# OBFF功能介绍

## OBFF背景

随着电源管理机制越来越完善，处理器和其他系统组件的功耗越来越低，而像 PCIe 组件这样的外围设备在 PC 系统功耗中占据的比重越来越大。虽然早期的 PCIe 允许一些软件和硬件电源管理，但是并没有把电源管理策略与系统的协调放在一个高优先级，所以软件的控制也很有限。

缺乏系统与电源管理策略之间的协调会导致一个很明显的问题：某些时候，系统可能已经进入睡眠状态 (sleep state)了，而 PCIe 设备可能还处于工作状态。这些 PCIe 设备可能会产生中断或者 DMA 数据传输请求，虽然这些请求的优先级可能并不高，但也会导致系统从睡眠状态唤醒，这样就破坏了节能的目标。

还有一种情况，PCIe 设备在向系统提交请求后，并不一定需要即时响应，只要在某个时间范围内得到响应即可。如果系统不知道 PCIe 设备能够容忍的响应时间，那就会默认立即响应，这样会导致系统进入低功耗模式的时间比较短或者频繁的在低功耗与非低功耗模式之间来回切换。如果系统能知道 PCIe 设备能够容忍的响应时间，就可以合理的安排对设备请求的响应，在允许的时间范围内，可以把请求积攒起来，在某一个时刻统一进行响应。这样，系统就能有比较集中的时间处于低功耗模式。

## OBFF（Optimized [Buffer](https://so.csdn.net/so/search?q=Buffer&spm=1001.2101.3001.7020) Flush Fill）

OBFF 给 PCIe 设备提供了一种获取系统电源状态的机制。利用 OBFF，PCIe 设备就可以选择与系统进行数据交互的最优时间。

### 问题

对于具有总线主控功能的设备（bus-master capable devices）来说，如果不知道系统当前的电源状态，那么它们可能会在某些不合适的时间提交请求。

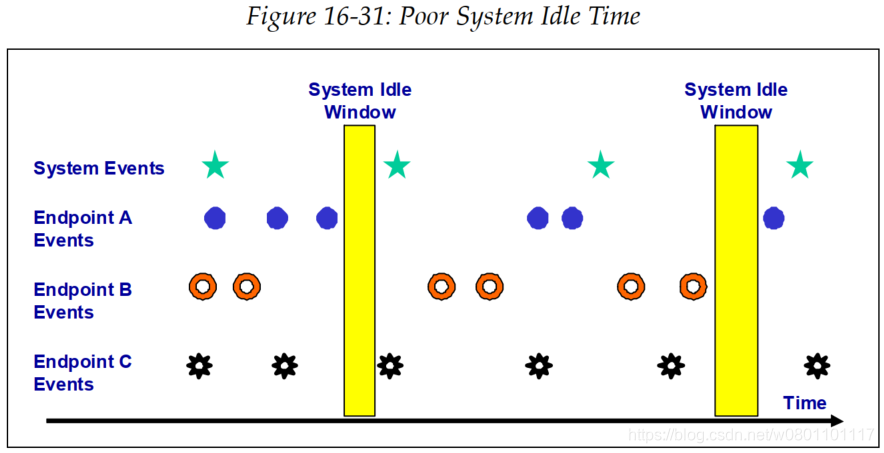


Figure 16-31 展示一个比较差的电源管理系统场景：系统中有 3 个 PCIe 终端设备（Endpoint），而且每个设备提交申请的时间比较散乱。这种场景下，系统能够进入 Idle 状态的时间就很短。

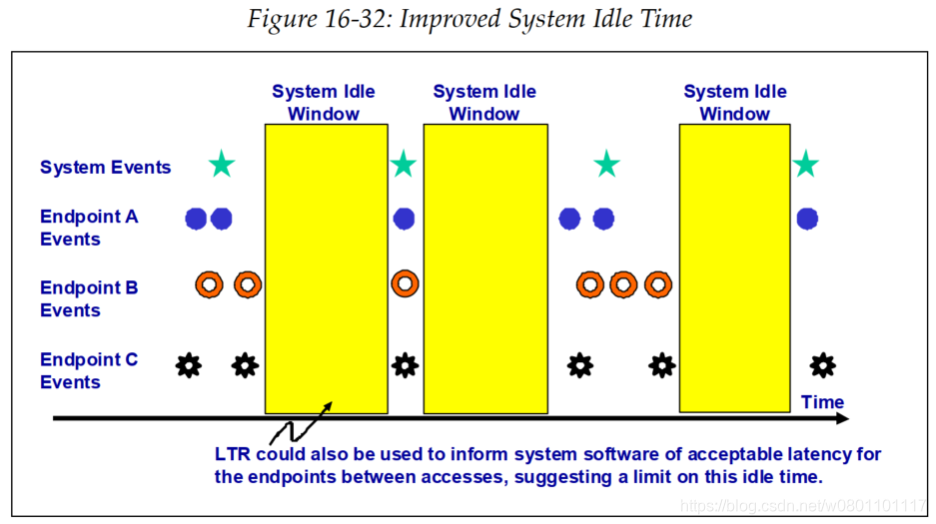


Figure 16-32 展示一个改善后的电源管理系统场景：该场景下，相同类型的请求会被攒到一起进行统一响应。这样，系统就可以有大块的时间进入 Idle 状态。很明显，这种模式能够达到不错的电源管理效果，而且实现起来并不复杂。只要 PCIe 设备能够查询当前系统的电源状态，并且知道在当前状态下该做些什么就可以。

### 解决方案

通过 OBFF，系统告知 PCIe 设备在哪些时间进行数据传输。OBFF 只是提供一个建议时间，PCIe 设备可以忽略 OBFF 信息，在它想要发请求的任意时刻发送请求（这样就导致功耗比较大，所以最好还是按照 OBFF 的建议来做）。有两种方式完成 OBFF 信息的交互：（1）通过向 Endpoint 发送消息（message）；（2）控制 WAKE# 引脚。在两种方式都支持的情况下，优先选择 WAKE# 引脚的方式。只有在 WAKE# 引脚不能用的时候，再去选择发送消息（message）的方式。

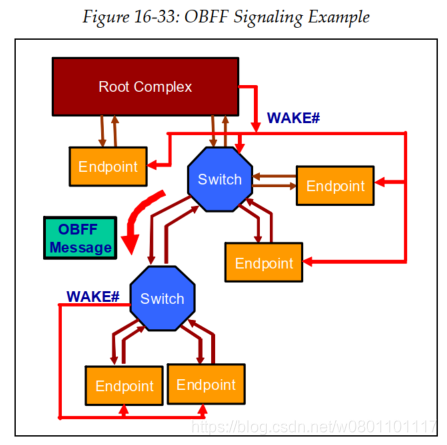


Figure 16-33 展示了 OBFF 的信息传输过程。在本例中，由于两个 Switch 之间不支持 WAKE# 引脚连接，所以同时用到了消息（message）和 WAKE# 引脚两种信息传输方式。上面的 Switch 收到来自 Root Complex WAKE# 引脚的信息后，会将其转化成消息（message）发送给下面的 Switch

#### 使用 WAKE# 引脚

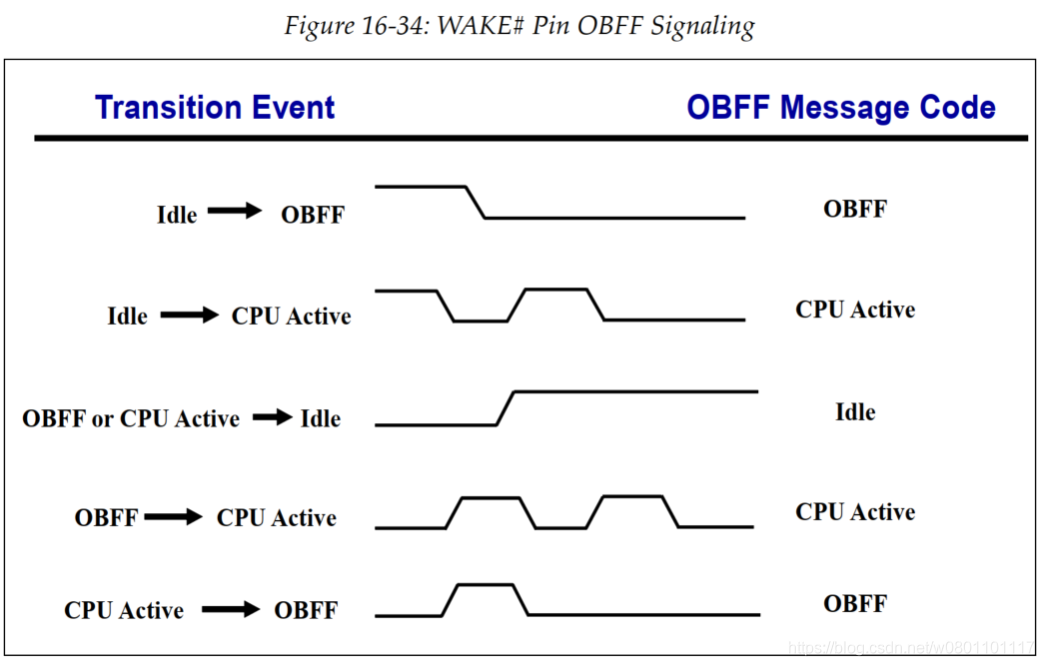
以前，WAKE# 只有一个功能：PCIe 组件使用该引脚通知系统恢复该 PCIe 组件的主电源供电（低功耗情况下，系统可能会移除 PCIe 组件的主电源）。

引入 OBFF 机制后，该引脚被用来向 PCIe 组件传递当前系统的电源状态。这种方式的协议很简单，只需要按照一定规律改变 WAKE# 引脚的电平，就可以表示出三种状态：

CPU Active：系统处于活跃状态，可以进行任意类型的事务交互。

OBFF：可以进行系统内存访问，可以读写系统内存；除此之外其他类型的事务交互都需要等到更高等级的电源状态。

Idle：等待进入更高等级的电源状态之后，启动。



当处于 CPU Active 或者 OBFF 状态时，建议在 10 us 之内不要进入 Idle 状态。这样可以给 PCIe 组件足够的时间去处理之前 Idle 状态下积攒的未被处理的事务。但是，由于 10 us 只是一个建议，所以 PCIe Endpoint 也不要盲目的认为系统会在 CPU Active 或者 OBFF 状态保持足够的时间。

系统可以在真正进入 Idle 状态之前，先通过 WAKE# 提前发出切换到 Idle 状态的信号，这样收到信号的 PCIe 组件就知道该结束事务传输了。

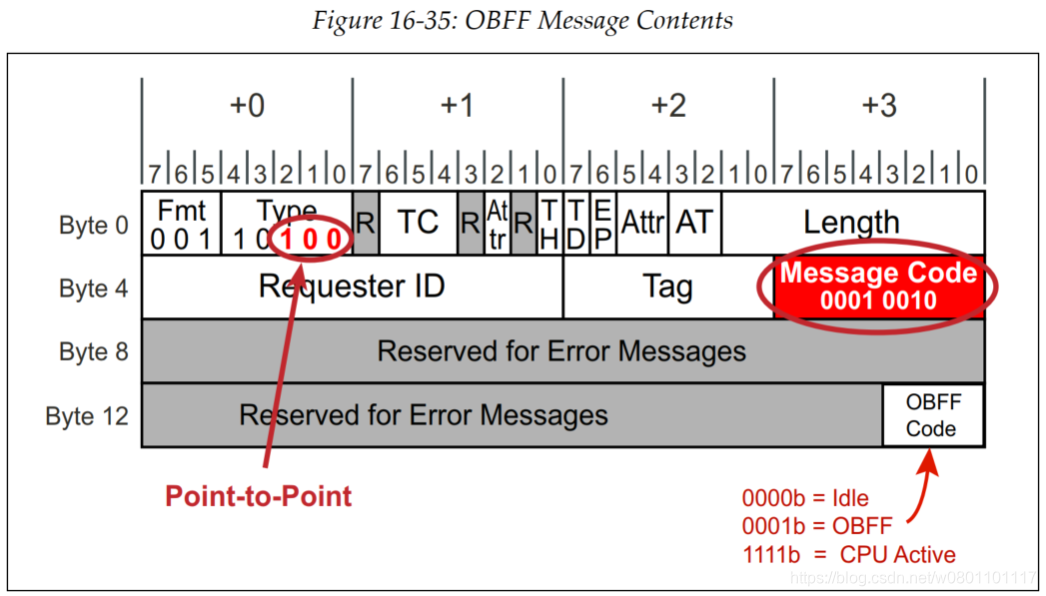
这种提前通知的机制，可以避免 PCIe 组件在系统刚进入 Idle 状态时就发起事务传输，导致系统从 Idle 状态唤醒。

按照协议的建议，系统的这种提前通知与真正进入 Idle 状态的通知之间的间隔时间要尽量的短一些。

有趣的是，WAKE# 引脚仍然能支持它最初的功能：PCIe 组件使用该引脚通知系统恢复该 PCIe 组件的主电源供电。这种情况下，当某个 PCIe 组件尝试通知系统恢复它的主电源供电时，可能会被其他监听 WAKE# 引脚的 PCIe 组件误认为是 OBFF 信息。为了解决这个问题，协议规定：如果 PCIe 组件无法清晰的理解 WAKE# 引脚的变化所指示的信息，那么该组件就可以认为当前系统处于 CPU Active 状态。

#### 使用OBFF 消息（OBFF Message）

OBFF 消息（OBFF Message）只能从 Root Complex 发送到其他 PCIe 组件。消息格式如下图：



Routing Type = 0b100，表示 Point-to-Point 类型。

OBFF Code 有 3 中有效值，其它都是 reserved code：

0b1111：CPU Active

0b0001：OBFF

0b000b：Idle

其他：reserved code

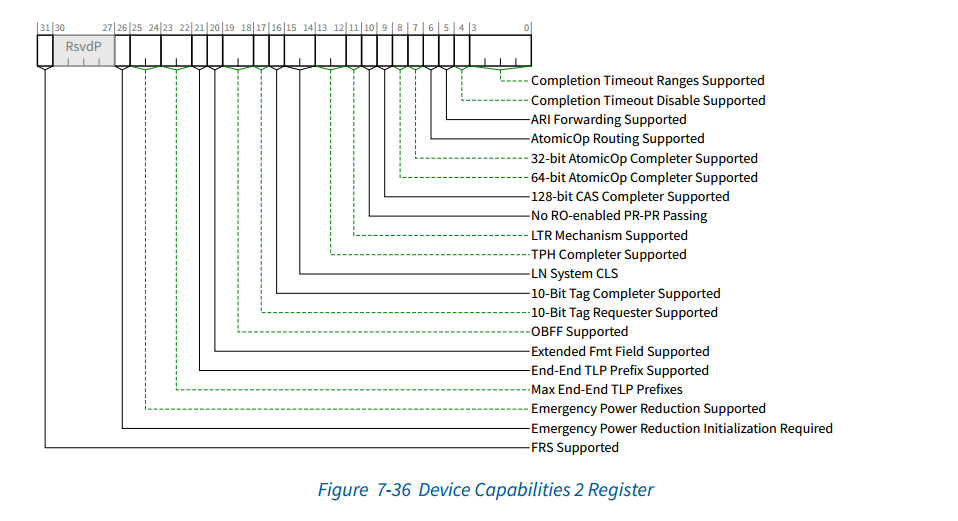
如果收到的 OBFF 消息中，OBFF Code的值是 reserved code，那么 PCIe 组件可以认为当前系统处于 CPU Active 状态。

如果 PCIe 组件不支持 OBFF 或者还没有使能 OBFF，那么当它收到 OBFF 消息时，需要按照 Unsupported Request类型来处理（返回的 Completion 中，将 UR 状态位设置为 1）。

# OBFF功能实现

## OBFF相关寄存器

### Device Capabilities 2 Register（已实现,RW）



OBFF Supported - This field indicates if OBFF is supported and, if so, what signaling mechanism is used.

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

The value reported in this field must indicate support for WAKE# signaling only if:

• for a Downstream Port, driving the WAKE# signal for OBFF is supported and the connector or component connected Downstream is known to receive that same WAKE# signal

• for an Upstream Port, receiving the WAKE# signal for OBFF is supported and, if the component is on an add-in-card, that the component is connected to the WAKE# signal on theconnector.

Root Ports, Switch Ports, and Endpoints are permitted to implement this capability.

For a Multi-Function Device associated with an Upstream Port, each Function must report the same value for this field.

For Bridges and Ports that do not implement this capability, this field must be hardwired to 00b.

此字段中报告的值必须指示仅在以下情况下支持WAKE#信令：

•对于下游端口，支持驱动OBFF的WAKE#信号，并且已知连接到下游的连接器或组件接收相同的WAKE#signal

•对于上行端口，支持接收OBFF的WAKE#信号，如果组件在外接板卡上，则表示该组件已连接到连接器上的WAKE#信号。

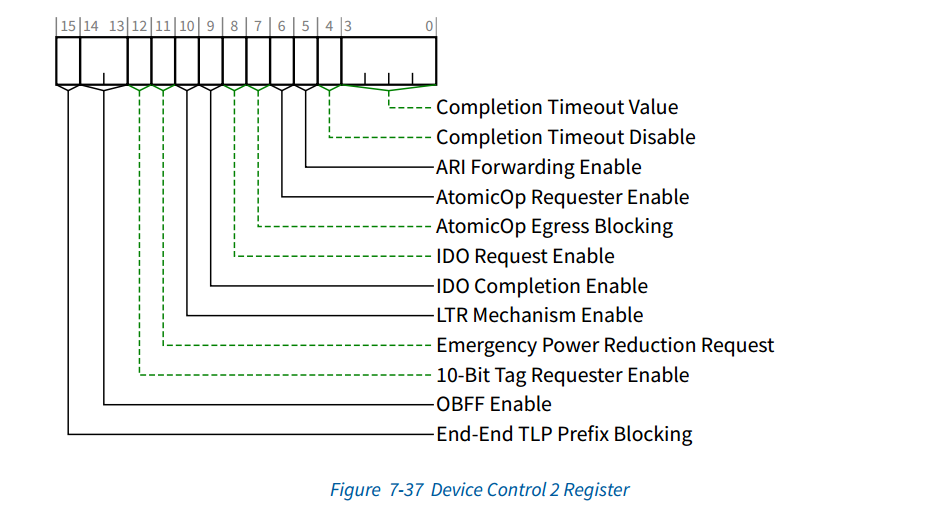
允许根端口、交换机端口和端点实现此功能。

对于与上游端口关联的多功能设备，每个功能必须报告该字段的相同值。

对于未实现此功能的网桥和端口，此字段必须硬接线到00b。

225已实现，具体在配置空间obffctrl模块中实现，支持WAKE和message两种方式，，通过k\_pexconf[51:50]进行配置，k\_pexconf[51:50]目前实际配置为00。

### Device Control 2 Register（已实现,RW）



OBFF Enable - This field enables the OBFF mechanism and selects the signaling method.

00b: Disabled

01b: Enabled using Message signaling [Variation A]

10b: Enabled using Message signaling [Variation B]

11b: Enabled using WAKE# signaling

See Section 6.19 for an explanation of the above encodings.

This field is required for all Ports that support the OBFF Capability.

For a Multi-Function Device associated with an Upstream Port of a Device that implements OBFF, the field in Function 0 is of type RW, and only Function 0 controls the Component’s behavior. In all other Functions of that Device, this field is of type RsvdP.

Ports that do not implement OBFF are permitted to hardwire this field to 00b.

Default value of this field is 00b.

有关上述编码的解释，请参见第6.19节。

所有支持OBFF能力的端口都需要此字段。

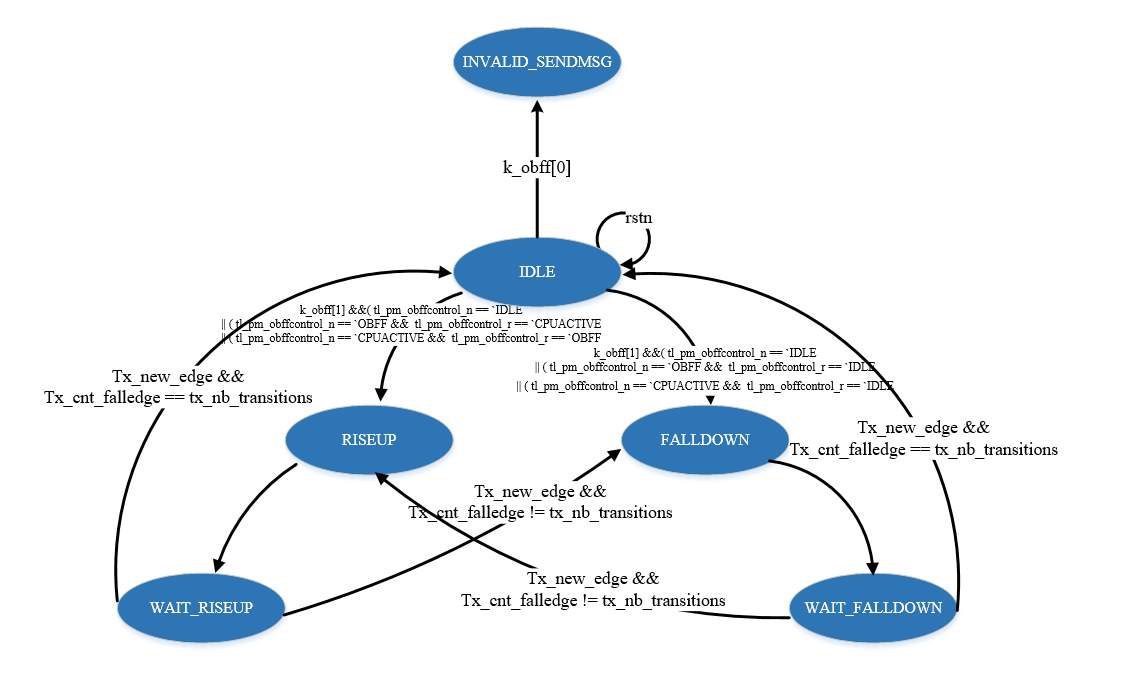
对于与实现OBFF的设备的上游端口相关联的多功能设备，功能0中的字段类型为RW，并且只有功能0控制组件的行为。在该设备的所有其他功能中，该字段的类型为RsvdP。

允许未实现OBFF的端口将此字段硬接线至00b。

此字段的默认值为00b。

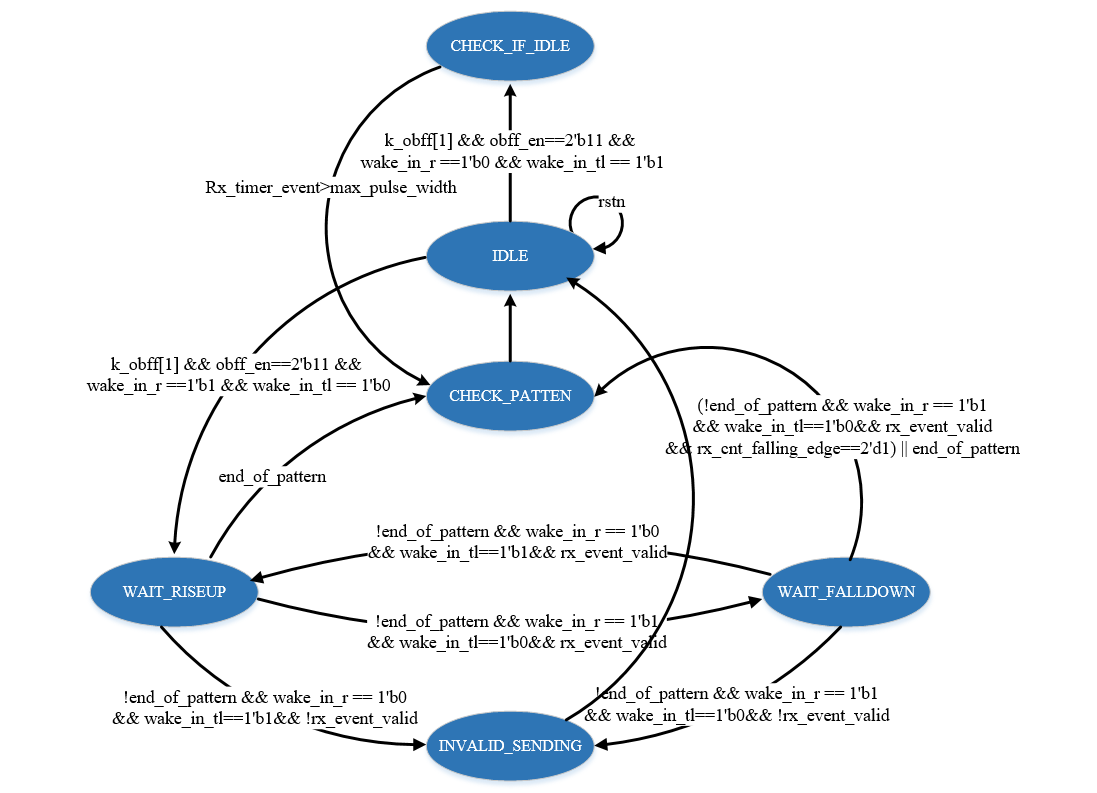
225已实现，具体在配置空间obffctrl模块中实现，支持A和B两种方式，通过配置空间pexreg模块写入Device Control 2 Register寄存器的OBFF Enable字段进行配置。

## OBFF状态机



下行端口状态机，主要功能为根据USP下发的OBFF状态信号tl\_pm\_obffcontrol（IDLE/OBFF/CPUACTIVE）产生对应的WAKE时序信号；

|  |  |  |  |
| --- | --- | --- | --- |
| 当前状态 | 下一个状态 | 转移条件 | 状态说明 |
| IDLE | RISEUP | Obff状态变化时，在WAKE模式，满足以下任意条件：  现在OBFF状态为IDLE；  现在OBFF状态为OBFF，之前OBFF状态为CPUACTIVE；  现在OBFF状态为CPUACTIVE，之前OBFF状态为OBFF； | 待机状态 |
| IDLE | FALLDOWN | Obff状态变化时，在WAKE模式，满足以下任意条件：  现在OBFF状态为OBFF，之前OBFF状态为IDLE；  现在OBFF状态为CPUACTIVE，之前OBFF状态为IDLE； | 待机状态 |
| IDLE | INVALID\_SENDMSG | Obff状态变化时，在MSG模式，满足以下任意条件：  MSG变量A配置，LTSSM状态机在L0状态；  MSG变量B配置，LTSSM状态机在L0、L0S、L0S、L1\_ENTRY、L1\_IDLE状态； | 待机状态 |
| RISEUP | WAIT\_RISEUP | 无条件，直接跳转 | 使能WAKE输出0 |
| FALLDOWN | WAIT\_FALLDOWN | 无条件，直接跳转 | 使能WAKE输出1 |
| WAIT\_RISEUP | IDLE | 等待400ns后，检测下降沿个数满足要求时跳转 | 等待输出WAKE |
| WAIT\_RISEUP | FALLDOWN | 等待400ns后，检测下降沿个数不满足要求时跳转 | 等待输出WAKE |
| WAIT\_FALLDOWN | IDLE | 等待400ns后，检测下降沿个数满足要求时跳转 | 等待输出WAKE |
| WAIT\_FALLDOWN | RISEUP | 等待400ns后，检测下降沿个数不满足要求时跳转 | 等待输出WAKE |
| INVALID\_SENDMSG | IDLE | OBFF消息发送完成即跳转 | 等待发送OBFF消息 |



上行端口状态机，主要功能为根据RC下发的WAKE信号，判断和输出当前USP的OBFF状态；(注，当判断不符合WAKE状态时序条件时，即进入CPUACTIVE状态)

|  |  |  |  |
| --- | --- | --- | --- |
| 当前状态 | 下一个状态 | 转移条件 | 状态说明 |
| IDLE | CHECK\_IF\_IDLE | 在WAKE模式，满足以下任意条件：  现在OBFF状态为IDLE；  现在OBFF状态为OBFF，之前OBFF状态为CPUACTIVE；  现在OBFF状态为CPUACTIVE，之前OBFF状态为OBFF； | 待机状态 |
| IDLE | WAIT\_RISEUP | 在WAKE模式，检测WAKE输入信号wake\_in\_tl下降沿时跳转 | 待机状态 |
| CHECK\_IF\_IDLE | CHECK\_PATTERN | 等待最大脉冲宽度700ns后跳转； | 待机状态 |
| CHECK\_IF\_IDLE | WAIT\_RISEUP | 事件发生后300~700ns内，检测WAKE输入信号wake\_in\_tl下降沿时跳转 | 使能WAKE输出0 |
| CHECK\_PATTERN | IDLE | 无条件，直接跳转 | 清上下沿计数器，并根据上下沿次数判断OBFF状态 |
| WAIT\_RISEUP | CHECK\_PATTERN | wake\_in\_tl下降沿开始等待1us后直接跳转 | 等待wake上升沿 |
| WAIT\_RISEUP | WAIT\_FALLDOWN | 检测WAKE输入信号wake\_in\_tl上升沿时，若上升沿事件发生在计时300~700ns内跳转 | 等待wake上升沿 |
| WAIT\_RISEUP | INVALID\_SENDMSG | 检测WAKE输入信号wake\_in\_tl上升沿时，若上升/下降沿事件不发生在计时300~700ns内即跳转 | 等待wake上升沿 |
| WAIT\_FALLDOWN | CHECK\_PATTERN | wake\_in\_tl下降沿开始等待1us后直接跳转，或以下条件：检测WAKE输入信号wake\_in\_tl下降沿时，若事件发生在计时300~700ns内，下降沿发生1次即跳转 | 等待wake下降沿 |
| WAIT\_FALLDOWN | WAIT\_RISEUP | 检测WAKE输入信号wake\_in\_tl下降沿时，若事件发生在计时300~700ns内，下降沿发生非1次即跳转 | 等待wake下降沿 |
| WAIT\_FALLDOWN | INVALID\_SENDMSG | 检测WAKE输入信号wake\_in\_tl上升沿时，若上升/下降沿事件不发生在计时300~700ns内即跳转 | 等待wake下降沿 |