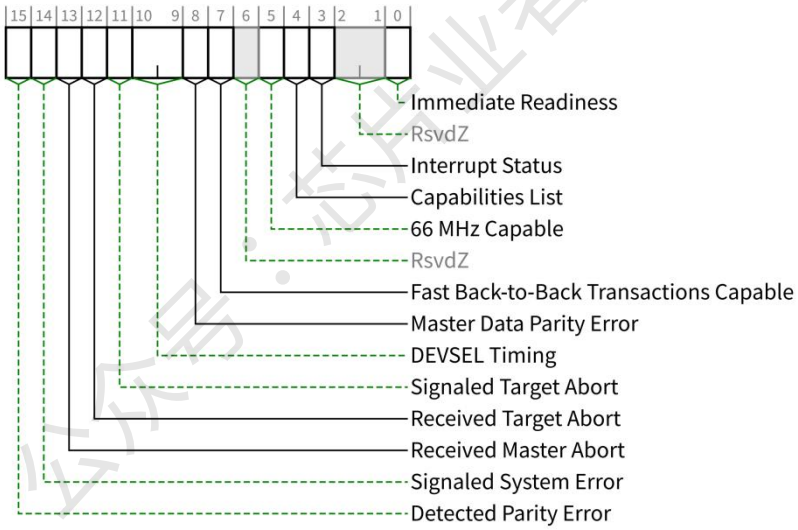

Figure 7-6 Status Register

Bit Location	Register Description	Attributes
0	**Immediate Readiness** - 为 1 时，此可选位表示已保证该 Function 准备就绪，可以随时成功完成有效的配置请求。允许该指示基于特定于实现的知识，即主机准备发出配置请求要花费多长时间。 当该位置 1 时，对于访问此 Function，软件不受任何要求以免在任何类型的复位后延迟配置访问，这包括但不限于 Section 6.6 中定义的时序要求。 如何建立这样的"保证"超出了本文档的范围。 允许系统软件/固件提供取代此位提供的指示的机制，但是此类软件/固件机制不在本规范的范围	RO

之内。

3	**Interrupt Status** - 为 1 时，表明 INTx 中断在 Function 内部挂起。 请注意，具有 Type 1 配置空间头的 Function 转发的来自 secondary side 的 INTx 中断不会在此字段中指示。 将 Interrupt Disable 字段置 1 不会影响该位的状态。 允许不产生 INTx 中断的 Function 将该位硬接线为 0b。 默认值为 0b。	RO
4	**Capabilities List** - 表示是否存在 Extended Capability 列表项。由于所有 PCI Express device Function 都要实现 PCI Express Capability structure，所以该位必须硬连线至 1b。	RO
5	**66 MHz Capable** - 该位在 PCI 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
7	**Fast Back-to-Back Transactions Capable** - 该位在 PCI 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
8	**Master Data Parity Error** - See Section 7.5.1.1.14。 如果 Command 寄存器中的 Parity Error Response 字段为 1b 并且发生以下两种情况之一，则此位由具有 Type 0 配置空间头的 Function 设置为 1b： ● Function 收到一个 Poisoned Completion。 ● Function 发送一个 Poisoned Request。 如果 Command 寄存器中的 Parity Error Response 字段为 1b 并且发生以下两种情况之一，则此位由具有 Type 1 配置空间头的 Function 设置为 1b： ● 端口收到一个向下游的 Poisoned Completion。 ● 端口发送一个向上游的 Poisoned Request。 如果 Parity Error Response 字段为 0b，则该字段永远不会设置为 1b。 默认值为 0b。	RW1C
10:9	**DEVSEL Timing** - 该位在 PCI 规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 00b。	RO
11	**Signaled Target Abort** - See Section 7.5.1.1.14。 当 Function 以 Completer Abort 错误完成 Posted 或 Non-Posted Request 时，该位置 1。当应用于具有 Type 1 配置空间头的 Function 时，此 Completer Abort 错误由 Primary Side 生成。 不发出 Completer Abort 错误的具有 Type 0 配置空间头的 Function 时可以把该字段硬连线为 0b。 默认值为 0b。	RW1C

公众号·芯片业者相承双

12	**Received Target Abort** - See Section 7.5.1.1.14。 当 Requester 收到一个 Completer Abort 的 Completion 时此字段置为 1b。对于具有 Type 1 配置空间头的 Function，在其 Primary Side 收到 Completer Abort 时该字段置为 1b。 不能自己制造 Non-Posted Request 的具有 Type 0 配置空间头的 Function 可以把该位硬连线为 0b。 默认值为 0b。	RW1C
13	**Received Master Abort** - See Section 7.5.1.1.14。 当 Requester 收到一个 Unsupported Request 的 Completion 时此字段置为 1b。对于具有 Type 1 配置空间头的 Function，在其 Primary Side 收到 Unsupported Request 时该字段置为 1b。 不能自己制造 Non-Posted Request 的具有 Type 0 配置空间头的 Function 可以把该位硬连线为 0b。 默认值为 0b。	RW1C
14	**Signaled System Error** - See Section 7.5.1.1.14。 当 Function 发送一个 ERR_FATAL 或 ERR_NONFATAL Message 且 Command Register 的 SERR# Enable 字段为 1b 时该字段置为 1b。 不发送 ERR_FATAL 或 ERR_NONFATAL Message 的 Function 可以把该字段硬连线为 0b。 默认值为 0b。	RW1C
15	**Detected Parity Error** - See Section 7.5.1.1.14。 不论 Command Register 中的 Parity Error Response 字段处于何种状态，只要该 Function 接收到中毒的 TLP，该字段就由该 Function 设置为 1b。对于具有 Type 1 配置空间头的 Function，在其 Primary Side 收到中毒 TLP 时该字段置为 1b。 默认值为 0b。	RW1C

7.5.1.1.5 Revision ID Register (Offset 08h)

Revision ID 寄存器是 HwInit，此寄存器中的值表示 Function 特定的版本标识符。该值由供应商提供。零是可接受的值。Device ID、Vendor ID 和 Revision ID 一起用作软件确定应加载哪个驱动程序的一种机制。供应商必须确保所选的值不会导致使用不兼容的设备驱动程序。

7.5.1.1.6 Class Code Register (Offset 09h)

Class Code 寄存器是只读的，用于标识 Function 的通用操作，在某些情况下还用于标识特定的寄存器级编程接口。寄存器布局如 Figure 7-7 所示，并在 Table 7-5 中进行了描述。PCI-Code-and-ID 规范中提供了 Base Class Code、Sub-Class Code 和 Programming Interface 的编码。所有未指定的编码均为保留。

公众号：志片业者相承双

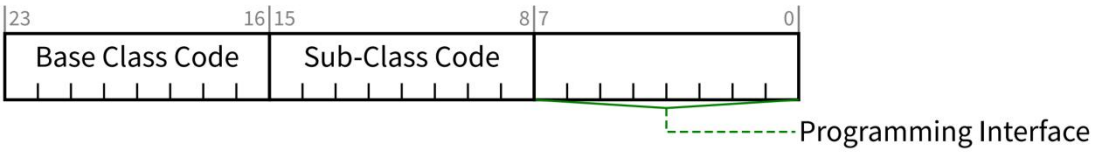

Figure 7-7 Class Code Register

Bit Location	Register Description	Attributes
7:0	**Programming Interface** - 该字段标识特定的寄存器级编程接口（如果有），以便与设备无关的软件可以与该 Function 进行交互。 [PCI-Code-and-ID]规范中提供了此字段的编码。所有未指定的编码均为保留。	RO
15:8	**Sub-Class Code** - 指定一个基类子类，该子类可以更具体地标识 Function 的操作。 [PCI-Code-and-ID]规范中提供了此字段的编码。所有未指定的编码均为保留。	RO
23:16	**Base Class Code** - 该编码可以对 Function 执行的操作类型进行大致分类。 [PCI-Code-and-ID]规范中提供了此字段的编码。所有未指定的编码均为保留。	RO

7.5.1.1.7 Cache Line Size Register (Offset 0Ch)

Cache Line Size Register 由系统固件或操作系统编程为系统高速缓存行的大小。但是，请注意，特别是在热插拔设备的情况下，传统的 PCI 兼容软件可能并不总是能够正确编程此寄存器。实现此可读写寄存器是为了实现旧版兼容性，但对任何 PCI Express 设备行为都没有影响。对于 PCI Express to PCI/PCI-X Bridge，请参阅[PCIe-to-PCI-PCI-X-Bridge]规范以了解此寄存器的要求。该寄存器的默认值为 00h。

7.5.1.1.8 Latency Timer Register (Offset 0Dh)

该寄存器也称为 Type 1 配置空间头 Function 的 Primary Latency Timer。该延迟计时器最初是在[PCI]和[PCI-to-PCI-Bridge]中描述的。其功能不适用于 PCI Express。该寄存器必须硬连线到 00h。

7.5.1.1.9 Header Type Register (Offset 0Eh)

该寄存器标识预定义配置头的第二部分的布局（从配置空间中的字节 10h 开始），以及设备是否可能包含多个 Function。寄存器布局如 Figure 7-8 所示，Table 7-6 说明了寄存器中的字段。

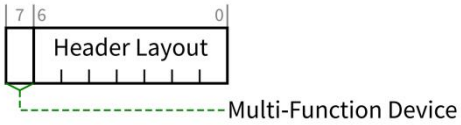

Figure 7-8 Header Type Register

公众号·芯片业者相承双

Bit Location	Register Description	Attributes
6:0	**Header Layout** - 该字段标识预定义配置头的第二部分的布局。对于实现 Type 0 配置空间头的 Function，必须使用 000 0000b 编码。 对于实现 Type 1 配置空间头的 Function，必须使用 000 0001b 编码。 编码 000 0010b 是保留的。此编码最初在[PC-Card]中进行了描述，并在以前的编程模型版本中使用。任何重新调整用途的尝试都应仔细考虑。 所有其他编码都是保留的。	RO
7	**Multi-Function Device** - 置位时表示设备可能包含多个 Function，但不是必须的。 允许软件探查 Function 0 以外的 Function。清除后，除非有其他机制（如 ARI 或 SR-IOV 扩展功能结构）明确指出，否则软件不得探查 Function 0 以外的 Function。 除另有说明外，建议在有多个 Function 的情况下将该位置 1，在只有一个 Function 的情况下清 0。	RO

7.5.1.1.10 BIST Register (Offset 0Fh)

该寄存器用于 BIST 的控制和状态描述。不支持 BIST 的 Function 必须将该寄存器硬连线到 00h。实现了 BIST 的 Function 不得阻碍 PCI Express 链路的正常运行。Table 7-7 说明了寄存器中的字段，Figure 7-9 显示了寄存器布局。

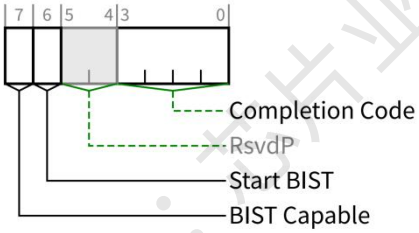

Figure 7-9 BIST Register

Bit Location	Register Description	Attributes
3:0	**Completion Code** - 此字段编码最近测试的状态。值 0000b 表示该 Function 已通过测试。非零值表示 Function 测试失败。特定于 Function 的故障代码可以编码为非零值。 仅当设置了 BIST Capable 并且 Start BIST 为 Clear 时，此字段的值才有意义。 默认值为 0000b。 如果 BIST Capable 为 0，则此字段必须硬连线至 0000b。	RO
6	**Start BIST** - 如果设置了 BIST Capable，则将该位置 1 以启用 BIST。BIST 完成后，Function 将重置该位。如果在设置后 2 秒钟未清除该位（BIST 未完成），则允许软件使设备发生故障。 此字段写 0 没影响。 如果 BIST Capable 为 0，则此字段必须硬连线至 0b。	RW/RO

公众号：芯片业者柏承双

7	**BIST Capable** - 为 1b 时，表示该 Function 支持 BIST。为 0b 时，表示 Function 不支持 BIST。	HwInit

7.5.1.1.11 Capabilities Pointer (Offset 34h)

该寄存器用于指向该 Function 实现的 Capabilities List 中的第一个 Capability。所有 PCI Express Function 都需要实现 PCI Power Management Capability 和 PCI Express Capability Structure，这两个 Structure 必须包含在链表中的某个位置。该寄存器可以指向这两个 Capability Structure 中的任何一个，或者指向由 Function 实现的可选的 Capability Structure。该寄存器的最低两位保留，必须设置为 00b。在将此寄存器用作配置空间中指向新 Capabilities List 的第一个条目的指针之前，软件必须屏蔽掉这两位。

7.5.1.1.12 Interrupt Line Register (Offset 3Ch)

Interrupt Line 寄存器表示中断线路由信息。该寄存器是可读写的，必须由需要使用中断引脚的 Function 实现（请参见以下说明）。该寄存器中的值由系统软件编程，并且特定于系统架构。Function 本身不使用该值；该寄存器中的值由设备驱动程序和操作系统使用。

7.5.1.1.13 Interrupt Pin Register (Offset 3Dh)

Interrupt Pin 寄存器是一个只读寄存器，用于标识 Function 使用的传统中断消息（更多信息，请参见第 6.1 节）。有效值为 01h，02h，03h 和 04h，它们分别映射到 INTA，INTB，INTC 和 INTD 传统中断消息。值 00h 表示该 Function 不使用传统中断消息。值 05h 至 FFh 为保留值。

PCI Express 为 single Function device 定义一个传统中断消息，为一个 Multi-Function Device 定义多达四个传统中断消息。对于 single Function device，只能使用 INTA。

Multi-Function Device 上的任何 Function 都可以使用任何 INTx 消息。如果设备实现单个传统中断消息，则它必须是 INTA。如果它实现了两个传统中断消息，则它们必须是 INTA 和 INTB；以此类推。对于 Multi-Function Device，所有 Function 都可以使用相同的 INTx 消息，或者每个 Function 都可以拥有自己的（最多四个功能）或其任意组合。单个 Function 永远不会产生多个 INTx 消息的中断请求。

7.5.1.1.14 Error Registers

Command and Status 寄存器中的 Error Control/Status 字段(分别参阅 Section 7.5.1.1.3 和 Section 7.5.1.1.4)以及 Type 1 Configuration Space Header Function 的 Bridge Control 和 Secondary Status 寄存器（see Section 7.5.1.3.10 and Section 7.5.1.3.7 respectively）用于控制 PCI 和 PCI Express device Function 的 PCI 兼容错误的报告。在 Section 6.2.7.1 中还将讨论了如何将 PCI Express 错误映射到 PCI 错误。除了 PCI 兼容的错误控制和状态之外，还可以通过 Section 7.5.3 中所述的 PCI Express Capability 结构控制 PCI Express 错误报告。PCI 兼容的错误控制和状态寄存器的字段对通过 PCI Express Capability 结构启用的 PCI Express 错误报告没有任何影响。PCI Express device Function 可以实现 Section 7.8.4 所述的可选高级错误报告。

对于实现了 Type 1 配置空间头的 PCI Express 根端口：

- primary side Error Control/Status 寄存器适用于在与 RC 相关联的内部逻辑上检测到的错误。
- secondary side Error Control/Status 寄存器适用于在链路上检测到的源自根端口的错误。

对于实现了 Type 1 配置空间头的 PCI Express Switch Upstream Port：

- primary side Error Control/Status 寄存器适用于在 Switch 上行链路上检测到的错误。

公众号：芯片业者相承双

- secondary side Error Control/Status 寄存器适用于 Switch 内部检测到的错误。

对于实现了 Type 1 配置空间头的 PCI Express Switch Downstream Port：

- primary side Error Control/Status 寄存器适用于在 Switch 内部检测到的错误。
- secondary side Error Control/Status 寄存器适用于 Switch 下行链路上检测到的错误。

7.5.1.2 Type 0 Configuration Space Header

Figure 7-10 详细说明了 PCI Express device Function 的 Type 0 Configuration Space Header 的寄存器的分配。

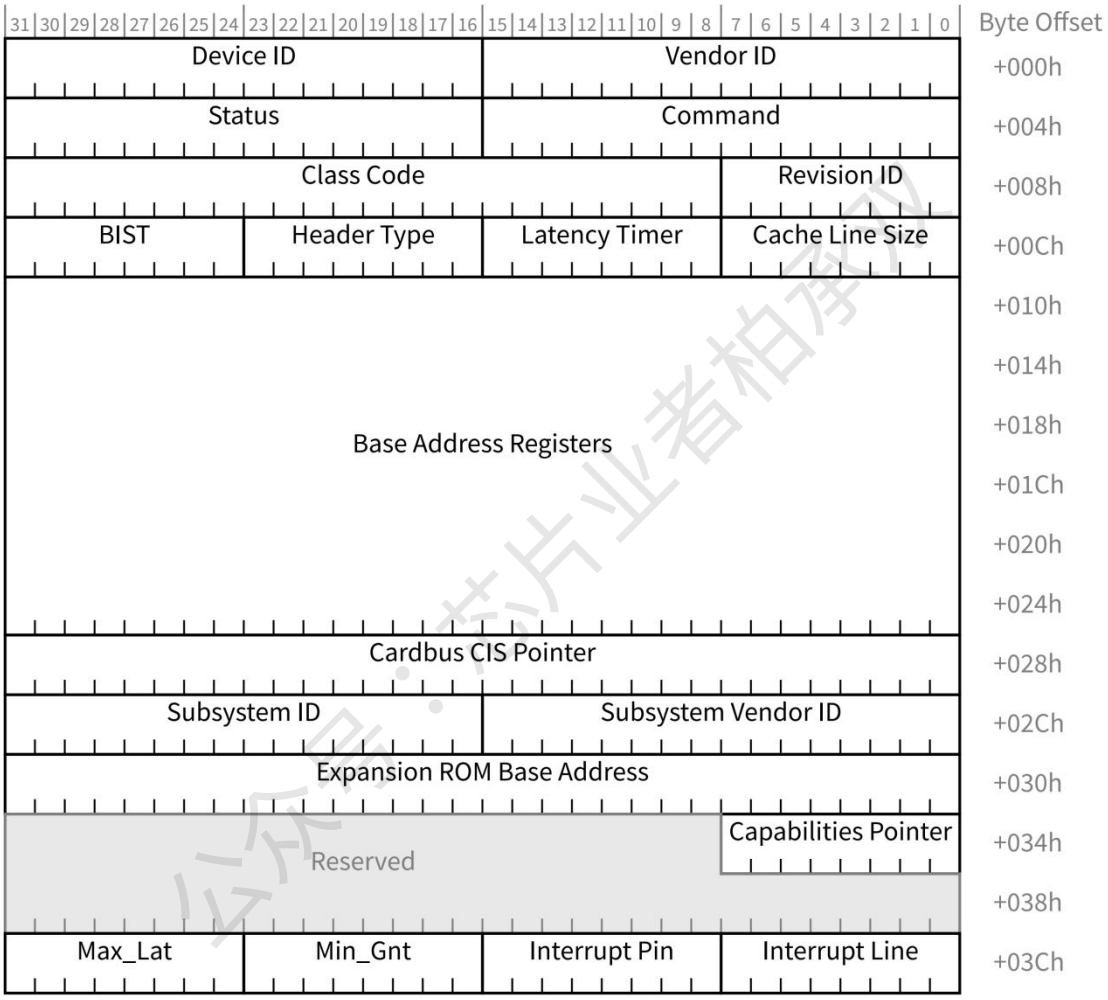

Figure 7-10 Type 0 Configuration Space Header

7.5.1.2.1 Base Address Registers (Offset 10h - 24h)

在将计算机 Boot 至操作系统之前，系统软件必须构建一致的地址映射。这意味着必须确定系统中有多大内存，以及系统中的 Function 需要多少地址空间。确定此信息后，系统软件可以将 Function 映射到合理的位置，然后继续执行系统 Boot。为了以与设备无关的方式进行此映射，将此映射的基址寄存器放置在配置空间的预定义头中。强烈建议上电之后固件/软件还支持可选的增强配置访问机制（ECAM）。

所有基址寄存器中的 bit 0 都是只读的，用于确定该寄存器是映射到存储器还是 I/O 空间。映射到存储器空间的基址寄存器的 bit

0 必须为 0b（见 Figure 7-11）。映射到 I/O 空间的基址寄存器的 bit 0 必须为 1b（见 Figure 7-12）。

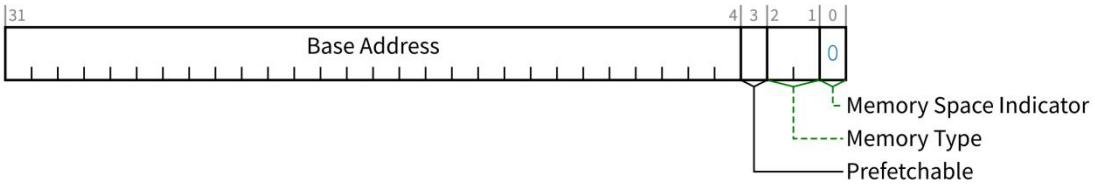

Figure 7-11 Base Address Register for Memory

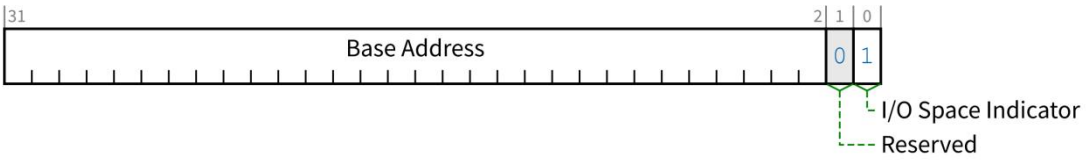

Figure 7-12 Base Address Register for I/O

映射到 I/O 空间的基址寄存器始终为 32 位宽，其 bit 0 硬连线为 1b，bit 1 是保留位，读取时必须返回 0b，其他位用于将 Function 映射到 I/O 空间。

映射到内存空间的基址寄存器可以是 32 位或 64 位宽（以支持映射到 64 位地址空间），其中 bit 0 硬连线到 0b。 对于存储器基址寄存器，bit 2 和 1 的编码含义如 Table 7-8 所示。如果数据可预取，则 bit 3 应设置为 1b，否则应设置为 0b。如果读取没有任何副作用，则允许该 Function 将范围标记为可预取，该 Function 会在读取时返回所有字节，而与字节使能无关，并且主机桥可以将处理器写入合并到该范围内而不会引起错误。bit 3-0 是只读的。

Table 7-8 Memory Base Address Register Bits 2:1 Encoding

Bits 2:1(b)	Meaning
00	Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.
01	Reserved[140]
10	Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.
11	Reserved

BAR 中某些比特的属性（如果已实现）受 Resizable BAR Capability 影响。

通过 BAR 请求内存空间的 PCI Express Function 必须设置 BAR 的 Prefetchable 字段，除非该范围包含具有读取副作用的地址或该 Function 不允许写入合并的地址。 强烈建议映射到内存空间中的资源应尽可能设计为可预取。Legacy Endpoint 以外的 PCI Express Function 必须为请求可预取内存空间的任何基址寄存器支持 64 位寻址。BAR 请求的最小存储空间地址范围是 128 个字节。 BAR 中某些比特的属性受 Resizable BAR Capability（如果已实现）影响。

Additional Guidance on the Prefetchable Bit in Memory Space BARs

带有请求大量不可预取内存空间（例如，超过 64 MB）的 Memory Space BAR 的 PCI Express 适配器可能会导致某些可伸缩平台上该空间的短缺，因为许多平台仅支持总共 1 GB 或更少的可预取的内存空间。这可能会限制这些平台上可以支持的此类适配器的数量。 因此，特别建议 BAR 请求大量存储空间时设置其 Prefetchable 字段，因为在大多数可扩展平台上，可预取存储空间更为丰富。

如果 Memory Space BAR 在其范围内的任何位置都没有读取副作用并且所有位置都允许写入合并时需要将其 Prefetchable 位置 1，在某些系统配置中，即使不满足这些条件，将 Prefetchable 位置 1 仍然允许正确的操作。 对于那些情况，适配器在某些候选 BAR 中设置 Prefetchable 位是有意义的，以便系统可以将这些 BAR 映射到可预取 Memory Space 中，以避免不可预取的 Memory Space

公众号：芯片业者拍承双

短缺。

在满足以下列举标准的 PCI Express 系统上，即使 BAR 的范围包括某些具有读取副作用或不能容忍写入合并的位置，在 BAR 中设置 Prefetchable 位仍将允许正确操作。这主要是由于 PCI Express 内存读取始终包含一个明确的长度，并且 PCI Express Switch 从不预取或进行字节合并。通常，只有 64 位 BAR 是很好的候选者，因为只允许 Legacy Endpoint 设置 32 位 BAR 中的 Prefetchable 位，并且大多数可扩展平台将所有 32 位 Memory BAR 映射到不可预取的内存空间，而不管 Preprefable 位的值如何。

以下是足以保证给定候选 BAR 正确性的标准：

- 从主机到适配器的整个路径都是 PCI Express。
- 没有传统的 PCI 或 PCI-X 设备对等读取 BAR 映射的范围。
- PCI Express Host Bridge 不执行任何字节的合并。（大多数平台都是如此。）
- TH 位置 1 的任何具有读取副作用的位置绝不是存储器读取的目标。参见 Section 2.2.5。
- BAR 映射的范围永远不是主机发起的或对等的推测性内存读取的目标。

以上标准是一个简单的设置，足以保证正确性。其他限制较少但较复杂的标准也可以保证正确性，但不属于本规范的范围。

7.5.1.2.2 Cardbus CIS Pointer Register (Offset 28h)

该寄存器最初在[PC-Card]规范中进行了描述。其功能不适用于 PCI Express。它必须是只读的并且硬连线到 0000 0000h。

7.5.1.2.3 Subsystem Vendor ID Register/Subsystem ID Register (Offset 2Ch/2Eh)

Subsystem Vendor ID 和 Subsystem ID 寄存器用于唯一标识 PCI Express 组件所在的适配器或子系统。它们为供应商提供了一种机制，使它们的产品彼此区分，即使这些组件上可能具有相同的 PCI Express 组件（因此也具有相同的 Vendor ID 和 Device ID）。

除 Base Class 为 06h 且其 Sub-Class 为 00h-04h（00h，01h，02h，03h，04h），或者 Base Class 为 08h 且其 Sub-Class 为 00h-03h（00h，01h，02h，03h）的 Function 外，所有 Function 都要实现这些寄存器。Subsystem Vendor ID 可以从 PCI SIG 获得，并用于标识适配器、主板或子系统的供应商 [141]。Subsystem Vendor ID（SVID）必须是 PCI-SIG 分配给子系统供应商的 Vendor ID。与 PCI-SIG 程序保持一致，必须从 PCI-SIG 获得有效的供应商标识符，以确保唯一性。

注 141：一家公司仅需要一个 Vendor ID。该值可用于配置空间的 Vendor ID 寄存器或 Subsystem Vendor ID 寄存器中。如果公司制造芯片，则将其用于 Vendor ID 寄存器）。如果公司负责组装，则在 Subsystem Vendor ID 寄存器中使用该值。如果公司同时生产芯片和组件，则两个寄存器中将使用相同的值。

Subsystem ID 的值由供应商分配。Subsystem ID 值与 Subsystem Vendor ID 一起构成 PCI 产品的唯一标识符。Subsystem ID 和 Device ID 的值是不同的且彼此无关，并且软件不应假定它们之间有任何关系。

Function 变为 Configuration-Ready 之前，必须先加载这些寄存器中的值。这些值的加载方式未指定，但可以在制造过程中完成，也可以从外部逻辑（例如，串行 ROM 等）加载。不能使用 Expansion ROM 软件加载这些值，因为不能保证在所有系统的 POST 期间都运行 Expansion ROM 软件。

如果将设备设计为专用于主板，则系统供应商可以在每次开机后使用系统特定的软件来初始化这些寄存器。

Subsystem Vendor ID and Subsystem ID

Subsystem Vendor ID 和 Subsystem ID 字段一起使用，使软件可以唯一地标识 PCI 电路板产品。因此，供应商不应在共享同一 Subsystem Vendor ID 的多种产品类型之间重用 Subsystem ID。如果产品属于同一族和同一代，并且仅在容量或性能上有所不同，

则可以在具有相同 Subsystem Vendor ID 的产品上重用 Subsystem ID 值，尽管不是优选的做法。还请注意，对于单个产品类型，供应商可以随时使用多个唯一的 Subsystem ID 值，以表明某些内部差异，例如组件选择。

7.5.1.2.4 Expansion ROM Base Address Register (Offset 30h)

某些 Function，尤其是打算用于附加卡的 Function，需要为 Expansion ROM 提供本地 EPROM（有关 ROM 内容的定义，请参阅 [PCI-Firmware]）。定义 Expansion ROM Base Address Register 用以处理此 Expansion ROM 的基地址和大小信息。寄存器布局如 Figure 7-13 所示，Table 7-9 说明了寄存器中的各个字段。

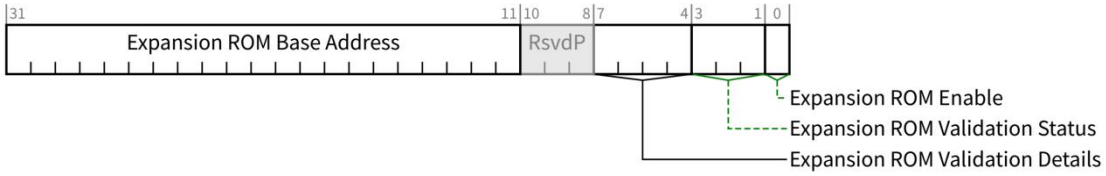

Figure 7-13 Expansion ROM Base Address Register

Bit Location	Register Description	Attributes
0	**Expansion ROM Enable** - 该位控制 Function 是否能通过 Expansion ROM Base Address Register 接受对 Expansion ROM 的访问。支持可通过该寄存器访问 Expansion ROM 的 Function 必须实现该字段。如果 Function 实现了 Enhanced Allocation Capability，其中包括用于 Expansion ROM 的 EA 条目，则该字段必须硬连线至 0b（请参见 Section 7.5.1.2.4）。允许不支持 Expansion ROM 的 Function 将该位硬接线为 0b。当该位为 0b 时，将禁用通过 Expansion ROM 基址寄存器的访问 Function 的 Expansion ROM 地址空间。当该位为 1b 时，该寄存器中的 Expansion ROM Base Address 字段被用于地址解码。Command 寄存器中的 Memory Space Enable 字段优先于 Expansion ROM Enable 位。仅当同时设置了 Memory Space Enable 和 Expansion ROM Enable 时，Function 才可以通过 Expansion ROM Base Address Register 访问其 Expansion ROM。该位的默认值为 0b。 为了最小化所需的地址解码器的数量，允许 Function 在 Expansion ROM Base Address Register 和 Enhanced Allocation Capability 中的基址寄存器条目之间共享解码器。设置 Expansion ROM Enable 后，解码器将用于访问 Expansion ROM，并且与设备无关的软件不得通过任何其他基址寄存器或 Enhanced Allocation Capability 中的条目来访问。 PF 不允许地址解码共享（请参阅 Section 9.3.4.1.15），或者该 Function 包含带有 Expansion ROM 的 EA 条目的 Enhanced Allocation Capability。	RO/RW
3:1	**Expansion ROM Validation Status** - Expansion ROM 验证是可选实现的。当此字段为非零时，表示硬件处于 Expansion ROM 内容的验证状态。 ● 如果 Expansion ROM 通过了特定于实现的完整性检查，则它被视为有效。 ● 如果通过了特定于实现的完整性检查，但被指示为可能存在风险，则 Expansion ROM 被视为有效警告。 ● 如果通过了可选的特定于实现的信任测试（例如，由受信任的证书签名），则有效或有效警告的 Expansion ROM 也被视为受信任的。 ● 硬件验证指的是验证 Expansion ROM 的内容。还允许此验证状态涵盖其他内部信	HwInit/ROS

息（例如，内部固件）。验证不包括重要产品数据（请参阅 Section 6.28）。

- 实现是否能够返回验证状态值 011b，101b，110b 或 111b 是可选的。定义的验证状态编码为：

000b 不支持验证。

001b 正在验证过程中。

010b 内容验证通过，未执行信任测试。

011b 验证通过，信任。

100b 内容验证失败。

101b 验证失败有效但不受信任的内容（例如，超时、过期或吊销的证书）。

110b 警告通过验证通过特定于实现的警告。有效内容，未执行信任测试。

111b 警告通过验证通过特定于实现的警告。有效和受信任的内容。

- 如果 Function 不支持验证，则必须将该字段硬连线到 000b。
- 如果 Function 支持验证并且实现了针对 Expansion ROM 的 EA 条目的 EA 功能，则此字段为 HwInit，其值必须在 010b 和 111b 之间（请参见 Section 7.8.5.3）。
- 否则，此字段为 Read Only Sticky，默认值为 001b。验证完成后，此字段必须包含介于 010b 和 111b 之间的值。
- 软件假定当该字段为 001b 并且在该 Function 基本复位释放后经过 1 分钟验证完成。此字段仅通过基本复位复位，而不受其他复位的影响。

7:4 **Expansion ROM Validation Details** - 包含与 Expansion ROM Validation 相关的可选的、特定于实现的详细信息。 HwInit/ROS/RsvdP

- 如果该 Function 不支持 Expansion ROM Validation，则此字段为 RsvdP。
- 该字段是可选的。如果支持验证且未实现此字段，则必须将该字段硬连线到 0000b。允许将该字段中任何未使用的位硬连线为 0b。
- 如果验证正在进行中（Expansion ROM Validation Status 为 001b），则此字段的非零值表示验证进度阶段的特定于实现的指示（例如 50％完成）。值 0000b 表示未提供验证进度信息。
- 如果验证完成（Expansion ROM Validation Status 为 010b 至 111b），则此字段中的非零值表示其他特定于实现的信息。值 0000b 表示未提供任何信息。
- 如果 Function 支持验证，并且具有针对 Expansion ROM 的 EA 功能，则此字段为 HwInit。
- 否则，此字段为 Read Only Sticky。此字段仅通过基本复位复位，而不受其他复位

公众号·芯片业者相承双

	的影响。 • 验证过程完成后，该字段不得更改值。 • 建议系统软件在报告验证状态（例如错误日志）时包括此字段的值。	
31:11	**Expansion ROM Base Address** - Expansion ROM 的起始存储器地址的高 21 位。Expansion ROM 基址寄存器的低 11 位由软件屏蔽（设置为零）以形成 32 位地址。 该字段类似于 32 位基地址寄存器的地址部分。该字段对应于扩展 ROM 基址的高 21 位。Function 实际实现的位数（在这 21 个中）取决于 Function 所需的 Expansion ROM 地址空间。例如，一个需要 64 KB 来映射其 Expansion ROM 的 Function 会将该字段中的前 16 位实现为可写状态，而将剩下的后 5 位（这 21 位中的最后 5 位）硬连线为 0b。Function 请求的地址空间的大小不得大于 16 MB。支持可通过该寄存器访问的 Expansion ROM 的 Function 必须实现此字段。如果 Function 实现了 Expansion ROM 的 EA 功能，则此字段必须硬连线为 0（请参见 Section 7.8.5.3）。不支持 Expansion ROM 的 Function 允许将该字段硬连线为 0。 与设备无关的配置软件可以通过将全 1 的值写入该字段，然后将其读回来确定该 Function 所需的地址空间。该 Function 将在所有无关位中返回 0，有效地指定了大小和对齐要求。Function 请求的地址空间量不能大于 16 MB。	RO/RW

7.5.1.2.5 Min_Gnt Register/Max_Lat Register (Offset 3Eh/3Fh)

这些寄存器不适用于 PCI Express。它们必须是只读的，并且必须硬连线到 00h。

7.5.1.3 Type 1 Configuration Space Header

Figure 7-14 详细说明了用于 Switch 和 Root Port 的 Type 1 Configuration Space Header 的寄存器的分配。

公众号·芯片业者相关权

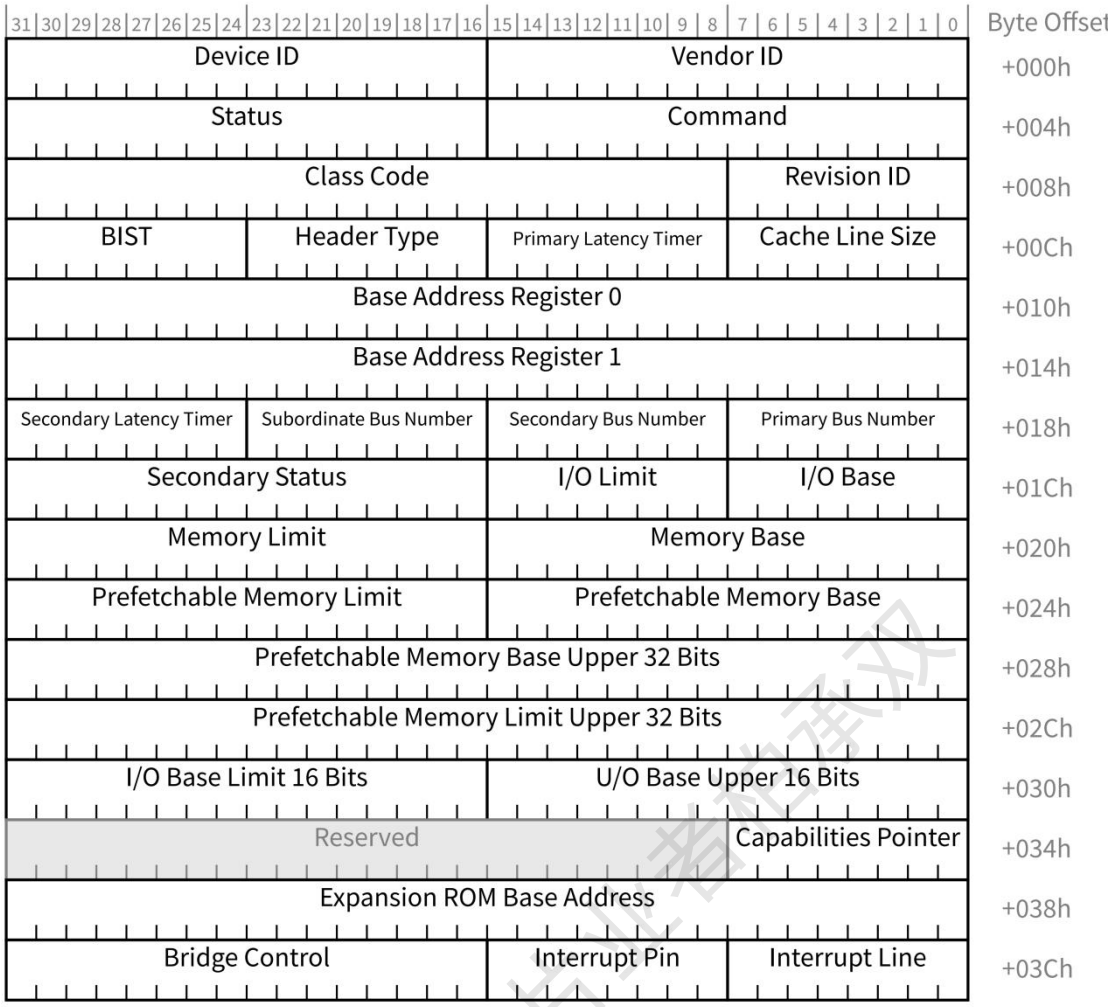

Figure 7-14 Type 1 Configuration Space Header

7.5.1.3.1 Type 1 Base Address Registers (Offset 10h-14h)

Type 1 配置空间头的基址寄存器与 Section 7.5.1.2.1 中定义的 Type 0 配置空间头的基址寄存器一致，只不过数量不同。

7.5.1.3.2 Primary Bus Number Register (Offset 18h)

除非另有说明，否则 PCI Express Function 不使用该寄存器，但必须将其实现为可读写，并且默认值必须为 00h，以与旧版软件兼容。PCI Express Function 按 Section 2.2.6 中所述（包括例外）捕获 Bus Number 和 Device Number。有关此要求的例外，请参考[PCIe-to-PCI-PCI-X-Bridge]规范。

主总线号（Primary Bus Number）指的是与端口上游一侧相连的总线的编号。二级总线号指的是与端口下游一侧相连的总线的编号。从属总线号指的是从该端口看去所连最大的 PCI 总线号，或者说最远的 PCI 的总线号，因而对于 RC 设备或 Switch 设备，这个值在枚举过程中是不断增大变化的。

7.5.1.3.3 Secondary Bus Number Register (Offset 19h)

Secondary Bus Number 寄存器用于记录与桥的二级接口连接的 PCI 总线的总线号。配置软件对该寄存器中的值进行编程。桥使用此寄存器来确定何时以及如何响应在其主接口上观察到的基于 ID 路由的 TLP，特别是在执行某些转换后的某些情况下，何时将

TLP 转发到其二级接口。有关配置请求路由和转换规则，请参见 Section 7.3.3。该寄存器必须实现为可读/写，默认值必须为 00h。

7.5.1.3.4 Subordinate Bus Number Register (Offset 1Ah)

Subordinate Bus Number 寄存器用于记录桥后面（或从属于）的编号最高的 PCI 总线的总线号。配置软件对该寄存器中的值进行编程。桥使用该寄存器来确定何时以及如何响应在其主接口上观察到的基于 ID 路由的 TLP，特别是何时将 TLP 转发到其二级接口。有关配置请求路由和转换规则，请参见 Section 7.3.3。该寄存器必须实现为可读/写，默认值必须为 00h。

7.5.1.3.5 Secondary Latency Timer (Offset 1Bh)

该寄存器不适用于 PCI Express。它必须是只读的并且硬连线到 00h。对于 PCI Express to PCI/PCI-X Bridge，请参阅[PCIe-to-PCI-PCI-X-Bridge]规范以了解此寄存器的要求。

7.5.1.3.6 I/O Base/I/O Limit Registers(Offset 1Ch/1Dh)

I/O Base 和 I/O Limit 寄存器是可选实现的，它们定义一个地址范围，桥使用该地址范围来确定如何将 I/O 事务从一个接口转发到另一个接口。

如果桥未实现 I/O 地址范围，则 I/O Base 和 I/O Limit 寄存器都必须实现为只读寄存器，读取时返回零。如果桥支持 I/O 地址范围，则这些寄存器必须由配置软件初始化，因此未指定默认状态。

如果桥实现了 I/O 地址范围，则 I/O Base 和 I/O Limit 寄存器的高 4 位都是可写的，并且对应于地址 Address [15:12]。出于地址解码的目的，桥假定 I/O 基址（未在 I/O 基址寄存器中实现）的低 12 位地址 Address [11：0]为零。类似地，桥假定 I/O 限址（未在 I / O 限制寄存器中实现）的低 12 地址位 Address [11:0]为 FFFh。因此，已定义的 I/O 地址范围的底部将与 4 KB 边界对齐，而已定义的 I/O 地址范围的顶部将比 4 KB 边界小 1。

如果桥的二级侧没有 I/O 地址，则可以将 I/O Limit 寄存器编程为小于 I/O Base 寄存器的值。在这种情况下，桥不会将任何 I/O 事务从主总线转发到二级总线，而是将所有 I/O 事务从二级总线转发到主总线。

I/O Base 和 I/O Limit 寄存器的低四位均为只读，包含相同的值，并根据 Table 7-10 编码桥的 I/O 寻址能力。

Table 7-10 I/O Addressing Capability

Bits 3:0	I/O Addressing Capability
0h	16-bit I/O addressing
1h	32-bit I/O addressing
2h-Fh	Reserved

如果 I/O Base 和 I/O Limit 寄存器的低四位的值为 0000b，则桥仅支持 16 位 I/O 寻址（出于 ISA 兼容性），并且出于地址解码的目的， 桥假定 I/O Base 和 I/O Limit 地址（未在 I/O Base 和 I/O Limit 寄存器中实现）的高 16 地址位 Address [31:16]为零。 请注意，桥仍必须对 I/O 地址执行完整的 32 位解码（即，检查 Address [31:16]为 0000h）。在这种情况下，桥支持的 I/O 地址范围将限制为 I/O 空间的前 64 KB（0000 0000h 至 0000 FFFFh）。

如果 I/O Base 和 I/O Limit 寄存器的低四位是 0001b，则桥支持 32 位 I/O 地址解码，并且 I/O Base 高 16 位和 I/O Limit 高 16 位，这些位分别保留 32 位 I/O Base 和 I/O Limit 地址的高 16 位，分别对应于地址[31:16]。在这种情况下，允许系统配置软件在 4 GB I/O 空间中的任何位置找到桥支持的 I/O 地址范围。请注意，当桥支持 32 位 I/O 寻址时，仍然适用 4 KB 对齐和粒度限制。

公众号：芯片业者柏承双

7.5.1.3.7 Secondary Status Register (Offset 1Eh)

Table 7-11 定义了二级状态寄存器，Figure 7-15 提供了寄存器布局。对于 PCI Express to PCI/PCI-X Bridge，请参阅[PCIe-to-PCI-PCI-X-Bridge]规范以了解此寄存器的要求。

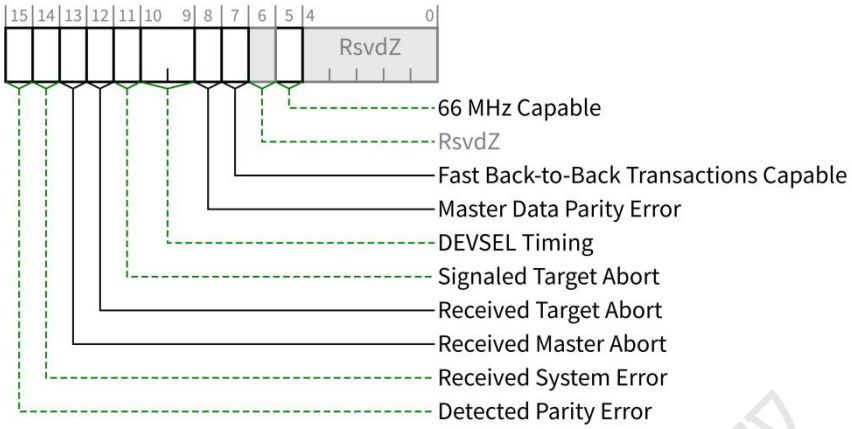

Figure 7-15 Secondary Status Register

Bit Location	Register Description	Attributes
5	**66 MHz Capable** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
7	**Fast Back-to-Back Transactions Capable** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
8	**Master Data Parity Error** - See Section 7.5.1.1.14。 如果 Bridge Control 寄存器中的 Parity Error Response Enable 字段为 1b 并且发生以下两种情况之一，则此位由具有 Type 1 配置空间头的 Function 设置为 1b： ● 端口收到一个来自上游的 Poisoned Completion。 ● 端口发送一个向下游的 Poisoned Request。 如果 Parity Error Response Enable 字段为 0b，则该字段永远不会设置为 1b。 默认值为 0b。	RW1C
10:9	**DEVSEL Timing** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 00b。	RO
11	**Signaled Target Abort** - See Section 7.5.1.1.14。 当具有 Type 1 配置空间头的 Function（由 Type 1 头的 Function 自己完成的请求）的 Secondary Side 以 Completer Abort 错误完成 Posted 或 Non-Posted Request 时，会把该字段置为 1b。 默认值为 0b。	RW1C

公众号·芯片业老柏承双

12	**Received Target Abort** - See Section 7.5.1.1.14。 当具有 Type 1 配置空间头的 Function（由 Type 1 头的 Function 自己发出的请求）的 Secondary Side 收到一个 Completer Abort 的 Completion 时，会把该字段置为 1b。 默认值为 0b。	RW1C
13	**Received Master Abort** - See Section 7.5.1.1.14。 当具有 Type 1 配置空间头的 Function（由 Type 1 头的 Function 自己发出的请求）的 Secondary Side 收到一个 Unsupported Request 的 Completion 时，会把该字段置为 1b。 默认值为 0b。	RW1C
14	**Signaled System Error** - See Section 7.5.1.1.14。 当具有 Type 1 配置空间头的 Function 的 Secondary Side 收到一个 ERR_FATAL 或 ERR_NONFATAL Message 时，会把该字段置为 1b。 默认值为 0b。	RW1C
15	**Detected Parity Error** - See Section 7.5.1.1.14。 不论 Bridge Control 寄存器中的 Parity Error Response Enable 字段处于何种状态，只要具有 Type 1 配置空间头的 Function 的 Secondary Side 收到一个中毒的 TLP 时，会把该字段置为 1b。 默认值为 0b。	RW1C

7.5.1.3.8 Memory Base Register/Memory Limit Register(Offset 20h/22h)

Memory Base 和 Memory Limit 寄存器定义了一个内存映射的地址范围，桥使用该地址范围来确定如何将内存事务从一个接口转发到另一个接口（有关更多详细信息，请参见[PCI-to-PCI-Bridge]规范）。

Memory Base 和 Memory Limit 寄存器的高 12 位都是可读/写的，它们对应于 32 位地址的高 12 地址位 Address [31:20]。 出于地址解码的目的，桥假定存储器基址（未在存储器基址寄存器中实现）的低 20 地址位 Address [19:0]为零。类似地，桥假定存储器限址（未在存储器限制寄存器中实现）的低 20 地址位 Address [19:0]为 F FFFFh。因此，定义的内存地址范围的底部将与 1 MB 边界对齐，而定义的内存地址范围的顶部将比 1 MB 边界小 1。

如果桥的二级侧没有存储器映射的地址空间，则必须将 Memory Limit 寄存器的值编程为小于 Memory Base 寄存器的值。

如果没有可预取的内存空间，并且桥的二级侧上没有内存映射的空间，则桥将不会将任何内存事务从主总线转发到二级总线，并将所有内存事务从二级总线转发主总线。

Memory Base 和 Memory Limit 寄存器的低四位均为只读，读取时返回零。

这些寄存器必须由配置软件初始化，因此未指定默认值。

7.5.1.3.9 Prefetchable Memory Base/Prefetchable Memory Limit Registers (Offset 24h/26h)

Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器必须在支持 64 位地址的场景下才能使用。

公众号：芯片业务相承双

Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器是可选实现的。它们定义了一个可预取的内存地址范围，桥使用该范围来确定何时将内存事务从一个接口转发到另一个接口。

如果桥未实现可预取存储器地址范围，则必须将 Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器寄存器都实现为只读寄存器，在读取时返回零。如果桥实现了可预取的存储器地址范围，则这两个寄存器都必须实现为可读/写寄存器。如果桥支持可预取的存储器地址范围，则这些寄存器必须由配置软件初始化，因此未指定默认状态。

如果桥实现了可预取的存储器地址范围，则将对寄存器的高 12 位进行读/写操作，并与 32 位地址的高 12 位地址 Address [31:20]相对应。出于地址解码的目的，桥假定可预取存储器基址（未在可预取存储器基址寄存器中实现）的低 20 地址位 Address [19:0]为零。类似地，桥假定可预取存储器限制地址（未在可预取存储器限制寄存器中实现）的低 20 地址位 Address [19:0]为 F FFFFh。因此，定义的可预取存储器地址范围的底部将与 1 MB 边界对齐，并且定义的存储器地址范围的顶部将比 1 MB 边界小 1。

如果桥的二级侧没有可预取的存储器，则必须将 Prefetchable Memory Limit 寄存器编程为小于 Prefetchable Memory Base 寄存器的值。如果桥的二级侧上没有可预取的内存，并且没有内存映射的地址空间（请参阅[PCI-to-PCI-Bridge]规范），则桥将不会转发来自主总线的任何内存事务到二级总线，但会将所有内存事务从二级总线转发到主总线。

Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器的低 4 位是只读的，包含相同的值，并对桥是否支持 64 位地址进行编码。如果这四个位的值为 0h，则桥仅支持 32 位地址。如果这四个位的值均为 01h，则桥支持 64 位地址，并且 Prefetchable Base Upper 32 Bits 和 Prefetchable Limit Upper 32 Bits 寄存器分别保存其余的 64 位可预取基本地址和限制地址。所有其他编码都是保留的。

7.5.1.3.10 Prefetchable Base Upper 32 Bits/Prefetchable Limit Upper 32 Bits Registers (Offset 28h/2Ch)

Prefetchable Base Upper 32 Bits 和 Prefetchable Limit Upper 32 Bits 寄存器是对 Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器的可选扩展。

如果 Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器指示支持 32 位寻址，则 Prefetchable Base Upper 32 Bits 和 Prefetchable Limit Upper 32 Bits 寄存器均被实现为只读寄存器，读取时返回零。如果 Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器指示支持 64 位寻址，则 Prefetchable Base Upper 32 Bits 和 Prefetchable Limit Upper 32 Bits 寄存器被实现为可读/写寄存器。如果这些寄存器被实现为读/写寄存器，则必须由配置软件对其进行初始化，因此未指定默认状态。

如果支持 64 位可预取的内存地址范围，则 Prefetchable Base Upper 32 Bits 和 Prefetchable Limit Upper 32 Bits 寄存器指定 64 位基址和限制地址的高 32 位，对应于地址[63:32]（有关更多详细信息，请参见[PCI-to-PCI-Bridge]规范）。

7.5.1.3.11 I/O Base Upper 16 Bits/I/O Limit Upper 16 Bits Registers (Offset 30h/32h)

I/O Base Upper 16 Bits 和 I/O Limit Upper 16 Bits 寄存器是对 I/O Base 和 I/O Limit 寄存器的可选扩展。如果 I/O Base 和 I/O Limit 寄存器指示支持 16 位 I/O 地址解码，则 I/O Base Upper 16 Bits 和 I/O Limit Upper 16 Bits 寄存器被实现为只读寄存器，读取时返回零。

如果 I/O Base 和 I/O Limit 寄存器指示支持 32 位 I/O 寻址，则必须由配置软件初始化 I/O Base Upper 16 Bits 和 I/O Limit Upper 16 Bits 寄存器。

如果支持 32 位 I/O 地址解码，则 I/O Base Upper 16 Bits 和 I/O Limit Upper 16 Bits 寄存器指定 32 位的高 16 位，对应于地址[31:16]基本地址和限制地址（有关更多详细信息，请参见[PCI-to-PCI-Bridge]规范）。

7.5.1.3.12 Expansion ROM Base Address Register (Offset 38h)

此寄存器的意义与 Section 7.5.1.2.4 中 Type 0 Configuration Space Header 定义的 Expansion ROM Base Address Register 的一致。

7.5.1.3.13 Bridge Control Register (Offset 3Eh)

Bridge Control Register提供了对Command Register的扩展,这些扩展只对Type 1 Configuration Space Header的Function有效。Bridge Control Register 为二级接口提供了许多与 Command Register 相同的控制主接口的操作。有些位会影响桥两个接口的操作。

Table 7-12 定义了 Bridge Control 寄存器， Figure 7-16 描述了寄存器布局。对于 PCI Express to PCI/PCI-X Bridge， 请参阅[PCIe-to-PCI-PCI-X-Bridge]规范以了解此寄存器的要求。

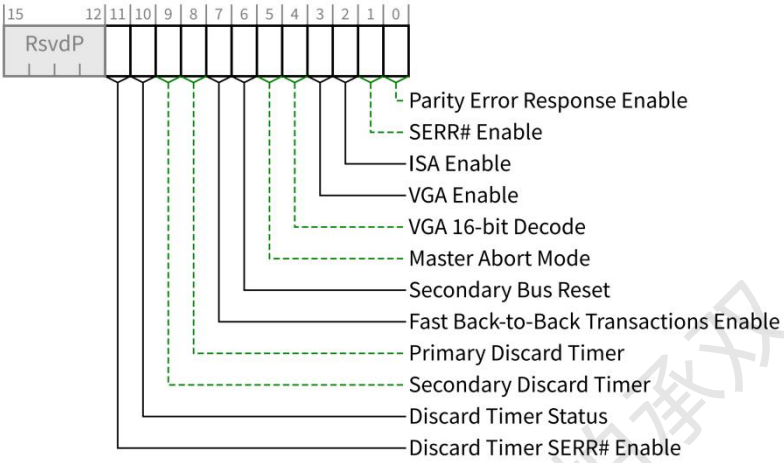

Figure 7-16 Bridge Control Register

Bit Location	Register Description	Attributes
0	**Parity Error Response Enable**- see Section 7.5.1.1.14。 该位控制 Secondary Status 寄存器中的 Master Data Parity Error 字段关于中毒 TLP 的记录。 默认值为 0b。	RW
1	**SERR# Enable** - see Section 7.5.1.1.14。 该位控制 ERR_COR、ERR_NONFATAL 和 ERR_FATAL 从二级接口往主接口的转发。 默认值为 0b。	RW
2	**ISA Enable** - 修改桥到 ISA I/O 地址的响应。这仅适用于由 I/O Base 和 I/O Limit 寄存器启用且位于 I/O 地址空间的前 64 KB（0000 0000h 至 0000 FFFFh）中的 I/O 地址。如果该位置 1，则桥将阻止从 I/O 事务从主接口到二级接口的任何转发，这些转发将寻址每个 1 KB 块中的最后 768 个字节。在相反的方向（从二级接口到主接口），如果 I/O 事务处理每个 1 KB 块中的最后 768 个字节，则将转发它们。 0b - 在 I/O Base 和 I/O Limit 寄存器定义的地址范围内向下游转发所有 I/O 地址。 1b - 在 PCI I/O 地址空间的前 64 KB（每个 1 KB 块的前 768 个字节）中，由 I/O Base 和 I/O Limit 寄存器定义的地址范围内转发前向 ISA 的 I/O 上游地址。 默认值为 0b。	RW

公众号：芯片业者相承双

3	**VGA Enable** - 通过桥修改对 VGA 兼容地址的响应。如果将 VGA Enable 置 1，则桥将对主接口上的以下访问进行肯解码并将其转发到二级接口（相反，将阻止这些地址从二级接口转发到主要接口）： ● Memory 访问地址为 000A0000h ~ 000BFFFFh。 ● I/O 地址空间的前 64 KB 中的 I/O 地址（地址[31:16]为 0000h），其中地址[9：0]的范围为 3B0h 至 3BBh 和 3C0h 至 3DFh（包括 ISA 地址别名，由 VGA 16 位解码设置决定）。 如果VGA Enable位置1，则这些访问的转发与I/O Base和I/O Limit寄存器，Memory Base和Memory Limit 寄存器以及 Prefetchable Memory Base 和 Prefetchable Memory Limit 寄存器定义的 I/O 地址范围和内存地址范围无关。（当 VGA Enable 位被置位时，这些访问的转发也独立于 ISA Enable 位的设置。这些访问的转发由 Command 寄存器的 I/O Space Enable 和 Memory Space Enable 位来限定。） 0b - 请勿将 VGA 兼容的内存和 I/O 地址从主接口转发到二级接口（上面定义的地址），除非已通过定义的 I/O 和内存地址范围启用它们。 1b - 将 VGA 兼容的内存和 I/O 地址（上面定义的地址）从主接口转发到二级接口（如果设置了 I/O Space Enable 和 Memory Space Enable），而与 I/O 和内存地址范围无关，并且独立于 ISA Enable 位。 不支持 VGA 的 Function 可以把该字段硬连线为 0b。 默认值为 0b。	RW
4	**VGA 16-bit Enable** - 仅当该寄存器的第 3 位（VGA Enable）也被置位时，该位才有意义，从而使桥能够进行 VGA I/O 解码和转发。 该位使系统配置软件可以为从主接口转发到二级接口的所有 VGA I/O 寄存器访问选择 10-bit 和 16-bit I/O 地址解码。 0b - 在 VGA I/O 访问上执行 10-bit 地址解码。 1b - 在 VGA I/O 访问上执行 16-bit 地址解码。 不支持 VGA 的 Function 可以把该字段硬连线为 0b。 默认值为 0b。	RW
5	**Master Abort Mode** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
6	**Secondary Bus Reset** -将该位置 1 会触发相应的 PCI Express 端口上的热复位。软件必须确保最小的复位持续时间（T_{rst}）。除非使用 Readiness Notifications 机制（参阅 Section 6.23）或相关 Function 的 Status 寄存器中的 Immediate Readiness 字段被置位，否则软件和系统必须遵守 6.6 节中定义的 first-access-following-reset 时序要求。 除非更改端口状态所需，否则不得更改端口配置寄存器。 默认值为 0b。	RW
7	**Fast Back-to-Back Transactions Enable** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于	RO

公众号·芯片业者柏承

	PCI Express，并且该位必须硬连线到 0b。	
8	**Primary Discard Timer** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
9	**Secondary Discard Timer** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
10	**Discard Timer Status** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO
11	**Discard Timer SERR# Enable** - 该位在[PCI-to-PCI-Bridge]规范中描述。它的功能不适用于 PCI Express，并且该位必须硬连线到 0b。	RO

7.5.2 PCI Power Management Capability Structure

Figure 7-17 给出了 PCI Power Management Capability structure 的组织。所有 PCI Express Function 都需要实现此结构。

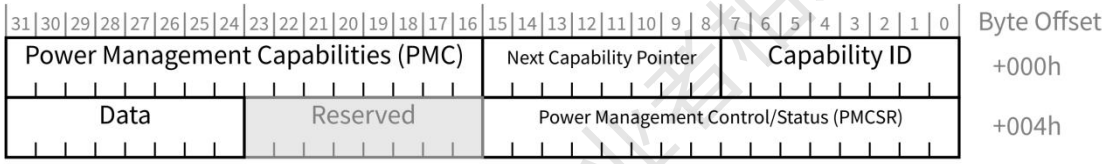

Figure 7-17 Power Management Capability Structure

注：对于 Type 0 和 Type 1 Function，8-bit Data（偏移量 07h）字段是可选的。

PCI Express device Function 需要支持 D0 和 D3 设备状态。由于 PCI Express 的 PME 消息传递的带内特性，Section 7.1 中所述的表示 PCI Express 端口的 PCI-PCI 桥结构需要指明对 PME 消息的传递能力。

但是，仅当 PCI-PCI 桥本身正在生成 PME 时，PME_Status 字段才表示 PCI Express 端口的 PCI-PCI 桥结构传递 PME 的能力。当桥可以传送 PME 消息 PCI-PCI 桥本身未在内部生成 PME，则 PME_Status 位置 0。

7.5.2.1 Power Management Capabilities Register (Offset 00h)

Figure 7-18 详细说明了 Power Management Capabilities Register 字段分配，Table 7-13 说明了此寄存器的规则。

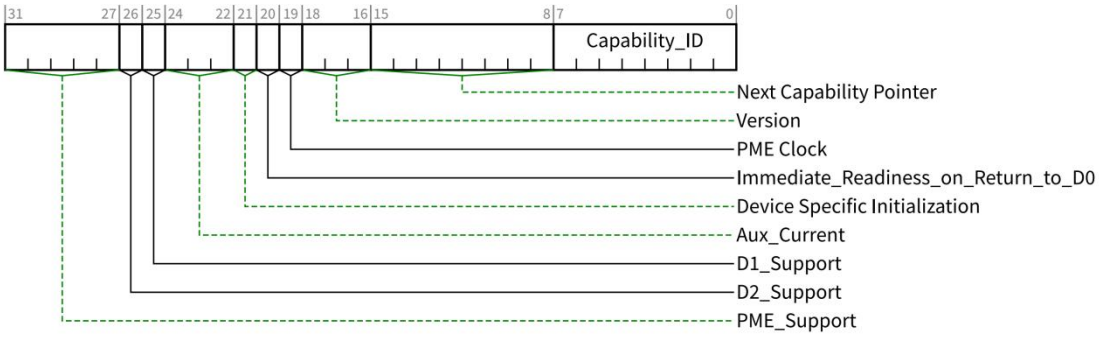

Figure 7-18 Power Management Capabilities Register

Bit Location	Register Description	Attributes
7:0	**Capability_ID** - Power Management Capability 的 Capability_ID 为 01h。每个 Function 都可能会实现一个 Power Management Capability。	RO
15:8	**Next Capability Pointer** - 指向该 Function 的 Capabilities List 中下一个 Capability 的偏移地址。如果没有下一个 Capability，此字段值为 00h。	RO
18:16	**Version** - 这个版本的规范中该值必须硬连线为 011b。	RO
19	**PME Clock** - 该位不适用于 PCI Express，必须硬连线为 0b。	RO
20	**Immediate_Readiness_on_Return_to_D0** - 如果该位为 1，可以保证该 Function 已准备就绪，可以在将其设置为 D0 后立即成功完成有效的访问。这些访问包括配置访问，如果功能返回到 $D0_{active}$，则它们还包括内存访问和 I/O 访问。 当该位为 1 时，在访问该 Function 时，软件无需遵循转换到 D0 后的延迟的所有要求，这包括但不限于 Section 5.9 中定义的 10 ms 延迟。 如何建立此保证超出了本文档的范围。 允许系统软件/固件提供取代此位提供的指示的机制，但是此类软件/固件机制不在本规范的范围之内。	RO
21	**Device Specific Initialization** - DSI 字段指示是否需要对此 Function 进行特殊初始化。 为 1 时表示在转换到 $D0_{uninitialized}$ 状态后，该 Function 需要特定于设备的初始化序列。	RO
24:22	**Aux_Current** - 此 3-bit 字段用于报告此 Functon 的 Vaux 辅助电流要求。 如果此 Functon 实现了 Data 寄存器，则该字段必须硬连线至 000b。 如果 PME_Support 为 0xxxxb，则此字段必须硬连线至 0000b。 如果 PME_Support 为 1xxxxb 且未实现 Data 寄存器的，则引用以下编码： Encoding Vaux Max. Current Required **111b** 375mA **110b** 320mA **101b** 270mA **100b** 220mA **011b** 160mA **010b** 100mA **001b** 55mA **000b** 0(self powered)	RO

公众号：芯片业老柏承双

25	**D1_Support** - 为 1 时表示此 Function 支持 D1 状态。	RO
26	**D2_Support** - 为 1 时表示此 Function 支持 D2 状态。	RO
31:27	**PME_Support** - 该 5-bit 字段指示此 Function 是否可以在当前电源状态下生成 PME 和/或转发 PME 消息。 任何位的值为 0b 表示此 Function 在该电源状态下无法发出 PME。 **bit(27) X XXX1b** PME can be generated from D0 **bit(28) X XX1Xb** PME can be generated from D1 **bit(29) X X1XXb** PME can be generated from D2 **bit(30) X 1XXXb** PME can be generated from $D3_{Hot}$ **bit(31) 1 XXXXb** PME can be generated from $D0_{Cold}$ Bit 31 代表一种特殊情况。设置该位的 Function 需要某种辅助电源。在设置此位之前，建议使用特定于实现的机制来验证电源是否可用。 必须为代表 RC/Switch 上的端口的 PCI-PCI Bridge 结构设置与支持的 D 状态相对应的每个位，以指示桥可以转发 PME 消息。仅当在主电源不可用时端口仍能够转发 PME 消息时，才必须设置 bit 31。	RO

7.5.2.2 Power Management Control/Status Register (Offset 04h)

该寄存器用于管理 PCI Function 的电源管理状态以及使能/监控 PME。

PME 上下文包括 PME_Status 和 PME_En 位的值，以及在 $D3_{Cold}$ 期间实现唤醒功能所需的特定于实现的状态（例如，识别出 LAN 上的唤醒数据包并生成 PME 消息），以及在转换到未初始化状态期间必须保留的任何其他特定于实现的状态。

如果 Function 支持在 $D3_{Cold}$ 下生成 PME，则其 PME 上下文不受 Reset 的影响。这是因为 Function 的 PME 功能本身可能是导致转换回 D0 的唤醒事件的原因。因此，必须为系统软件处理保留 PME 上下文。

如果 $D3_{Cold}$ 不支持 PME 生成，那么所有 PME 上下文都用 Reset 初始化。

Figure 7-19 详细说明了 Power Management Control/Status 寄存器各字段的分配，Table 7-14 描述了该寄存器的要求。

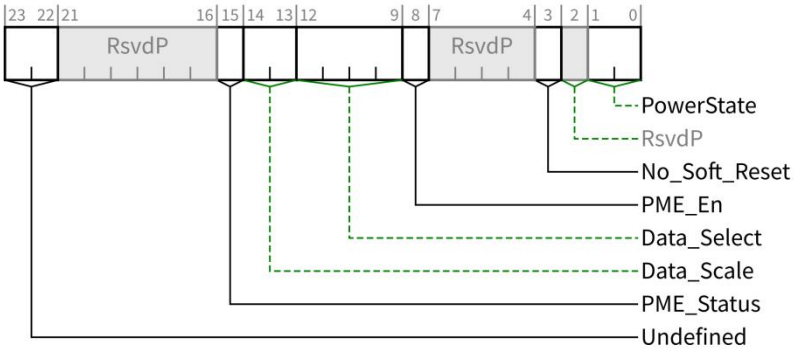

Figure 7-19 Power Management Control/Status Register

公众号：芯片业者自承双

Bit Location	Register Description	Attributes
1:0	**PowerState** - 这 2 比特用于确定 Function 的当前电源状态，并将 Function 设置为新的电源状态。字段值的定义如下。 **00b** D0 **01b** D1 **10b** D2 **11b** $D3_{Hot}$ 如果软件试图将不受支持的可选状态写入此字段，则写入操作必须正常完成；但是，数据将被丢弃，并且不会发生状态更改。 默认置为 00b。	RW
3	**No_Soft_Reset** - 此位表示写入 PowerState 字段以将 Function 从 $D3_{Hot}$ 转换为 D0 后 Function 的状态。 为 1，此转换将保留内部 Function 状态。该 Function 处于 $D0_{active}$ 状态并且不需要额外的软件干预。 为 0，表示此转换导致了一个未定义的内部 Function 状态。 不管该位如何，如果支持并启用 PME，通过基本复位从 $D3_{Hot}$ 转换到 D0 的 Function 将返回到 $D0_{uninitialized}$，并且只保留 PME 上下文。	RO
8	**PME_En** - 设置为 1 则允许 Function 生成 PME。清 0 时，不允许 Function 生成 PME。 如果 PME_Support 为 1xxxxb（指允许从 $D3_{Cold}$ 生成 PME），或者该 Function 消耗了辅助电源并且有可用的辅助电源，则该位为 RWS，并且该位不会被常规复位或 FLR 修改。 如果 PME_Support 为 0xxxxb，则此字段不是粘性字段（即为 RW）。 如果 PME_Support 为 00000b，则允许将该位硬接线为 0b。	RW/RWS
12:9	**Data_Select** - 该 4bit 字段用于选择要通过 Data 寄存器和 Data_Scale 字段报告的数据。 如果未实现 Data 寄存器，则该字段必须硬连线至 0000b。 参阅 Section 7.5.2.3 获取更详细的信息。 默认值为 0000b。	RW
14:13	**Data_Scale** - 该字段指示在解释 Data 寄存器的值时要使用的比例因子。该字段的值和含义将根据 Data_Select 字段选择的数据值而有所不同。 该字段是 Data 寄存器的必需组成部分，如果实现了 Data 寄存器，则必须实现该字段。 如果未实现 Data 寄存器，则该字段必须硬连线至 00b。 参阅 Section 7.5.2.3 获取更详细的信息。	RO

公众号·芯片业者相承及

15	**PME_Status** - 当 Function 正常生成 PME 信令时，该位置 1。 该位的值不受 PME_En 的值影响。 如果 Power Management Capabilities 寄存器的 PME_Support 字段的 bit31 被清 0，则允许将该位硬连线到 0b。 当辅助电源可用时，消耗辅助电源的 Function 必须保留此粘滞寄存器的值。在此类 Function 中，常规复位或 FLR 不会修改该寄存器的值。	RW1CS
23:22	**Undefined** - 这些位在先前的规范中定义。它们应该被软件忽略。	RO

7.5.2.3 Data (Offset 07h)

Data 寄存器是一个可选的 8 位只读寄存器，它为 Function 提供了一种机制，用于报告与状态有关的功耗。

如果实现了 Data 寄存器，则还必须实现 Data_Select 和 Data_Scale 字段。如果未实现该寄存器，则必须将其硬连线至 00h。

软件可以通过将不同的值写入 Data_Select 字段中，在 Data Register 和/或 Data_Scale 字段中查找非零返回数据来检查 Data 寄存器的存在。任何非零的 Data 寄存器/ Data_Select 读取数据都表明已实现 Data 寄存器组合。

Bit Location	Register Description	Attributes
7:0	**Data** - 该寄存器用于报告 Data_Select 字段请求的状态相关数据。该寄存器的值由 Data_Scale 字段报告的值进行缩放。	RO

通过将适当的值写入 PMCSR 中的 Data_Select 字段，然后读取 Data_Scale 字段和 Data 寄存器，来使用 Data 寄存器。然后，将从 Data 中读取的二进制值乘以 Data_Scale 所指示的比例因子即可得出所需测量值。Table 7-16 列出了定义的测量值以及如何解释每个寄存器的值。

Table 7-16 Power Consumption/Dissipation Reporting

Value in Data_Select	Data Reported	Data_Scale Interpretation	Units/ Accuracy
0	D0 Power Consumed	0 = Unknown 1 = 0.1x 2 = 0.01x 3 = 0.001x	Watts
1	D1 Power Consumed		
2	D2 Power Consumed		
3	D3 Power Consumed		
4	D0 Power Dissipated		
5	D1 Power Dissipated		
6	D2 Power Dissipated		
7	D3 Power Dissipated		
8	Common logic power consumption (Multi-Function Devices, Function 0 only) **Function 0 of a Multi-Function Device:** Power consumption that is not associated with a specific Function. **All other Functions:** Reserved		
9-15	Reserved	Reserved	TBD

公众号·芯片业者相关权

上面定义的“Power Consumed”值必须包括电源平面通过连接器引脚消耗的所有功率。 如果附加卡为外部设备供电，则还必须包括该电源。它不得包含来自电池或外部电源的任何电源。此信息对于电源或电池的管理很有用。

“Power Dissipated”值必须提供将散发到计算机底座上的热量。这不包括传递给外部设备的任何电源，但必须包括来自电池或外部电源并耗散在计算机底座上的任何电源。此信息对于细粒度的热管理很有用。

建议 Multi-Function Device 在每个相应 Function 的配置空间中报告每个 Function 消耗的功率。在 Multi-Function Device 中，一旦将 Function 0 的 Power Management Control/Status 寄存器的 Data_Select 字段编程为 1000b，就会通过 Data 寄存器在 Function 0 的配置空间中报告多个 Function 所共有的电路的功耗。 对于 Multi-Function Device，设备的功耗是该值的总和，对于设备的每个 Function，则是与该 Function 的当前电源状态相关联的报告值。

实现功率报告的多个组件附加卡（即，switch/bridge 后面的多个组件）必须让 switch/bridge 报告其自身使用的功率。附加卡上每个组件的每个 Function 负责报告该 Function 消耗的功率。

7.5.3 PCI Express Capability Structure

PCI Express 在 PCI 兼容配置空间（前 256 个字节）中定义了此功能结构，如 Figure 7-3 所示。该结构允许识别 PCI Express device Function，并表示支持新的 PCI Express feature。 PCI Express device Function 需要实现 PCI Express Capability Structure。 Capability structure 是一种用于启用 PCI 软件透明功能的机制，该功能需要对旧版操作系统的支持。除了识别 PCI Express device Function 之外，PCI Express Capability Structure 还用于提供对 PCI Express 特定的控制/状态寄存器和相关电源管理增强功能的访问。

Figure 7-21 详细介绍了 PCI Express Capability Structure 中的寄存器字段分配。

所有的 PCI Express device Function 都要实现 PCI Express Capabilities、Device Capabilities、Device Status 和 Device Control 寄存器。所有的 PCI Express device Function 都要实现需要相应功能的 Device Capabilities 2、 Device Status 2 和 Device Control 2 寄存器。对于不需要实现 Device Capabilities 2、 Device Status 2 和 Device Control 2 寄存器的 device Function，这些寄存器全部硬连线为 0。

所有 Root Port、Switch Port、Bridge 和 Endpoint（并非 RCiEP）都需要实现 Link Capabilities、Link Status 和 Link Control 寄存器。对于不需要实现 Link Capabilities、Link Status 和 Link Control 寄存器的 Function，这些寄存器全部硬连线为 0。所有 Root Port、Switch Port、 Bridge 和 Endpoint（并非 RCiEP）都要实现需要相应功能的 Link Capabilities 2、 Link Status 2 和 Link Control 2 寄存器。对于不需要实现 Link Capabilities 2、 Link Status 2 和 Link Control 2 寄存器的 Function，这些寄存器全部硬连线为 0。

某些 Switch Downstream 和 Root Port 需要实现 Slot Capabilities、Slot Status 和 Slot Control 寄存器。如果 Slot Implemented 字段设置为 1b，则需要实现 Slot Capabilities 寄存器（Section 7.5.3.2）。如果 Slot Implemented 字段设置为 1b 或者 Data Link Layer Link Active Reporting Capable 字段（Section 7.5.3.6）设置为 1b，则需要实现 Slot Status 和 Slot Control 寄存器。Switch Downstream 和 Root Port 可以在不需要的情况下实现这些寄存器，这种情况下这些寄存器的行为是未定义的。Section 7.5.3.9、Section 7.5.3.10 和 Section 7.5.3.11 给出了更详细的描述。对于不需要实现 Slot Capabilities、Slot Status 和 Slot Control 寄存器的 Function，除 Slot Status 的 Presence Detect State 字段硬连线为 1b 之外，其他的字段都硬连线为 0。Switch Downstream 和 Root Port 在需要相应功能的时候可以实现对应的 Slot Capabilities 2、Slot Status 2 和 Slot Control 2 寄存器。对于不需要实现 Slot Capabilities 2、Slot Status 2 和 Slot Control 2 寄存器的 Function，这些寄存器全部硬连线为 0。

Root Port 和 Root Complex Event Collector 必须实现 Root Capabilities、 Root Status 和 Root Control 寄存器。对于没有实现 Root Capabilities、 Root Status 和 Root Control 寄存器的 Function，这些字段全部硬连线为 0。

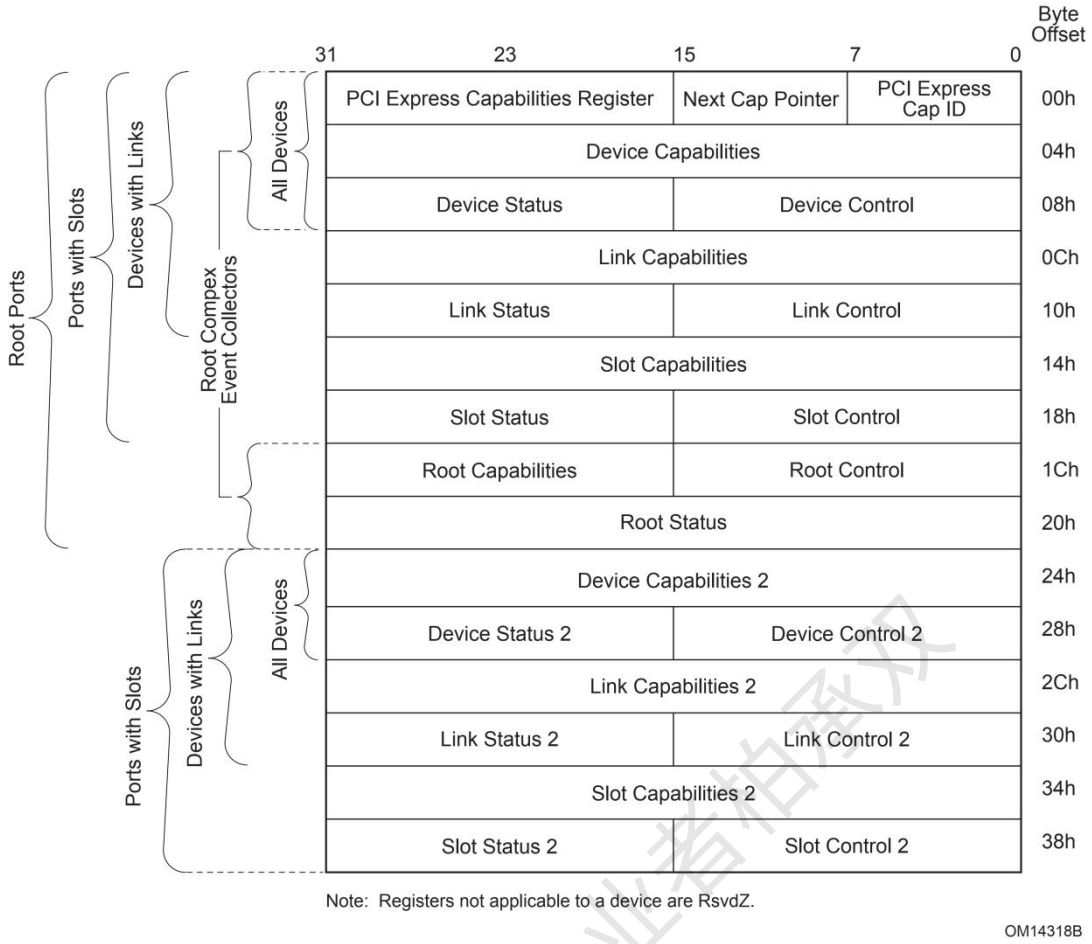

Figure 7-21 PCI Express Capability Structure

7.5.3.1 PCI Express Capability List Register (Offset 00h)

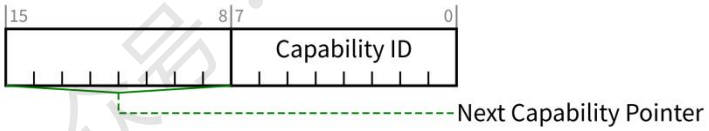

Figure 7-22 PCI Express Capability List Register

Bit Location	Register Description	Attributes
7:0	**Capability_ID** -PCI Express Capability 的 Capability_ID 为 10h。	RO
15:8	**Next Capability Pointer** - 指向该 Function 的 Capabilities List 中下一个 Capability 的偏移地址。如果没有下一个 Capability，此字段值为 00h。	RO

公众号·芯片业者拍成双

7.5.3.2 PCI Express Capabilities Register (Offset 02h)

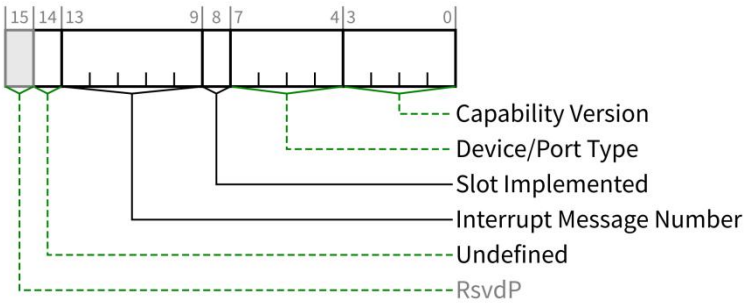

Figure 7-23 PCI Express Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Capability Version** - 表示 PCI-SIG 定义的 PCI Express Capability structure 的版本号。 更改 PCI Express Capability structure 的规范版本来增加此字段。对 PCI Express Capability structure 的所有更改必须与软件兼容。软件必须检查功能版本号是否大于或等于编写软件时定义的最高版本号，因为报告任何此类功能版本号的 Function 将包含与该软件兼容的 PCI Express Capability structure。 此版本规范中的 Function 中的该字段必须为 2h。	RO
7:4	**Device/Port Type** - 表示此 PCI Express Function 的类型。注意，多功能设备中的不同 Function 通常有不同的类型。 实现 Type 0 PCI Configuration Space header 的 Function 此字段的值可以是： **0000b** PCI Express Endpoint **0001b** Legacy PCI Express Endpoint **1001b** RCiEP **1010b** Root Complex Event Collector 实现 Type 1 PCI Configuration Space header 的 Function 此字段的值可以是： **0100b** Root Port of PCI Express RC **0101b** Upstream Port of PCI Express Switch **0110b** Downstream Port of PCI Express Switch **0111b** PCI Express to PCI/PCI-X Bridge **1000b** PCI/PCI-X to PCI Express Bridge 所有其他编码值都是保留的。	RO
8	**Slot Implemented** - 置为 1 时，此位指示与此端口关联的链路已连接到插槽（相对于连接到系统集成设备或被禁用而言）。	HwInit

公众号·芯片业者程序员

	该位对下游端口有效。上游端口未定义此位。	
13:9	**Interrupt Message Number -** 该字段指示哪个 MSI/MSI-X 向量用于与此 Capability structure 的状态字段相关的生成的中断消息。 对于 MSI，此字段中的值指示基本消息数据和所生成的中断消息之间的偏移量。 硬件需要更新此字段的值，以便当软件写 MSI 的 Message Control Register 中的 Multiple Message Enable 字段时，分配给该 Function 的 MSI 消息的数目发生更改时，它是正确的。 对于 MSI-X，此字段中的值指示使用哪个 MSI-X 表条目来生成中断消息。即使 Function 实现了 32 个以上的条目，该条目也必须是前 32 个条目之一。对于给定的 MSI-X 实现，该条目必须保持不变。 如果同时实现了 MSI 和 MSI-X，则允许它们使用不同的向量，尽管允许软件一次仅启用一种机制。如果启用了 MSI-X，则此字段中的值必须指示 MSI-X 的向量。 如果启用了 MSI 或都不启用，则此字段中的值必须指示 MSI 的向量。如果软件同时启用 MSI 和 MSI-X，则此字段中的值为未定义。	RO
14	**Undefined -** 从该位读取的值是不确定的。在本规范的先前版本中，该位用于指示对 TCS 路由的支持。系统软件应忽略从该位读取的值。允许系统软件向该位写入任何值。	RO

7.5.3.3 Device Capabilities Register (Offset 04h)

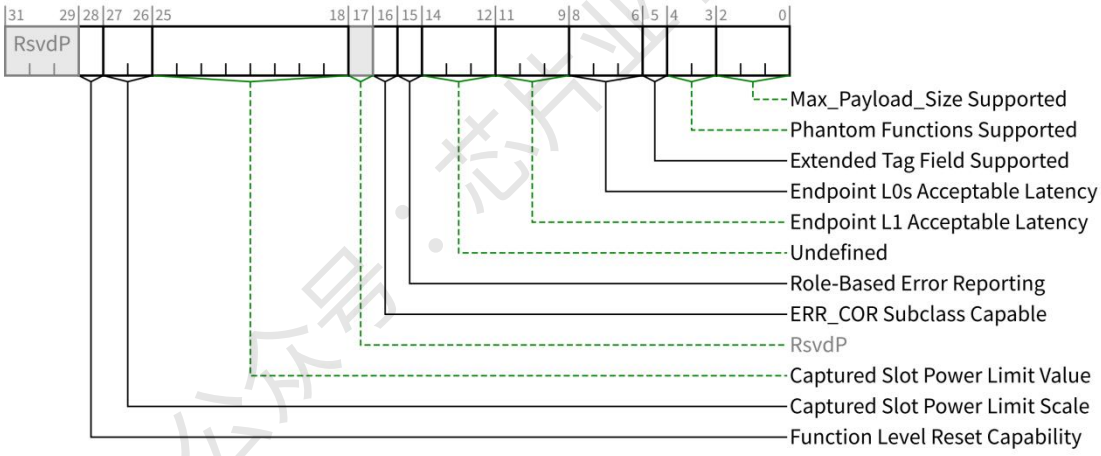

Figure 7-24 Device Capabilities Register

Bit Location	Register Description	Attributes
2:0	**Max_Payload_Size Supported** - 该字段指示此 Function 可以支持 TLP 的最大有效载荷大小。 **000b** 128 bytes max payload size **001b** 256 bytes max payload size **010b** 512 bytes max payload size **011b** 1024 bytes max payload size	RO

公众号·芯片业者相承及

100b 2048 bytes max payload size

101b 4096 bytes max payload size

110b Reserved

111b Reserved

允许 Multi-Function Device 的各个 Function 为此字段报告不同的值。

4:3 **Phantom Functions Supported** - 该字段表示支持使用无人认领的 Function Number 来扩展允许的 outstanding transaction 的数目。此数目是用无人认领的 Function Number（称为 Phantom Function）与 Tag 标识符进行逻辑组合出来的（有关 Tag Extension 的说明，请参见 Section 2.2.6.2）。 RO

对于 ARI 设备中的每个 Function，必须将 Phantom Functions Supported 字段设置为 00b。该字段说明的其余部分仅适用于 non-ARI Multi-Function Device。

该字段指示在逻辑上与 Tag 标识符组合的 Requester ID 的 Function Number 部分的最高有效位数。

定义的编码值为：

00b 没有多余 Function Number 比特用于 Phantom Function。Multi-Function Device 最多可实现 8 个独立 Function。

01b Requester ID 中 Function Number 的最高有效位用于 Phantom Function；允许 Multi-Function Device 实现 Function 0-3。允许 Function 0、1、2 和 3 分别使用 Function Number 4、5、6 和 7 用于 Phantom Function。

10b Requester ID 中 Function Number 的最高两位有效位用于 Phantom Function。允许 Multi-Function Device 实现 Function 0-1。允许 Function 0 将 Function Number 2、4 和 6 用于 Phantom Function。允许 Function 1 将 Function Number 3、5 和 7 用于 Phantom Function。

11b Requester ID 中所有 3 比特用于 Phantom Function。设备只有 Function 0，该 Function 允许将所有其他 Function Number 用于 Phantom Function。

请注意，在允许 Function 使用 Requester ID 中的 Function Number 字段用于 Phantom Function 之前，必须通过 Device Control 寄存器中的 Phantom Functions Enable 字段启用对 Phantom Function 的支持。

5 **Extended Tag Field Supported** - 该位与 Device Capabilities 2 寄存器中支持的 10-Bit Tag Requester Supported 字段结合在一起，指示作为请求者的 Tag 字段的最大支持大小。如果设置了 10-Bit Tag Requester Supported 字段，则必须将该位置 1。 RO

定义的编码为：

0b 5-bit Tag field supported

1b 8-bit Tag field supported

请注意，在请求者可以生成 8-Bit Tag 之前，必须把请求者的 Function 的 Device Control 寄存器中

公众号：芯片业者相承双

的 Extended Tag Field Enable 字段置 1 来启用 8-Bit Tag 字段的生成。有关启用 10-Bit Tag 的交互过程，请参见 Section 2.2.6.2。

8:6 **Endpoint L0s Acceptable Latency** - 此字段指示从 L0s 状态过渡到 L0 状态时，Endpoint 可以接受的总延迟。它本质上是对 Endpoint 内部缓冲的间接度量。 RO

电源管理软件使用此报告的 L0s Acceptable Latency 值与所有组件（包括从此 Endpoint 到 RC Root Port 的数据路径）报告的 L0s 退出延迟进行比较，以确定是否可以进入 ASPM L0s 而不损失性能。

定义的编码值为：

000b Maximum of 64 ns

001b Maximum of 128ns

010b Maximum of 256ns

011b Maximum of 512ns

100b Maximum of 1 μs

101b Maximum of 2 μs

110b Maximum of 4 μs

111b No Limit

对于 Endpoint 以外的 Function，此字段为保留字段，必须硬连线至 000b。

11:9 **Endpoint L1 Acceptable Latency** - 此字段指示从 L1 状态过渡到 L0 状态时，Endpoint 可以接受的总延迟。它本质上是对 Endpoint 内部缓冲的间接度量。 RO

电源管理软件使用此报告的 L1 Acceptable Latency 值与所有组件（包括从此 Endpoint 到 RC Root Port 的数据路径）报告的 L1 退出延迟进行比较，以确定是否可以进入 ASPM L1 而不损失性能。

定义的编码值为：

000b Maximum of 1 us

001b Maximum of 2 us

010b Maximum of 4 us

011b Maximum of 8 us

100b Maximum of 16 us

101b Maximum of 32 us

110b Maximum of 64 us

111b No Limit

对于 Endpoint 以外的 Function，此字段为保留字段，必须硬连线至 000b。

14:12 **Undefined** - 从这些位读取的值是不确定的。在本规范的先前版本中，该位用于指示在适配器上实现了 Attention Button\Attention Indicator 或 Power Indicator，并由适配器上的组件进行电气控 RO

制。系统软件必须忽略从该位读取的值。允许系统软件向该位写入任何值。

15	**Role-Based Error Reporting** - 设置为 1 时，该位表示该 Function 实现了最初在[PCIe-1.0a]的 Error Reporting ECN 中定义后来又集成到[PCIe-1.1]中的功能。必须由符合 ECN、[PCIe-1.1]或后续 PCIe 版本的所有 Function 将该位置 1。	RO
16	**ERR_COR Subclass Capable** - 为 1 时，该位指示该 Function 支持 ERR_COR 消息中的 ERR_COR Subclass 字段，从而可以区分不同的子类。参见 Section 2.2.8.3。 实现 System Firmware Intermediary（SFI）capability 的下游端口必须将该位置 1。强烈建议实现 Downstream Port Containment（DPC）的下游端口将该位置 1。	RO
25:18	**Captured Slot Power Limit Value**(Upstream Ports only) - 与 Captured Slot Power Limit Scale 值结合使用，指定适配器可用的功率上限。 功率限制（以 Watts 为单位）是通过将该字段中的值乘以 Captured Slot Power Limit Scale 字段中的值来计算的，除非 Captured Slot Power Limit Scale 字段等于 00b（1.0x）并且 Captured Slot Power Limit Value 超过 EFh ，在这种情况下会使用其他编码（请参见 Section 7.5.3.9）。 该值由 Set_Slot_Power_Limit 消息设置或硬连线至 00h（请参见 Section 6.9）。默认值为 00h。	RO
27:26	**Captured Slot Power Limit Scale**(Upstream Ports only) - 指定用于插槽功率限制值的比例因子。 该值的范围为： **00b** 1.0 x **01b** 0.1 x **10b** 0.01 x **11b** 0.001 x 该值由 Set_Slot_Power_Limit 消息设置或硬连线至 00b（请参见 Section 6.9）。默认值为 00b。	RO
28	**Function Level Reset Capability** - 值为 1b 表示 Function 支持 Section 6.6.2 中所述的可选实现的 FLR 机制。 该位仅适用于 Endpoint。对于所有其他 Function 类型，该位必须硬连线至 0b。	RO

公众号：芯片业者相承双

7.5.3.4 Device Control Register (Offset 08h)

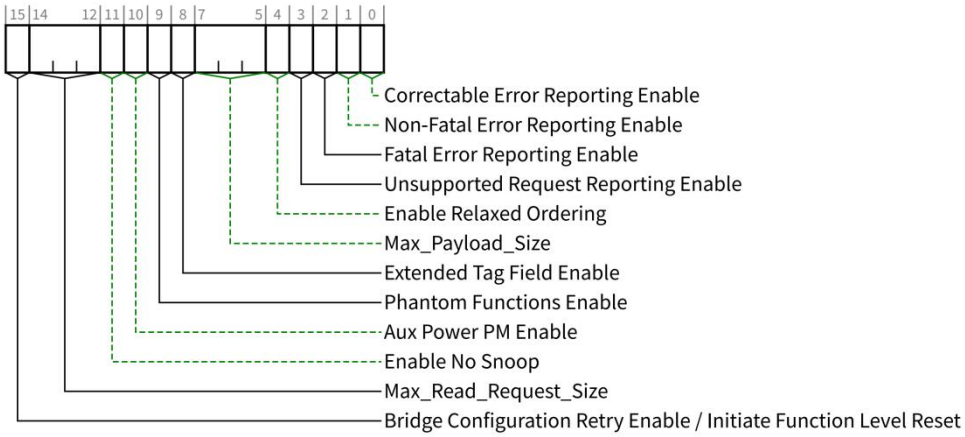

Figure 7-25 Device Control Register

Bit Location	Register Description	Attributes
0	**Correctable Error Reporting Enable** - 该位与其他位一起，用于控制 ERR_COR 消息的发送（有关详细信息，请参见 Section 6.2.5、6.2.6 和 6.2.10.2）。对于 Multi-Function Device，此位从相应 Function 的角度控制每个 Function 的错误报告。 对于根端口，可纠正错误的报告是根内部的。不会生成外部 ERR_COR 消息。 与 Root Complex Event Collector 无关的 RCiEP 被允许将该位硬接线为 0b。 默认值为 0b。	RW
1	**Non-Fatal Error Reporting Enable** - 该位与其他位一起，用于控制 ERR_FATAL 消息的发送（有关详细信息，请参见 Section 6.2.5 和 6.2.6）。对于 Multi-Function Device，此位从相应 Function 的角度控制每个 Function 的错误报告。 对于根端口，致命错误的报告是根内部的。不会生成外部 ERR_FATAL 消息。 与 Root Complex Event Collector 无关的 RCiEP 被允许将该位硬接线为 0b。 默认值为 0b。	RW
2	**Fatal Error Reporting Enable** - 该位与其他位一起，用于控制 ERR_NONFATAL 消息的发送（有关详细信息，请参见 Section 6.2.5 和 6.2.6）。对于 Multi-Function Device，此位从相应 Function 的角度控制每个 Function 的错误报告。 对于根端口，非致命错误的报告是根内部的。不会生成外部 ERR_NONFATAL 消息。 与 Root Complex Event Collector 无关的 RCiEP 被允许将该位硬接线为 0b。 默认值为 0b。	RW
3	**Unsupported Request Reporting Enable** - 该位与其他位一起，用于控制 Unsupported Request Error 信令的错误消息的发送（有关详细信息，请参见 Section 6.2.5 和 6.2.6）。对于 Multi-Function Device，	RW

公众号·芯片业者相爱双

此位从相应 Function 的角度控制每个 Function 的错误报告。

与 Root Complex Event Collector 无关的 RCiEP 被允许将该位硬接线为 0b。

默认值为 0b。

4 **Enable Relaxed Ordering** - 如果该位置 1，则允许该 Function 在其发送的不需要强写顺序的事务的 Attributes 字段中设置 Relaxed Ordering 字段（请参阅 Section 2.2.6.4 和 Section 2.4）。 RW

如果某个 Function 不支持在其发送的 TLP 中设置 Relaxed Ordering 属性，则该 Function 可以将该位硬连线为 0b。

如果没有硬连线为 0b，则默认值为 1b。

7:5 **Max_Payload_Size** - 该字段设置 Function 的最大 TLP 有效负载大小。作为接收器，Function 能处理的 TLP 的最大值为此设置值（指收到的 MemWr、IOWr 和 CplD 携带的数据量不得超过此设置值）。作为发送器，Function 不得生成超过此设置值的 TLP（指 MemWr、IOWr 和 CplD 携带的数据量不得超过此设置值）。Device Capabilities Register 中的 Max_Payload_Size Supported 字段指示了允许的编程值（Section 7.5.3.3）。 RW

- **000b** 128 bytes max payload size
- **001b** 256 bytes max payload size
- **010b** 512 bytes max payload size
- **011b** 1024 bytes max payload size
- **100b** 2048 bytes max payload size
- **101b** 4096 bytes max payload size
- **110b** Reserved
- **111b** Reserved

允许仅支持最大 128 字节有效负载大小的 Function 可以将此字段硬连线到 000b。

允许 Multi-Function Device 的各个 Function 为此字段报告不同的值。有关重要指导，请参阅 Section 2.2.2。

对于 ARI 设备，Max_Payload_Size 仅由 Function 0 中的设置确定。其他 Function 该字段可以被写入，但是组件会忽略它们。

默认值为 000b。

8 **Extended Tag Field Enable** - 该位与 Device Control 2 寄存器中的 10-Bit Tag Requester Enable 字段一起，确定允许请求者使用多少比特的 Tag 字段。 RW

当 10-Bit Tag Requester Enable 被清 0 时，以下情况适用。如果 Extended Tag Field Enable 字段被置 1，则允许作为请求者的 Function 使用 8-Bit Tag 字段。如果该位为 0b，则该 Function 仅限于使用 5-Bit Tag 字段。

有关设置 10-Bit Tag Requester Enable 所需的行为，请参阅 Section 2.2.6.2。

公众号：芯片业者相承双

如果软件在 Function 有 outstanding Non-Posted Request 时更改 Extended Tag Field Enable 的值，则结果未定义。

未实现此 capability 的 Function 把此字段硬连线为 0b。

默认值与实现相关。

9 **Phantom Functions Enable** - 该位与 Device Control 2 寄存器中的 10-Bit Tag Requester Enable 字段结合，用于确定请求者可以生成多少个 outstanding Non-Posted Request。完整细节见 Section 2.2.6.2。 RW

设置时，此位使 Function 能够将无人认领的 Function 用作 Phantom Function，以扩展 outstanding transaction identifier 的数量。如果此位清 0，则 Function 不允许使用 Phantom Function。

当 Function 有 outstanding Non-Posted Request 时，软件不应更改此位的值；否则，结果未定义。

未实现此 capability 的 Function 把此字段硬连线为 0b。

默认值为 0b。

10 **Aux Power PM Enable** - 当设置为 1 时，Function 会提取与 PME Aux power 无关的辅助电源。在旧版操作系统上需要辅助电源的 Function 应继续指示 PME Aux 电源要求。根据 Power Management Capabilities 寄存器（PMC）的 Aux_Current 字段中的请求分配辅助电源，与 Power Management Control/Status 寄存器（PMCSR）中的 PME_En 位无关（请参阅 Chapter 5）。对于 Multi-Function Device，如果至少一个 Function 设置了该位，就允许组件提取辅助电源。 RWS

注意：当辅助电源可用时，消耗辅助电源的 Function 必须保留此粘滞寄存器的值。在此类 Function 中，常规复位不会修改该位。

未实现此 capability 的 Function 把此字段硬连线为 0b。

11 **Enable No Snoop** - 如果该位置为 1，则该 Function 被允许在其发起的不需要硬件强制缓存一致性的事务的 Attributes 中设置 No Snoop 字段（请参见 Section 2.2.6.5）。请注意，将此位设置为 1b 不会导致 Function 在它启动的所有事务上都设置 No Snoop 属性。即使设置了此位，也只能允许某个 Function 在其事务上设置 No Snoop 属性，前提是它可以确保事务的地址未存储在系统的任何高速缓存中。 RW

如果某个 Function 永远不会在其发起的事务中设置 No Snoop 属性，则可以将该位硬连线为 0b。

默认值为 1b。

14:12 **Max_Read_Request_Size** - 该字段表示 Function 不得生成大小超过此设置值的读请求（指 MemRd 和 IORd 的 Length 字段值不超过此设置值）。定义的编码为： RW

000b	128 bytes max Read Request size
001b	256 bytes max Read Request size
010b	512 bytes max Read Request size
011b	1024 bytes max Read Request size
100b	2048 bytes max Read Request size

公众号：芯片业老柏承双

101b 4096 bytes max Read Request size

110b Reserved

111b Reserved

如果 Function 不生成大于 128 字节的读请求或者 Function 没有自行生成读请求的行为，则此字段实现为 000b 的只读字段。

默认值为 000b。

15 **Bridge Configuration Retry Enable/Initiate Function Level Reset** - 根据不同的 Function 类型，此位具有不同的含义：

- **PCI Express to PCI/PCI-X Bridges**：

Bridge Configuration Retry Enable - 置为 1 时，此位使 PCI Express to PCI/PCI-X Bridge 可以返回 Configuration Request Retry Status (CRS)，以响应以桥下面的设备为目标的配置请求。有关更多详细信息，请参阅[PCIe-to-PCI-PCI-X-Bridge]规范。

默认值为 0b。

- **Endpoints with Function Level Reset Capability set to 1b**：

Initiate Function Level Reset - 写 1b 将对该 Function 发起一个 FLR。软件从该位读取的值始终为 0b。

- **All others**：

Reservd - 必须硬连线为 0b。

PCI Express to PCI/PCI-X Bridges: RW
FLR Capable Endpoints: RW
All others: RsvdP

Software UR Reporting Compatibility with 1.0a Devices

对于 1.0a 设备功能，如果设置了 Unsupported Request Reporting Enable 字段，则当检测到 UR 错误时，该 Function 作为完成者运行时将发送不可纠正的错误消息（如果启用）。在将不可纠正的错误消息作为系统错误处理的平台上，这将破坏 PC 兼容的配置空间探测，因此在此类平台上的软件/固件可能需要避免将 Unsupported Request Reporting Enable 设置为 1。

注：在这种情况下，“1.0a 设备”指未实现 Role-Based Error Reporting 的设备。

使用实现 Role-Based Error Reporting 的设备功能，假设 UR 的严重性保留为默认值，则设置 Unsupported Request Reporting Enable 将不会干扰与 PC 兼容的配置空间探测。但是，设置 Unsupported Request Reporting Enable 将使该 Function 能够报告通过 posted Request 检测到的 UR 错误，从而有助于避免这种情况导致潜在的静默数据损坏。

注：对于 Role-Based Error Reporting 的设备，将 Command 寄存器中的 SERR # Enable 位置 1 也会隐式启用 UR 报告。

在需要鲁棒错误处理和 PC 兼容配置空间探测的平台上，建议软件或固件为 Role-Based Error Reporting 的 Function 设置 Unsupported Request Reporting Enable，但对于 1.0a 的 Function 则清除。通过检查 Device Capabilities Register 中的 Role-Based Error Reporting 字段，软件或固件可以区分这两类 Function。

Use of Max_Payload_Size

Max_Payload_Size 机制允许软件控制 Endpoint 发送的数据包中的最大有效负载，以平衡延迟和带宽权衡，尤其是对于定时业务流。

公众号·芯片业者相承双

如果软件选择将各种系统元件的 Max_Payload_Size 编程为非默认值，则必须注意确保每个数据包沿数据包路径的长度不超过任何系统元件的 Max_Payload_Size 参数。否则，数据包将被 Max_Payload_Size 参数过小的系统元件拒绝接收。

用于配置 Max_Payload_Size 以满足此要求的特定算法的讨论超出了本规范的范围，但是软件应基于以下因素来建立其算法：

- 拓扑结构中每个系统元件的 Max_Payload_Size 能力。
- 要关注系统元件何时通过热插拔操作被添加或删除。
- 了解哪些系统元件相互发送数据包，承载哪种类型的业务流，使用哪种类型的事务，或者是否由其他机制限制了数据包的大小。

对于固件配置系统元件以准备运行旧版操作系统环境的情况，固件可能需要避免将 Max_Payload_Size 编程为高于 128 字节的默认值，这是 Endpoint 支持的最小值。

例如，如果操作系统环境不包含 PCI Express，则固件可能不应该为支持热插拔操作的拓扑结构编程非默认的 Max_Payload_Size。否则，如果在添加新元件时不存在用于管理 Max_Payload_Size 设置的软件，则可能会导致操作不正确。请注意，新添加的元件甚至可能不支持与其余拓扑结构一样大的 Max_Payload_Size 设置，在这种情况下，软件可能需要拒绝启用新元件或减少其他元件的 Max_Payload_Size 设置。

Use of Max_Read_Request_Size

Max_Read_Request_Size 机制允许在服务质量（QoS）对于目标应用程序很重要的系统中改善带宽分配控制。 例如，当一些请求者使用比其他请求大得多的 Size 时，基于计数请求（而不是基于请求的大小）的仲裁方案会提供不精确的带宽分配。通过限制读请求的上限，可以使用 Max_Read_Request_Size 机制强制更均匀地分配带宽。

7.5.3.5 Device Status Register (Offset 0Ah)

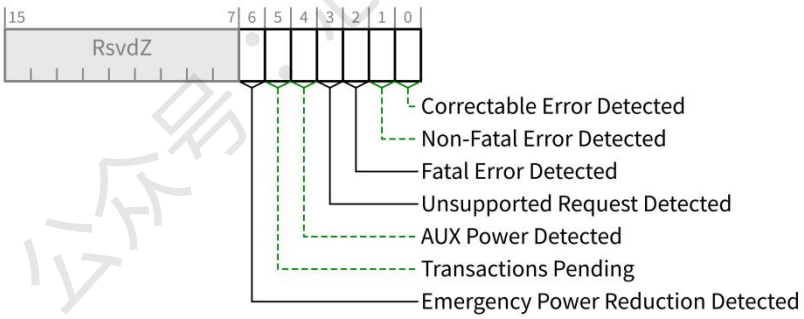

Figure 7-26 Device Status Register

Bit Location	Register Description	Attributes
0	**Correctable Error Detected** - 该位用于指示是否检测到可纠正错误。不管是否在 Device Control 寄存器中使能了错误报告，都会在该寄存器中记录错误。对于 Multi-Function Device，每个 Function 都指示相应 Function 是否检测到可纠正错误。 对于支持高级错误处理的 Function，无论 Correctable Error Mask 寄存器的设置如何，都会在该寄存器中记录错误。 默认值为 0b。	RW1C

公众号·芯片业者相承双

1	**Non-Fatal Error Detected** - 该位用于指示是否检测到非致命错误。不管是否在 Device Control 寄存器中使能了错误报告，都会在该寄存器中记录错误。对于 Multi-Function Device，每个 Function 都指示相应 Function 是否检测到非致命错误。 对于支持高级错误处理的 Function，无论 Uncorrectable Error Mask 寄存器的设置如何，都会在该寄存器中记录错误。 默认值为 0b。	RW1C
2	**Fatal Error Detected** - 该位用于指示是否检测到致命错误。不管是否在 Device Control 寄存器中使能了错误报告，都会在该寄存器中记录错误。对于 Multi-Function Device，每个 Function 都指示相应 Function 是否检测到致命错误。 对于支持高级错误处理的 Function，无论 Uncorrectable Error Mask 寄存器的设置如何，都会在该寄存器中记录错误。 默认值为 0b。	RW1C
3	**Unsupported Request Detected** - 该位用于指示 Function 是否收到 Unsupported Request。不管是否在 Device Control 寄存器中使能了错误报告，都会在该寄存器中记录错误。对于 Multi-Function Device，每个 Function 都指示相应 Function 是否收到 Unsupported Request。 默认值为 0b。	
4	**AUX Power Detected** - 需要辅助电源的 Function 如果检测到辅助电源　如果 Function 检测到辅助电源就把该位置 1。	RW1C
5	**Transactions Pending** - 对 Endpoint： 为 1 时，表示 Function 已发出的 Non-Posted Request 尚未完成。仅当所有未完成的 Non-Posted Request 已完成或已由完成超时机制终止时，Function 才报告清除该位。FLR 完成时也必须清除该位。 对 Root 和 Switch Port： 为 1 时，表示端口自己（使用端口自己的 Request ID）已发出的 Non-Posted Request 尚未完成。仅当所有此类未完成的 Non-Posted Request 都已完成或已由完成超时机制终止时，端口才报告清除此位。请注意，仅实现本文档所需功能的 Root Port 和 Switch Port 不会代表自己发出 Non-Posted Request，因此不受此情况的影响。 不代表自己发出 Non-Posted Request 的 Root 和 Switch Port 将该位硬连线为 0b。	RW1C
6	**Emergency Power Reduction Detected** - 当 Function 处于紧急断电状态时，该位置 1。只要存在任何会导致进入 Emergency Power Reduction State 的条件，该 Function 便会保持在 Emergency Power Reduction State，并且此时对该位的写无效。有关更多详细信息，请参见 Section 6.25。 与上游端口关联的 Multi-Function Device 必须在所有支持 Emergency Power Reduction State 的 Function 中将该位置 1。	RW1C

公众号：芯片业者程序员

如果 Emergency Power Reduction Supported 字段为 00b，则此位为 RsvdZ（请参见 Section 7.5.3.15）。

与上游端口无关的 Function 中的该位是 RsvdZ。

默认值为 0b。

7.5.3.6 Link Capabilities Register (Offset 0Ch)

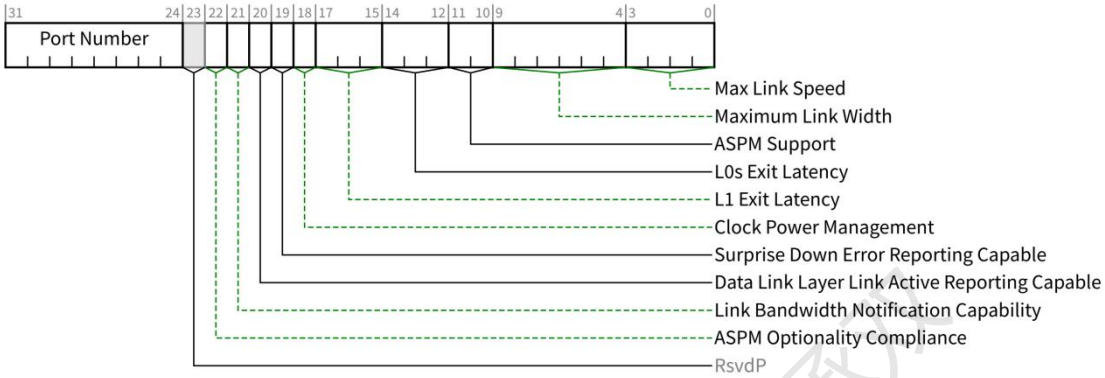

Figure 7-27 Link Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Max Link Speed** - 此字段值表示相关 Port 的最大 Link speed。 此字段的编码值指向 Link Capabilities 2 寄存器中 Supported Link Speeds Vector 中对应比特的位置。 定义的编码值为： **0001b** Supported Link Speeds Vector field bit 0 **0010b** Supported Link Speeds Vector field bit 1 **0011b** Supported Link Speeds Vector field bit 2 **0100b** Supported Link Speeds Vector field bit 3 **0101b** Supported Link Speeds Vector field bit 4 **0110b** Supported Link Speeds Vector field bit 5 **0111b** Supported Link Speeds Vector field bit 6 所有其他编码值保留。 与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。	RO
9:4	**Maximum Link Width** - 该字段定义组件支持的最大链路宽度。允许该值超过路由到插槽（下游端口）以及适配器连接器（上游端口）的通道数，或者超过组件间连接的实际连接宽度。 定义的编码值为：	RO

公众号·芯片业者相承汉

000001b x1

000010b x2

000100b x4

001000b x8

001100b x12

010000b x16

100000b x32

所有其他编码值保留。

与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。

11:10 **ASPM Support/Active State Power Management Support** - 该字段表示该 PCIe 链路支持的 ASPM 的 Level。参阅 Section 5.4.1 获取 ASPM 相关介绍。 RO

定义的编码值有：

00b No ASPM Support

01b L0s Supported

10b L1 Support

11b L0s and L1 Support

与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。

14:12 **L0s Exit Latency** - 该字段指示给定 PCI Express 链路的 L0s 退出延迟。报告的值指示此端口完成从 L0s 到 L0 过渡所需的时间。如果不支持 L0s，则该值不确定。但是，有关建议值，请参见 Section 5.4.1.1 中的实现说明"Potential Issues With Legacy Software When L0s is Not Supported"。 RO

定义的编码值有：

000b Less than 64 ns

001b 64 ns to less than 128 ns

010b 128 ns to less than 256 ns

011b 256 ns to less than 512 ns

100b 512 ns to less than 1 us

101b 1 us to less than 2 us

110b 2 us - 4 us

111b More than 4 us

请注意，退出延迟可能会受 PCI Express 参考时钟配置的影响，具体取决于组件使用的是公共参考时钟还是独立的参考时钟。

公众号：芯片业者相承双

与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。

17:15 **L1 Exit Latency** - 该字段指示给定 PCI Express 链路的 L1 退出延迟。报告的值指示此端口完成从 L1 到 L0 过渡所需的时间。如果不支持 L1，则该值不确定。但是，有关建议值，请参见 Section 5.4.1.1 中的实现说明"Potential Issues With Legacy Software When L0s is Not Supported"。 RO

定义的编码值有：

000b Less than 1 us

001b 1 us to less than 2 us

010b 2 us to less than 4 us

011b 4 us to less than 8 us

100b 8 us to less than 16 us

101b 16 us to less than 32 us

110b 32 us - 64 us

111b More than 64 us

请注意，退出延迟可能会受 PCI Express 参考时钟配置的影响，具体取决于组件使用的是公共参考时钟还是独立的参考时钟。

与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。

18 **Clock Power Management** - 对于上游端口，该位的值为 1b 表示当链路处于 L1 和 L2/L3 Ready 链路状态时，组件通过"clock request"（CLKREQ＃）机制允许删除参考时钟。值为 0b 表示该组件不具备此功能，并且在该链路状态下不得删除参考时钟。简称 Clock PM 功能。 RO

L1 PM Substate 为 CLKREQ＃信号定义了其他语义，这些语义的实现独立于 Clock Power Management 功能。

此功能仅在支持"clock request"（CLKREQ＃）Capability 的外形规格中适用。

对于与上游端口关联的 Multi-Function Device，每个 Function 均独立指示此 Capability。如果 Multi-Function Device 的所有 Function 在该位中均指示 1b，则电源管理配置软件必须允许移除参考时钟。对于 ARI 设备，所有 Function 必须在此位中指示相同的值。

对于下游端口，该位必须硬连线至 0b。

19 **Surprise Down Error Reporting Capable** - 对于下游端口，如果组件支持可选实现的检测和报告 Surprise Down 错误条件的 Capability，则必须将该位置 1。 RO

对于不支持此可选 Capability 的上游端口和组件，此位必须硬连线至 0b。

20 **Data Link Layer Link Active Reporting Capable** - 对于下游端口，如果组件支持可选实现的报告 Data Link Control and Management State Machine 的 DL_Active 状态的 Capability，则该位必须硬连线至 1b。对于具有热插拔功能的下游端口（由 Slot Capabilities Register 的 Hot-Plug Capable 字段指示）或支持链路速度大于 5.0 GT/s 的下游端口，此位必须硬连线至 1b。 RO

公众号·芯片业者怕承双

	对于不支持此可选 Capability 的上游端口和组件，此位必须硬连线至 0b。	
21	**Link Bandwidth Notification Capability** - 值 1b 表示支持 Link Bandwidth Notification 状态和中断机制。所有支持大于 x1 的链路和/或多个链路速率的 Root Port 和 Switch Downstream Port 都需要实现此 Capability。 该字段不适用于 Endpoint、 PCI Express to PCI/PCI-X Bridge 和 Upstream Ports of Switch，对这些 Function 该字段为保留值。 没有实现 Link Bandwidth Notification Capability 的 Function 此字段必须硬连线为 0b。	HwInit
22	**ASPM Optionality Compliance** - 在所有 Function 中，必须将该位置 1b。针对本规范的某些较早版本实现的组件会将其设置为 0b。 允许软件使用该位的值来帮助确定是启用 ASPM 还是运行 ASPM 一致性测试。	HwInit
31:24	**Port Number** - 该字段指示给定 PCI Express 链路的 PCI Express 端口号。 与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。	

Use of the ASPM Optionality Compliance Bit

正确实现和利用 ASPM 可以大大降低链路功耗。但是，ASPM 功能实现可能很复杂，并且从历史上看，某些实现尚未符合该规范。为了解决这个问题，本文档早期版本中的一些 ASPM 可选性和 ASPM 条目的要求已被放宽。但是，也支持并期望对 ASPM 功能进行明确的通过/失败的一致性性测试。

定义 ASPM Optionality Compliance 字段是为了建立对硬件和软件的明确期望的工具。该位置 1 表示硬件符合当前规范，并且必须在符合该规范的组件中将该位置 1。

系统软件以及一致性软件可以假定如果该位置 1，则关联的硬件符合当前规范。硬件应完全有能力支持 ASPM 的配置管理，而无需系统软件对组件进行特定处理。

对于没有设置此位的较旧的硬件，强烈建议系统软件提供机制以在与 ASPM 正常工作的组件上启用 ASPM，并在不与 ASPM 正常工作的组件上禁用 ASPM。

公众号·芯片业者相濡以沫

7.5.3.7 Link Control Register (Offset 10h)

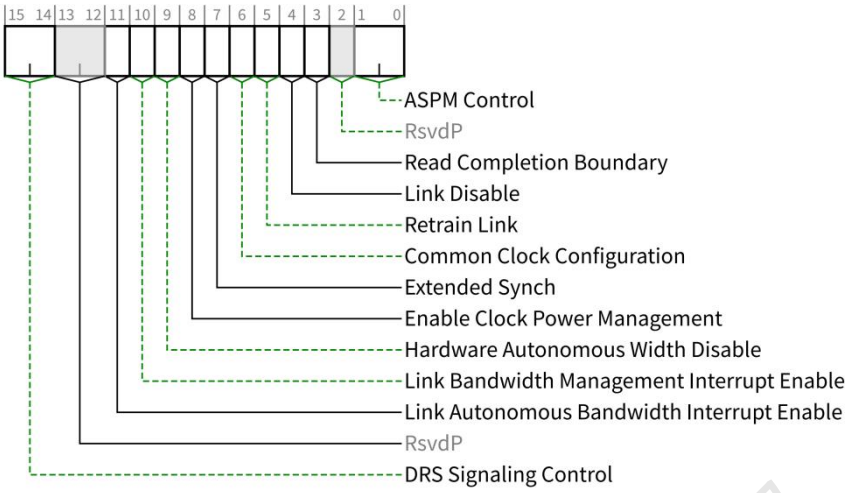

Figure 7-28 Link Control Register

Bit Location	Register Description	Attributes
1:0	**ASPM Control/Active State Power Management Control** - 参阅 Section 5.4.1.3 查阅如何使能 ASPM。 定义的编码值有: **00b** Disabled **01b** L0s Entry Enabled **10b** L1 Entry Enabled **11b** L0s and L1 Entry Enabled 注意："L0s Entry Enabled"使发送器可以进入 L0s。如果支持 L0s，即使发送器被禁止进入 L0s（00b 或 10b），接收器也必须能够进入 L0s。 必须先在链路的上游组件中通过软件启用 ASPM L1，然后才能在该链路的下游组件中启用 ASPM L1。 禁用 ASPM L1 时，软件必须先禁用链路上下游组件中的 ASPM L1，然后再禁用该链路上游组件中的 ASPM L1。仅当链路上的两个组件都支持 ASPM L1 时，才必须在下游组件上启用 ASPM L1。 对于 Multi-Function Device（包括 ARI 设备），建议软件在所有 Function 中为此字段编程相同的值。对于 non-ARI Multi-Function Device，仅当所有 Function 都支持此 Capability 时才为此组件启用该功能。 对于 ARI 设备，ASPM 控制完全由 Function 0 中的设置决定，而与 Function 0 的 D 状态无关。软件可以为每个 Function 编程不同的值，但组件会忽略它们。 除非特殊格式要求，否则此字段的默认值为 00b。	RW
3	**Read Completion Boundary** - 在 Root Port、Endpoint 和 Bridge 中该字段才是有意义的。当有意义	Root Ports:

公众号：芯片业者相承双

	时，定义的编码为： **0b** 64 bytes **1b** 128 bytes	RO Endpoints and Bridges: RW Switch Ports: RO
	Root Port：有关 RCB 的定义，请参阅 Section 2.3.1.1。 Root Port 的该字段要硬连线为其 RCB 能支持的 Capability。 Endpoint and Bridge：可选地由配置软件进行设置，以指示 Endpoint 或桥上游的根端口的 RCB 值。有关参数 RCB 的定义，请参见 Section 2.3.1.1。 仅当 Endpoint 或桥的上游根端口报告 RCB 值为 128 字节（Read Completion Boundary 字段的值为 1b）时，配置软件才必须将该位置 1。 默认值为 0b。 不实现此 feature 的 Function 此字段硬连线为 0b。 Switch Port：不使用此字段，必须硬连线为 0b。	
4	**Link Disable** - **设**置为 1 时，通过将 LTSSM 定向为 Disable 状态来禁用链路。该字段在 Endpoint、PCI Express to PCI/PCI-X 桥以及 Switch 的上游端口上为保留值。 不管实际的链路状态如何，对该位的写操作都会立即反映在从该位读取的值中。 清 0 后，软件必须遵守 Section 6.6.1 定义的有关常规复位后的第一个配置读的时序要求。 默认值为 0b。	RW
5	**Retrain Link** - 向该位写 1b 可让物理层 LTSSM 重新定向到 Recovery 状态来启动链路重新训练。如果 LTSSM 已经在 Recovery 或 Configuration 状态中，则允许重新进入 Recovery 状态，但不是必需的。如果在向该位写入 1b 时端口位于 DPC，则结果不确定。读该位始终返回 0b。 允许向该位写入 1b，同时向此寄存器（其实也包括 Link Control 2 寄存器）的其他字段写入修改值。如果 LTSSM 尚未处于 Recovery 或 Configuration 状态中，则触发的链路训练使用修改后的值。如果 LTSSM 已在 Recovery 或 Configuration 状态中，则修改的值不影响已经在进行的链路训练。 该位不适用于 Endpoint、PCI Express to PCI/PCI-X bridge 和 Upstream Ports of Switche，对于这些器件为保留值。 读该位始终返回 0b。	RW
6	**Common Clock Configuration** - 当设置为 1b 时，表示该组件和链路另一端的组件使用分布式公共参考时钟（我理解就是两侧设备的时钟同源）。设置为 0b 表示此组件和链路另一端的组件使用异步参考时钟。 对于 non-ARI Multi-Function Device，软件必须在所有 Function 中为此位编程相同的值。如果所有 Function 设置不一样的值，则该组件认为其参考时钟与上游组件使用的不是公共的时钟。 对于 ARI 设备，Common Clock Configuration 仅由 Function 0 中的设置值决定。其他 Function 中的	RW

公众号·芯片业务相承双

设置值将被忽略。

组件利用此 Common Clock Configuration 信息来报告正确的 L0s 和 L1 退出延迟。

更改链路上两个组件中该位的值后，软件必须通过向下游端口的 Retrain Link 位写入 1b 触发链路重新训练。

默认值为 0b。

7 **Extended Synch** - 设置为 1b 时，该位在退出 L0s 状态（Section 4.2.4.6）和处于 Recovery 状态（Section 4.2.6.4.1）时强制传输额外的有序集。此模式使外部设备（例如逻辑分析仪）在链路进入 L0 状态并恢复通信之前监视链路时间以实现 Bit Lock 和 Symbol Lock。 RW

对于 Multi-Function Device，如果任何一个 Function 把该位置为 1b，则组件必须在退出 L0s 或处于 Recovery 状态时传输额外的有序集。

默认值为 0b。

8 **Enable Clock Power Management** - 该位仅适用于上游端口，并且其外形规格支持"Clock Request"（CLKREQ＃）机制，其操作如下： RW

0b 时钟电源管理已禁用，并且设备必须将 CLKREQ＃信号保持低电平。

1b 当该位置 1 时，设备允许使用 CLKREQ＃信号根据适当的外形规格规范中定义的协议为链路时钟供电。

对于 non-ARI Multi-Function Device，仅当 Multi-Function Device 的所有 Function 的 Link Capabilities 寄存器的 Clock Power Management 字段为 1b 时，电源管理配置软件才必须将该位置 1。仅当所有 Function 的该位置 1 时，才允许该组件使用 CLKREQ＃信号进行电源管理链路时钟。

对于 ARI 设备，仅通过 Function 0 中的设置来启用时钟电源管理，其他 Function 中的设置将被忽略。

CLKREQ＃信号也可以通过 L1 PM Substates 机制来控制。这种控制不受该位设置的影响。

不支持时钟功率管理的下游端口和组件（Link Capabilities 寄存器的 Clock Power Management 字段为 0b）必须将该位硬接线为 0b。

除非外形规格规范另有规定，否则该位的默认值为 0b。

9 **Hardware Autonomous Width Disable** - 设置为 1 时使硬件无法更改链路宽度，这不是尝试通过减小链路宽度来纠正不可靠的链路操作而导致的。可以理解为就是强制硬件无法更改链路宽度。 RW/RsvdP

对于与上游端口关联的 Multi-Function Device，Function 0 中的该位为 RW 类型，且只有 Function 0 控制组件的链路行为。在该类设备的所有其他 Function 中，该位的类型为 RsvdP。

允许未实现自主更改链路宽度功能的组件将该位硬接线为 0b。

默认值为 0b。

10 **Link Bandwidth Management Interrupt Enable** - 当该位置 1 时，该位使能产生中断，以指示 Link Bandwidth Management Status 字段已被置为 1b。 RW

该位不适用于 Endpoint、PCI Express-to-PCI/PCI-X bridge 和 Upstream Ports of Switch，对这些组件

为保留值。

没有实现 Link Bandwidth Notification Capability 的 Function 必须把该字段硬连线为 0b。

默认值为 0b。

11	**Link Autonomous Bandwidth Interrupt Enable** - 置为 1 时，此位使能中断产生，以指示 Link Autonomous Bandwidth Status 字段已被置 1。 没有实现 Link Bandwidth Notification Capability 的 Function 必须把该字段硬连线为 0b。 默认值为 0b。	RW
15:14	**DRS Signaling Control** - 表示用于报告 DRS 消息接收的机制。必须通过在 Link Capabilities 2 寄存器中 DRS Supported 字段设置为 1 的下游端口来实现。编码为： **00b** DRS not Reported：如果 DRS Supported 设置为 1b，则接收一个 DRS 消息会将 Link Status 2 寄存器的 DRS Message Received 字段设置为 1，否则没有影响。 **01b** DRS Interrupt Enabled：如果 Link Status 2 寄存器的 DRS Message Received 字段从 0 变成 1，且 MSI 或 MSI-X 使能，则会使用 Interrupt Message Number（Section 7.5.3.2）中的向量生成一个 MSI 或 MSI-X 中断。 **10b** DRS to FRS Signaling Enabled：如果 Link Status 2 寄存器的 DRS Message Received 字段从 0 变成 1，该端口会向上游发送一个 FRS Reason 字段设置为 DRS Message Received 值的 FRS Message。 如果此字段设置为 10b，并且 Link Status 2 寄存器的 DRS Message Received 字段为 0，则行为未定义。 此字段设置为 11b 的行为也未定义。 Link Status 2 寄存器的 DRS Message Received 字段为 0 的下游端口必须将该字段硬连线到 00b。 对上有端口该字段保留。 默认值为 00b。	RW/RsvdP

Software Compatibility with ARI Devices

对于 Link Control 寄存器中的 ASPM Control、Common Clock Configuration 和 Enable Clock Power Management 字段而言，ARI 设备存在潜在的软件兼容性问题，因为这些元件严格按照 Function 0 中设置的值而不是所有 Function 中设置的值操作。

对于兼容的软件，Common Clock Configuration 字段应该没有问题，因为软件必须把所有 Function 中的该字段设置为相同值。

对于 Enable Clock Power Management 字段，在所有 Function 中将该字段设置为相同值的软件应该不会有兼容性问题。但是，如果软件没有在所有 Function 中将此字段设置为相同值，而是依赖于每个 Function 均具有防止启用 lock Power Management 的能力，则此类软件可能会与 ARI 设备存在兼容性问题。

对于 ASPM Control 字段，在所有 Function 中将此字段设置为相同值的软件应该不存在兼容性问题。但是，如果软件没有在所有 Function 中此字段设置为相同值，并且依赖于处于 D0 状态的每个 Function 具有阻止启用 ASPM 的能力，则此类软件可能会与 ARI

设备存在兼容性问题。

Avoiding Race Conditions When Using the Retrain Link Bit

当软件更改链路控制参数并在 Retrain Link 中写入 1b 以便使用新的参数启动链路训练时，需要特别小心，以免出现某些竞争状况。在任何时候，由于正常的链路活动，LTSSM 可能会转换到 Recovery 或 Configuration 状态，而无需软件感知。如果 LTSSM 已经处于 Recovery 或 Configuration 状态，在此情况下软件将更新的参数写入 Link Control 寄存器其把 Retrain Link 置为 1b，则 LTSSM 可能不将更新的参数与当前的链路训练一起使用。链路训练可能无法达到软件预期的结果。

为避免这种潜在的竞争状况，强烈建议软件使用以下算法或类似方法：

1. 软件无需将 1b 写入 Retrain Link 就可以将相关的链路控制参数设置为所需的设置。
2. 软件轮询 Link Status Register 中的 Link Training 字段，直到返回的值为 0b。
3. 软件将 1b 写入 Retrain Link，而不更改 Link Control 寄存器中的其他字段。

上面的算法保证链路训练将基于软件预期的链路控制参数进行。

Use of the Slot Clock Configuration and Common Clock Configuration Bits

为了确定跨连接器的链路两端上组件的公共时钟配置，需要两条信息。以下描述定义了这些要求。

第一条必要的信息是连接到插槽的端口是否使用了具有公共源的时钟，并因此导致插槽上提供的时钟信号具有恒定的相位关系。系统端组件（主机侧）通过其 Link Status 寄存器中需要硬件初始化的 Slot Clock Configuration 字段提供此信息。请注意，某些机械外形规格规范可能要求连接到插槽的端口使用时钟与插槽上提供的时钟信号相同的源。

第二个必要的信息是适配器上的组件是使用插槽上提供的时钟还是适配器上本地生成的时钟。 适配器的设计和布局将确定组件是否连接到插槽提供的时钟源。连接此适配器的组件应具有一些针对适配器的硬件初始化方法，以指示用于此特定适配器设计的配置。此信息由上游端口中每个 Function 的 Link Status 寄存器中的 Slot Clock Configuration 字段报告。请注意，某些机械外形规格可能要求适配器上的端口使用连接器上提供的时钟信号。

系统固件或软件将从物理链路两端的组件读取此 Slot Clock Configuration 值。如果此值都为 1，则此固件/软件会将 Link Control 寄存器上的两个组件上的链接控制寄存器的 Common Clock Configuration 编程为 1b。每个组件使用此位来确定退出 L0s 时将其接收器重新与对端组件的发送器重新同步所需的时间。

该值在 Link Capabilities Register 的第 12-14 位（偏移量为 0Ch）中作为时间值报告，并作为初始化过程的一部分作为 N_FTS 发送给相对的发送器。预计组件将需要更长的同步时间而无需公共时钟，因此会在 Link Capabilities Register 的第 12-14 位中报告更长的 L0s 退出延迟，并且在训练期间将为 N_FTS 发送更大的数量。 这就要求无论软件如何更改此位都应强制进行链路重新训练，以便为链路两端的接收器设置正确的 N_FTS。

7.5.3.8 Link Status Register (Offset 12h)

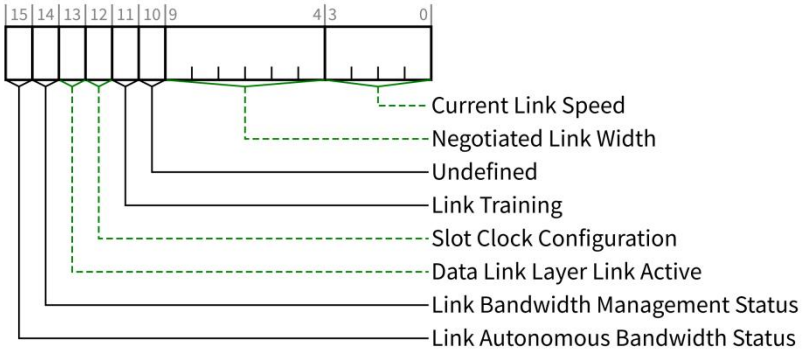

Figure 7-29 Link Status Register

Bit Location	Register Description	Attributes
3:0	**Current Link Speed** - 此字段值表示协商好的 Link speed。 此字段的编码值指向 Link Capabilities 2 寄存器中 Supported Link Speeds Vector 中对应比特的位置。 定义的编码值为： **0001b** Supported Link Speeds Vector field bit 0 **0010b** Supported Link Speeds Vector field bit 1 **0011b** Supported Link Speeds Vector field bit 2 **0100b** Supported Link Speeds Vector field bit 3 **0101b** Supported Link Speeds Vector field bit 4 **0110b** Supported Link Speeds Vector field bit 5 **0111b** Supported Link Speeds Vector field bit 6 所有其他编码值保留。链路没有 Link up 时此值是未定义的。	RO
9:4	**Negotiated Link Width** - 此字段值表示协商好的链路宽度。 定义的编码值为： **000001b** x1 **000010b** x2 **000100b** x4 **001000b** x8 **001100b** x12	RO

公众号·芯片业者相承双

010000b x16

100000b x32

所有其他编码值保留。链路没有 Link up 时此值是未定义的。

10 **Undefined** - 从该位读取的值是不确定的。在本规范的先前版本中，此位用于指示链路训练错误。系统软件必须忽略从该位读取的值。允许系统软件向该位写入任何值。 RO

11 **Link Training** - 此只读位指示物理层 LTSSM 处于 Recovery 或 Configuration 状态，或者 Retrain Link 字段写入 1b，但尚未开始链路训练。当 LTSSM 退出 Recovery 或 Configuration 状态时，硬件将清除该位。 RO

12 **Slot Clock Configuration** - 该位指示组件使用平台在连接器上提供的相同物理参考时钟。如果设备使用独立时钟而不考虑连接器上是否存在参考时钟，则该位必须清零。 HwInit

对于 Multi-Function Device，每个 Function 必须为此位报告相同的值。

13 **Data Link Layer Link Active** - 该位表示 Data Link Control and Management State Machine 的状态。1b 表示 DL_Active 状态，其他状态为 0b。 RO

如果 Data Link Layer Link Active Reporting Capable 字段为 1b，则必须实现此位。否则，该位必须硬连线至 0b。

14 **Link Bandwidth Management Status** - 此位由硬件设置，以指示在端口未通过 DL_Down 状态转换的情况下发生了以下任一情况： RW1C

- 在向 Retrain Link 字段写入 1b 后，链路重训练完成。

  Note：该位是在向 Retrain Link 位写入 1b 之后设置的，包括当链路出于其他原因正在进行重新训练时。

- 硬件改变了链路速率或宽度，试图通过 LTSSM 超时或更高级别的进程来纠正不可靠的链路操作。

如果物理层报告速率或宽度更改不是由下游组件自主发起的，则必须设置此位。

该位不适用于 Endpoint、PCI Express-to-PCI/PCI-X bridge 和 Upstream Ports of Switch，对这些组件为保留值。

没有实现 Link Bandwidth Notification Capability 的 Function 此字段需要硬连线为 0b。

默认值为 0b。

15 **Link Autonomous Bandwidth Status** - 此位由硬件设置，以指示硬件已经自主地改变了链路速率或宽度，而不是由于尝试纠正不可靠的链路操作以外的原因而使端口通过 DL_Down 状态转换而导致。 RW1C

如果物理层报告速率或宽度更改是由指示为自主更改的下游组件发起的，则必须设置此位。

该位不适用于 Endpoint、PCI Express-to-PCI/PCI-X bridge 和 Upstream Ports of Switch，对这些组件为保留值。

公众号·志片业者怕承双

没有实现 Link Bandwidth Notification Capability 的 Function 此字段需要硬连线为 0b。

默认值为 0b。

7.5.3.9 Slot Capabilities Register (Offset 14h)

如果实现了该寄存器但 Slot Implemented 字段为 0b，则整个寄存器的字段行为是未定义的。

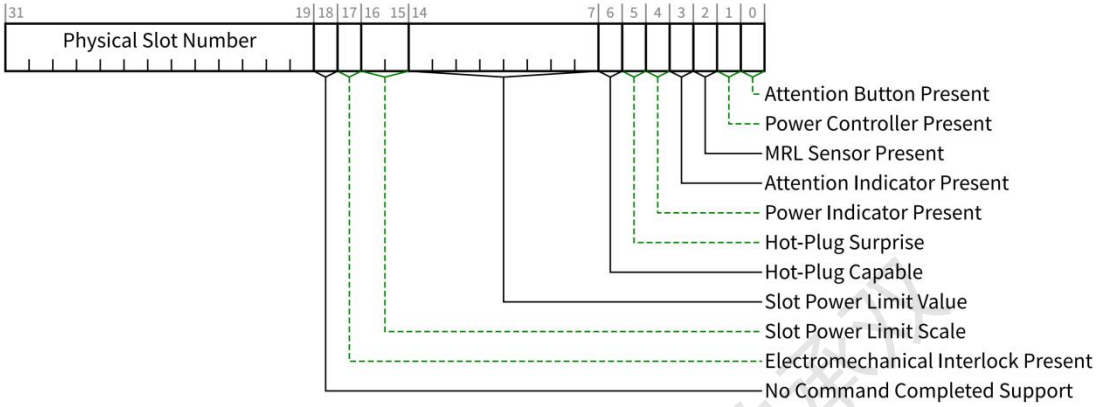

Figure 7-30 Slot Capabilities Register

Bit Location	Register Description	Attributes
0	**Attention Button Present** - 为 1 时，表示该插槽的 Attention Button 由底座进行电气控制。	HwInit
1	**Power Controller Present** - 为 1 时，表示为此插槽/适配器实现了软件可编程的电源控制器（取决于外形规格）。	HwInit
2	**MRL Sensor Present** - 为 1 时，表示为此底座上的插槽实现了 MRL Sensor。	HwInit
3	**Attention Indicator Present** - 为 1 时，表示该插槽的 Attention Indicator 由底座进行电气控制。	HwInit
4	**Power Indicator Present** - 为 1 时，表示该插槽的 **Power Indicator** 由底座进行电气控制。	HwInit
5	**Hot-Plug Surprise** - 为 1 时，表示该插槽中存在的适配器可能会在不事先通知的情况下从系统中删除。这是特定于外形规格的功能。该位为操作系统提供信息，以便在不影响软件继续运行的情况下进行此类删除。 如果 SFI Control Register 的 SFI HPS Suppress 字段清 0，则读取 Hot-Plug Surprise 将返回 HwInit 值。如果 SFI HPS Suppress 字段置 1，则读取返回 0b。请参阅 Section 7.9.23.3。	HwInit/RO
6	**Hot-Plug Capable** - 为 1 时表示该插槽支持热插拔操作。	HwInit
14:7	**Slot Power Limit Value** - 与 Slot Power Limit Scale 值结合使用，指定插槽提供的电源上限（see Section 6.9）或通过其他方式提供给适配器的电源。 功率限制（以瓦特为单位）是通过将该字段中的值乘以 Slot Power Limit Scale 字段中的值来计算	HwInit

的，除非 Slot Power Limit Scale 字段等于 00b 且 Slot Power Limit Value 超过 EFh，编码值为：

F0h	250 W Slot Power Limit
F1h	275 W Slot Power Limit
F2h	300 W Slot Power Limit
F3h to FFh	Reserved for Slot Power Limit Values above 300 W

Slot Implemented 字段为 1 时必须实现此字段。

写此寄存器会导致 Port 发送一个 Set_Slot_Power_Limit Message。

硬件/固件初始化之前的默认值为 0000 0000b。

16:15 **Slot Power Limit Scale** - 用于 Slot Power Limit Value 的比例因子（see Section 6.9）。 HwInit

编码值为：

00h	1.0 x
01h	0.1 x
10h	0.01 x
11h	0.001 x

Slot Implemented 字段为 1 时必须实现此字段。

写此寄存器会导致 Port 发送一个 Set_Slot_Power_Limit Message。

硬件/固件初始化之前的默认值为 00b。

17 **Electromechanical Interlock Present** - 为 1 时，表示在底座上为此插槽实现了 Electromechanical Interlock。 HwInit

18 **No Command Completed Support** - 为 1 时，表示当 Hot-Plug Controller 完成发出的命令时，该插槽不生成软件通知。仅当具有热插拔功能的端口能够接受对 Slot Control Register 的所有字段的写操作且连续写操作之间没有延迟时，才允许将该位置 1。 HwInit

31:19 **Physical Slot Number** - 该字段表示连接到该端口的物理插槽号。必须将此字段硬件初始化为一个值，该值为底座上的插槽分配唯一的插槽号，而不管与该插槽关联的外形规格如何。对于连接到集成在系统板上或与 Switch 设备或根端口位于同一芯片内的设备的端口，此字段必须初始化为零。 HwInit

7.5.3.10 Slot Control Register (Offset 18h)

Slot Control Register 控制 PCI Express 插槽的特定参数。Figure 7-31 详细说明了 Slot Control Register 中寄存器字段的分配；Table 7-26 提供了相应的字段的定义。

Slot Control Register 的 Attention Indicator Control、 Power Indicator Control 和 Power Controller Control 字段没有定义的默认值。如果实现了这些字段，则链路复位后，系统固件或操作系统软件必须初始化这些字段。

公众号·芯片业者相承双

在支持热插拔的下游端口中，对 Slot Control Register 的写操作必须导致生成热插拔命令（有关热插拔命令的详细信息，请参见 Section 6.7.3.2）。对不具有热插拔功能的下游端口中的 Slot Control Register 进行写操作，不得导致执行热插拔命令。

如果已实现该寄存器，但 Slot Implemented 字段清 0，则除 Data Link Layer State Changed Enable 字段外，整个寄存器的字段行为均未定义。

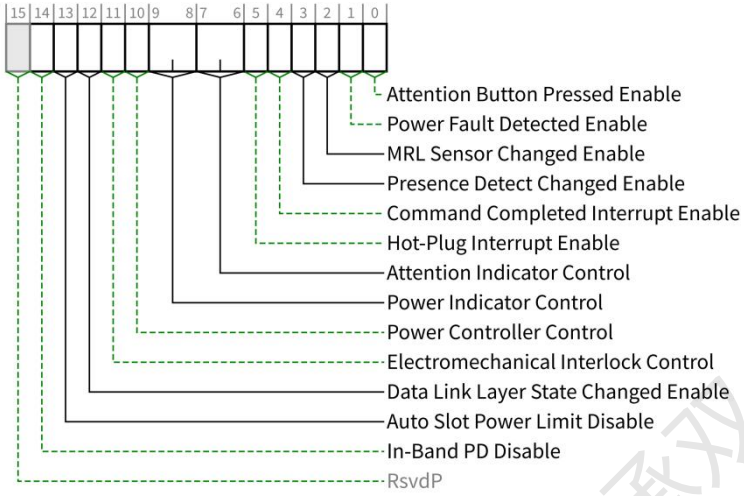

Figure 7-31 Slot Control Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed Enable** - 当设置为 1b 时，该位使能在 Attention Button pressed event 时的软件通知（see Section 6.7.3）。 如果 Slot Capabilities Register 的 Attention Button Present 字段为 0，则此字段可以实现为只读且读回值为 0b。 默认值为 0b。	RW
1	**Power Fault Detected Enable** - 当设置为 1b 时，该位使能在 power fault event 时的软件通知（see Section 6.7.3）。 如果未实现支持电源故障检测的电源控制器，则此字段可以实现为只读且读回值为 0b。 默认值为 0b。	RW
2	**MRL Sensor Changed Enable** - 当设置为 1b 时，该位使能在 MRL sensor changed event 时的软件通知（see Section 6.7.3）。 如果 Slot Capabilities Register 的 MRL Sensor Present 字段为 0，则此字段可以实现为只读且读回值为 0b。 默认值为 0b。	RW
3	**Presence Detect Changed Enable** - 当设置为 1b 时，该位使能在 presence detect changed event 时的软件通知（see Section 6.7.3）。 如果 Slot Capabilities Register 的 Hot-Plug Capable 字段为 0，则此字段可以实现为只读且读回值为	RW

公众号：芯片业者相承双

0b。

默认值为 0b。

4	**Command Completed Interrupt Enable** - 如果支持命令完成通知（如果 Slot Capabilities Register 中的 No Command Completed Support 为 0b），则在此字段置 1 时，当 Hot-Plug Controller 完成热插拔命令时，该位将使能软件通知。 如果不支持命令完成通知，此字段必须硬连线为 0b。 默认值为 0b。	RW
5	**Hot-Plug Interrupt Enable** - 当设置为 1b 时，该位使能在发生热插拔事件时生成一个中断。 如果 Slot Capabilities Register 的 Hot-Plug Capable 字段为 0，则此字段可以实现为只读且读回值为 0b。 默认值为 0b。	RW
7:6	**Attention Indicator Control** - 如果实现了 Attention Indicator，则写入字段中的值会将 Attention Indicator 设置为写入值表示的状态。 即使相应的热插拔命令未完成，该字段的读取也必须反映最新写入的值，除非软件发出写入而无需等待（如果需要）前一个命令完成，在这种情况下，读取值为 未定义。 定义的编码值有： **00b** Reserved **01b** On **10b** Blink（闪烁） **11b** Off 注意：此字段的默认值必须是非保留值之一。如果 Slot Capabilities Register 的 Attention Indicator Present 字段为 0，则此字段可以实现为只读且读回值为 0b。	RW
9:8	**Power Indicator Control** - 如果实现了 Power Indicator，则写入字段中的值会将 Power Indicator 设置为写入值表示的状态。 即使相应的热插拔命令未完成，该字段的读取也必须反映最新写入的值，除非软件发出写入而无需等待（如果需要）前一个命令完成，在这种情况下，读取值为 未定义。 定义的编码值有： **00b** Reserved **01b** On **10b** Blink（闪烁） **11b** Off 注意：此字段的默认值必须是非保留值之一。如果 Slot Capabilities Register 的 Power Indicator	RW

公众号：芯片业老拍承双

Present 字段为 0，则此字段可以实现为只读且读回值为 0b。

10	**Power Controller Control** - 如果实现了电源控制器，则在写入时该位将根据定义的编码设置插槽的电源状态。即使相应的热插拔命令未完成，读取该位也必须返回最新写入的值，除非软件发出写入（如果需要），而无需等待上一个命令完成，在这种情况下，读取的值为未定义。 请注意，在某些情况下，电源控制器可能会根据检测到的故障状况自动移除插槽电源或不响应加电请求，而与电源控制器控制设置无关。 定义的编码值为： **0b** Power On **1b** Power Off 如果 Slot Capabilities Register 中的 Power Controller Present 为 0，则对该位的写入无效，并且该位的读取值不确定。	RW
11	**Electromechanical Interlock Control** - 如果实现了 Electromechanical Interlock，则对该位写 1b 会导致机械锁的状态切换。向该位写入 0b 无效。读该位总是返回 0b。	RW
12	**Data Link Layer State Changed Enable** - 如果 Data Link Layer Link Active Reporting Capable 为 1b，则当 Data Link Layer Link Active 字段变化时，此位将启用软件通知。 如果 Data Link Layer Link Active Reporting Capable 为 0b，则该字段允许实现为只读且读回值为 0b。 默认值为 0b。	RW
13	**Auto Slot Power Limit Disable** - 设置为 1 时，当链路从非 DL_Up 状态转换为 DL_Up 状态时，这将禁用自动发送 Set_Slot_Power_Limit 消息。 允许不支持 DPC 的下游端口将该位硬接线为 0b。 默认值是实现相关的。	RW
14	**In-Band PD Disable** - 为 1 时，此位禁止 in-band presence detect mechanism 影响 Presence Detect State 字段的值，从而允许该位专门报告 out-of-band presence detect。否则，Presence Detect State 将反映带内和带外状态检测机制的逻辑或。 in-band presence detect mechanism（带内存在监测机制）和 out-of-band presence detect（带外存在检测机制）用于表示在热插拔发生时用带内信号通知还是带外信号通知。 另外，In-Band PD Disable 字段控制 Link Status 2 Register 中 Downstream Component Presence 字段表示的组件存在状态。参见 Section 7.5.3.20。 如果 In-Band PD Disable Supported 字段为 1b，则必须实现该位。否则，该位必须硬连线到 0b。 默认值为 0b。	RW

公众号·芯片业者相承双

7.5.3.11 Slot Status Register (Offset 1Ah)

Slot Status Register 提供有关 PCI Express 插槽特定参数的信息。Figure 7-32 详细说明了 Slot Status Register 中寄存器字段的分配；Table 7-27 提供了相应的位定义。设备未实现的状态位的寄存器字段具有 RsvdZ 属性。

如果实现了该寄存器，但 Slot Implemented 为 0，则整个寄存器的字段行为（数据链路层状态更改位除外）是未定义的。

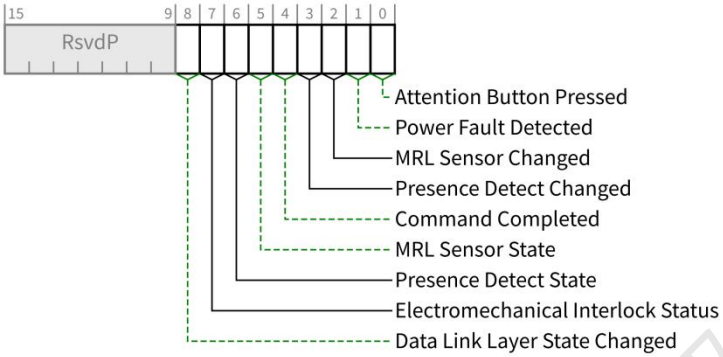

Figure 7-32 Slot Status Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed** - 如果实现了 Attention Button，则在按下 Attention Button 时该位置 1。否则为 0。	RW1C
1	**Power Fault Detected** - 如果实现了支持电源故障检测的电源控制器，则在电源控制器检测到此插槽的电源故障把该位置 1。注意，根据硬件能力，可以随时检测到电源故障，这与电源控制器控制设置或插槽占用无关。如果不支持电源故障检测，则不得设置此位。	RW1C
2	**MRL Sensor Changed** - 如果实现了 MRL Sensor，则在检测到 MRL Sensor 状态变化时把该位置 1。反之为 0。	RW1C
3	**Presence Detect Changed** - 当 Presence Detect State 字段值改变时此字段置为 1。	RW1C
4	**Command Completed** - 如果支持命令完成通知（如果 Slot Capabilities Register 中的 No Command Completed Support 为 0b），则在热插拔命令完成且热插拔控制器准备接受后续命令时，此位置 1。Command Completed 状态位置 1，表示主机软件热插拔控制器已处理上一条命令并准备接收下一条命令。它不能保证与命令相对应的动作已经完成。 如果不支持命令完成通知，则该位必须硬连线为 0b。	RW1C
5	**MRL Sensor State** - 如果实现了 MRL Sensor，此字段用于报告 MRL Sensor 的状态。 定义的编码值为： **0b** MRL Closed **1b** MRL Open	RO
6	**Presence Detect State** - 该位为 1 表示插槽中存在适配器。当 In-Band PD Disable 字段为 0 时，这	RO

公众号·志先业者相承双

	将通过物理层带内存在检测机制的逻辑"或"来反映，如果存在，则为插槽的对应外形规格的任何带外存在检测机制也将反映出来。请注意，带内存在检测机制要求对适配器施加电源以检测其存在。因此，需要电源控制器进行热插拔的外形规格必须实现带外存在检测机制。当 In-Band PD Disable 字段为 1 时，带内存在检测机制对此位无效。 定义的编码为： **0b** Adapter not Present **1b** Adapter Present 此位必须在实现插槽的所有下游端口上实现。对于不连接到插槽的下游端口（PCI Express Capabilities Register 的 Slot Implemented 字段为 0b），此位必须硬连线至 1b。	
7	**Electromechanical Interlock Status** - 如果实现了机械锁，此字段表示机械锁的状态。 定义的编码值为： **0b** 机械锁空闲 **1b** 机械锁占用	RO
8	**Data Link Layer State Changed** - 当 Link Status Register 的 Data Link Layer Link Active 字段值变化时该位置 1。 为响应数据链路层状态更改事件，软件必须在启动对热插拔设备的配置周期之前，读取 Link Status Register 的 Data Link Layer Link Active 字段，以确定链路是否处于激活状态。	RW1C

No Slot Power Controller

对于未实现电源控制器的插槽，软件必须确保在读取状态检测状态之前，已启用系统电源平面为插槽提供电源。

7.5.3.12 Root Control Register (Offset 1Ch)

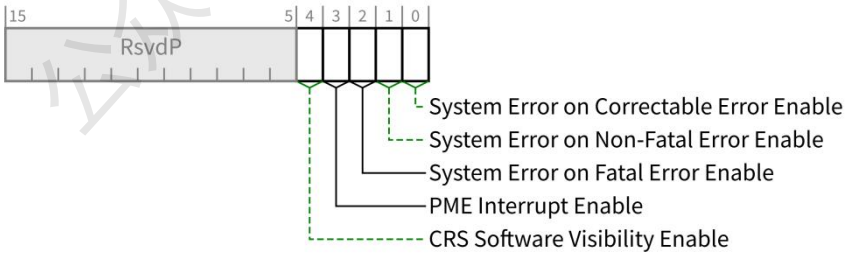

Figure 7-33 Root Control Register

Bit Location	Register Description	Attributes
0	**System Error on Correctable Error Enable** - 为 1 时，该位表示如果与此根端口关联的层次结构域中的任何设备或根端口本身报告了可纠正错误（ERR_COR），则应生成系统错误。向系统发送系统错误信令的机制是系统特定的。 Root Complex Event Collector 为 RCiEP 的这个功能提供支持。	RW

公众号·芯片工程师相承双

	默认值为 0b。	
1	**System Error on Non-Fatal Error Enable** - 为 1 时，该位表示如果与此根端口关联的层次结构域中的任何设备或根端口本身报告了非致命错误（ERR_NONFATAL），则应生成系统错误。向系统发送系统错误信令的机制是系统特定的。 Root Complex Event Collector 为 RCiEP 的这个功能提供支持。 默认值为 0b。	RW
2	**System Error on Fatal Error Enable** - 为 1 时，该位表示如果与此根端口关联的层次结构域中的任何设备或根端口本身报告了致命错误（ERR_FATAL），则应生成系统错误。向系统发送系统错误信令的机制是系统特定的。 Root Complex Event Collector 为 RCiEP 的这个功能提供支持。 默认值为 0b。	RW
3	**PME Interrupt Enable** - 为 1 时，此位使能接收到 PME 消息且反映在 PME Status 中时生成 PME 中断（见 Table 7-30）。如果将该位从 0 改为 1（see Section 5.3.3），则如果 PME Status 字段被置 1，也会生成 PME 中断。 默认值为 0b。	RW
4	**CRS Software Visibility Enable** - 为 1 时，该位使根端口可以将 Configuration Request Retry Status (CRS) Completion Status 返回给软件（see Section 2.3.1）。 未实现此功能的根端口必须将此位硬连线到 0b。 默认值为 0b。	RW

7.5.3.13 Root Capabilities Register (Offset 1Eh)

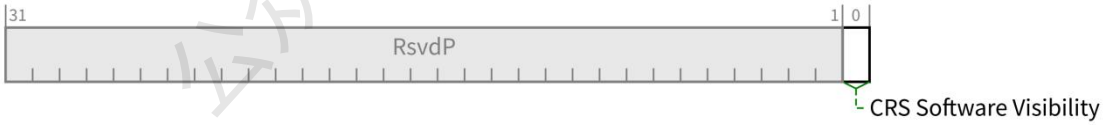

Figure 7-34 Root Capabilities Register

Bit Location	Register Description	Attributes
0	**CRS Software Visibility** - 为 1 时，表示根端口能够将 Configuration Request Retry Status (CRS) Completion Status 返回给软件（see Section 2.3.1）。	RO

公众号·芯片业者相关双

7.5.3.14 Root Status Register (Offset 20h)

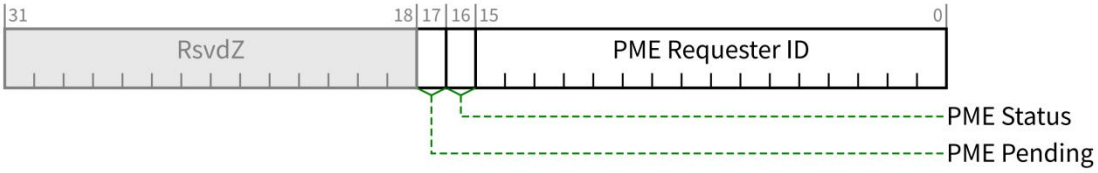

Figure 7-35 Root Status Register

Bit Location	Register Description	Attributes
15:0	**PME Requester ID** - 该字段表示最后一个 PME 请求者的 PCI Requester ID。仅当 PME Status 字段设置为 1 时，此字段才有效。	RO
16	**PME Status** - 该位表示 PME 由 PME Requester ID 字段中指示的 PME 请求者置起。 后续的 PME 保持待处理状态，直到该状态寄存器通过写 1b 将其清零为止。	RW1C
17	**PME Pending** - 当 PME Status 字段被置 1 时，该位表示存在另一个 PME 待处理。 通过软件清除 PME Status 字段时； 通过再次设置 PME Status 字段并适当更新 PME Requester ID 字段，硬件会产生 PME。如果没有其他 PME 挂起，则 PME Pending 字段由硬件清除。	RO

7.5.3.15 Device Capabilities 2 Register (Offset 24h)

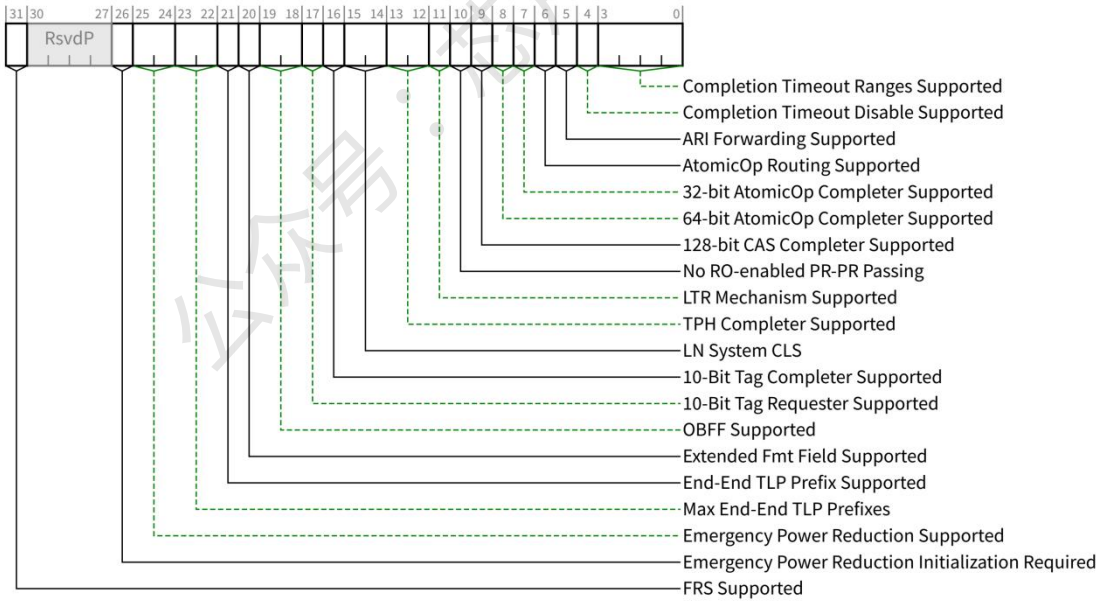

Figure 7-36 Device Capabilities 2 Register

Bit Location	Register Description	Attributes
3:0	**Completion Timeout Ranges Supported** - 该字段表示设备的 Function 对可选的完成超时可编程机制的支持能力。该机制允许系统软件修改完成超时值。	HwInit

该字段仅适用于 Root Port、能自行发出请求的 Endpoint，以及能在 PCI Express 上自行发出请求的 PCI Express to PCI/PCI-X Bridge。对于所有其他功能，此字段为保留字段，必须硬连线至 0000b。

有四个时间范围值：

Range A 50 us to 10 ms

Range B 10 ms to 250 ms

Range C 250 ms to 4 s

Range D 4 s to 64 s

以下编码给出了选择的完成超时时间范围：

0000b 不支持完成超时编程-Function 需要实现 50us ~ 50ms 的超时时间值。

0001b Range A

0010b Range B

0011b Range A and Range B

0110b Range B and Range C

0111b Range A ，B，and C

1110b Range B ，C，and D

1111b Range A ，B，C，and D

所有其他值为保留值。

强烈建议"完成超时"机制在 10 ms 内不过期。

4 **Completion Timeout Disable Supported** - 值 1b 表示支持"完成超时禁用"机制。 RO

完成超时禁用机制仅适用于能自行发出请求的 Endpoint，以及能在 PCI Express 上自行发出请求的 PCI Express to PCI/PCI-X Bridge。对于所有其他功能，此字段为保留字段，必须硬连线至 0000b。

此机制对 Root Port 是可选的。

对于其他 Function 此字段是保留的并且必须硬连线为 0b。

5 **ARI Forwarding Supported** - 仅适用于 Switch Downstream Port 和 Root Port；对于其他 Function，必须为 0b。如果 Switch Downstream Port 和 Root Port 支持此可选功能，则必须将该位设置为 1b。有关更多详细信息，请参见 Section 6.13。 RO

6 **AtomicOp Routing Supported** - 仅适用于 Switch Upstream Port、Switch Downstream Port 和 Root Port；对于其他 Function，必须为 0b。如果 Switch Downstream Port 和 Root Port 支持此可选功能，则必须将该位设置为 1b。有关更多详细信息，请参见 Section 6.15。 RO

7 **32-bit AtomicOp Completer Supported** - 适用于实现了 Memory Space BAR 以及所有 Root Port 的 Function；否则必须为 0b。 包括 FetchAdd、Swap 和 CAS AtomicOp。 如果 Function 支持此可选功能，则必须将该位置 1b。有关其他 RC 要求，请参见 Section 6.15.3.1。 RO

公众号：芯片业者相承双

8	**64-bit AtomicOp Completer Supported** - 适用于实现了 Memory Space BAR 以及所有 Root Port 的 Function；否则必须为 0b。 包括 FetchAdd、Swap 和 CAS AtomicOp。 如果 Function 支持此可选功能，则必须将该位置 1b。有关其他 RC 要求，请参见 Section 6.15.3.1。	RO
9	**128-bit CAS Completer Supported** - 适用于实现了 Memory Space BAR 以及所有 Root Port 的 Function；否则必须为 0b。如果 Function 支持此可选功能，则必须将该位置 1b。有关更多详细信息，请参见 Section 6.15。	RO
10	**No RO-enabled PR-PR Passing** - 如果该位置 1，则路由元件将永远不会执行 Table 2-40 中设置了 Relaxed Ordering Attribute 字段的条目 A2b 所允许的路由顺序。 此位仅适用于支持根端口之间有对等业务的 Switch 和 RC。此位仅适用于通过 Switch 或 RC 转发的 Posted Request，不适用于在 Switch 或 RC 本身内发起或终止的业务流。Switch 或 RC 上的所有端口必须为此位报告相同的值。 对所有其他 Function，该字段需要硬连线为 0b。	HwInit
11	**LTR Mechanism Supported** - 为 1b 时表示支持可选的延迟容忍报告（LTR）机制。 允许 Root Port、Switch 和 Endpoint 实现此 Capability。 对于与一个上游端口关联的 Multi-Function Device，其每个 Function 必须为此位报告相同的值。 对于未实现此 Capability 的桥和其他 Function，此位必须硬连线至 0b。	RO
13:12	**TPH Completer Supported** - 该字段的值表示 Completer 对 TPH 或 Extended TPH 的支持。此字段仅适用于根端口和 Endpoint。对于所有其他 Function，此字段为保留字段。 定义的编码值为： **00b** Completer 不支持 TPH 和 Extended TPH。 **01b** Completer 支持 TPH，不支持 Extended TPH。 **10b** 保留。 **11b** Completer 支持 TPH 和 Extended TPH。 参阅 Section 6.17 获取更详细信息。	RO
15:14	**LN System CLS** - 仅适用于根端口和 RCRB； 对于所有其他 Function 类型，必须为 00b。该字段指示根端口或 RCRB 是否支持 LN 协议作为 LN Completer，如果支持，则会作用于高速缓存行大小。 编码值为： **00b** LN Completer either not supported or not in effect **01b** LN Completer with 64-byte cachelines in effect **10b** LN Completer with 128-byte cachelines in effect **11b** Reserved	HwInit

公众号·芯片业者相承双

16	**10-Bit Tag Completer Supported** - 如果该位置 1，则 Function 支持 10-Bit Tag Completer；否则，不支持。参阅 Section 2.2.6.2。	HwInit
17	**10-Bit Tag Requester Supported** - 如果该位置 1，则 Function 支持 10-Bit Tag Requester；否则，不支持。 如果 10-Bit Tag Completer Supported 为 0，则该字段不能为 1。 如果 Function 是个 RCiEP，则如果 RC 不支持来自此 RCiEP 的请求的 10-Bit Tag Completer Supported，则该位必须为清零。 请注意，在请求者可以生成 10 位 Tag 之前，必须通过作为请求者的 Function 的 Device Control 2 寄存器中的 10-Bit Tag Requester Enable 字段来启用 10-Bit Tag 字段。参阅 Section 2.2.6.2。	HwInit
19:18	**OBFF Supported** - 该字段表示是否支持 OBFF，以及使用哪种信令机制。 **00b** OBFF Not Supported **01b** OBFF supported using Message signaling only **10b** OBFF supported using WAKE# signaling only **11b** OBFF supported using WAKE# and Message signaling 仅在以下情况下，此字段中报告的值必须表示支持 WAKE # 信令： ● 对于下游端口，支持为 OBFF 驱动 WAKE # 信号，并且已知连接到下游的连接器或组件会接收相同的 WAKE # 信号。 ● 对于上游端口，支持接收 OBFF 的 WAKE # 信号，并且如果该组件在附加卡上，则表明该组件已连接到连接器上的 WAKE # 信号。 允许 Root Port、Switch Port 和 Endpoint 实现此 Capability。 对于与上游端口关联的 Multi-Function Device，每个 Function 必须为此字段报告相同的值。 对于未实现此 Capability 的桥和端口，必须将该字段硬连线到 00b。	HwInit
20	**Extended Fmt Field Supported** - 为 1 时，表示该 Function 支持 3-Bit 的 Fmt 字段的定义。如果为 0，则 Function 支持 2-Bit 的 Fmt 字段的定义。参阅 Section 2.2。 支持 End-End TLP Prefix 的 Function 此字段必须设置为 1。上游端口中的所有 Function 的该位必须具有相同的值。组件的每个下游端口对此位可能具有不同的值。 强烈建议 Function 支持 3-Bit 的 Fmt 字段的定义。	RO
21	**End-End TLP Prefix Supported** - 表示 Function 是否支持 End-End TLP Prefix。 **0b** 不支持 **1b** 支持接收包含 End-End TLP Prefix 的 TLP。 所有的端口或 Switch 的此字段应该有相同的值。	HwInit

公众号·芯片业者相承双

23:22 **Max End-End TLP Prefixes** - 表示此 Function 支持的最大 End-End TLP Prefix 数。参阅 Section 2.2.10.2 获取重要信息。支持的值是：HwInit

01b 最大支持 1 个 End-End TLP Prefix

10b 最大支持 2 个 End-End TLP Prefix

11b 最大支持 3 个 End-End TLP Prefix

00b 最大支持 4 个 End-End TLP Prefix

如果 End-End TLP Prefix Supported 字段为 0，则该字段为 RsvdP。

如果不同的 Root Port 的 End-End TLP Prefix Supported 字段的值不同，则此字段值也可以设置的不同。

对于 End-End TLP Prefix Supported 字段设置为 1 的 Switch，此字段需要设置为 00b 表示支持 4 个 End-End TLP Prefix。

25:24 **Emergency Power Reduction Supported** - 表示对可选实现的 Emergency Power Reduction State feature 的支持级别。Function 可以自主进入应急功率降低状态，也可以基于相关的外形规格规范定义的两种机制之一进入应急功率降低状态。 处于紧急断电状态的 Function 消耗的功率更少。紧急功率降低机制允许底座请求附加卡快速进入紧急功率应急状态，而无需使用系统软件。有关更多详细信息，请参见 Section 6.25。HwInit

支持的值为：

00b 不支持应急功率降低状态

01b 支持应急功率降低状态并由设备相关的实现来触发

10b 支持应急功率降低状态并由设备相关的实现或者相关外形规格规范定义的机制触发。

11b 保留

在与上游端口不相关的 Function 中，此字段为 RsvdP。

对于与上游端口关联的 Multi-Function Device，所有为此字段报告非零值的 Function 必须为此字段报告相同的非零值。

默认值为 00b。

复位后，此字段返回非零值后，必须继续返回相同的非零值，直到下一次复位。

26 **Emergency Power Reduction Initialization Required** - 如果设置为 1，则退出紧急功率降低状态后，Function 需要完全或部分初始化。如果清 0，则退出紧急功率降低状态后，Function 无需软件干预即可恢复正常运行。有关更多详细信息，请参见 Section 6.25。HwInit

在与上游端口不相关的 Function 中，此字段为 RsvdP。

对于与上游端口关联的 Multi-Function Device，所有 Function 必须为此字段报告相同的值。

默认值为 00b。

复位后，此字段返回非零值后，必须继续返回相同的非零值，直到下一次复位。

31	**FRS Supported** - 为 1 时表示支持可选的 Function Readiness Status（FRS） Capability。 支持 FRS 消息的生成或接收的所有 Function 此字段必须设置为 1。 不能自己生成 FRS 消息的 Switch Function 此字段不能设置为 1。	HwInit

Use of the No RO-enabled PR-PR Passing Bit

No RO-enabled PR-PR Passing 字段允许平台在请求者和完成者之间的路径上利用 PCI Express Switch 元件，以使请求者可以从稍微宽松的排序模型中受益。一个示例是无法确保对同一地址的多个重复的 Posted Write 都在同一时间处于 outstanding 状态的设备。使这种设备能够利用此模型的方法超出了本规范的范围。

7.5.3.16 Device Control 2 Register (Offset 28h)

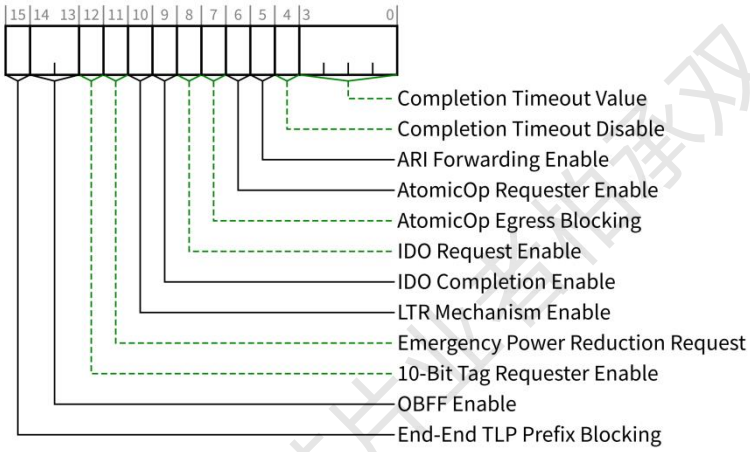

Figure 7-37 Device Control 2 Register

Bit Location	Register Description	Attributes
3:0	**Completion Timeout Value** - 在实现了完成超时机制的设备中，该字段用于完成超时值的编程。该字段仅适用于 Root Port、能自行发出请求的 Endpoint，以及能在 PCI Express 上自行发出请求的 PCI Express to PCI/PCI-X Bridge。对于所有其他功能，此字段为保留字段，必须硬连线至 0000b。 不支持此可选功能的 Function 必须将该字段硬接线为 0000b，并且要求实现 50 µs 至 50 ms 范围内的超时值。支持完成超时可编程性的 Function 必须支持下面编码给出的值，这些值与 Completion Timeout Ranges Supported 字段指示的可编程范围相对应。 定义的编码值为： **0000b** Default range: 50 us to 50 ms 强烈建议“完成超时”机制在 10 毫秒内不过期。 如果支持 Range A（50µs to 10 ms）的可编程范围，则可用的值为： **0001b** 50 us to 100 us **0010b** 1 ms to 10 ms	RW

如果支持 Range B（10 ms to 250 ms）的可编程范围，则可用的值为：

0101b 16 ms to 55 ms

0110b 65 ms to 210 ms

如果支持 Range C（250 ms to 4 s）的可编程范围，则可用的值为：

1001b 260 ms to 900 ms

1010b 1 s to 3.5 ms

如果支持 Range D（4 s to 64 s）的可编程范围，则可用的值为：

1101b 4 s to 13 s

1110b 17 s to 64 s

没有定义的值为保留值。

允许软件随时更改此字段中的值。对于更改完成超时值时已经挂起的请求，允许硬件对 outstanding Request 使用新值或旧值，并允许每个请求的开始时间基于该值更改的时间或基于每个请求发出的时间。

默认值为 0000b。

4 **Completion Timeout Disable** - 为 1 时禁用完成超时机制。 RW

所有支持 Completion Timeout Disable 功能的 Function 都需要实现此位。允许不支持此可选功能的 Function 将该位硬接线为 0b。

允许软件随时设置或清除该位。设置为 1 时，将禁用完成超时检测机制。如果在清除该位时还有 outstanding Request，则允许但不要求硬件将完成超时机制应用于这些 outstanding Request。如果这样做，则允许以清除该位的时间或发出每个请求的时间为每个请求的开始时间。

默认值为 0b。

5 **ARI Forwarding Enable** - 设置后，当将 Type 1 Configuration Request 转换为 Type 0 Configuration Request 时，下游端口将禁用其传统 Device Number 字段，将其强制为 0，从而允许访问端口下方的 ARI 设备中的 Extended Function。See Section 6.13。 RW/RsvdP

该位的默认值为 0b。如果 ARI Forwarding Supported 字段为 0b，则该位必须硬连线至 0b。

对于上游端口该位不适用并且保留。

6 **AtomicOp Requester Enable** - 仅适用于 Endpoint 和 Root Port；对于其他 Function 类型，必须硬连线至 0b。仅当该位和 Command 寄存器中的 Bus Master Enable 字段都置为 1 时，才允许该 Function 发出 AtomicOp Request。 RW

如果 Endpoint 和 Root Port 能够发起 AtomicOp Request，则该位必须为 RW，否则允许其硬连线至 0b。

该位不用作 Capability 位。即使 Endpoint 和 Root Port 不支持作为 AtomicOp Requester，也可以将该位设为 RW。

公众号：芯片业者柏承双

默认值为 0b。

7	**AtomicOp Egress Blocking** - 对实现 AtomicOp 路由功能的 Switch Upstream Port、Switch Downstream Port 和 Root Port 要强制实现该字段；否则必须硬连线至 0b。 设置此位后，必须阻止目标为从该 Egress Port 出的 AtomicOp Request。See Section 6.15.2。 默认值为 0b。	RW
8	**IDO Request Enable** - 如果该位设置为 1，则允许 Function 把它发出的请求中的 ID-Based Ordering (IDO) bit (Attr[2]) 字段设置为 1（see Section 2.2.6.3 and 2.4）。 Endpoint、RC Integrated Endpoint 和 Root Port 都可以实现此 Capability。 如果某个 Function 从未在其请求中设置 IDO 属性，则允许将该位硬连线为 0b。 默认值为 0b。	RW
9	**IDO Completion Enable** - 如果该位设置为 1，则允许 Function 把它返回的 Completion 中的 ID-Based Ordering (IDO) bit (Attr[2]) 字段设置为 1（see Section 2.2.6.3 and 2.4）。 如果某个 Function 从未在其 Completion 中设置 IDO 属性，则允许将该位硬连线为 0b。 默认值为 0b。	RW
10	**LTR Mechanism Enable** - 当设置为 1b 时，该位开启上游端口发送 LTR 消息的功能，开启下游端口处理 LTR 消息的功能。 对于与实现 LTR 的设备的上游端口相关联的 Multi-Function Device，Function 0 中的该位为 RW，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，该位为 RsvdP。 没有实现 LTR 机制的 Function 该位硬连线为 0b。 默认值为 0b。 对于下游端口，如果端口进入 DL_Down 状态，则必须将该位重置为默认值。	RW/RsvdP
11	**Emergency Power Reduction Request** - 如果设置为 1，则支持紧急功率降低状态的组件中的所有 Function 都必须进入紧急功率降低状态。如果清 0，则在没有其他原因阻止退出此状态的情况下，这些 Function 必须退出紧急节能状态。有关更多详细信息，请参见 Section 6.25。 在与 Emergency Power Reduction Supported 字段中具有非零值的上游端口关联的编号最小的 Function 中实现此位。在所有其他 Function 中，该位为 RsvdP。 默认值为 0b。	RW/RsvdP
12	**10-Bit Tag Requester Enable** - 该位与 Device Control 寄存器中的 Extended Tag Field Enable 字段一起，确定允许请求者使用多少比特的 Tag 字段。当 10-Bit Tag Requester Enable 字段被设置为 1 时，允许请求者使用 10-Bit Tag。有关完整的详细信息，请参见 Section 2.2.6.2。 如果在 Function 还有 outstanding Non-Posted Request 时软件更改了该位的值，则结果是不确定的。 允许未实现 10-Bit Tag Requester capability 的 Function 的功能将该位硬接线为 0b。	RW

公众号：芯片业者相承及

默认值为 0b。

14:13 **OBFF Enable** - 该字段启用 OBFF 机制并选择信令方法。 RW/RsvdP

00b Disabled

01b Enabled using Message signaling [Variation A]

10b Enabled using Message signaling [Variation B]

11b Enabled using WAKE# signaling

有关上述编码的说明，请参见 Section 6.19。

所有支持 OBFF Capability 的端口都需要此字段。

对于与实现 OBFF 的设备的上游端口相关联的 Multi-Function Device，Function 0 中的字段为 RW 类型，只有 Function 0 控制组件的行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

允许未实现 OBFF 的端口将此字段硬接线为 00b。

默认值为 00b。

15 **End-End TLP Prefix Blocking** - 控制是否允许路由功能转发包含 End-End TLP Prefix 的 TLP。编码值为： RW

0b Forwarding Enabled - 允许 Function 发送带 End-End TLP Prefix 的 TLP。

1b Forwarding Blocked - 不允许 Function 发送带 End-End TLP Prefix 的 TLP。

该位影响从相关 Switch/RC 端口退出时的 TLP。它不会影响在 Switch/RC 内部转发的 TLP。它不影响通过关联端口进入，源自关联端口或源自与关联端口集成的 RC 集成设备的 TLP。如 Section 2.2.10.2 中所述，被阻止的 TLP 会报告 TLP Prefix Blocked Error。

该字段默认值为 0b。

该位在支持 End-End TLP Prefix 但不支持转发 End-End TLP Prefix 的根端口中硬连线为 1b。

此位适用于设置了 End-End TLP Prefix Supported 的根端口和 Switch 端口。该位不适用于所有其他情况，这些情况下均为 RsvdP。

公众号·芯片业者相承双

7.5.3.17 Device Status 2 Register (Offset 2Ah)

目前还没有为该寄存器设置一些字段。

对软件来说这个寄存器应当被当做 RsvdZ。

7.5.3.18 Link Capabilities 2 Register (Offset 2Ch)

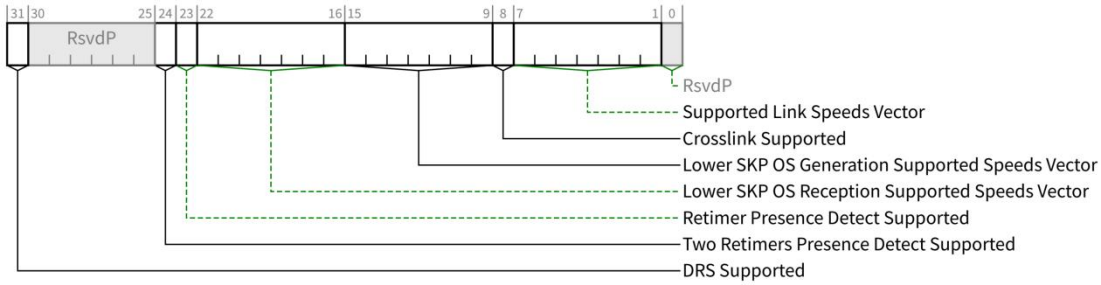

Figure 7-38 Link Capabilities 2 Register

Bit Location	Register Description	Attributes
7:1	**Supported Link Speeds Vector** - 该字段指示关联端口支持的 Link speed。对于每一比特，值 1b 表示支持相应的 Link speed；否则，不支持该 Link speed。有关更多要求，请参见 Section 8.2.1。 各 Bit 定义为： **Bit 0** 2.5GT/s **Bit 1** 5.0GT/s **Bit 2** 8.0GT/s **Bit 3** 16.0GT/s **Bit 4** 32.0GT/s **Bit 6:5** RsvdP 与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。	HwInit/RsvdP
8	**Crosslink Supported** - 设置为 1b 时，该位表示关联的端口支持 Crosslink（see Section 4.2.6.3.1）。当在支持 8.0 GT/s 或更高的链路速率的端口上将其设置为 0b 时，此位表示关联的端口不支持 Crosslink。在仅支持 2.5 GT/s 或 5.0 GT/s 链路速率的端口上设置为 0b 时，此位不提供有关端口的 Crosslink 支持级别的信息。 建议即使在也支持以 8.0 GT/s 或更高的链路速率运行的端口中要求这样做，也要在任何支持 Crosslink 的端口中将该位置 1。 注意：软件在其引用的字段取决于端口是否支持 Crosslink 时需要用到该字段（see Section 7.7.3.4）。 与上游端口关联的 Multi-Function Device 必须在此字段中为所有 Function 报告相同的值。	RO
15:9	**Lower SKP OS Generation Supported Speeds Vector** - 如果此字段非零，则表示端口以指定的速度运行时支持 SRIS，还支持 SKP 有序集传输调度速率的软件控制。 每 Bit 的定义为：	HwInit/RsvdP

公众号·芯片业者相承双

Bit 0 2.5GT/s

Bit 1 5.0GT/s

Bit 2 8.0GT/s

Bit 3 16.0GT/s

Bit 4 32.0GT/s

Bit 6:5 RsvdP

与上游端口关联的 Multi-Function Device 必须在此字段中为所有 Function 报告相同的值。

如果在此字段中设置的 Bit 位，并且在 Supported Link Speeds Vector 中未设置相应的位，则行为是不确定的。

22:16 **Lower SKP OS Reception Supported Speeds Vector** - 如果此字段非零，则表示端口以指定的速度运行时支持 SRIS，并且还支持在 SRIS 中运行时以 SRNS 定义的速率接收 SKP OS。 HwInit/RsvdP

每 Bit 的定义为：

Bit 0 2.5GT/s

Bit 1 5.0GT/s

Bit 2 8.0GT/s

Bit 3 16.0GT/s

Bit 4 32.0GT/s

Bit 6:5 RsvdP

与上游端口关联的 Multi-Function Device 必须在此字段中为所有 Function 报告相同的值。

如果在此字段中设置的 Bit 位，并且在 Supported Link Speeds Vector 中未设置相应的位，则行为是不确定的。

23 **Retimer Presence Detect Supported** - 当设置为 1b 时，该位表示关联的端口支持检测和报告 Retimer 存在。 HwInit/RsvdP

当 Link Capabilities 2 Register 的 Supported Link Speeds Vector 表示支持 16.0 GT/s 或更高的链路速率时，必须在端口中将该位置 1b。

无论支持的链路速率如何，都可以将其设置为 1b。

与上游端口关联的 Multi-Function Device 必须在此字段中为所有 Function 报告相同的值。

24 **Two Retimers Presence Detect Supported** - 当设置为 1b 时，该位表示关联的端口支持检测和报告两个 Retimer 存在。 HwInit/RsvdP

当 Link Capabilities 2 Register 的 Supported Link Speeds Vector 表示支持 16.0 GT/s 或更高的链路速率时，必须在端口中将该位置 1b。

如果 Retimer Presence Detect Supported 字段也设置为 1b，则无论支持的链路速率如何，都可

公众号：芯片业者相承双

以将其设置为 1b。

与上游端口关联的 Multi-Function Device 必须在此字段中为所有 Function 报告相同的值。

31	**DRS Supported** - 置位时表示支持可选的 Device Readiness Status (DRS) capability。 支持 DRS 的 Downstream Port 需要把该位置 1。 支持 FRS 的 Downstream Port 需要把该位置 1。 对于支持 DRS 的 Upstream Port，强烈建议在 Function 0 中将该位置 1。对于与 Upstream Port 相关的所有其他 Function，该位必须为 0。 注：明确允许上游端口发送 DRS 消息，即使 DRS Supported 位为 0。 在与端口无关的 Function 中必须清 0。 其他 Function 此字段为 RsvdP。	HwInit/RsvdP

Software Management of Link Speeds With Earlier Hardware

兼容[PCIe-3.0]之前版本的硬件组件未实现 Link Capabilities 2 寄存器，或者该寄存器为保留。

对于确定不实现 Link Capabilities 2 寄存器或其支持的 Supported Link Speeds Vector 的值为 0000000b 的组件，如果软件想确定支持的链路速率，软件可以读取 Link Capabilities 寄存器的 3:0 位（现在定义为 Max Link Speed），并按以下方式解释该值：

0001b 2.5 GT/s Link speed supported

0010b 5.0GT/s and 2.5 GT/s Link speed supported

对于此类组件，Link Status Register 的 Current Link Speed 字段和 Link Control 2 Register 的 Target Link Speed 字段的值编码与上述相同。

Software Management of Link Speeds With Future Hardware

强烈建议软件首先使用 Supported Link Speeds Vector 而不是 Max Link Speed 字段，以便软件可以确定当前和将来硬件上支持的速度的确切集合。如果以后的规范定义了不需要支持所有较慢速度的链路，则可以避免软件引起混淆。

7.5.3.19 Link Control 2 Register (Offset 30h)

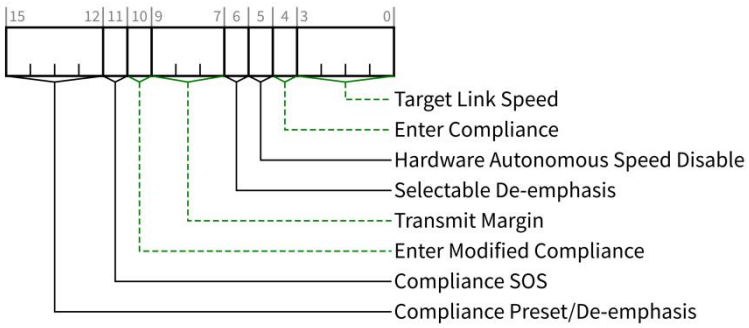

Figure 7-39 Link Control 2 Register

公众号：芯片业者相承双

Bit Location	Register Description	Attributes
3:0	**Target Link Speed** - 对于下游端口，此字段通过限制上游组件在其训练序列中通告的值来设置链路工作速率的上限。 此字段的编码值指向 Link Capabilities 2 寄存器中 Supported Link Speeds Vector 中对应比特的位置。 定义的编码值为： **0001b** Supported Link Speeds Vector field bit 0 **0010b** Supported Link Speeds Vector field bit 1 **0011b** Supported Link Speeds Vector field bit 2 **0100b** Supported Link Speeds Vector field bit 3 **0101b** Supported Link Speeds Vector field bit 4 **0110b** Supported Link Speeds Vector field bit 5 **0111b** Supported Link Speeds Vector field bit 6 所有其他编码值保留。 如果写入的值与 Supported Link Speeds Vector 支持的值不一致，则结果不确定。 如果实现了 Enter Compliance 或 Enter Modified Compliance 字段，则也必须实现此字段。 除非相应的平台/规格要求不同的默认值，否则此字段的默认值是组件支持的最高链路速率（由 Link Capabilities Register 的 Max Link Speed 字段报告）。 对于上游端口和下游端口，当软件使用 Enter Compliance 字段强制链路进入一致性模式时，此字段用于设置目标一致性模式速率。 对于上游端口，如果 Enter Compliance 字段为 0，则此字段无效。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。 只支持 2.5GT/s 的组件此字段值可以硬连线为 0000b。	RWS/RsvdP
4	**Enter Compliance** - 软件可以设置链路两端两个组件的此字段为 1 强制链路进入一致性模式（速率由 Target Link Speed 指示，de-emphasis/preset level 由 Compliance Preset/De-emphasis 字段指示），然后在链路上发送一个 Hot Reset。 基本复位之后的默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。 只支持 2.5GT/s 的组件此字段值可以硬连线为 0b。 该字段仅用于调试，一致性测试目的。仅在调试或一致性测试期间，才允许系统固件和软件修	RWS/RsvdP

公众号：志告业者相承双

改此位。在所有其他情况下，系统必须确保将此位设置为默认值。

5 **Hardware Autonomous Speed Disable** - 设置为 1 时，该位禁止硬件出于设备特定的原因更改链路速率，而不是尝试通过降低链路速率来纠正不可靠的链路操作。此位不会阻止初始化时链路转换为支持的最高公共链路速率。 RWS/RsvdP

对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

允许未实现相关机制的 Function 将该位硬接线为 0b。

默认值为 0b。

6 **Selectable De-emphasis** - 当链路以 5.0 GT/s 的速度运行时，该位用于控制特定情况下链路的发送侧的去加重值。有关详细的使用信息，请参见 Section 4.2.6。 HwInit

编码值为：

1b -3.5 dB

0b -6 dB

当链路未以 5.0 GT/s 的速率运行时，该位的设置无效。仅支持 2.5 GT/s 速率的组件需要将该位硬接线为 0b。

对 Endpoint、PCI Express to PCI/PCI-X bridge 和 Upstream Ports of Switch 该字段不适用且为保留值。

9:7 **Transmit Margin** - 该字段控制发送器引脚上的未经过去加重的电压值。在进入 Polling.Configuration 子状态时，此字段将重置为 000b（有关如何在各种状态下确定发送器电压电平的详细信息，请参见 Chapter 4）。 RWS/RsvdP

编码值为：

000b Normal operating range

001b - 111b 如 Section 8.3.4 所定义，并非所有编码都需要实现。

对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

默认值为 000b。

只支持 2.5GT/s 的组件此字段值可以硬连线为 000b。

该字段仅用于调试，一致性测试目的。仅在调试或一致性测试期间，才允许系统固件和软件修改此字段。在所有其他情况下，系统必须确保将此字段设置为默认值。

10 **Enter Modified Compliance** - 当此位设置为 1b 时，如果 LTSSM 进入 Polling.Compliance 子状态，则设备将发送 Modified Compliance Pattern。 RWS/RsvdP

对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

公众号·芯片业者相承双

默认值为 0b。

只支持 2.5GT/s 的组件此字段值可以硬连线为 0b。

该字段仅用于调试，一致性测试目的。仅在调试或一致性测试期间，才允许系统固件和软件修改此字段。在所有其他情况下，系统必须确保将此字段设置为默认值。

11 **Compliance SOS** - 当设置为 1b 时，要求 LTSSM 在发送 Compliance Pattern 或 Modified Compliance Pattern 时在序列之间发送 SKP 有序集。 RWS/RsvdP

对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

默认值为 0b。

这个字段只在链路速率为 2.5GT/s 和 5.0GT/s 下使用

只支持 2.5GT/s 的组件此字段值可以硬连线为 0b。

15:12 **Compliance Preset/De-emphasis** - RWS/RsvdP

对于 8.0 GT/s 和更高的数据速率：此字段用于设置由于 Enter Compliance 字段为 1b 而导致链路进入的 Polling.Compliance 状态的发送器 Preset 值。 编码在 Section 4.2.3.2 中定义。如果以这种方式进入 Polling.Compliance 时使用保留的 Preset 编码，则结果不确定。

对于 5.0 GT/s 数据速率：此字段用于设置由于 Enter Compliance 字段为 1b 而导致链路进入的 Polling.Compliance 状态的发送器去加重级别。定义的编码为：

0001b -3.5 dB

0000b -6 dB

当链路以 2.5 GT/s 的速度运行时，此字段的设置无效。允许仅支持 2.5 GT/s 速率的组件将该字段硬连线为 0000b。

对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型，只有 Function 0 能控制组件的链路行为。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。

默认值为 000b。

该字段仅用于调试，一致性测试目的。仅在调试或一致性测试期间，才允许系统固件和软件修改此字段。在所有其他情况下，系统必须确保将此字段设置为默认值。

Selectable De-emphasis Usage

可选的 De-emphasis 的设置仅适用于 Root Port 和 Switch Downstream Port。De-emphasis 设置是特定于实现的，并且取决于 Root Port 和 Switch Downstream Port 所在的平台或底座。系统固件或硬件捆绑用于配置可选的去加重值。如果无法使用系统固件设置去加重值（例如，热插拔的 Switch），则必须使用硬件捆绑来设置去加重值。

公众号：芯片业者但承双

7.5.3.20 Link Status 2 Register (Offset 32h)

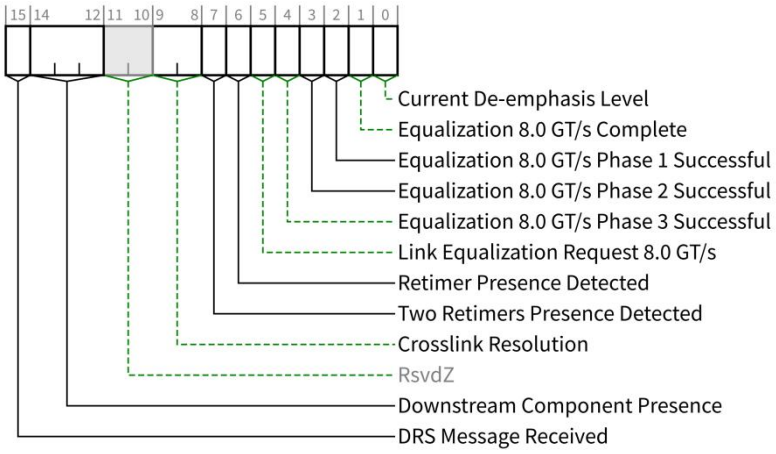

Figure 7-40 Link Status 2 Register

Bit Location	Register Description	Attributes
0	**Current De-emphasis Level** - 当链路以 5.0 GT/s 的速度运行时，该位反映了去加重的等级。 编码值为： **1b** -3.5 dB **0b** -6 dB 当链路未以 5.0 GT/s 的速度运行时，该位的值不确定。 仅支持 2.5 GT/s 速率的组件需要将该位硬接线为 0b。 对于支持速度大于 2.5 GT/s 的组件，与上游端口关联的 Multi-Function Device 必须为所有 Function 的此字段报告相同的值。	RO
1	**Equalization 8.0 GT/s Complete** - 当设置为 1b 时，该位表示以 8.0 GT/s 的数据速率完成发送器均衡过程。Section 4.2.6.4.2 节提供了发送器均衡过程的详细信息以及何时需要将该位设置为 1b 的细节。 默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。支持 8.0GT/s 以下速率的设备需要把该字段硬连线为 0b。	ROS
2	**Equalization 8.0 GT/s Phase 1 Successful** - 当设置为 1b 时，该位表示 8.0 GT/s 发送器均衡过程的 Phase 1 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程的详细信息以及何时需要将该位设置为 1b 的细节。 默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型。在该设备	ROS

公众号：芯片业者相承双

	的所有其他 Function 中，此字段的类型为 RsvdP。支持 8.0GT/s 以下速率的设备需要把该字段硬连线为 0b。	
3	**Equalization 8.0 GT/s Phase 2 Successful** - 当设置为 1b 时，该位表示 8.0 GT/s 发送器均衡过程的 Phase 2 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程的详细信息以及何时需要将该位设置为 1b 的细节。 默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。支持 8.0GT/s 以下速率的设备需要把该字段硬连线为 0b。	ROS
4	**Equalization 8.0 GT/s Phase 3 Successful** - 当设置为 1b 时，该位表示 8.0 GT/s 发送器均衡过程的 Phase 3 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程的详细信息以及何时需要将该位设置为 1b 的细节。 默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。支持 8.0GT/s 以下速率的设备需要把该字段硬连线为 0b。	ROS
5	**Link Equalization Request 8.0 GT/s** - 该位由硬件置 1，以请求在链路上执行 8.0 GT/s 链路均衡过程。有关详细信息，请参见 Section 4.2.3 和 Section 4.2.6.4.2。 默认值为 0b。 对于与上游端口关联的 Multi-Function Device，Function 0 中的该字段为 RWS 类型。在该设备的所有其他 Function 中，此字段的类型为 RsvdP。支持 8.0GT/s 以下速率的设备需要把该字段硬连线为 0b。	RW1CS
6	**Retimer Presence Detected** - 设置为 1b 时，该位表示在最近的链路协商期间存在 Retimer。有关详细信息，请参见 Section 4.2.6.3.5.1。 默认值为 0b。 Link Capabilities 2 Register 的 Retimer Presence Detect Supported 字段设置为 1b 的端口需要实现此字段。 将 Retimer Presence Detect Supported 字段设置为 0b 的端口需要将该位硬接线为 0b。 对于与上游端口关联的 Multi-Function Device，该位必须在 Function 0 中实现，并且在所有其他 Function 中均为 RsvdZ。	ROS/RsvdZ
7	**Two Retimers Presence Detected** - 设置为 1b 时，该位表示在最近的链路协商期间存在两个 Retimer。有关详细信息，请参见 Section 4.2.6.3.5.1。 默认值为 0b。 Link Capabilities 2 Register 的 Two Retimer Presence Detect Supported 字段设置为 1b 的端口需要	ROS/RsvdZ

公众号·芯片业者相互双

实现此字段。

将 Two Retimer Presence Detect Supported 字段设置为 0b 的端口需要将该位硬接线为 0b。

对于与上游端口关联的 Multi-Function Device，该位必须在 Function 0 中实现，并且在所有其他 Function 中均为 RsvdZ。

9:8	**Crosslink Resolution** - 该字段表示 Crosslink 协商的状态。如果 Crosslink Supported 设置为 1 并且端口支持 16.0 GT/s 或更高的数据速率，则必须实现该协议。允许在所有其他端口中实现该字段。如果 Crosslink Supported 设置为 0，则此字段可以硬连线到 01b 或 10b。 编码为： **00b** 不支持 Crosslink Resolution。没有提供有关 Crosslink 协商状态的信息。 **01b** Crosslink 协商解析为上游端口。 **10b** Crosslink 协商解析为下游端口。 **11b** Crosslink 协商未完成。 一旦在该字段中返回 01b 或 10b，则在 LinkUp 期间必须继续返回该值。	RO
14:12	**Downstream Component Presence** - 此字段表示连接到链路的下游组件的存在和 DRS 状态；定义的值是： **000b** **Link Down - 存在状态不确定** **001b** **Link Down - 组件不存在**，意味着 Downstream Port（DP）已确定不存在下游组件。 **010b** **Link Down - 组件存在**，意味着 DP 确定存在下游组件，但数据链路层还不是 Active 状态。 **011b** Reserved **100b** **Link Up - 组件存在**，意味着 DP 确定存在下游组件，但是数据链路层在进入 Active 状态后还未收到 DRS Message。 **101b** **Link Up - 组件存在且收到 DRS**，意味着数据链路层在进入 Active 状态后收到了 DRS Message。 **110b** Reserved **111b** Reserved Downstream Component Presence 状态必须由物理层带内存在检测机制的逻辑“或”确定，如果存在，则还应通过为链路实现的任何带外存在检测机制来确定。 如果未实现带外存在检测机制，则 Downstream Component Presence 必须仅由物理层带内存在检测机制确定。 如果 Slot Control Register 中的 In-Band PD Disable 字段被置 1，则物理层带内存在检测机制必须	RO/RsvdZ

公众号·芯片业者怕承双

	始终指示不存在任何组件。 该字段指示的 Component Presence、 Link Up 和 DRS Received 状态必须反映其可屏蔽状态，这些状态由 SFI Control Register 中的 SFI PD State Mask、SFI DLL State Mask 和 SFI DRS Mask 字段控制。参阅 Section 7.9.23.3。 必须在 Link Capabilities 2 Register 中 DRS Supported 字段设置为 1 的任何下游端口中实现此字段。 其他 Function 类型此字段为 RsvdP。 默认值为 000b。	
15	**DRS Message Received** - 每当端口接收到 DRS Message 时，必须将该位置 1。 在 DL_Down 状态该字段清 0。 必须在 Link Capabilities 2 Register 中 DRS Supported 字段设置为 1 的任何下游端口中实现此字段。 其他 Function 类型此字段为 RsvdP。 默认值为 000b。	RW1C/RsvdZ

7.5.3.21 Slot Capabilities 2 Register (Offset 34h)

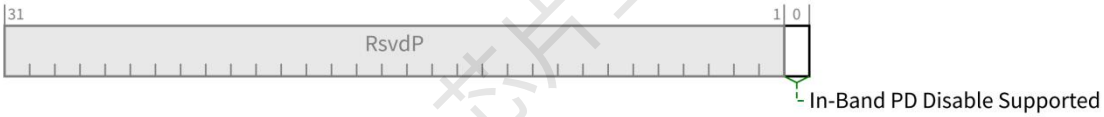

Figure 7-41 Slot Capabilities 2 Register

Bit Location	Register Description	Attributes
0	**In-Band PD Disable Supported** - 为 1 时，此位表示此插槽支持禁用由 Slot Control Register 的 In-Band PD Disable 字段控制的带内存在检测状态的报告。如果插槽不支持带外状态检测机制，则该位必须为 0。	HwInit

7.5.3.22 Slot Control 2 Register (Offset 38h)

目前还没有为该寄存器设置一些字段。

对软件来说这个寄存器应当被当做 RsvdZ。

7.5.3.23 Slot Status 2 Register (Offset 3Ah)

目前还没有为该寄存器设置一些字段。

公众号·芯片业者相承双

对软件来说这个寄存器应当被当做 RsvdZ。

7.6 PCI Express Extended Capabilities

PCI Express Extended Capability 寄存器位于偏移量为 256 或更大的配置空间中，如 Figure 7-42 所示，或者位于 RC 寄存器块（RCRB）中。这些寄存器位于配置空间中时，只能使用 PCI Express 增强配置访问机制（ECAM）进行访问。

使用类似于 PCI Capability structure 的格式的可选或必需 PCI Express Extended Capability 的链表来分配 PCI Express Extended Capability structure。Capability structure 的第一个 DWORD 用于标识 Capability 和其版本，并指向下一个 Capability ，如 Figure 7-42 所示。

每个 Capability structure 必须是 DWORD 对齐的。

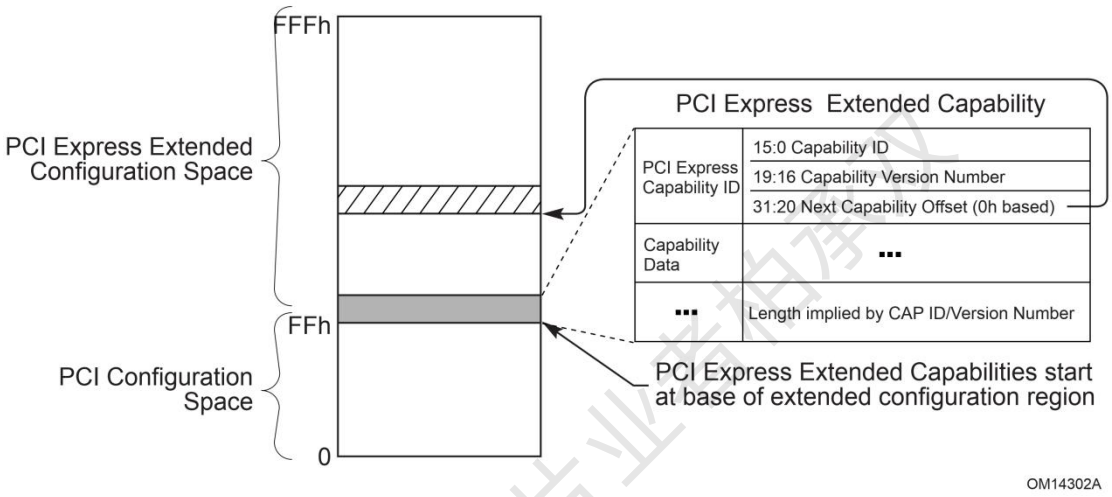

Figure 7-42 PCI Express Extended Configuration Space Layout

7.6.1 Extended Capabilities in Configuration Space

配置空间中的 Extended Capability 始终从 PCI Express Extended Capability header（Section 7.6.3）的偏移为 100h 处开始。当某 Extended Capability header 的 Capability ID 为 0000h、Capability Version 为 0h 且 Next Capability Offset 为 000h，则表示没有下一个 Extended Capability。

7.6.2 Extended Capabilities in the RC Register Block

RC Register Block(RCRB)中的 Extended Capability 始终从具有 PCI Express Extended Capability header（Section 7.6.3）的偏移量 000h 开始。当某 Extended Capability header 的 Capability ID 为 0000h、Capability Version 为 0h 且 Next Capability Offset 为 000h，则表示没有下一个 Extended Capability。

7.6.3 PCI Express Extended Capability Header

所有 PCI Express Extended Capability 必须以 PCI Express Extended Capability header 开头。Figure 7-43 详细说明了 PCI Express Extended Capability header 的寄存器字段的分配；Table 7-37 提供了相应的位定义。

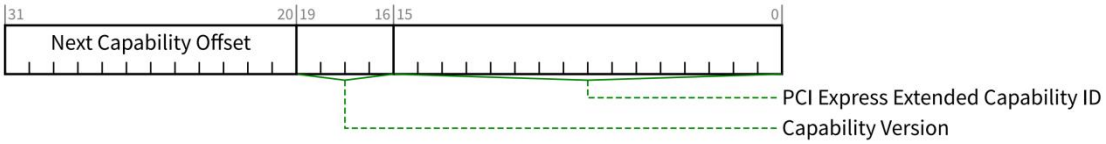

Figure 7-43 PCI Express Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** - 该字段是 PCI-SIG 定义的 ID 号，用于指示扩展功能的性质和格式。	RO
19:16	**Capability Version** - 该字段是 PCI-SIG 定义的版本号，指示存在的功能结构的版本。 （例如，增加新的功能字段）更改 Extended Capability 的规范版本可以增加该字段的值。对 Capability structure 的所有此类更改必须与软件兼容。软件必须检查 Capability Version 是否大于或等于编写软件时定义的最高版本号，因为报告任何此类 Capability Version 的 Function 将包含与该软件兼容的功能结构。	ROS
31:20	**Next Capability Offset** - 此字段指向下一个 PCI Express Capability structure 的偏移量；如果在 Capability 的链表中不存在其他项，则为 000h。 对于在配置空间中实现的扩展功能，此偏移量是相对于与 PCI 兼容的配置空间的开始的，因此必须始终为 000h 或大于 0FFh。 该偏移量的后 2 位保留，并且必须实现为 00b，尽管软件必须屏蔽它们以允许将来使用这些位。	ROS

7.7 PCI and PCIe Capabilities Required by the Base Spec in Some Situations

本规范要求某些 Function 具有以下功能。例如，支持特定数据速率，生成中断等。

7.7.1 MSI Capability Structures

所有能够产生中断的 PCI Express device Function 都必须实现 MSI 或 MSI-X 或两者都实现。

本节介绍了 MSI Capability structure 。Section 7.7.2 节介绍了 MSI-X Capability structure 。

MSI Capability structure 如 Figure 7-44 和 Figure 7-45 所示。每个支持 MSI 的 device Function（在 Multi-Function Device 中）都必须实现自己的 MSI Capability structure 。每个 Function 禁止使用多个 MSI Capability structure，但允许 Function 同时实现 MSI 和 MSI-X Capability structure。

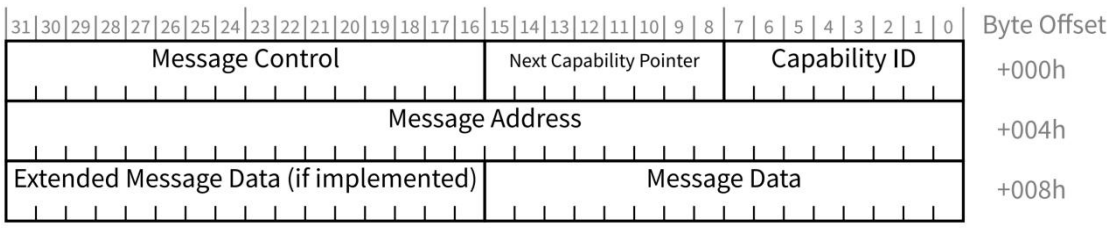

Figure 7-44 MSI Capability Structure for 32-bit Message Address

公众号·芯片业者相承双

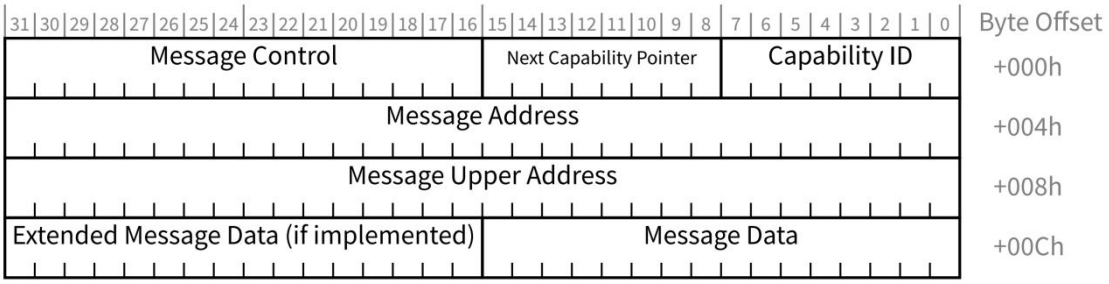

Figure 7-45 MSI Capability Structure for 64-bit Message Address

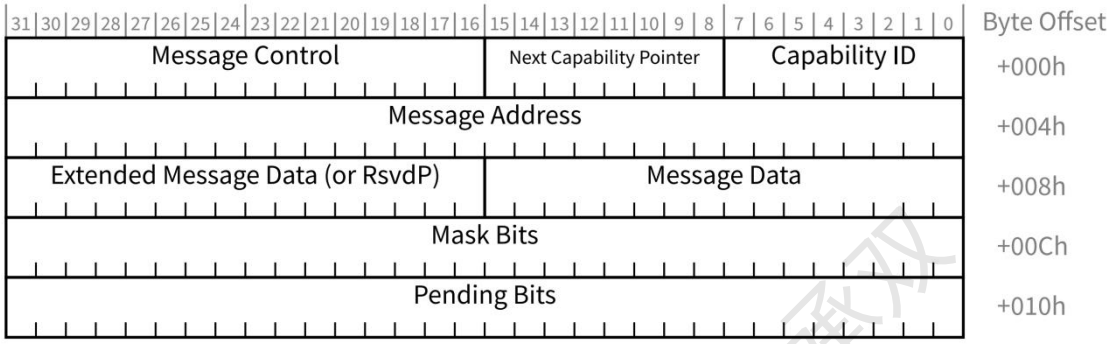

Figure 7-46 MSI Capability Structure for 32-bit Message Address and PVM

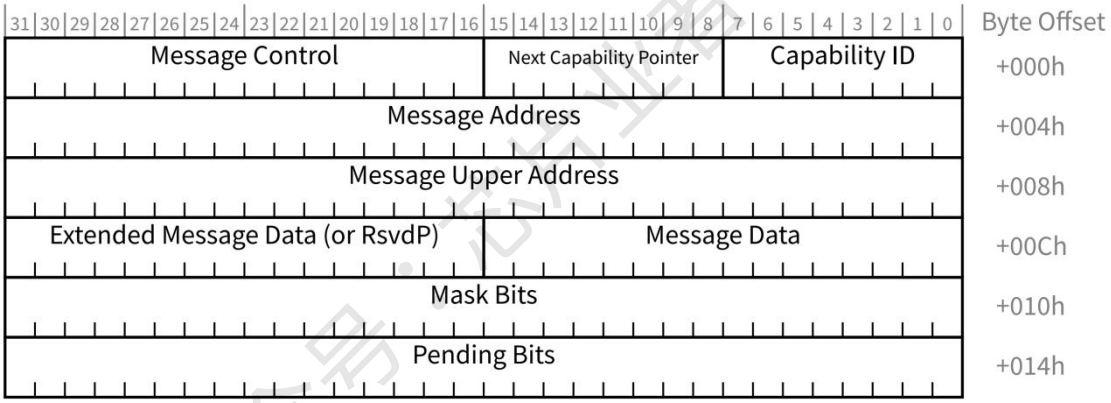

Figure 7-47 MSI Capability Structure for 64-bit Message Address and PVM

为了请求服务，实现了 MSI 的 Function 将 MSI 的 Message Data Register 的内容以及 MSI 的 Extended Message Data Register（如果启用)写入由 MSI 的 Message Address Register 的内容指定的地址(并且,当使用 64 位消息地址时,还可以选择写入 MSI 的 Message Upper Address Register）。读取由消息地址寄存器的内容指定的地址会产生未定义的结果。

支持 MSI 的 Function 实现了 Figure 7-44 至 Figure 7-47 中所示的四个 MSI Capability structure layout 之一,具体取决于实现的需求。一个实现了 MSI 的 Legacy Endpoint 需要实现 32 位或 64 位 Message Address 版本。一个实现了 MSI 的 PCI Express Endpoint 需要实现 64 位 Message Address 版本。MSI 的 Message Control Register 指示了 Function 的 Capability，并提供对 MSI 的系统软件控制。

以下各节将进一步描述每个字段。

7.7.1.1 MSI Capability Header (Offset 00h)

MSI Capability Header 用于在 PCI 配置空间功能列表中枚举 MSI Capability structure。Figure 7-48 详细说明了 MSI Capability Header 中寄存器字段的分配；Table 7-38 提供了相应的字段的定义。

公众号：芯片业者相承双

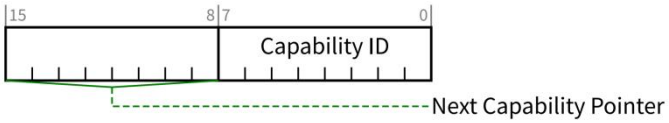

Figure 7-48 MSI Capability Header

Bit Location	Register Description	Attributes
7:0	**Capability_ID** - MSI Capability structure 的 Capability_ID 为 05h。	RO
15:8	**Next Capability Pointer** - 指向该 Function 的 Capabilities List 中下一个 Capability 的偏移地址。如果没有下一个 Capability，此字段值为 00h。	RO

7.7.1.2 Message Control Register for MSI (Offset 02h)

该寄存器提供了对 MSI 的系统软件控制。默认情况下，MSI 被禁用。如果 MSI 和 MSI-X 都被禁用，则该 Function 使用 INTx 中断（如果支持）请求服务。系统软件可以通过将该寄存器的 Bit-0 置 1 来启用 MSI。允许系统软件修改 MSI 的 Message Control Register 的可读写字段。不允许设备驱动程序修改 MSI 的 Message Control Register 的可读写字段。

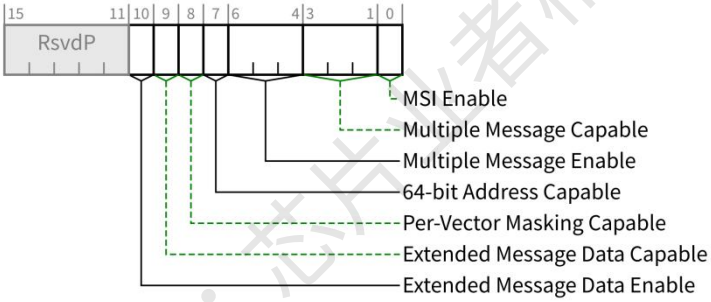

Figure 7-49 Message Control Register for MSI

Bit Location	Register Description	Attributes
0	**MSI Enable** - 如果该字段置 1 且 MSI-X 的 Message Control Register 中的 MSI-X Enable（Section 7.9.2）字段为 0，则允许该 Function 使用 MSI 请求服务，并禁止使用 INTx 中断。系统配置软件将该位置 1 以启用 MSI。禁止设备驱动程序写入此字段以屏蔽 Function 的服务请求。有关 INTx 中断的控制，请参见 Section 7.5.1.1。 如果该字段清 0，则禁止该 Function 使用 MSI 请求服务。 默认值为 0b 。	RW
3:1	**Multiple Message Capable** - 系统软件读取该字段以确定请求的向量数。请求向量的数量必须以 2 的幂（如果一个 Function 需要三个向量，则通过将此字段初始化为 010b 来请求四个）对齐。 编码定义为： **000b** 1 vector requested **001b** 2 vector requested	RO

	010b 4 vector requested	
	011b 8 vector requested	
	100b 16 vector requested	
	101b 32 vector requested	
	110b reserved	
	111b reserved	
	该字段对系统软件只读，表示当前 PCIe 设备可以使用几个中断向量号，在不同的 PCIe 设备中该字段的值并不不同。该字段与 Multiple Message Enable 字段的含义不同，该字段表示，当前 PCIe 设备支持的中断向量个数，而 Multiple Message Enable 字段是系统软件分配给 PCIe 设备实际使用的中断向量个数。	
6:4	**Multiple Message Enable** - 软件会写入此字段以指示分配的向量数（等于或小于请求的向量数）。分配向量的数量与 2 的幂对齐。如果某个 Function 请求四个向量（Multiple Message Capable 编码为 010b），则系统软件可以通过分别向该字段写入 010b，001b 或 000b 来分配四个，两个或一个向量。 当设置了 MSI Enable 时，将为一个 Function 分配至少 1 个向量。编码定义为： **000b** 1 vector allocated **001b** 2 vector allocated **010b** 4 vector allocated **011b** 8 vector allocated **100b** 16 vector allocated **101b** 32 vector allocated **110b** reserved **111b** reserved 默认值为 000b 。 在系统的中断向量资源并不紧张时，Multiple Message Capable 字段和该字段的值相等；而资源紧张时，该字段的值可能小于 Multiple Message Capable 字段的值。	RW
7	**64-bit Address Capable** - 如果置 1，则该 Function 能够发送 64 位消息地址。如果清 0，则该 Function 将无法发送 64 位消息地址。如果 Function 是 PCI Express Endpoint，则必须将该位置 1。	RO
8	**Per-Vector Masking Capable** - 如果置 1，则该 Function 支持 MSI 向量掩码。如果清 0，则该 Function 不支持 MSI 向量掩码。如果 Function 是 SR-IOV 设备中的 PF 或 VF，则必须将该位置 1。	RO
9	**Extended Message Data Capable** - 如果置 1，则该 Function 能够提供 Extended Message Data。如果清 0，则该 Function 不支持提供 Extended Message Data。	RO
10	**Extended Message Data Enable** - 如果置 1，则启用该 Function 提供 Extended Message Data 的功能。	RW/RO

公众号·芯片业者相关

如果清 0，则该 Function 不启用 Extended Message Data 功能。

该字段的默认值为 0b。

如果 Extended Message Data Capable 字段为 1b，则该字段必须为可读写状态；否则，必须将其硬连线到 0b。

7.7.1.3 Message Address Register for MSI (Offset 04h)

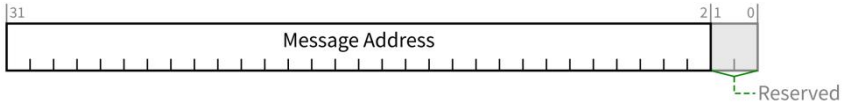

Figure 7-50 Message Address Register for MSI

Bit Location	Register Description	Attributes
1:0	**Reserved** - 读此字段永远返回 0。写此字段没影响。	RsvdP
31:2	**Message Address** - 系统指定的消息地址。 如果 MSI Enable 字段置 1，则此寄存器的内容为 MSI 事务指定 DWORD 对齐的地址（Address[31:02]）。Address[1:0]设置为 00b。 此字段的默认值为 undefined。	RW

7.7.1.4 Message Upper Address Register for MSI (Offset 08h)

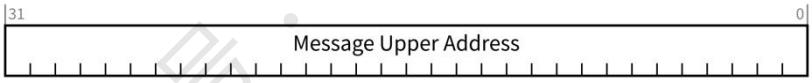

Figure 7-51 Message Upper Address Register for MSI

Bit Location	Register Description	Attributes
31:0	**Message Upper Address** - 系统指定的消息地址的高 32 bit。 仅当 Function 支持 64 位消息地址（64-bit Address Capable 置 1）时，才实现该寄存器。PCI Express Endpoint 必需实现该寄存器，而对于其他 Function 类型则是可选的。 如果 MSI Enable 置 1，则该寄存器的内容（如果非零）指定 64 位消息地址（Address [63:32]）的高 32 位。如果该寄存器的内容为零，则 Function 使用 Message Address 字段指定的 32 位地址。 此字段的默认值为 undefined。	RW

公众号·芯片业者怕承双

7.7.1.5 Message Data Register for MSI (Offset 08h or 0Ch)

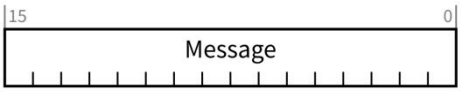

Figure 7-52 Message Data Register for MSI

Bit Location	Register Description	Attributes
31:0	**Message Data** - 系统指定的消息数据。 如果 MSI Enable 置 1，则 Function 使用消息数据的低 16 位发送 DWORD Memory Write transaction 。所有 4 Byte Enable 均已置 1。 Multiple Message Enable 字段定义了允许 Function 修改以生成其系统软件分配向量的低阶消息数据字段的数量。例如，Multiple Message Enable 为 010b 表示该 Function 已分配了四个向量，并允许修改消息数据的 Bit 1 和 0（Function 修改较低的消息数据位以生成分配的向量数量）。如果 Multiple Message Enable 字段为 000b，则不允许该 Function 修改消息数据。 此字段的默认值为 undefined。 当 MSI Enable 位有效时，该字段存放 MSI 报文使用的数据。该字段保存的数值与处理器系统相关，在 PCIe 设备进行初始化时，处理器将初始化该字段，而且不同的处理器填写该字段的规则并不相同。如果 Multiple Message Enable 字段不为 0b000 时(即该设备支持多个中断请求时)，PCIe 设备可以通过改变 Message Data 字段的低位数据发送不同的中断请求。PCIe 总线规定当一个设备使用 MSI 中断机制时，最多可以使用 32 个中断向量，从而一个设备最多可以发送 32 种中断请求。	RW

7.7.1.6 Extended Message Data Register for MSI (Optional)

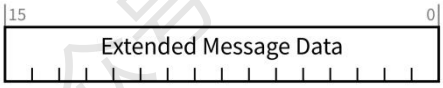

Figure 7-53 Extended Message Data Register for MSI

Bit Location	Register Description	Attributes
31:0	**Extended Message Data** - 系统指定的消息数据。 该寄存器是可选的。对于没有实现 Per-vector Masking 的 MSI Capability structure，如果 Extended Message Data Capable 被置 1 则必须实现它；否则，它不在 MSI Capability structure 之内且未定义。对于带有 Per-vector Masking 的 MSI Capability structure，如果 Extended Message Data Capable 被置 1 则必须实现它；否则为 RsvdP。如果 Extended Message Data Capable 置 1，则 DWORD Memory Write transaction 使用 Extended Message Data 的高 16 位；否则，其高 16 位为 0000h。 该字段的默认值为 0000h。	RW

公众号·芯片业者相承攻

7.7.1.7 Mask Bits Register for MSI (Offset 0Ch or 10h）

Mask Bits Register 由 32 位组成，其中每一位对应一种中断请求。当相应位为 1 时表示对应的中断请求被屏蔽，为 0 时表示允许该中断请求。系统软件可读写该字段，系统初始化时该字段为全 0，表示允许所有中断请求。该字段和 Pending Bits 字段对于 MSI 中断机制是可选字段，但是 PCIe 总线规范强烈建议所有 PCIe 设备支持这两个字段。

Pending Bits Register。该寄存器对于系统软件是只读位，PCIe 设备内部逻辑可以改变该字段的值。该字段由 32 位组成，并与 PCIe 设备使用的 MSI 中断一一对应。该字段需要与 Mask Bits 字段联合使用。

当 Mask Bits 字段的相应位为 1 时，如果 PCIe 设备需要发送对应的中断请求时，Pending Bits 字段的对应位将被 PCIe 设备的内部逻辑置 1，此时 PCIe 设备并不会使用 MSI 报文向中断控制器提交中断请求；当系统软件将 Mask Bits 字段的相应位从 1 改写为 0 时，PCIe 设备将发送 MSI 报文向处理器提交中断请求，同时将 Pending Bit 字段的对应位清零。在设备驱动程序的开发中，有时需要联合使用 Mask Bits 和 Pending Bits 字段防止处理器丢弃中断请求。

该寄存器是可选的。如果 Per-Vector Masking Capable 为 1，则实现该寄存器（Section 7.7.1.2）。该寄存器在能力范围内的偏移量取决于 64-bit Address Capable 的值（Section 7.7.1.2）。

屏蔽位和待处理位寄存器使软件可以禁用或推迟基于每个向量的消息发送。

MSI 向量的编号为 0 到 N-1，其中 N 是由软件分配的向量的数量。每个向量都与 Mask Bits 和 Pending Bits 寄存器中的相应编号位相关联。

Multiple Message Capable 字段指示实现了多少个矢量（带有关联的 Mask 和 Pending ）。所有未实现的 Mask 和 Pending 位均为保留位。

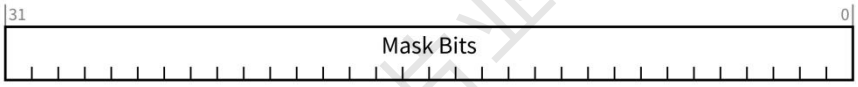

Figure 7-54 Mask Bits Register for MSI

Bit Location	Register Description	Attributes
31:0	**Mask Bits** - 对于设置的每个 Mask 位，禁止该 Function 发送对应的消息。 默认值为 0。	RW

7.7.1.8 Pending Bits Register for MSI (Offset 10h or 14h)

该寄存器是可选的。如果 Per-Vector Masking Capable 为 1，则实现了该寄存器（Section 7.7.1.2）。

该寄存器在能力范围内的偏移量取决于 64-bit Address Capable 的值（Section 7.7.1.2）。

有关此寄存器的其他要求，请参见 Section 7.7.1.7。

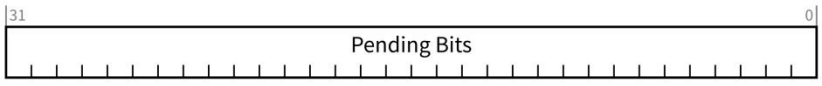

Figure 7-55 Pending Bits Register for MSI

公众号：芯片业者相承双

Bit Location	Register Description	Attributes
31:0	**Pending Bits** - 对于每个设置的 Pending 位，该 Function 都有一个待处理的关联消息。 默认值为 0。	RO

7.7.2 MSI-X Capability and Table Structure

MSI-X Capability 中断机制与 MSI Capability 的中断机制类似。PCIe 总线引出 MSI-X 机制的主要目的是为了**扩展 PCIe 设备使用中断向量的个数**，同时**解决 MSI 中断机制要求使用中断向量号连续所带来的问题**。

MSI 中断机制最多只能使用 32 个中断向量，而 MSI-X 可以使用更多的中断向量。目前 Intel 的许多 PCIe 设备支持 MSI-X 中断机制。与 MSI 中断机制相比，MSI-X 机制更为合理。首先 MSI-X 可以支持更多的中断请求，但是这并不是引入 MSI-X 中断机制最重要的原因。因为对于多数 PCIe 设备，32 种中断请求已经足够了。而引入 MSI-X 中断机制的主要原因是，使用该机制不需要中断控制器分配给该设备的中断向量号连续。

如果一个 PCIe 设备需要使用 8 个中断请求时，如果使用 MSI 机制时，Message Data 的[2:0]字段可以为 0b000~0b111，因此可以发送 8 种中断请求，但是这 8 种中断请求的 Message Data 字段必须连续。在许多中断控制器中，Message Data 字段连续也意味着中断控制器需要为这个 PCIe 设备分配 8 个连续的中断向量号。

有时在一个中断控制器中，虽然具有 8 个以上的中断向量号，但是很难保证这些中断向量号是连续的。因此中断控制器将无法为这些 PCIe 设备分配足够的中断请求，此时该设备的“Multiple Message Enable”字段将小于“Multiple Message Capable”。

而使用 MSI-X 机制可以合理解决该问题。在 MSI-X Capability 结构中，每一个中断请求都使用独立的 Message Address 字段和 Message Data 字段，从而中断控制器可以更加合理地为该设备分配中断资源。

与 MSI Capability 寄存器相比，MSI-X Capability 寄存器使用一个数组存放 Message Address 字段和 Message Data 字段，而不是将这两个字段放入 Capability 寄存器中，本篇将这个数组称为 MSI-X Table。从而当 PCIe 设备使用 MSI-X 机制时，每一个中断请求可以使用独立的 Message Address 字段和 Message Data 字段。

除此之外 MSI-X 中断机制还使用了独立的 Pending Table 表，该表用来存放与每一个中断向量对应的 Pending 位。这个 Pending 位的定义与 MSI Capability 寄存器的 Pending 位类似。MSI-X Table 和 Pending Table 存放在 PCIe 设备的 BAR 空间中。MSI-X 机制必须支持这个 Pending Table，而 MSI 机制的 Pending Bits 字段是可选的。

MSI-X Capability structure 如 Figure 7-56 所示。每个 Function 禁止使用多 MSI-X Capability structure，但允许 Function 同时实现 MSI Capability structure 和 MSI-X Capability structure。

与直接包含该函数向量的所有控制/状态信息的 MSI Capability 结构相反，MSI-X Capability 结构则指向 MSI-X 表结构和 MSI-X PBA 结构（待处理的位数组结构）），每个都驻留在内存空间中（请参见图 7-57 和图 7-58）。

MSI Capability structure 直接给出 Function 的向量的所有控制/状态信息，与此相反，MSI-X Capability structure 指向 MSI-X Table structure 和 MSI-X PBA structure（Pending Bit Array structure），每个结构都驻留在内存空间中（Figure 7-57 和 Figure 7-58）。

每个 structure 由属于该 Function 的基址寄存器（BAR）映射，该地址位于 Configuration Space 中的 10h 开始，或者位于 Enhanced Allocation capability 中。BAR Indicator register（BIR）指示哪个 BAR（或使用 Enhanced Allocation 时的 BEI），以及 QWORD 对齐的偏移量指示结构相对于与 BAR 关联的基址开始的位置。BAR 允许为 32 位或 64 位，但必须映射内存空间。允许一个 Function 使用相同的 BAR 映射两个 structure，或使用不同的 BAR 映射每个 structure。

MSI-X Table structure（Figure 7-57）通常包含多个表项，每个表项都由几个字段组成：Message Address、Message Upper Address、Message Data 和 Vector Control 。每个表项都可以指定一个唯一的向量。

公众号：芯片业者柏承双

Pending Bit Array (PBA) structure 如 Figure 7-58 所示，其中包含 Function 的 Pending Bit，每个表项一个，组织为 QWORD 中的位的打包数组。最后一个 QWORD 不一定完全填充。

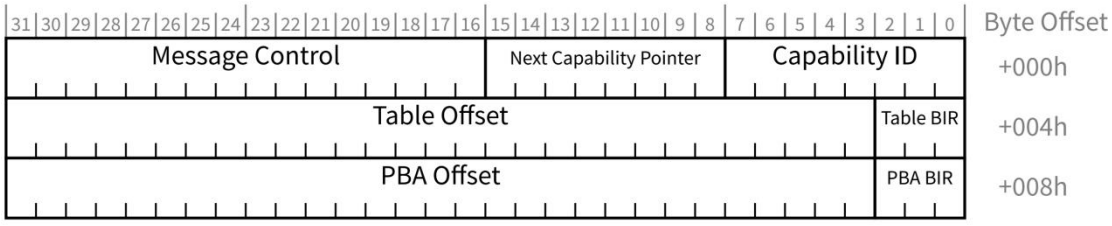

Figure 7-56 MSI-X Capability Structure

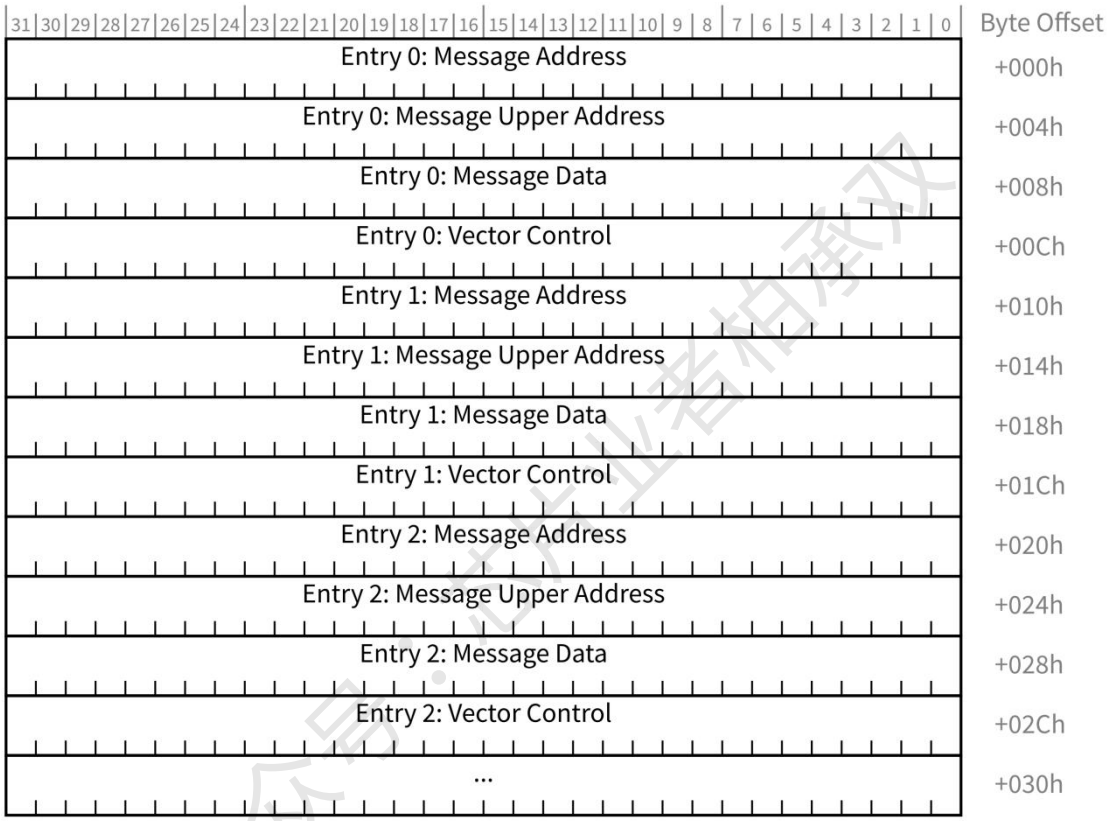

Figure 7-57 MSI-X Table Structure

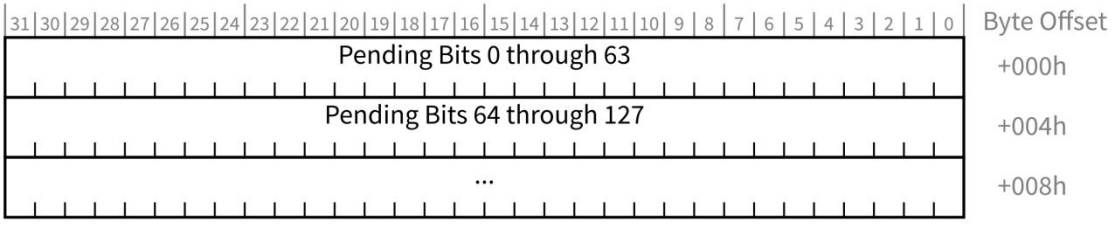

Figure 7-58 MSI-X PBA Structure

为了使用给定的 MSI-X Table entry 请求服务，Function 使用 Message Data、Message Upper Address 以及 Message Address 字段的内容来发起一个 DWORD Memory Write transaction。对 MSI-X 消息目标地址的 Memory Read 会产生未定义的结果。

如果为 MSI-X Table 或 MSI-X PBA 映射地址空间的 Enhanced Allocation capability 中的 Base Address Register 或条目还映射了与 MSI-X

structure、位置（例如 CSR）不相关的其他可用地址空间，则 MSI-X structure 所在的地址空间中使用的另一个地址不得与任何自然对齐的 4 KB 地址范围共享。这允许系统软件在合适的情况下针对 MSI-X structure 和其他地址空间使用不同的处理器属性。（某些处理器体系结构不支持具有与同一自然对齐的 4 KB 物理地址范围关联的不同处理器属性。）允许 MSI-X Table 和 MSI-X PBA 在自然对齐的 4 KB 地址范围内共存，但它们不得相互重叠。

IMPLEMENTATION NOTE

Dedicated BARs and Address Range Isolation

为了使系统软件能够将 MSI-X 结构映射到不同的处理器页面上以改善访问控制，建议 Function 为 MSI-X Table 和 MSI-X PBA 专门分配单独的 BAR，或者提供比最低要求更多的地址范围隔离策略。

如果专用的单独基址寄存器不可行，建议 Function 为 MSI-X Table 和 MSI-X PBA 使用一个 BAR。

如果专用的基址寄存器不可行，则建议 Function 将 MSI-X structure 与非 MSI-X structure 隔离，对齐范围为 8 KB，而不是强制对齐的 4 KB 范围。

例如，如果基址寄存器需要为包含 128 个条目的 MSI-X 表映射 2 KB，为包含 128 位的 MSI-X PBA 映射 16 字节，为与 MSI-X 无关的寄存器映射 64 字节，则以下是可接受的实现。基址寄存器请求 8 KB 的总地址空间，映射非 MSI-X 寄存器的前 64 个字节，映射从 4 KB 偏移量开始的 MSI-X Table，映射从 6 KB 偏移量开始的 MSI-X PBA 。

共享基址寄存器的一种更好的实现方式是，它请求 16 KB 的总地址空间，映射非 MSI-X 寄存器的前 64 个字节，映射以 8 KB 的偏移量开始的 MSI-X Table，并映射 MSI-X PBA 以 12 KB 的偏移量开始。

IMPLEMENTATION NOTE

MSI-X Memory Space Structures in Read/Write Memory

MSI-X Table 和 MSI-X PBA 结构被定义为它们可以驻留在设备上的通用读/写存储器中，以便于实现和增加灵活性。为此，所有包含的字段都不要求是只读的，并且对事务对齐和大小也有限制。

对于对 MSI-X Table 和 MSI-X PBA 字段的所有访问，软件必须使用对齐的完整 DWORD 或对齐的完整 QWORD 事务；否则，结果是不确定的。

MSI-X Table 条目和 Pending Bits 分别用 0 到 N-1 编号，其中 N-1 由 MSI-X 的 Message Control Register 中的 Table Size 字段指示。对于给定的任意 MSI-X Table 条目 k，可以使用以下公式计算其起始地址：

$$\text{entry starting address} = \text{Table base} + k \times 16$$

Equation 7-1 MSI-X Starting Address

对于关联的 Pending Bits k，可以使用以下公式计算其用于 QWORD 访问的地址和该 QWORD 中比特数：

$$\text{QWORD address} = \text{PBA base} + (k \operatorname{div} 64) \times 8$$

$$\text{QWORD bit\#} = k \bmod 64$$

Equation 7-2 MSI-X PBA QWORD Access

$$\text{QWORD address} = \text{PBA base} + (k \operatorname{div} 64) \times 8$$

$$\text{QWORD bit\#} = k \bmod 64$$

Equation 7-2 MSI-X PBA QWORD Access

选择读取具有 DWORD 访问权限的 Pending Bits K 的软件可以使用以下公式：

公众号·志片业者相承双

$$\text{DWORD address} = \text{PBA base} + (k \operatorname{div} 32) \times 4$$

$$\text{DWORD bit\#} = k \bmod 32$$

Equation 7-3 MSI-X PBA DWORD Access

在以下各节中将进一步描述 MSI-X Capability、MSI-X Table 和 MSI-X PBA 结构中的每个字段。在 MSI-X Capability structure 中，保留寄存器和比特位在读取时始终返回 0，而写入操作无效。在 MSI-X Table 和 PBA 结构中，保留字段具有特殊的规则。

7.7.2.1 MSI-X Capability Header (Offset 00h)

MSI-X Capability Header 用于在 PCI 配置空间功能列表中枚举 MSI-X Capability structure 。Figure 7-56 详细说明了 MSI-X Capability Header 中寄存器字段的分配；Table 7-46 提供了相应的比特位定义。

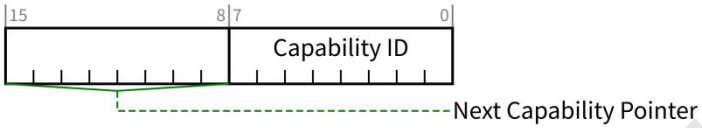

Figure 7-59 MSI-X Capability Header

Bit Location	Register Description	Attributes
7:0	**Capability_ID** - MSI-X Capability structure 的 Capability_ID 为 11h。	RO
15:8	**Next Capability Pointer** - 指向该 Function 的 Capabilities List 中下一个 Capability 的偏移地址。如果没有下一个 Capability，此字段值为 00h。	RO

7.7.2.2 Message Control Register for MSI-X (Offset 02h)

默认情况下，MSI-X 被禁用。如果 MSI 和 MSI-X 都被禁用，则该 Function 通过 INTx 中断（如果支持）请求服务。系统软件可以通过将该寄存器的 Bit-15 设置位 1 来启用 MSI-X。允许系统软件修改消息控制寄存器的可读写字段。不允许设备驱动程序修改消息控制寄存器的可读写字段。

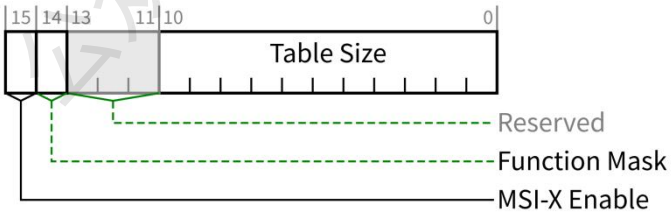

Figure 7-60 Message Control Register for MSI-X

Bit Location	Register Description	Attributes
10:0	**Table Size** - 系统软件读取此字段，以确定 MSI-X Table 的大小 N（编码为 N-1）。 例如，返回值 000 0000 0011b 表示 MSI-X Table 大小为 4。	RO
13:11	**Reserved** - 读返回 0；写无影响。	RsvdP

公众号·芯片业者相承双

14	**Function Mask** - 如果置 1，则与 Function 关联的所有向量都将被屏蔽，无论其每个向量的 Mask 比特位的值如何。 如果清 0，则每个向量的 Mask 比特位都会确定向量是否被掩蔽。 MSI-X Function Mask 设置为 1 或清 0 对每个向量的 Mask 比特位的值没有影响。 该位的默认值为 0b。	RW
15	**MSI-X Enable** - 如果置 1 且 MSI 的 Message Control Register 中的 MSI Enable 字段（Section 7.7.1.2）被清 0，则该 Function 允许使用 MSI-X 来请求服务，并且禁止使用 INTx 中断（如果已实现）。系统配置软件将该位置 1 以启用 MSI-X。禁止设备驱动程序写入此位以屏蔽 Function 的服务请求。 如果清 0，则禁止该 Function 使用 MSI-X 请求服务。 该位的默认值为 0b。	RW

7.7.2.3 Table Offset/Table BIR Register for MSI-X (Offset 04h)

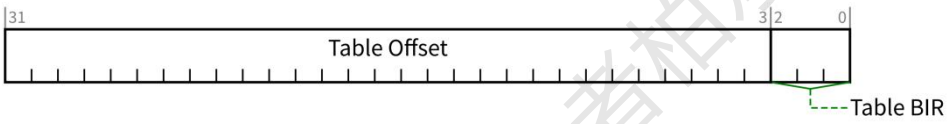

Figure 7-61 Table Offset/Table BIR Register for MSI-X

Bit Location	Register Description	Attributes
2:0	**Table BIR** - 指示哪个 Function 的基址寄存器(始于配置空间中的 10h)或具有匹配的 BAR Equivalent Indicator（BEI）的 Enhanced Allocation capability 中的条目用于将 Function 的 MSI-X 表映射到内存空间。 定义的编码为： **0** Base Address Register 10h **1** Base Address Register 14h **2** Base Address Register 18h **3** Base Address Register 1Ch **4** Base Address Register 20h **5** Base Address Register 24h **6** Reserved **7** Reserved 对于 64 位基址寄存器，Table BIR 指示较低的 DWORD。对于具有 Type 1 Configuration Space header 的 Function，BIR 值 2 到 5 也被保留。	RO

公众号·芯片业者相承双

31:3	**Table Offset** - 用作与 Function 的基址寄存器之一所包含的地址的偏移量，以指向 MSI-X Table 的基址。Table BIR 的低 3 位由软件屏蔽（设置为零），以形成 32 位 QWORD 对齐的偏移量。	RO

7.7.2.4 PBA Offset/PBA BIR Register for MSI-X (Offset 08h)

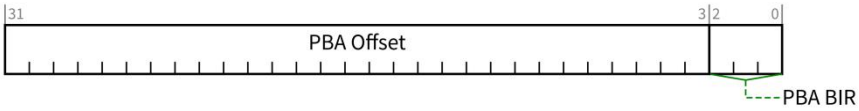

Figure 7-62 PBA Offset/PBA BIR Register for MSI-X

Bit Location	Register Description	Attributes
2:0	**PBA BIR** - 指示哪个 Function 的基址寄存器（位于配置空间中的 10h 开始）或具有匹配 BEI 的增强型分配功能中的条目用于将 Function 的 MSI-X PBA 映射到内存空间。PBA BIR 值定义与 Table BIR 相同。	RO
31:3	**PBA Offset** - 用作与该 Function 的基址寄存器之一所包含的地址的偏移量，以指向 MSI-X PBA 的基址。软件将较低的 3 个 PBA BIR 位屏蔽掉（设置为零），以形成 32 位 QWORD 对齐的偏移量。	RO

7.7.2.5 Message Address Register for MSI-X Table Entries

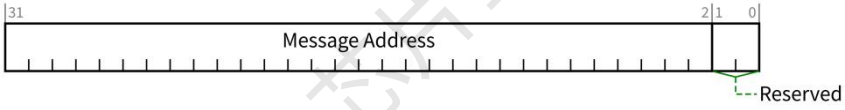

Figure 7-63 Message Address Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
1:0	**Reserved** - 为使 DWORD 正确对齐，软件必须始终向这两个位写入零;否则结果是不确定的。 该字段的默认值为 00b。 这些位允许为只读或读写。	RO or RW
31:2	**Message Address** - 系统指定的消息低 32 位地址。 对于 MSI-X 消息，MSI-X Table 条目中该字段的内容指定 Memory Write transaction 的 DWORD 对齐地址的低 32 位地址。 此字段的默认值为 undefined。	RW

公众号：芯片业者程承双

7.7.2.6 Message Upper Address Register for MSI-X Table Entries

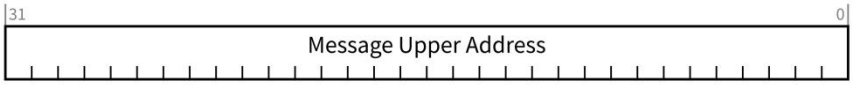

Figure 7-64 Message Upper Address Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
31:0	**Message Upper Address** - 系统指定的消息高 32 位地址。 如果该字段为零，则使用 32 位地址消息。如果此字段为非零，则使用 64 位地址消息。 此字段的默认值为 undefined。	RW

7.7.2.7 Message Data Register for MSI-X Table Entries

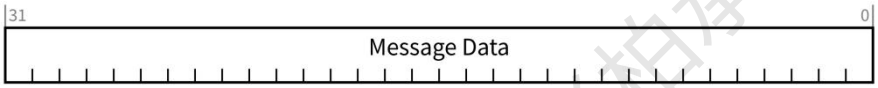

Figure 7-65 Message Data Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
31:0	**Message Data** - 系统指定的消息数据。 对于 MSI-X 消息，MSI-X Table 条目中此字段的内容指定 DWORD Memory Write transaction 的 32 位数据有效载荷。所有 4 Byte Enables 均已置 1。 与用于 MSI 消息的消息数据相反，MSI-X 消息中的低阶消息数据位未被 Function 修改。 该字段是可读写的。 此字段的默认值为 undefined。	RW

7.7.2.8 Vector Control Register for MSI-X Table Entries

如果某个 Function 实现了 TPH Requester Extended Capability structure 和 MSI-X Capability structure，则该 Function 可以选择在每个条目中使用 MSI-X Table 条目的 Vector Control Register 来存储 Steering Tag。见 Section 6.17。

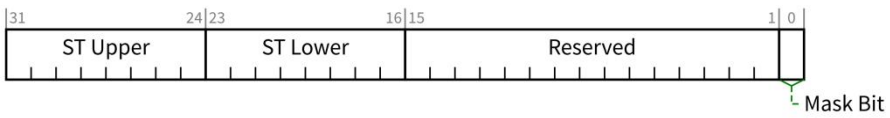

Figure 7-66 Vector Control Register for MSI-X Table Entries

公众号：芯片业者相承双

Bit Location	Register Description	Attributes
0	**Mask Bit** - 当该位置 1 时，禁止该 Function 使用该 MSI-X Table 条目发送消息。但是，用相同向量编程的任何其他 MSI-X Table 条目仍将能够发送等效消息，除非它们也被屏蔽。 该位的默认值为 1b（条目被屏蔽）。	RW
15:1	**Reserved** - 默认情况下，这些位的值必须为 0。但是，为了将来可能使用，软件在修改其他矢量控制位的值时必须保留这些保留位的值。如果软件修改了这些保留位的值，则结果不确定。 这些位允许为 RsvdP 或可读写。	RW or RsvdP
23:16	**ST Lower** - 如果 Function 实现了 TPH Requester Extended Capability structure，并且 ST Table Location 指示值为 10b，则此字段包含 Steering Tag 的低 8 位，并且必须是可读写的。 否则，允许此字段为可读写或 RsvdP，并且为了将来可能的使用，软件在修改其他向量控制位的值时必须保留这些保留位的值，否则结果不确定。 该字段的默认值为 00h。	RW/RsvdP
31:24	**ST Upper** - 如果 Function 实现了 TPH Requester Extended Capability structure，并且 ST Table Location 指示值为 10b，并且 Extended TPH Requester Supported 被置 1，则此字段包含 Steering Tag 的高 8 位，并且必须是可读写的 。 否则，允许此字段为可读写或 RsvdP，并且为了将来可能的使用，软件在修改其他矢量控制位的值时必须保留这些保留位的值，否则结果不确定。 该字段的默认值为 00h。	RW/RsvdP

7.7.2.9 Pending Bits Register for MSI-X PBA Entries

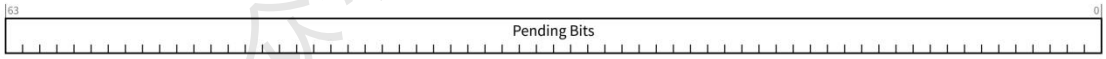

Figure 7-67 Pending Bits Register for MSI-X PBA Entries

Bit Location	Register Description	Attributes
31:0	**Pending Bits** - 对于每个已设置为 1 的 Pending Bit，该 Function 都有一个用于关联的 MSI-X Table 条目的待处理消息。 没有关联的 MSI-X Table 条目的待处理位被保留。默认情况下，Reserved Pending 字段的值必须为 0b。 软件从不应该写入，而应该仅读取 Pending Bit。如果软件写入 Pending Bit，则结果不确定。 每个 Pending Bit 的默认值为 0b。 这些位允许为只读或读写。	RW or RO

公众号·芯片业者相承双

7.7.3 Secondary PCI Express Extended Capability

在满足以下任一条件的任何 Function 或 RCRB 中，必须实现 Secondary PCI Express Extended Capability structure：

- Supported Link Speeds Vector 字段指示链路支持 8.0 GT/s 或更高的链路速度（Section 7.5.3.18 或 Section 7.9.9.2）。
- Lower SKP OS Generation Supported Speeds Vector 字段中的任何比特位都已设置为 1（Section 7.5.3.18）。
- 当需要在 Lane Error Status 寄存器中报告基于 Lane 的错误时（Section 4.2.6）。

为了支持将来在该 Capability 上添加新的 feature，与链路相关的任何 Function 或 RCRB 都允许实现此 Capability 。对于与 Upstream Port 关联的 Multi-Function Device，此 Capability 仅在设备的 Function 0 中才允许实现。

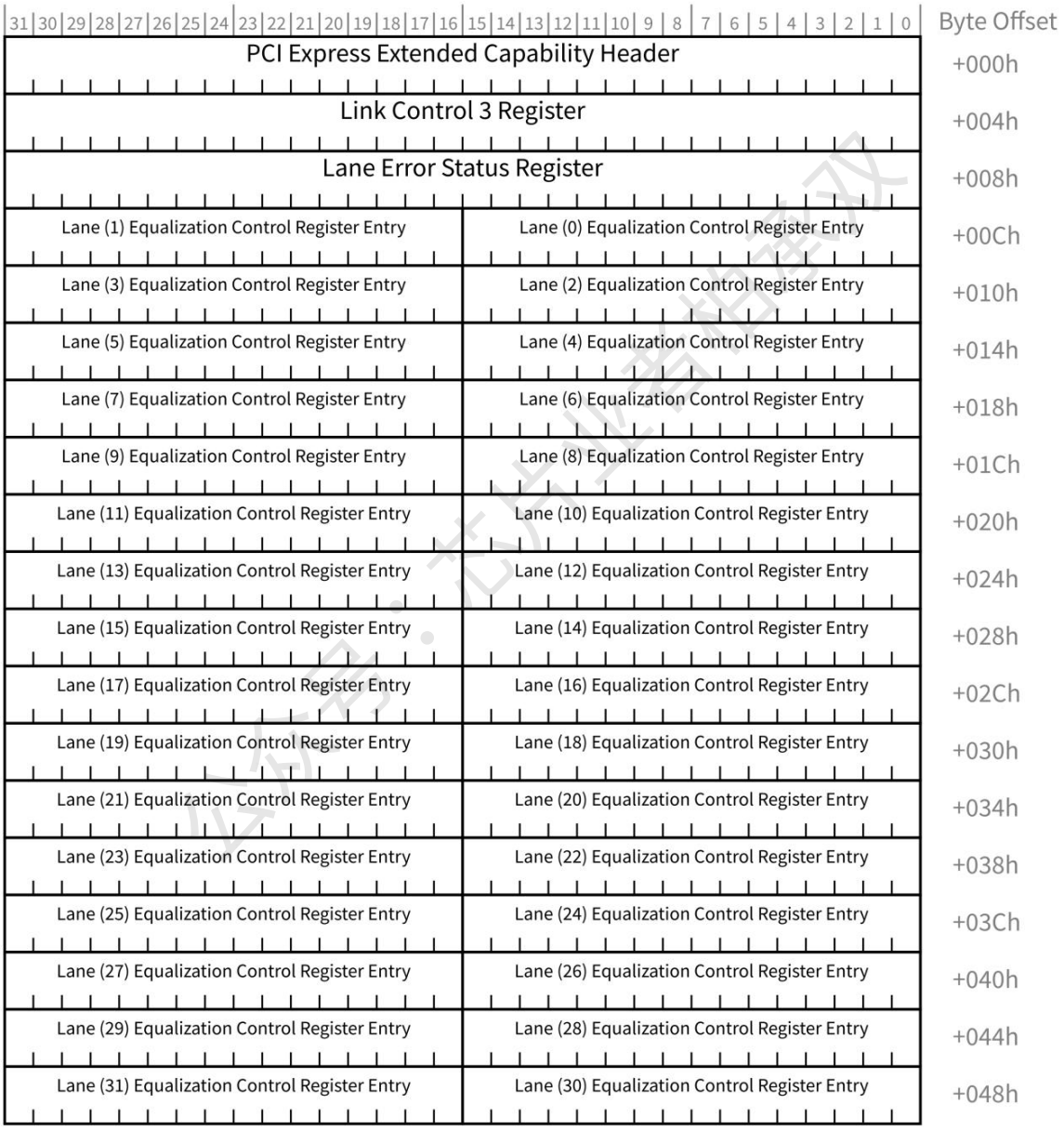

Figure 7-68 Secondary PCI Express Extended Capability Structure

7.7.3.1 Secondary PCI Express Extended Capability Header (Offset 00h)

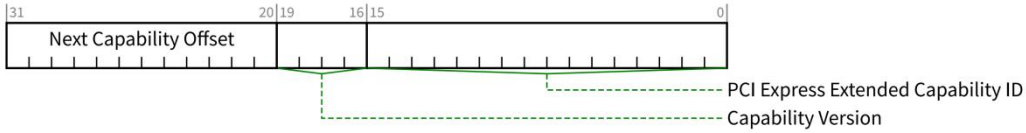

Figure 7-69 Secondary PCI Express Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** - 该字段是 PCI-SIG 定义的 ID 号，用于指示扩展功能的性质和格式。 Secondary PCI Express Extended Capability 的 PCI Express Extended Capability ID 为 0019h。	RO
19:16	**Capability Version** - 该字段是 PCI-SIG 定义的版本号，指示实现的 Capability structure 的版本。 对于该版本的规范，必须为 1h。	RO
31:20	**Next Capability Offset** - 此字段指示到下一个 PCI Express Extended Capability structure 的偏移量；如果在 Capability 的链接列表中不存在其他项，则为 000h。	RO

7.7.3.2 Link Control 3 Register (Offset 04h)

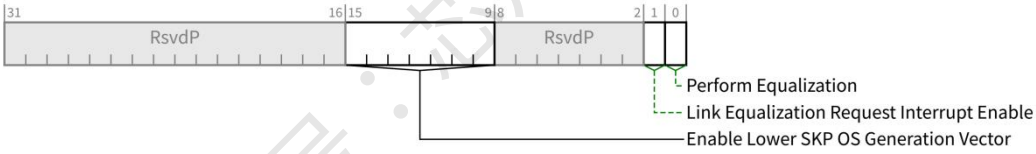

Figure 7-70 Link Control 3 Register

Bit Location	Register Description	Attributes
0	**Perform Equalization** - 当此比特位为 1b 且将 Target Link Speed 字段设置为 8.0 GT/s 或更高数据速率且 Retrain Link 字段写入 1b 时，Downstream Port 必须执行链路均衡。有关详细信息，请参见 Section 4.2.3 和 Section 4.2.6.4.2。 对 Downstream Port 和 Upstream Port，当 Crosslink Supported 为 1b 时，此比特位为 RW 属性（Section 7.5.3.18）。 当 Crosslink Supported 为 0b 时，此位不适用，并且对 Upstream Port 来说是 RsvdP 属性。 默认值为 0b。 如果端口不支持 8.0 GT/s，则允许将该位硬接线为 0b。	RW/RsvdP
1	**Link Equalization Request Interrupt Enable** - 当该比特位置 1 时，开启产生中断的功能，用来指示 Link Equalization Request 8.0 GT/s，Link Equalization Request 16.0 GT/s 或 Link Equalization Request	RW/RsvdP

公众号·芯片业者相承双

	32.0 GT/s 比特位已被置 1。 对 Downstream Port 和 Upstream Port，当 Crosslink Supported 为 1b 时，此比特位为 RW 属性（Section 7.5.3.18）。 当 Crosslink Supported 为 0b 时，此位不适用，并且对 Upstream Port 来说是 RsvdP 属性。 默认值为 0b。 如果端口不支持 8.0 GT/s，则允许将该位硬接线为 0b。	
15:9	**Enable Lower SKP OS Generation Vector** - 当链路处于 L0 且该字段中与当前链路速度相对应的比特位被置 1 时，将按为 SRNS 定义的速率来调度 SKP 有序集，从而取代基于时钟容限结构所需的速率。有关其他要求，请参见 Section 4.2.7。 该字段中的 bite 位定义为： **Bit 0** 2.5GT/s **Bit 1** 5.0GT/s **Bit 2** 8.0GT/s **Bit 3** 16.0GT/s **Bit 4** 32.0GT/s **Bit 6:5** RsvdP 如果 Lower SKP OS Generation Supported Speeds Vector 中的相应位设置为 1，则该字段中的每个未保留位都必须为 RW，否则该位必须为 RW 或硬接线为 0。 如果在此字段中设置了某个比特位，并且未设置 Lower SKP OS Generation Supported Speeds Vector 中的相应比特位，则行为未定义。 该字段的默认值为 000 0000b。	RW/RsvdP

7.7.3.3 Lane Error Status Register (Offset 08h)

Lane Error Status Register 由一个 32 位的向量组成，其中每个比特位指示具有相应 Lane number 的 Lane 是否检测到错误。该 Lane number 是默认的 Lane number，它与链路训练期间发生的 Link width 和 Lane reversal 协商的结果一致。

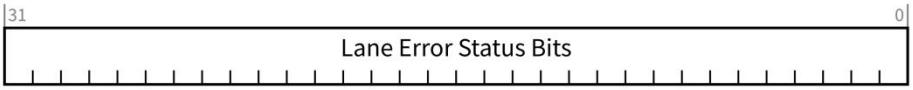

Figure 7-71 Lane Error Status Register

Bit Location	Register Description	Attributes
0	**Lane Error Status Bits** - 每个比特位指示相应的 Lane 是否检测到基于 Lane 的错误。值 1b 表示在相应的 Lane number 上检测到基于 Lane 的错误（有关详细信息，请参见 Section 4.2.2.3.3，Section 4.2.6 和 Section 4.2.7.2）。	RW/RsvdP

公众号：芯片业者相承双

每个比特位的默认值为 0b。

对于小于 32 Lane 的端口，未使用的[31:Maximum Link Width]为 RsvdZ。

对于不支持 8.0 GT/s 且未根据 8b/10b 错误设置这些位的端口（可选，请参见 Section 4.2.6），允许将该字段硬接线为 0。

7.7.3.4 Lane Equalization Control Register (Offset 0Ch)

Lane Equalization Control Register 由每 Lane 8.0 GT/s 均衡所需的控制字段组成，该寄存器中的条目数按 Maximum Link Width 确定大小（Section 7.5.3.6）。每个条目均包含带对应的默认 Lane number 的 Lane 值，该默认 Lane number 对于在链路训练期间发生的 Link width 和 Lane reversal 协商是不变的。

如果端口不支持 8.0 GT/s，则允许将该寄存器硬接线为 0。

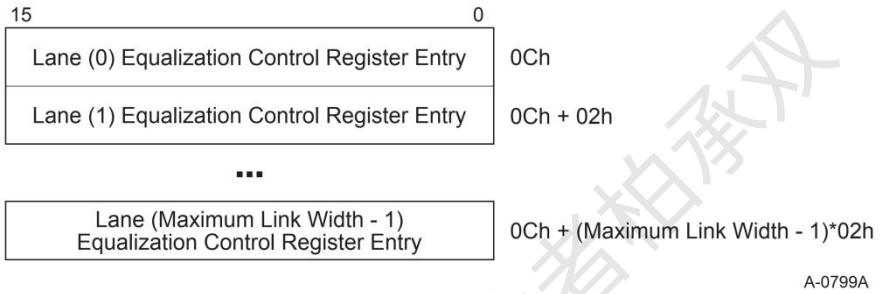

Figure 7-72 Lane Equalization Control Register

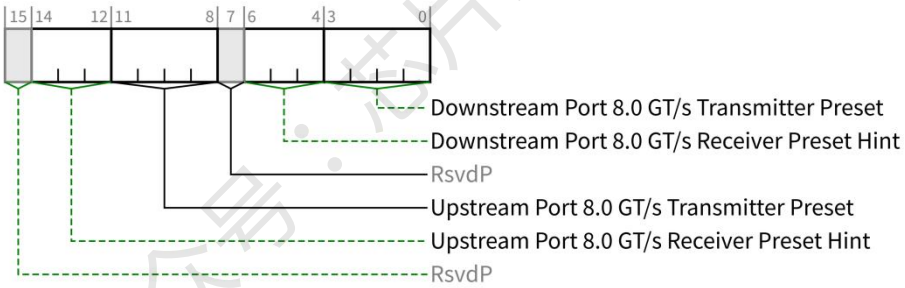

Figure 7-73 Lane Equalization Control Register Entry

Bit Location	Register Description	Attributes
3:0	**Downstream Port 8.0 GT/s Transmitter Preset** - 当端口用作 Downstream Port 时，此端口用于 8.0 GT/s 均衡的发送器 Preset 值。当端口作为 Upstream Port 运行时，将忽略此字段。有关详细信息，请参见 Chapter 8。字段编码在 Section 4.2.3.2 中定义。 对于 Upstream Port，如果 Crosslink Supported 为 0b，则此字段为 RsvdP。否则，此字段为 HwInit。参见 Section 7.5.3.18。 默认值为 1111b。	HwInit/RsvdP
6:4	**Downstream Port 8.0 GT/s Receiver Preset Hint** - 接收器 preset hint 值，当该端口作为 Downstream Port 运行时，可用作该端口的 8.0 GT/s 接收器均衡的建议设置。当端口作为	HwInit/RsvdP

公众号：芯片业者怕承双

Upstream Port 运行时，将忽略此字段。有关详细信息，请参见 Chapter 8。字段编码在 Section 4.2.3.2 中定义。

对于 Upstream Port，如果 Crosslink Supported 为 0b，则此字段为 RsvdP。否则，此字段为 HwInit。参见 Section 7.5.3.18。

默认值为 111b。

11:8 **Upstream Port 8.0 GT/s Transmitter Preset** - 该字段包含在 8.0 GT/s 链路均衡期间发送或接收的发送器 Preset 值。字段用法如下： HwInit/RO

	Operating Port Direction	Crosslink Support	Usage
A	Downstream Port	Any	此字段表示在链路均衡期间在相关 Lane 上发送的值。 此字段为 HwInit 属性。
B	Upstream Port	0b	该字段用于调试和诊断。它表示在链路均衡期间从相关的 Lane 上捕获的值。 字段是 RO 属性。 注意：当支持 crosslink 时，情况 C（如下）适用，并且此捕获的信息对软件不可见。鼓励供应商提供一种替代机制来获取此信息。
C	Upstream Port	1b	当前的链路均衡不使用或不影响该字段。如果将来的交叉链接协商切换了“工作端口方向”，则将使用该字段值，从而适用情况 A（上述）。 字段为 HwInit 属性。

有关详细信息，请参见 Section 4.2.3 和 Chapter 8。字段编码在 Section 4.2.3.2 中定义。

默认值为 1111b。

14:12 **Upstream Port 8.0 GT/s Receiver Preset Hint** - 该字段表示在 8.0 GT/s 链路均衡期间发送或接收的接收方 preset hint 值。字段用法如下： HwInit/RO

	Operating Port Direction	Crosslink Support	Usage
A	Downstream Port	Any	此字段表示在链路均衡期间在相关 Lane 上发送的值。 此字段为 HwInit 属性。

公众号·芯片业者相承双

B	Upstream Port	0b	该字段用于调试和诊断。它表示在链路均衡期间从相关的 Lane 上捕获的值。 字段是 RO 属性。 注意：当支持 crosslink 时，情况 C（如下）适用，并且此捕获的信息对软件不可见。鼓励供应商提供一种替代机制来获取此信息。
C	Upstream Port	1b	当前的链路均衡不使用或不影响该字段。如果将来的交叉链接协商切换了“工作端口方向”，则将使用该字段值，从而适用情况 A（上述）。 字段为 HwInit 属性。

有关详细信息，请参见 Section 4.2.3 和 Chapter 8。字段编码在 Section 4.2.3.2 中定义。

默认值为 111b。

7.7.4 Data Link Feature Extended Capability

Data Link Feature Capability 对于 Downstream Port 是一个可选的扩展功能。由于支持 16.0 GT/s 的端口需要实现 Scaled Flow Control Feature，因此支持 16.0 GT/s 的 Downstream Port 需要实现此 Capability（Section 3.4.2）。在其他 Downstream Port 中是可选的。与 Upstream Port 相关的 Function 中，它是可选的。与 Upstream Port 相关的 Multi-Function Device 中，所有 Function 中的 Capability 实例的字段中必须报告相同的信息。它不适用于与端口不相关的 Function（例如 RCiEP，Root Complex Event Collector）。Data Link Feature Capability 如 Figure 7-74 所示。

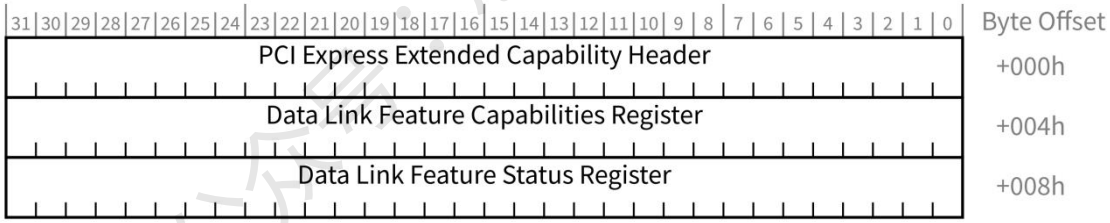

Figure 7-74 Data Link Feature Extended Capability

7.7.4.1 Data Link Feature Extended Capability Header (Offset 00h)

Figure 7-75 详细描述 Data Link Feature Extended Capability Header 中各字段的分配；Table 7-59 提供了相应的比特位定义。

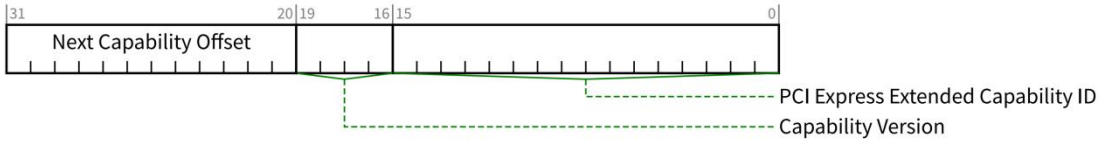

Figure 7-75 Data Link Feature Extended Capability Header

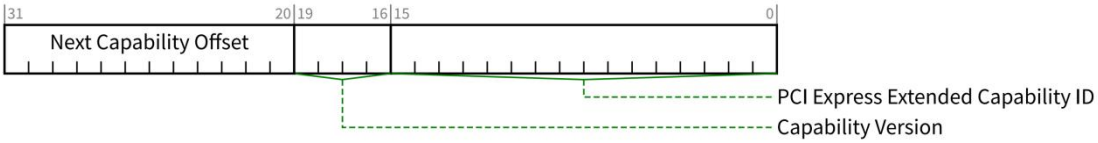

Figure 7-75 Data Link Feature Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** - 该字段是 PCI-SIG 定义的 ID 号，用于指示扩展功能的性质和格式。 Data Link Feature Extended Capability 的 PCI Express Extended Capability ID 为 0025h。	RO
19:16	**Capability Version** - 该字段是 PCI-SIG 定义的版本号，指示实现的 Capability structure 的版本。 对于该版本的规范，必须为 1h。	RO
31:20	**Next Capability Offset** - 此字段指示到下一个 PCI Express Extended Capability structure 的偏移量；如果在 Capability 的链接列表中不存在其他项，则为 000h。	RO

7.7.4.2 Data Link Feature Capabilities Register (Offset 04h)

Figure 7-76 详细描述了 Data Link Feature Capabilities Register 中的寄存器字段分配；Table 7-60 提供了相应的比特位定义。

当此端口发送一个 Data Link Feature DLLP 时，该 DLLP 的 Symbol 1、2 和 3 中的 Feature Support 字段分别对应此寄存器的位[22:16]，[15:8]和[7:0]（Figure 3-12）。

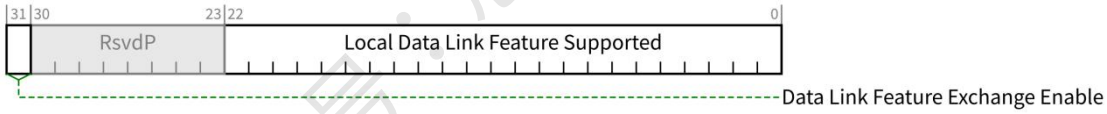

Figure 7-76 Data Link Feature Capabilities Register

Bit Location	Register Description	Attributes
22:0	**Local Data Link Feature Supported** - 该字段包含当此端口发送 Data Link Feature DLLP 时使用的 Feature Supported 值（Figure 3-12）。定义的 Feature 有： **Bit 1** **Local Scaled Flow Control Supported** - 该比特位表示该端口支持 Scaled Flow Control Feature（Section 3.4.2）。 **Bit 22:1** RsvdP 与该端口能够支持的 Feature 相关的比特位是 HwInit，默认为 1b。 该字段中的其他位是 RsvdP。	HwInit/RsvdP
31	**Data Link Feature Exchange Enable** - 如果置 1，则该比特位指示该端口将进入 DL_Feature 协商	HwInit

公众号：芯片业者相承双

状态（Section 3.2.1）。默认值为 1b。

7.7.4.3 Data Link Feature Status Register (Offset 08h)

Figure 7-77 详细说明了 Data Link Feature Status Register 中寄存器字段的分配； Table 7-61 提供了相应的比特位定义。

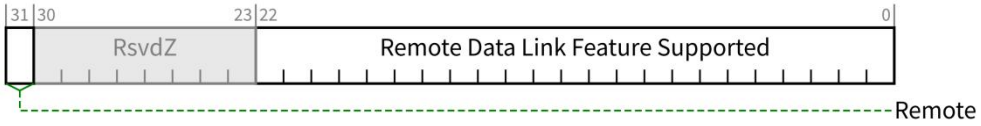

Figure 7-77 Data Link Feature Status Register

Bit Location	Register Description	Attributes
22:0	**Remote Data Link Feature Supported** - 这些比特位表示 Remote Port 支持相应的 Data Link Feature 。这些比特位从 Data Link Feature DLLP 的 Feature Supported 字段中捕获所有信息，即使该端口不支持相应的 Feature 也是如此。 该字段在进入状态 DL_Inactive 时被清 0（Section 3.2.1）。 **Bit 1** **Remote Scaled Flow Control Supported** - 该比特位表示 Remote Port 支持 Scaled Flow Control Feature（Section 3.4.2）。 **Bit 22:1** undefined 默认值为 00 0000h 。	RO
31	**Remote Data Link Feature Supported Valid** - 该比特位表示端口在 DL_Feature 状态收到 Data Link Feature DLLP（Section 3.2.1），并且 Remote Data Link Feature Supported 字段是有意义的。该位在进入状态 DL_Inactive 时被清 0（Section 3.2.1）。 默认值为 0b。	RO

7.7.5 Physical Layer 16.0 GT/s Extended Capability

以下情况需要实现 Physical Layer 16.0 GT/s Extended Capability structure：

- 与 Downstream Port 关联的 Function，其 Supported Link Speeds Vector 字段表示支持 16.0 GT/s 的链路速度。
- 与 Upstream Port 关联的 single-Function Device 的 Function，其 Supported Link Speeds Vector 字段表示支持 16.0 GT/s 的链路速度。
- 与 Upstream Port 关联的 Multi-Function Device 的 Function 0（仅 Function 0），其 Supported Link Speeds Vector 字段表示支持 16.0 GT/s 的链路速度。

即使不支持 16.0 GT/s 的链路速度，也可以在上面列出的任何 Function 中实现此 Capability。当不支持 16.0 GT/s 链路速度时，除 Capability Header 以外的寄存器的行为是不确定的。

Figure 7-79 详细说明了 Physical Layer 16.0 GT/s Extended Capability structure 中寄存器字段的分配。

公众号·芯片业者怕承认

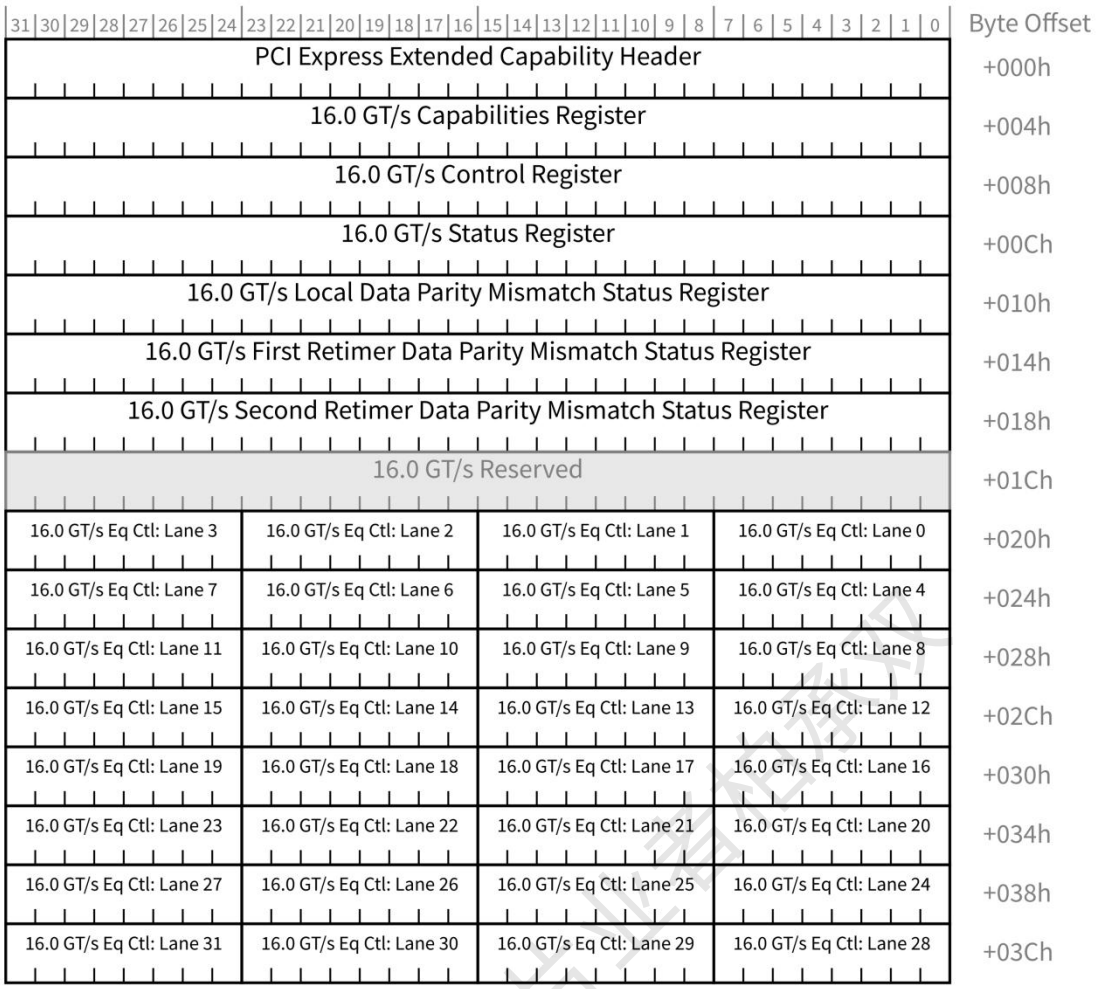

Figure 7-78 Physical Layer 16.0 GT/s Extended Capability

7.7.5.1 Physical Layer 16.0 GT/s Extended Capability Header (Offset 00h)

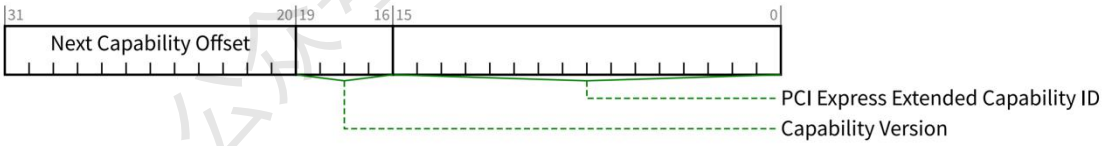

Figure 7-79 Physical Layer 16.0 GT/s Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** - 该字段是 PCI-SIG 定义的 ID 号，用于指示扩展功能的性质和格式。 Physical Layer 16.0 GT/s Extended Capability 的 PCI Express Extended Capability ID 为 0026h。	RO
19:16	**Capability Version** - 该字段是 PCI-SIG 定义的版本号，指示实现的 Capability structure 的版本。 对于该版本的规范，必须为 1h。	RO

31:20	**Next Capability Offset** - 此字段指示到下一个 PCI Express Extended Capability structure 的偏移量；如果在 Capability 的链接列表中不存在其他项，则为 000h。	RO

7.7.5.2 16.0 GT/s Capabilities Register (Offset 04h)

Figure 7-80 16.0 GT/s Capabilities Register

Bit Location	Register Description	Attributes
31:0	RsvdP	RsvdP

7.7.5.3 16.0 GT/s Control Register (Offset 08h)

Figure 7-81 16.0 GT/s Control Register

Bit Location	Register Description	Attributes
31:0	RsvdP	RsvdP

7.7.5.4 16.0 GT/s Status Register (Offset 0Ch)

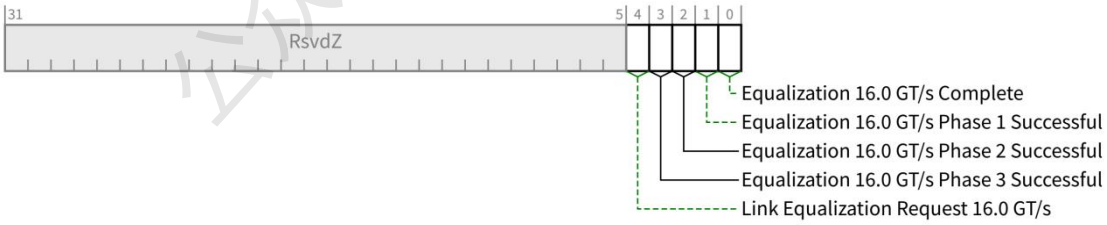

Figure 7-82 16.0 GT/s Status Register

Bit Location	Register Description	Attributes
0	**Equalization 16.0 GT/s Complete** - 置 1 时，该位表示 16.0 GT/s 发送器均衡过程已完成。Section 4.2.6.4.2 提供了发送器均衡过程以及何时需要将该位设置为 1b 的详细信息。 该位的默认值为 0b。 对于 Multi-Function Upstream Port，必须在 Function 0 中实现此比特位，而在其他 Function 中	ROS/RsvdZ

公众号·芯片业者相承双

	实现为 RsvdZ。	
1	**Equalization 16.0 GT/s Phase 1 Successful** - 当设置为 1b 时，该位表示 16.0 GT/s 发送器均衡过程的 Phase 1 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程以及何时需要将该位设置为 1b 的详细信息。 该位的默认值为 0b。 对于 Multi-Function Upstream Port，必须在 Function 0 中实现此比特位，而在其他 Function 中实现为 RsvdZ。	ROS/RsvdZ
2	**Equalization 16.0 GT/s Phase 2 Successful** - 当设置为 1b 时，该位表示 16.0 GT/s 发送器均衡过程的 Phase 2 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程以及何时需要将该位设置为 1b 的详细信息。 该位的默认值为 0b。 对于 Multi-Function Upstream Port，必须在 Function 0 中实现此比特位，而在其他 Function 中实现为 RsvdZ。	ROS/RsvdZ
3	**Equalization 16.0 GT/s Phase 3 Successful** - 当设置为 1b 时，该位表示 16.0 GT/s 发送器均衡过程的 Phase 3 已成功完成。Section 4.2.6.4.2 提供了发送器均衡过程以及何时需要将该位设置为 1b 的详细信息。 该位的默认值为 0b。 对于 Multi-Function Upstream Port，必须在 Function 0 中实现此比特位，而在其他 Function 中实现为 RsvdZ。	ROS/RsvdZ
4	**Link Equalization Request 16.0 GT/s** - 该位由硬件置 1，以请求在链路上执行 16.0 GT/s 链路均衡过程。有关详细信息，请参见 Section 4.2.3 和 Section 4.2.6.4.2。 该位的默认值为 0b。 对于 Multi-Function Upstream Port，必须在 Function 0 中实现此比特位，而在其他 Function 中实现为 RsvdZ。	RW1CS/RsvdZ

7.7.5.5 16.0 GT/s Local Data Parity Mismatch Status Register (Offset 10h)

Local Data Parity Mismatch Status 存器是一个 32 位向量，其中每个比特位表示本地接收器是否检测到具有相应 Lane number 的 Lane 上的 Data Parity 不匹配。该 Lane number 是默认的 Lane number，它与链路训练期间发生的链路宽度和通道反转协商保持不变。

该寄存器收集 16.0 GT/s 和更高数据速率的奇偶校验错误。当跟踪特定链接速度的错误时，软件应在速率发生变化时清除该寄存器。

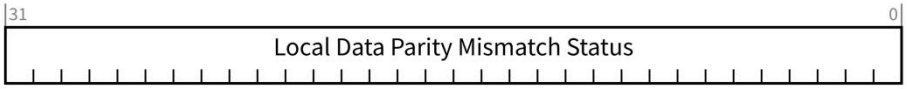

Figure 7-83 16.0 GT/s Local Data Parity Mismatch Status Register

公众号·芯片业者的承刃

Bit Location	Register Description	Attributes
0	**Local Data Parity Mismatch Status** - 每个比特位表示相应的 Lane 是否检测到数据奇偶校验不匹配。值 1b 表示在相应的 Lane number 上检测到不匹配。有关更多信息，请参见 Section 4.2.7.2。 每个比特位的默认值为 0b。 对于小于 32 Lane 的端口，未使用的高位[31:Maximum Link Width]为 RsvdZ。	RW1CS/RsvdZ

7.7.5.6 16.0 GT/s First Retimer Data Parity Mismatch Status Register (Offset 14h)

First Retimer Data Parity Status 寄存器是一个 32 位向量，其中每个比特位表示路径上的第一个 Retimer（有关更多信息，请参见 Figure 4-36）在 Lane 上检测到具有相应 Lane number 的数据奇偶校验不匹配。该 Lane number 是默认的 Lane number，它与链路训练期间发生的链路宽度和通道反转协商保持不变。

该寄存器收集 16.0 GT/s 和更高数据速率的奇偶校验错误。当跟踪特定链接速度的错误时，软件应在速率变化时清除该寄存器。

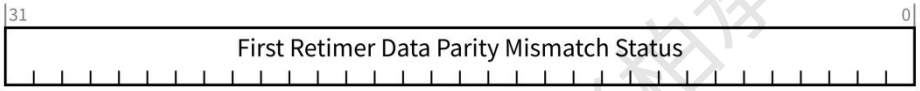

Figure 7-84 16.0 GT/s First Retimer Data Parity Mismatch Status Register

Bit Location	Register Description	Attributes
0	**First Retimer Data Parity Mismatch Status** - 每个位指示相应的 Lane 是否检测到数据奇偶校验不匹配。值 1b 表示在相应的 Lane number 上检测到不匹配。 有关更多信息，请参见 Section 4.2.7.2。 每个位的默认值为 0b。 没有 Retimer 时，此字段的值是不确定的。 对于小于 32 Lane 的端口，未使用的高位[31:Maximum Link Width]为 RsvdZ。	RW1CS/RsvdZ

7.7.5.7 16.0 GT/s Second Retimer Data Parity Mismatch Status Register (Offset 18h)

16.0 GT/s Second Retimer Data Parity Status 寄存器是一个 32 位向量，其中每个比特位表示路径上的第二个 Retimer（有关更多信息，请参见 Figure 4-36）在 Lane 上检测到具有相应 Lane number 的数据奇偶校验不匹配。该 Lane number 是默认的 Lane number，它与链路训练期间发生的链路宽度和通道反转协商保持不变。

该寄存器收集 16.0 GT/s 和更高数据速率的奇偶校验错误。当跟踪特定链接速度的错误时，软件应在速率变化时清除该寄存器。

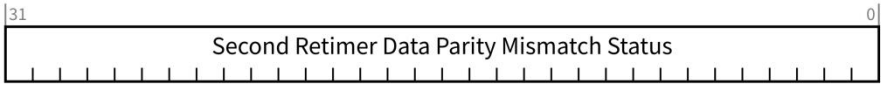

Figure 7-85 16.0 GT/s Second Retimer Data Parity Mismatch Status Register

Bit Location	Register Description	Attributes
0	**Second Retimer Data Parity Mismatch Status** - 每个位指示相应的 Lane 是否检测到数据奇偶校验不匹配。值 1b 表示在相应的 Lane number 上检测到不匹配。有关更多信息，请参见 Section 4.2.7.2。 每个位的默认值为 0b。 没有 Retimer 时，此字段的值是不确定的。 对于小于 32 Lane 的端口，未使用的高位[31:Maximum Link Width]为 RsvdZ。	RW1CS/RsvdZ

7.7.5.8 Physical Layer 16.0 GT/s Reserved (Offset 1Ch)

此寄存器为 RsvdP 。

7.7.5.9 16.0 GT/s Lane Equalization Control Register (Offsets 20h to 3Ch)

Lane Equalization Control Register 由 per-Lane 16.0 GT/s 均衡所需的控制字段组成。它包含至少由 Maximum Link Width（请参见 Section 7.5.3.6 或 Section 7.9.9.2）定义的 Lane 数个条目，必须以整个 DW 粒度实现（例如，如果 Maximum Link Width 为 x1，则寄存器将仍然包含 4 个 Lane 的条目，而 Lane 1、2 和 3 的条目未定义），并且无论 Maximum Link Width 如何，最多可以包含 32 个条目。超出 Maximum Link Width 的条目的值是不确定的。

每个条目均包含带对应 Lane number 的默认的 Lane 值，该默认 Lane number 对于在链路训练期间发生的 Link width 和 Lane reversal negotiation 是不变的。

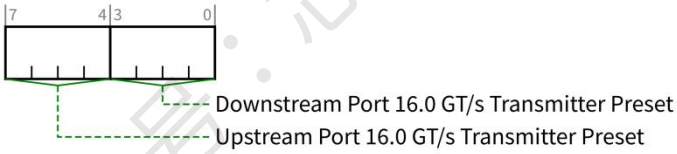

Figure 7-86 16.0 GT/s Lane Equalization Control Register Entry

Bit Location	Register Description	Attributes
3:0	**Downstream Port 16.0 GT/s Transmitter Preset** - 当端口作为 Downstream Port 运行时，此端口用于 16.0 GT/s 均衡的发送器 Preset 。当端口作为 Upstream Port 运行时，将忽略此字段。有关详细信息，请参见 Chapter 8。字段编码在 Section 4.2.3.2 中定义。 对于 Upstream Port，如果 Crosslink Supported 为 0b，则此字段为 RsvdP。否则，此字段为 HwInit。 参见 Section 7.5.3.18。 默认值为 1111b。	HwInit/RsvdZ
7:4	**Upstream Port 16.0 GT/s Transmitter Preset** - 该字段包含在 16.0 GT/s 链路均衡期间发送或接收的发送器 Preset 值。字段用法如下：	HwInit/RsvdZ

公众号·芯片业务的承双

	Operating Port Direction	Crosslink Support	Usage
A	Downstream Port	Any	此字段表示在链路均衡期间在相关 Lane 上发送的值。 此字段为 HwInit 属性。
B	Upstream Port	0b	该字段用于调试和诊断。它表示在链路均衡期间从相关的 Lane 上捕获的值。 字段是 RO 属性。 注意：当支持 crosslink 时，情况 C（如下）适用，并且此捕获的信息对软件不可见。鼓励供应商提供一种替代机制来获取此信息。
C	Upstream Port	1b	当前的链路均衡不使用或不影响该字段。如果将来的交叉链接协商切换了“工作端口方向”，则将使用该字段值，从而适用情况 A（上述）。 字段为 HwInit 属性。

有关详细信息，请参见 Section 4.2.3 和 Chapter 8。字段编码在 Section 4.2.3.2 中定义。

默认值为 1111b。

7.7.6 Physical Layer 32.0 GT/s Extended Capability

如果 Port 支持以下一个或多个 feature 则必须实现 Physical Layer 32.0 GT/s Extended Capability structure：

- Supported Link Speeds Vector 字段指示支持 32.0 GT/s 的链路速度。
- Function 支持发送和/或接收 Modified TS1/TS2 Ordered Set。

如果要实现，此 structure 必须在以下位置实现：

- 与 Downstream Port 关联的 Function 。
- 与 Upstream Port 关联的 single-Function Device 的 Function 。
- 与 Upstream Port 关联的 Multi-Function Device 的 Function 0（仅 Function 0）。

即使不支持 32.0 GT/s 链路速度，也可以在上面列出的任何 Function 中实现此功能。当不支持 32.0 GT/s 链路速度时，除 Capability Header 以外的寄存器的行为是不确定的。

Figure 7-87 详细说明了 Physical Layer 32.0 GT/s Extended Capability structure 中寄存器字段的分配。

请注意，32.0 GT/s 的奇偶校验错误记录在 16.0 GT/s Local Data Parity Mismatch Status Register、16.0 GT/s First Retimer Data Parity Mismatch Status Register 和 16.0 GT/s Second Retimer Data Parity Mismatch Status Register 中。当跟踪特定链路速率的错误时，软件应在速率变化时清除那些寄存器。

公众号·芯片业者柏承印

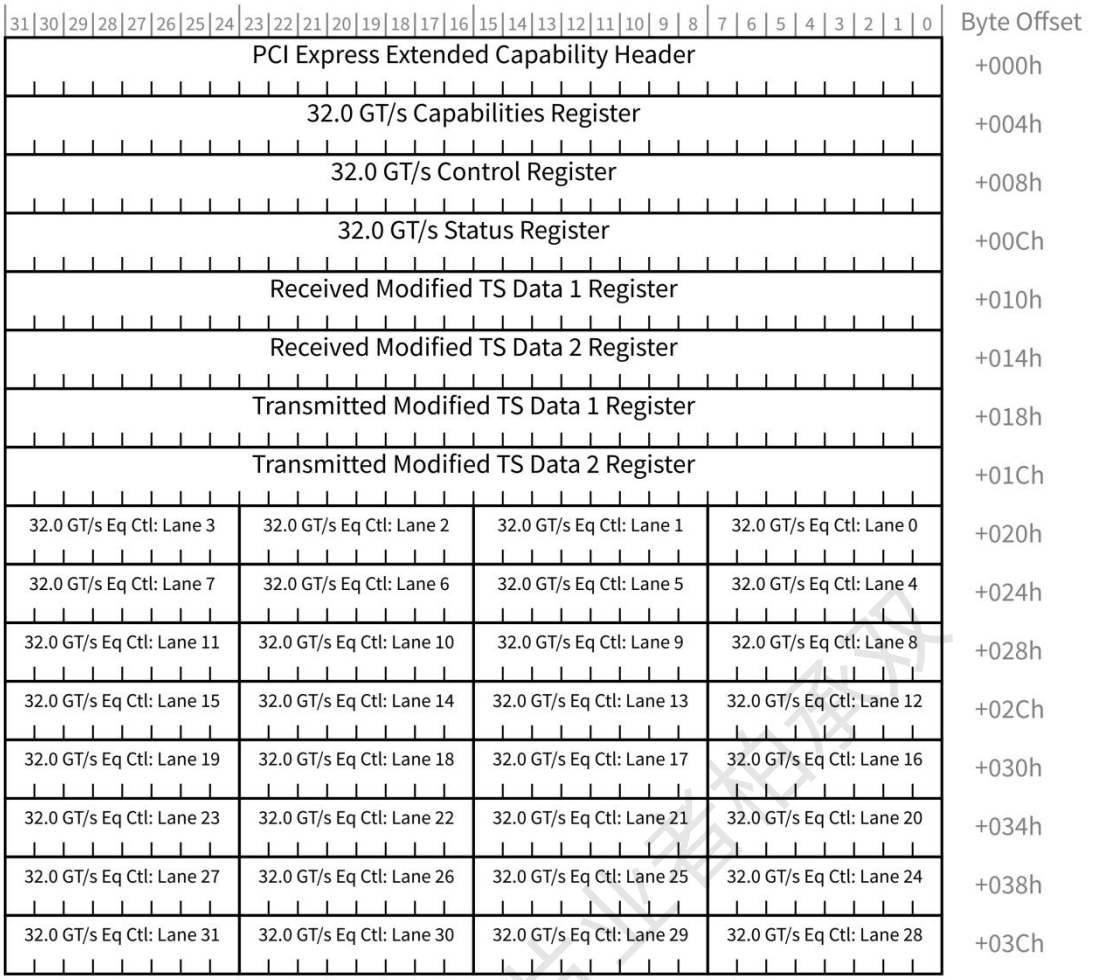

Figure 7-87 Physical Layer 32.0 GT/s Extended Capability

7.7.6.1 Physical Layer 32.0 GT/s Extended Capability Header (Offset 00h)

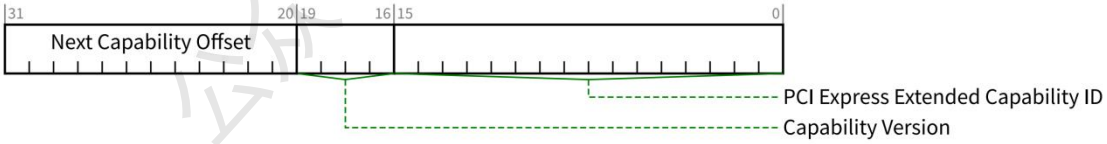

Figure 7-88 Physical Layer 32.0 GT/s Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** - 该字段是 PCI-SIG 定义的 ID 号，用于指示扩展功能的性质和格式。 Physical Layer 32.0 GT/s Extended Capability 的 PCI Express Extended Capability ID 为 002Ah。	RO
19:16	**Capability Version** - 该字段是 PCI-SIG 定义的版本号，指示实现的 Capability structure 的版本。 对于该版本的规范，必须为 1h。	RO

31:20	**Next Capability Offset** - 此字段指示到下一个 PCI Express Extended Capability structure 的偏移量；如果在 Capability 的链接列表中不存在其他项，则为 000h。	RO

7.7.6.2 32.0 GT/s Capabilities Register (Offset 04h)

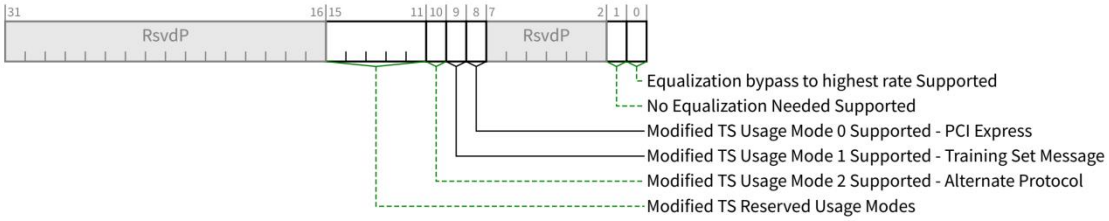

Figure 7-89 32.0 GT/s Capabilities Register

Bit Location	Register Description	Attributes
0	**Equalization bypass to highest rate Supported** - 设置为"1"时，此端口支持控制端口是否协商跳过最高支持的最高速度以外的其他速度的均衡。有关详细信息，请参见 Section 4.2.3。 对于支持 32.0 GT/s 或更高数据速率的端口，必须为 1b。	HwInit
1	**No Equalization Needed Supported** - 置 1 时，此端口支持控制是否需要均衡。	HwInit
8	**Modified TS Usage Mode 0 Supported - PCI Express** - 该比特位表示该端口支持 PCI Express（Modified TS Usage 编码为 000b）。该比特位必须为 1b。	RO
9	**Modified TS Usage Mode 1 Supported - Training Set Message** - 该比特位指示此端口支持发送和接收供应商特定的 Training Set Message（Modified TS Usage 编码为 001b）。有关详细信息，请参见 Section 4.2.4.2。	HwInit
10	**Modified TS Usage Mode 2 Supported - Alternate Protocol** - 该比特位指示此端口支持协商使用备用协议（Modified TS Usage 编码为 010b）。有关详细信息，请参见 Section 4.2.4.2。	HwInit
15:11	**Modified TS Reserved Usage Modes** - PCISIG 定义的将来使用模式的保留位。必须为 0 0000b。	RO

7.7.6.3 32.0 GT/s Control Register (Offset 08h)

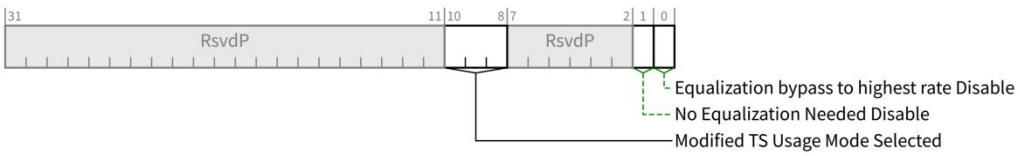

Figure 7-90 32.0 GT/s Control Register

公众号·芯片业者相关双