openEuler

Linux PCI子系统介绍

汪雄峰

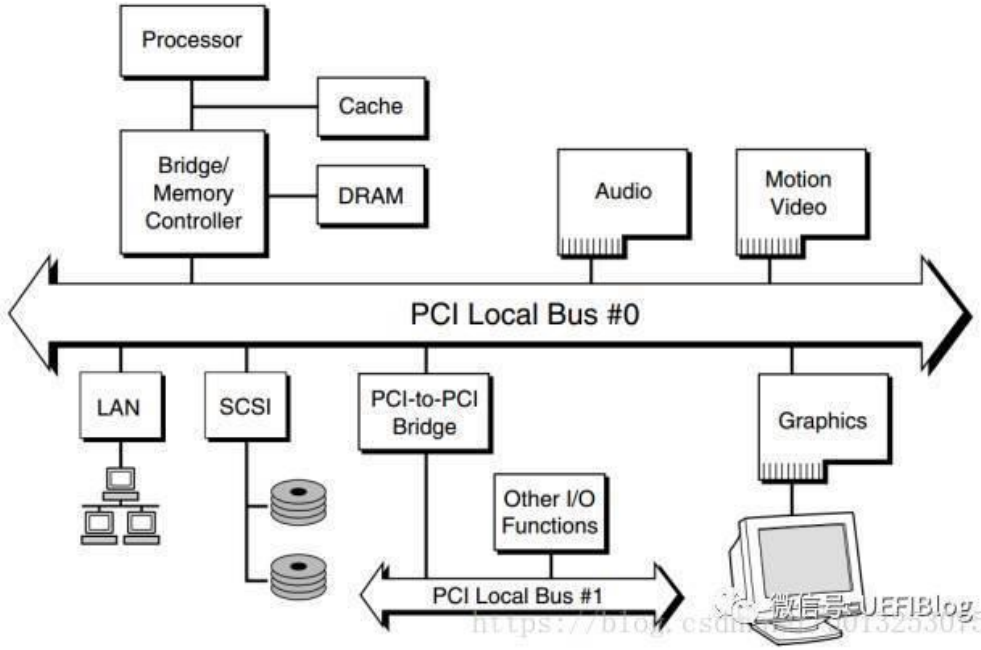
目录

CONTENT

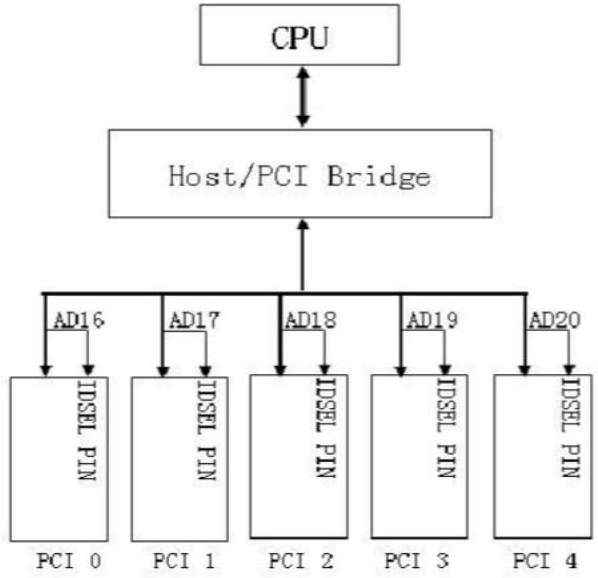
openEuler

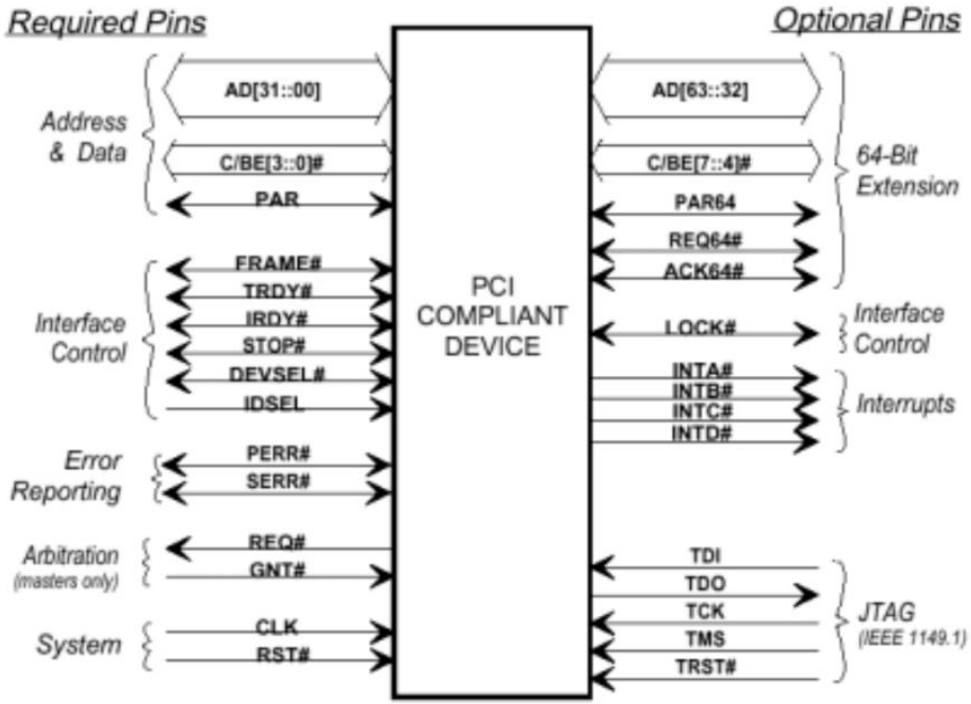
什么是PCI设备

- PCI总线
- 并行传输
- 仲裁机制
- 共享总线结构



一颗典型的PCI总线树





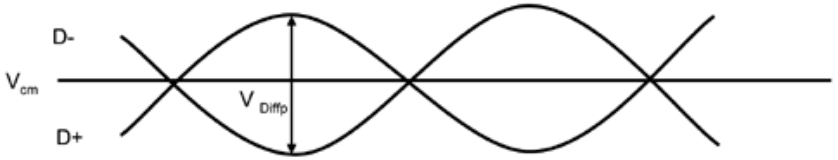
PCI总线引脚图

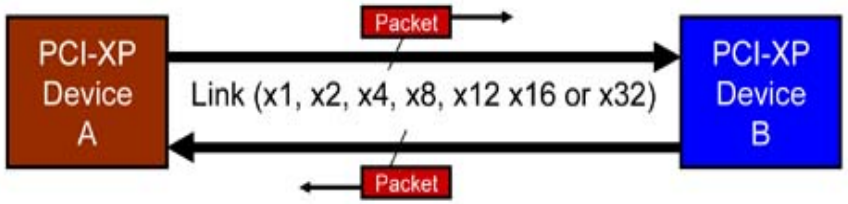
openEuler

PCI与PCIe

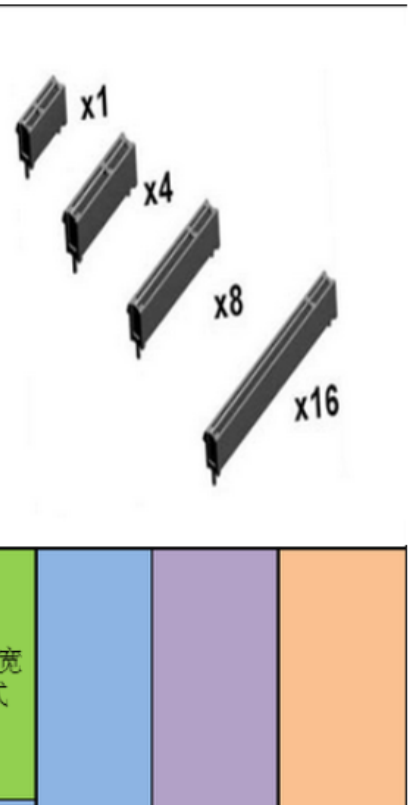
PCIe总线:

- 差分串行传输
- 点对点结构
- 多条lane





针脚号	定义（B）	说明	定义（A）	说明
1	+12V	+12V电压	PRSNT1#	热拔插存在检测
2	+12V	+12V电压	+12V	+12V电压
3	RSVD	保留针脚	+12V	+12V电压
4	GND	地	GND	地
5	SMCLK	系统管理总线时钟	JTAG2	测试时钟/TCK
6	SMDAT	系统管理总线数据	JTAG3	测试数据输入/TDI
7	GND	地	JTAG4	测试数据输出/TDO
8	+3.3V	+3.3V电压	JTAG5	测试模式选择/TMS
9	JTAG1	测试复位/TRST	+3.3V	+3.3V电压
10	3.3VAUX	3.3V辅助电源	+3.3V	+3.3V电压
11	WAKE#	链接激活信号	PWRGD	电源准备好信号
12	RSVD	保留针脚	GND	地
13	GND	地	REFCLK+	差分信号对参考时钟
14	HSOp(0)	0号信道发送差分信号对	REFCLK-	
15	HSOn(0)		GND	地
16	GND	地	HSIp(0)	0号信道接收差分信号对
17	PRSNT2#	热拔插存在检测	HSIn(0)	
18	GND	地	GND	地



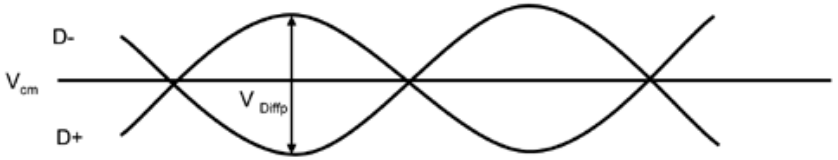
PCIe总线引脚图

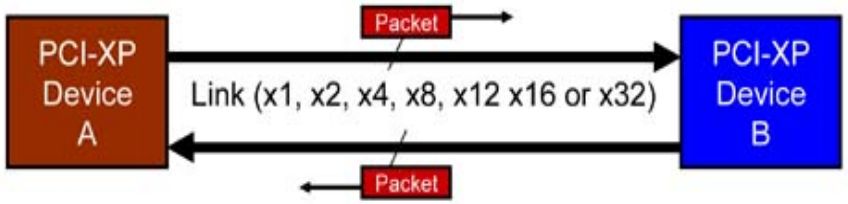
openEuler

PCI与PCIe

PCIe总线:

- 差分串行传输
- 点对点结构
- 多条lane





针脚号	定义（B）
1	+12V
2	+12V
3	RSVD
4	GND
5	SMCLK
6	SMDAT
7	GND
8	+3.3V
9	JTAG1
10	3.3VAUX
11	WAKE#
12	RSVD
13	GND
14	HSOp(0)
15	HSOn(0)
16	GND
17	PRSNT2#
18	GND

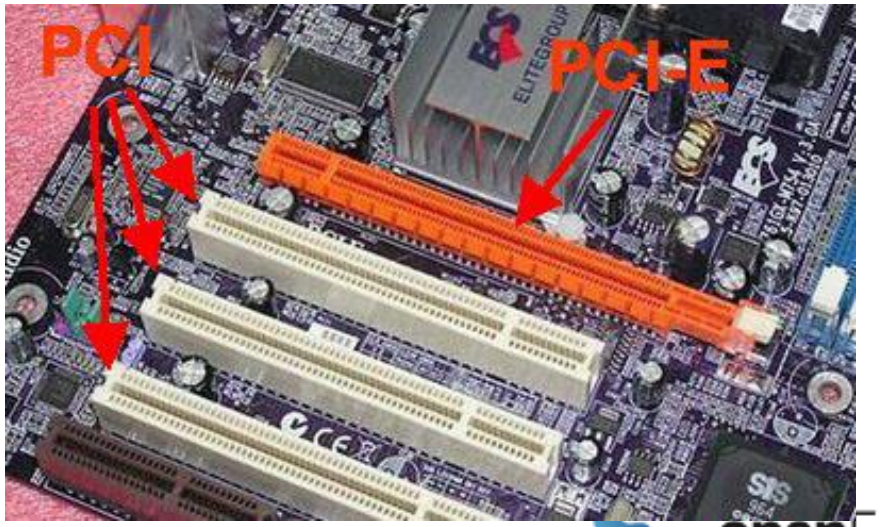
标准	时钟	传输位宽	每时钟数据	带宽
ISA	4.77 MHz	8	1	4.77 MB/s
ISA	8 MHz	16	0.5	8 MB/s
MCA	5 MHz	16	1	10 MB/s
MCA	5 MHz	32	1	20 MB/s
EISA	8.33 MHz	32	1	33.3 MB/s (16.7 MB/s typically)
VLB	33 MHz	32	1	133 MB/s
PCI	33 MHz	32	1	133 MB/s
PCI-X 66	66 MHz	64	1	533 MB/s
PCI-X 133	133 MHz	64	1	1,066 MB/s
PCI-X 266	133 MHz	64	2	2,132 MB/s
PCI-X 533	133 MHz	64	4	4,266 MB/s
AGP x1	66 MHz	32	1	266 MB/s
AGP x2	66 MHz	32	2	533 MB/s
AGP x4	66 MHz	32	4	1,066 MB/s
AGP x8	66 MHz	32	8	2,133 MB/s
PCIe 1.0 x1	2.5 GHz	1	1	250 MB/s
PCIe 1.0 x4	2.5 GHz	4	1	1,000 MB/s
PCIe 1.0 x8	2.5 GHz	8	1	2,000 MB/s
PCIe 1.0 x16	2.5 GHz	16	1	4,000 MB/s
PCIe 2.0 x1	5 GHz	1	1	500 MB/s
PCIe 2.0 x4	5 GHz	4	1	2,000 MB/s
PCIe 2.0 x8	5 GHz	8	1	4,000 MB/s
PCIe 2.0 x16	5 GHz	16	1	8,000 MB/s
PCIe 3.0 x1	8 GHz	1	1	1,000 MB/s
PCIe 3.0 x4	8 GHz	4	1	4,000 MB/s
PCIe 3.0 x8	8 GHz	8	1	8,000 MB/s
PCIe 3.0 x16	8 GHz	16	1	16,000 MB/s

PCI与PCIe总线速度对比

PCIe硬件形态

SSD

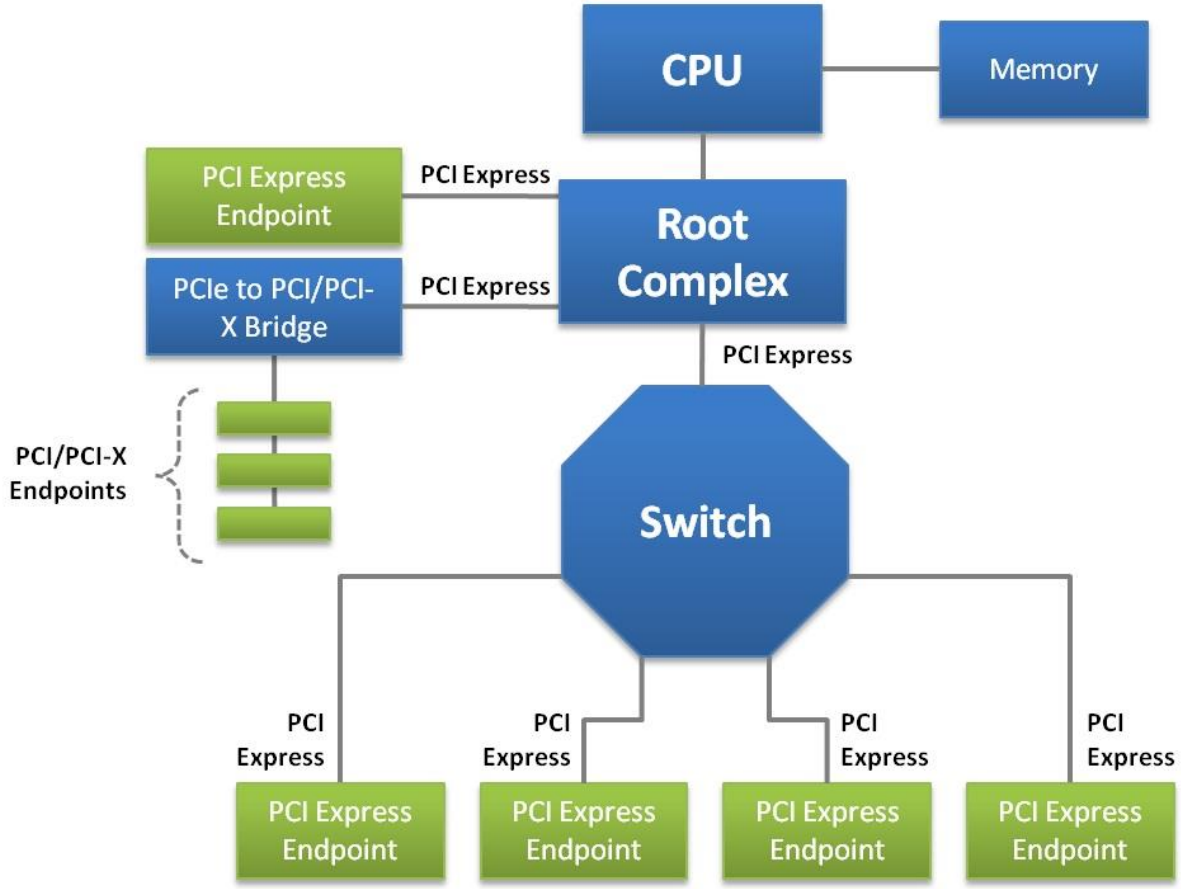
网卡



PCI与PCIe插槽

openEuler

PCIe总线拓扑



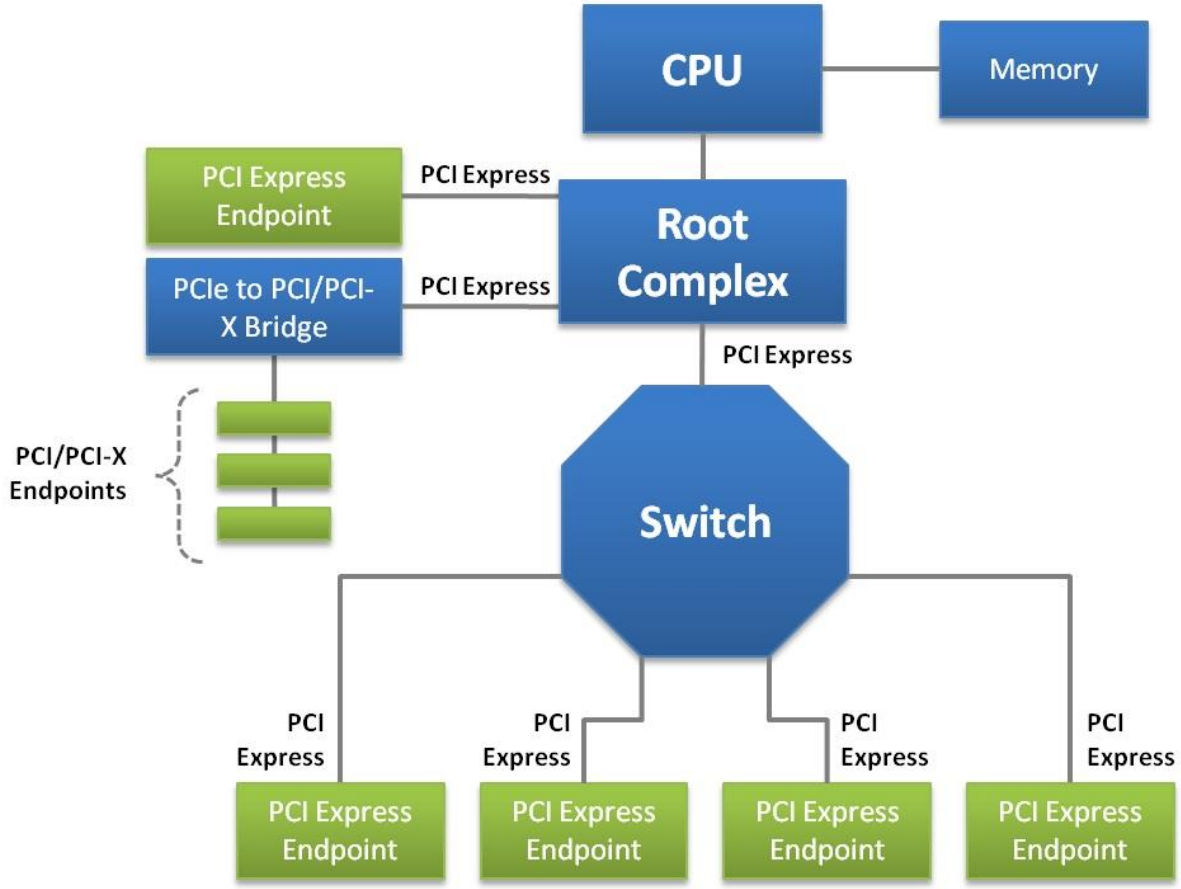
- Root Complex (RC)
- Switch
- Root Port (RP)
- Upstream Port
- Downstream Port
- Endpoint (EP)

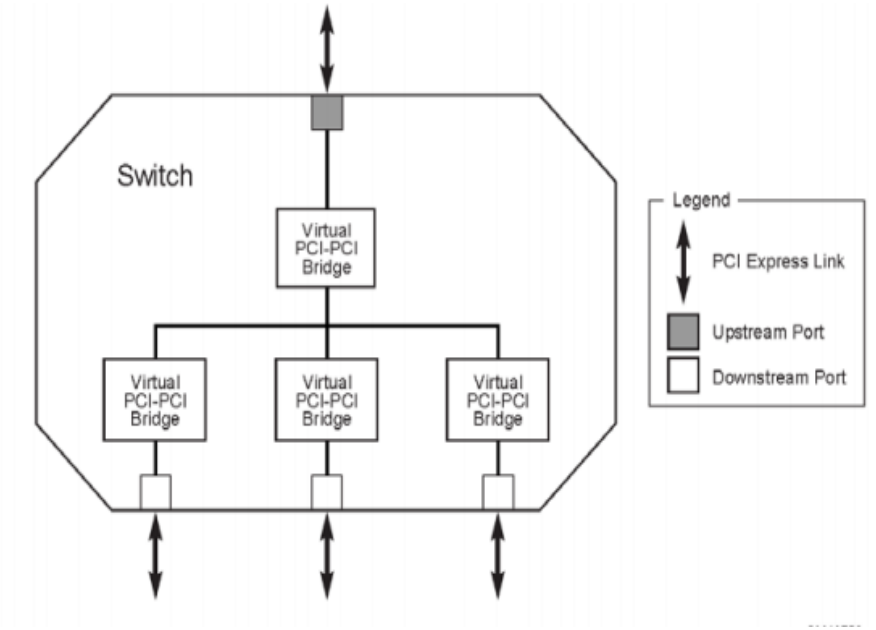
为什么需要Switch？

Switch内部结构？

openEuler

PCIe总线拓扑





为什么需要Switch？

Switch内部结构？

openEuler

CPU域与PCI域

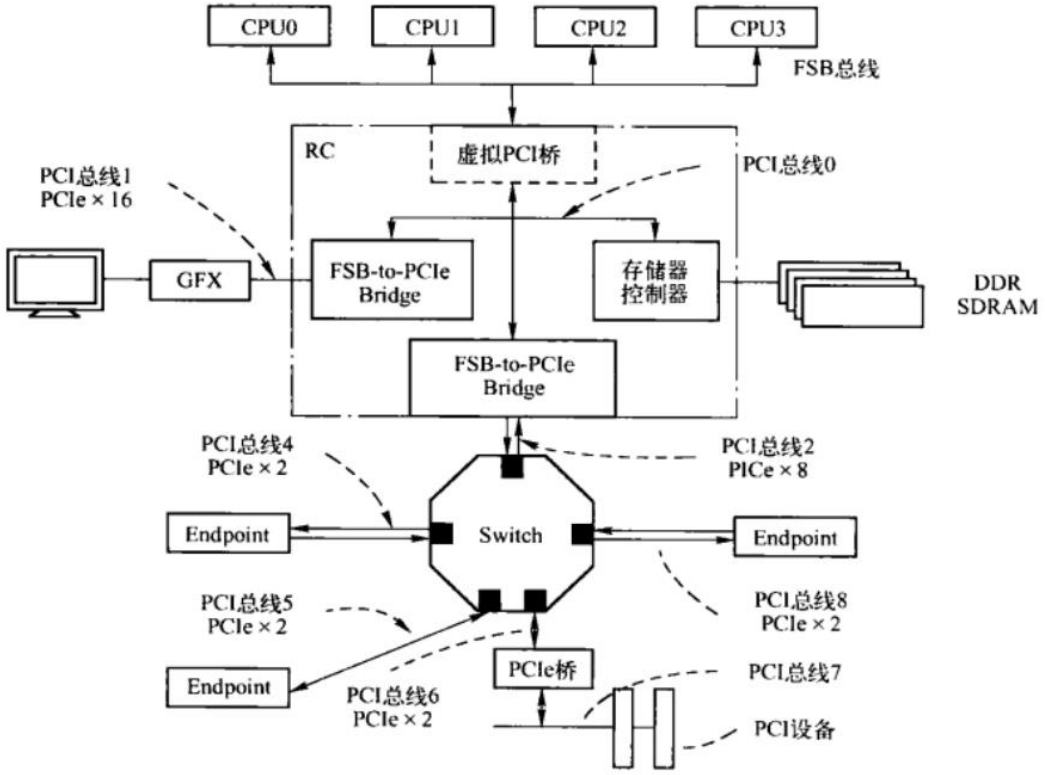
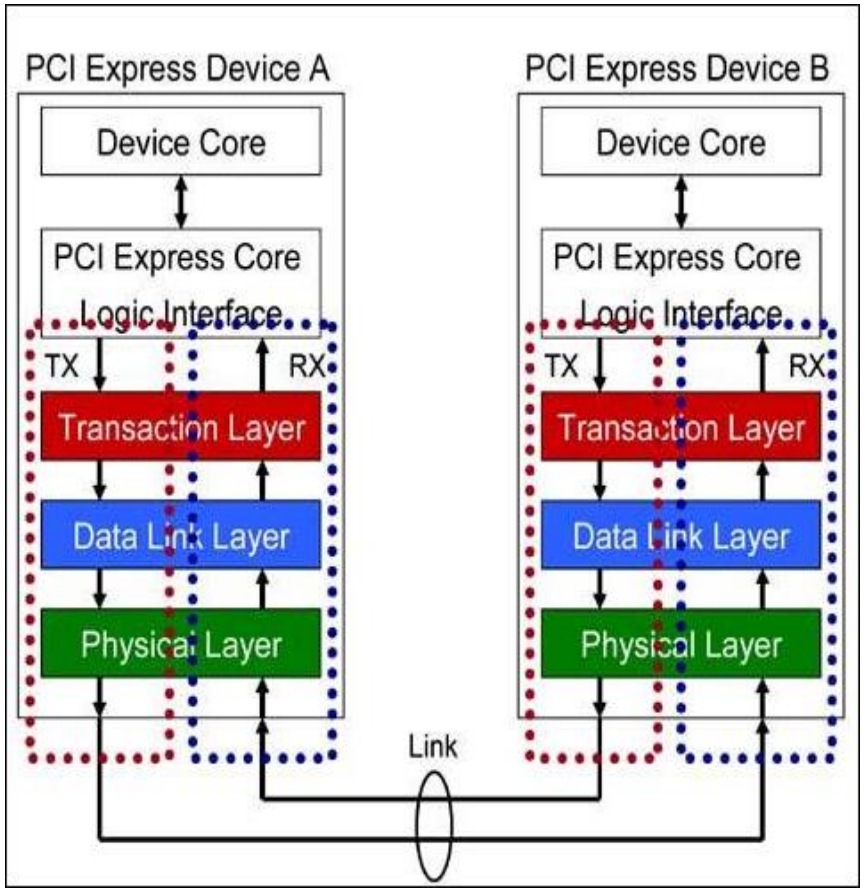


图 4-7　基于 PCIe 总线的处理器系统 A



- 事务层
- 数据链路层
- 物理层

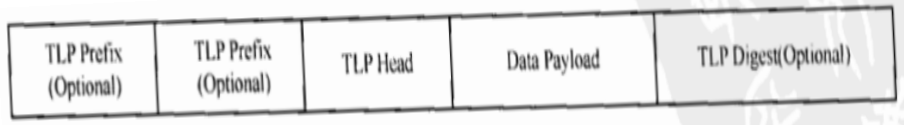


图 6-1　TLP 的格式

Switch怎么判断数据报文发到下行端口中的哪一个？

openEuler

查看系统PCI设备

- 查看pci设备拓扑： lspci –tv (-Dnn)
- 查看配置空间：lspci –vvv –s BDF号
- 以二进制方式查看配置空间：lspci –xxx –s BDF号 (-xxxx 查看扩展配置空间)

```
[root@server1 ~]# lspci -tv -Dnn
-+-[0000:7c]---00.0-[7d]--+-00.0  Huawei Technologies Co., Ltd. HNS GE/10GE/25GE RDMA Network Controller [19e5:a222]
 |                        +-00.1  Huawei Technologies Co., Ltd. HNS GE/10GE/25GE Network Controller [19e5:a221]
 |                        +-00.2  Huawei Technologies Co., Ltd. HNS GE/10GE/25GE RDMA Network Controller [19e5:a222]
 |                        \-00.3  Huawei Technologies Co., Ltd. HNS GE/10GE/25GE Network Controller [19e5:a221]
 +-[0000:7b]---00.0  Huawei Technologies Co., Ltd. HiSilicon Embedded DMA Engine [19e5:a122]
 +-[0000:7a]-+-00.0  Huawei Technologies Co., Ltd. HiSilicon USB 1.1 Host Controller [19e5:a23b]
 |           +-01.0  Huawei Technologies Co., Ltd. HiSilicon USB 2.0 2-port Host Controller [19e5:a239]
 |           \-02.0  Huawei Technologies Co., Ltd. HiSilicon USB 3.0 Host Controller [19e5:a238]
 +-[0000:74]-+-01.0-[76]--
 |           +-02.0  Huawei Technologies Co., Ltd. HiSilicon SAS 3.0 HBA [19e5:a230]
 |           +-03.0  Huawei Technologies Co., Ltd. HiSilicon AHCI HBA [19e5:a235]
 |           \-04.0  Huawei Technologies Co., Ltd. HiSilicon SAS 3.0 HBA [19e5:a230]
 \-[0000:00]-+-00.0-[01]--
             +-04.0-[02]--
             +-08.0-[03]----00.0  LSI Logic / Symbios Logic MegaRAID Tri-Mode SAS3408 [1000:0017]
             +-0c.0-[04-09]----00.0-[05-09]--+-00.0-[06]----00.0  Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) [19e5:1822]
             |                               +-01.0-[07]----00.0  Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) [19e5:1822]
             |                               +-02.0-[08]----00.0  Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) [19e5:1822]
             |                               \-03.0-[09]----00.0  Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) [19e5:1822]
             +-10.0-[0a]----00.0  Huawei Technologies Co., Ltd. iBMA Virtual Network Adapter [19e5:1710]
             +-11.0-[0b]----00.0  Huawei Technologies Co., Ltd. Hi1710 [iBMC Intelligent Management system chip w/VGA support] [19e5:1711]
             \-12.0-[0c]--
```

PCI域：总线号(8bit)：设备号(5bit)：功能号(3bit)

例: 0000：01：00. 0

openEuler

查看系统PCI设备

- 查看pci设备拓扑： lspci –tv (-Dnn)
- 查看配置空间：lspci –vvv –s BDF号
- 以二进制方式查看配置空间：lspci –xxx –s BDF号 (-xxxx 查看扩展配置空间)

```
[root@server1 ~]# lspci -vvv -s 08:00.0
08:00.0 Ethernet controller: Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) (rev 45)
        Subsystem: Huawei Technologies Co., Ltd. Device d136
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr+ Stepping- SERR+ FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx+
        Latency: 0, Cache Line Size: 32 bytes
        NUMA node: 0
        Region 0: Memory at 80003400000 (64-bit, prefetchable) [size=128K]
        Region 2: Memory at 800034e0000 (64-bit, prefetchable) [size=32K]
        Region 4: Memory at 80003300000 (64-bit, prefetchable) [size=1M]
        Capabilities: [40] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 512 bytes, PhantFunc 0, Latency L0s unlimited, L1 unlimited
                        ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset+ SlotPowerLimit 0.000W
                DevCtl: CorrErr+ NonFatalErr+ FatalErr+ UnsupReq+
                        RlxdOrd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+ FLReset-
                        MaxPayload 256 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr+ NonFatalErr- FatalErr- UnsupReq+ AuxPwr+ TransPend-
                LnkCap: Port #0, Speed 8GT/s, Width x16, ASPM not supported
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 128 bytes Disabled- CommClk-
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 8GT/s (ok), Width x16 (ok)
                        TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range B, TimeoutDis+, LTR-, OBFF Not Supported
                         AtomicOpsCap: 32bit+ 64bit+ 128bitCAS+
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
                         AtomicOpsCtl: ReqEn-
                LnkCtl2: Target Link Speed: 8GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete+, EqualizationPhase1+
                         EqualizationPhase2+, EqualizationPhase3+, LinkEqualizationRequest-
        Capabilities: [80] MSI: Enable- Count=1/32 Maskable+ 64bit+
                Address: 0000000000000000  Data: 0000
                Masking: 00000000  Pending: 00000000
        Capabilities: [a0] MSI-X: Enable+ Count=128 Masked-
                Vector table: BAR=2 offset=00000000
```

openEuler

查看系统PCI设备

- 查看pci设备拓扑： lspci –tv (-Dnn)
- 查看配置空间：lspci –vvv –s BDF号
- 以二进制方式查看配置空间：lspci –xxx –s BDF号 (-xxxx 查看扩展配置空间)

```
[root@server1 ~]# lspci -xxx -s 08:00.0
08:00.0 Ethernet controller: Huawei Technologies Co., Ltd. Hi1822
00: e5 19 22 18 46 05 18 00 45 00 00 02 08 00 00 00
10: 0c 00 40 03 00 08 00 00 0c 00 4e 03 00 08 00 00
20: 0c 00 30 03 00 08 00 00 00 00 00 00 e5 19 36 d1
30: 00 00 00 00 40 00 00 00 00 00 00 00 ff 00 00 00
40: 10 80 02 00 e2 8f 00 10 3f 29 19 00 03 f1 43 00
50: 08 00 03 01 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 92 03 00 00 00 00 00 00 0e 00 00 00
70: 03 00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 05 a0 8a 01 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0: 11 b0 7f 80 02 00 00 00 02 40 00 00 00 00 00 00
b0: 01 c0 03 f8 00 00 00 00 00 00 00 00 00 00 00 00
c0: 03 00 28 80 30 78 ff ff 00 00 00 00 00 00 00 00
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

怎样唯一标识一个PCI设备？

怎样访问一个PCI设备，设备驱动ioremap哪个地址呢？

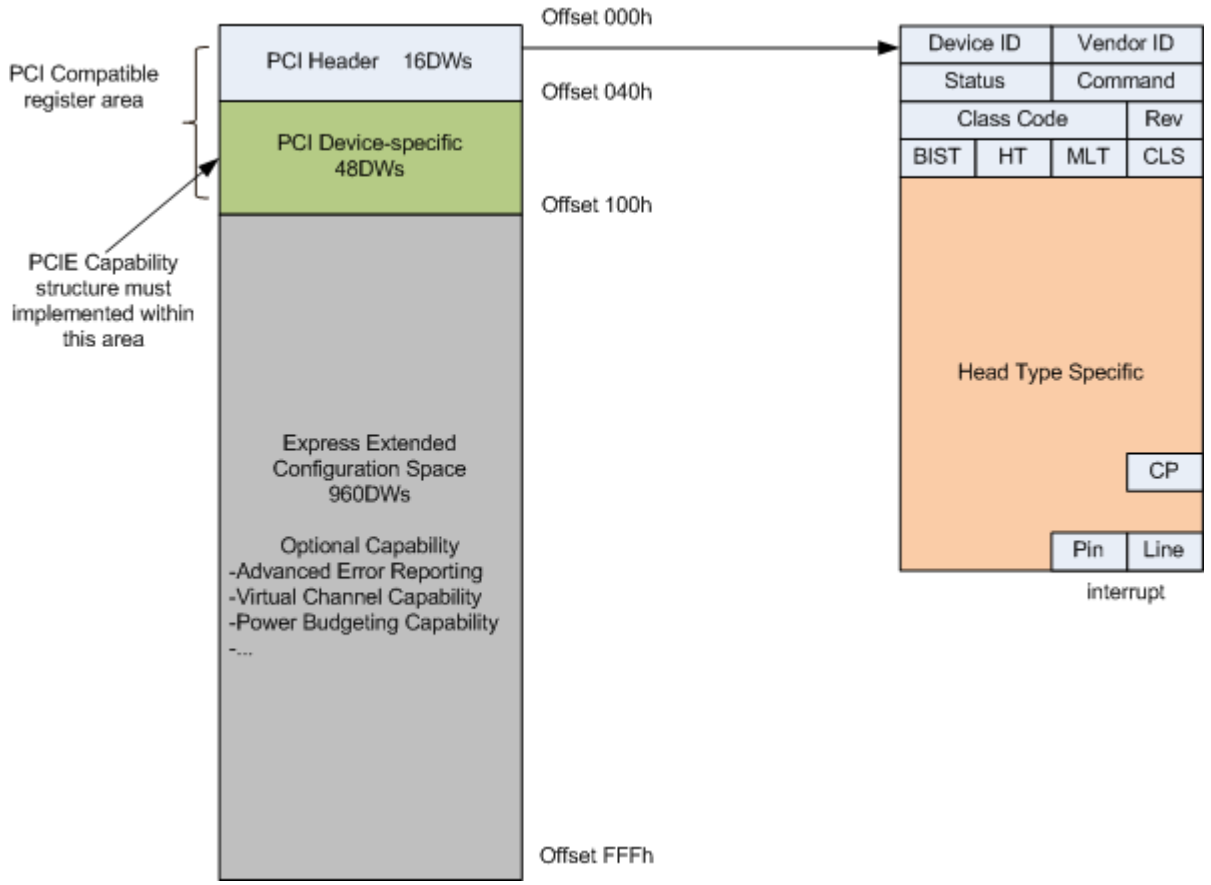
openEuler

PCI配置空间

- 配置空间
  - 可以直接访问
  - 用于发现设备
  - 用于分配mem/io地址
- MEM空间
  - 外设驱动访问
- IO空间
  - 外设驱动访问

PCI标准头空间	PCIe特性空间	设备专有寄存器空间

访问配置空间
- IO(CFC/CF8寄存器)
- ECAM方式

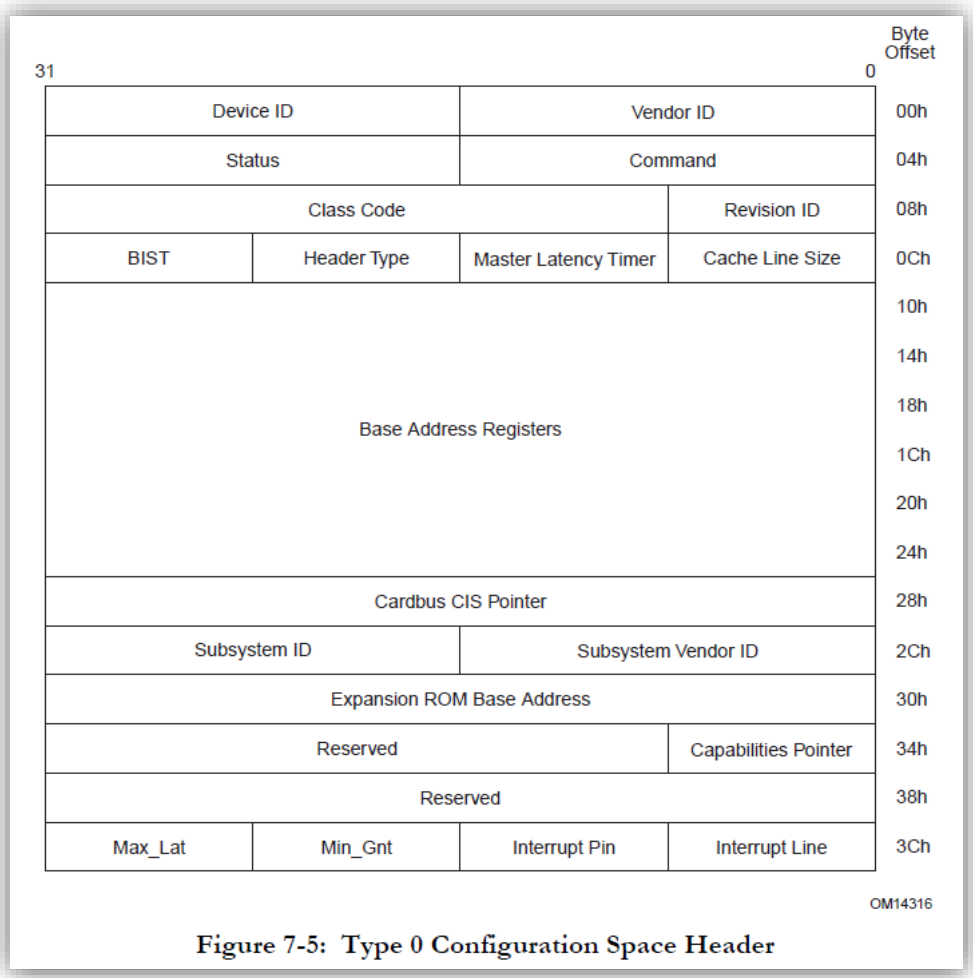


基本配置空间256字节，扩展配置空间4K

openEuler

PCI配置空间

64 Bytes



31			0	Byte Offset
Device ID			Vendor ID	00h
Status			Command	04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h–24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

OM14316

Figure 7-5: Type 0 Configuration Space Header

EndPoint 配置空间头

Device ID: 生产厂商指定的16位硬件设备编码

Vendor ID: 标识硬件制造商，如Intel 0x8086

Status: 设备相关状态位寄存器；

Commad： 设备控制寄存器，包括各种使能

Class: 设备所属类编码

Header Type: 设备所属PCI类型，端口设备或桥设备

Base : 基地址寄存器(min memory 128bytes)

Interrupt Pin: 对应PCI INTA#-INTD#四个中断引脚

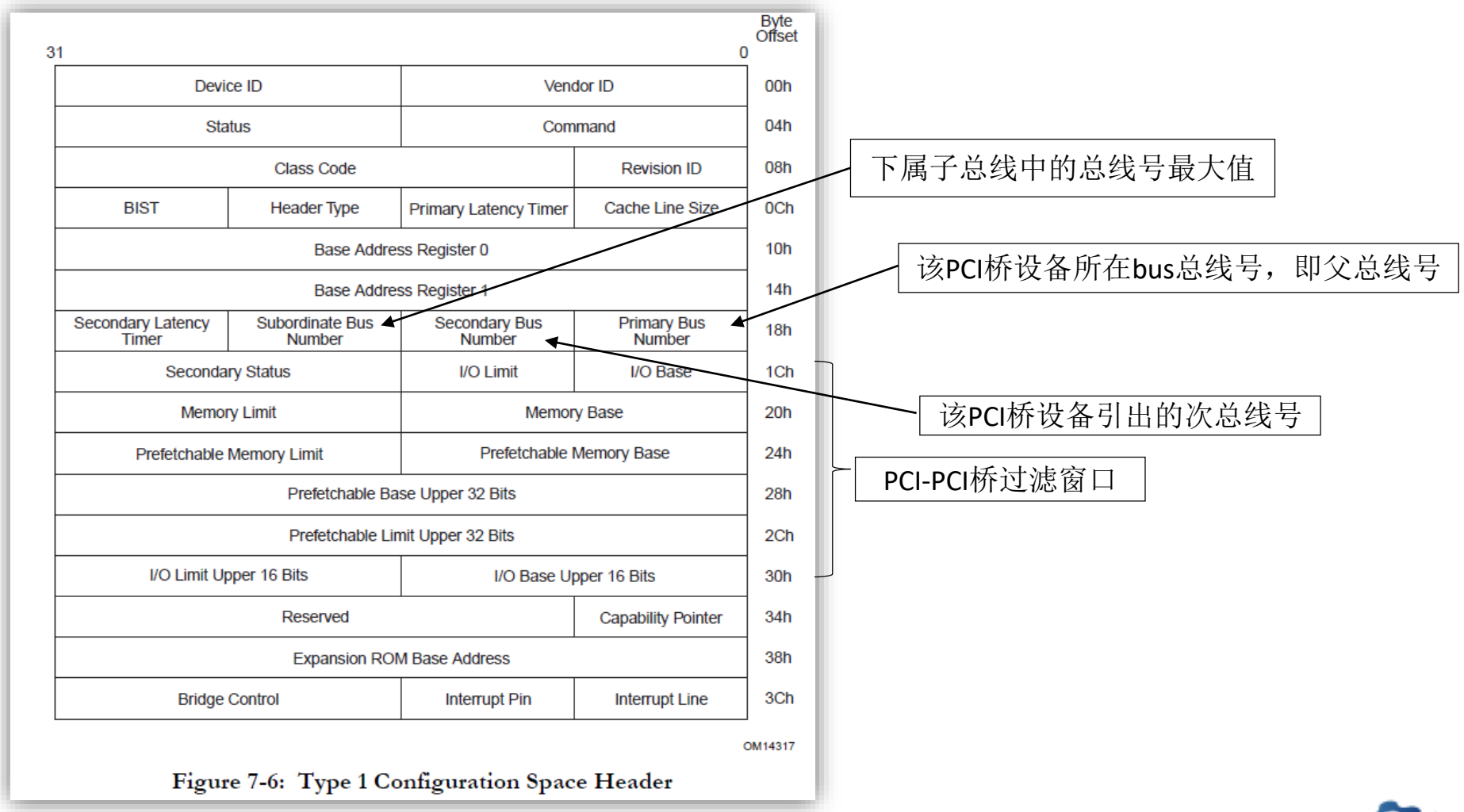
Interrupt Line: 保存中断路由信息，对应中断控制器的引脚(IRQ0-IRQ15)

Subsystem vendorId, subsystem deviceId: 进一步识别设备字段

Expansion ROM Base Register： 设备本身提供的Firmware映射信息的地址寄存器

openEuler

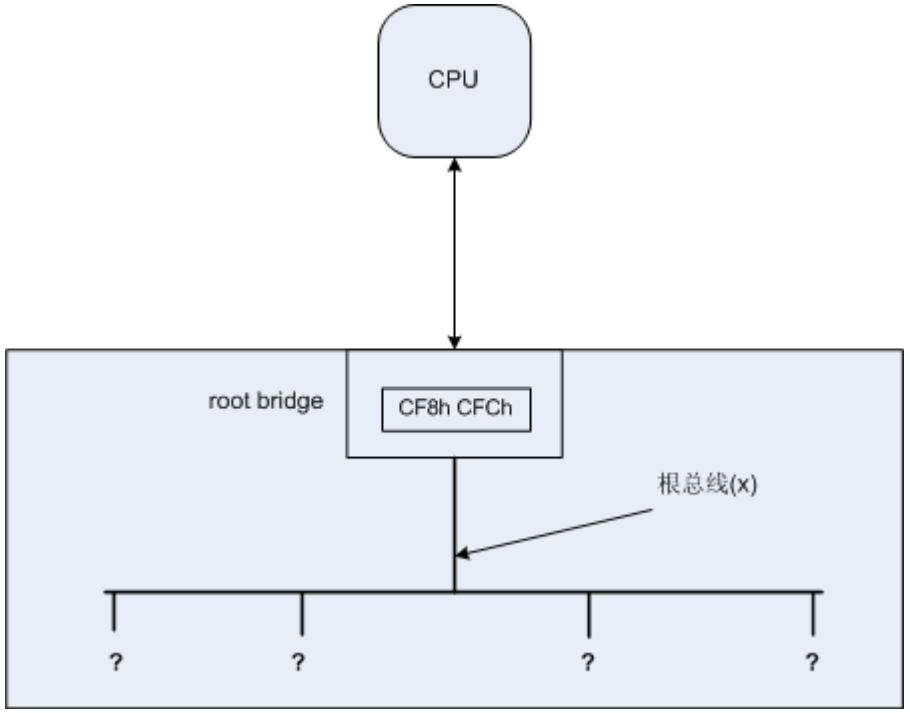
PCI配置空间



桥片配置空间头

openEuler

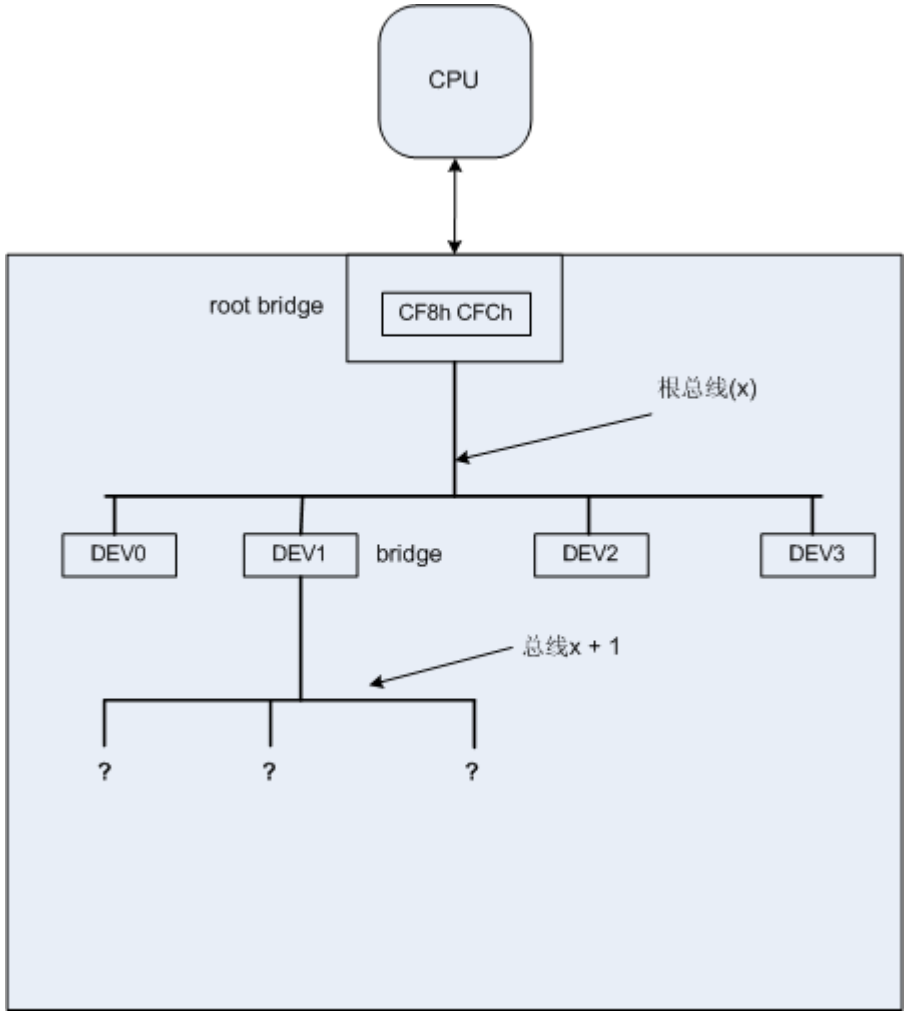
PCI初始化——设备枚举



- 遍历0号总线下的所有设备号功能号，device id和vendor id非全0或全f即存在设备
- 建立PCI设备对应的内核结构struct pci_dev

openEuler

PCI初始化——设备枚举



- 遍历0号总线下的所有设备号功能号，device id和vendor id非全0或全f即存在设备
- 建立PCI设备对应的内核结构struct pci_dev
- 如果某个设备为桥设备则引出1号总线
- 遍历1号总线下的所有设备号功能号，找出存在的PCI设备
- 依次递归

openEuler

PCI初始化代码分析

```
static const struct acpi_device_id root_device_ids[] = {
        {"PNP0A03", 0},
        {"", 0},
};

static struct acpi_scan_handler pci_root_handler = {
        .ids = root_device_ids,
        .attach = acpi_pci_root_add,
        .detach = acpi_pci_root_remove,
        .hotplug = {
                .enabled = true,
                .scan_dependent = acpi_pci_root_scan_dependent,
        },
};
```

```
Device (PCI2)
{
  Name (_HID, "PNP0A08") // PCI Express Root Bridge
  Name (_CID, "PNP0A03") // Compatible PCI Root Bridge
  Name(_SEG, 2) // Segment of this Root complex
  Name(_BBN, 0x80) // Base Bus Number
  Name(_CCA, 1)
  Method (_CRS, 0, Serialized) { // Root complex resources
    Name (RBUF, ResourceTemplate () {
      WordBusNumber ( // Bus numbers assigned to this root
        ResourceProducer, MinFixed, MaxFixed, PosDecode,
        0, // AddressGranularity
        0x80, // AddressMinimum - Minimum Bus Number
        0x87, // AddressMaximum - Maximum Bus Number
        0, // AddressTranslation - Set to 0
        0x8 // RangeLength - Number of Busses
      )
      QWordMemory ( // 64-bit BAR Windows
        ResourceProducer,
        PosDecode,
        MinFixed,
        MaxFixed,
        Cacheable,
        ReadWrite,
        0x0, // Granularity
        0xa8800000, // Min Base Address
        0xaffeffff, // Max Base Address
        0x0, // Translate
        0x77f0000 // Length
      )
      QWordIO (
        ResourceProducer,
        MinFixed,
        MaxFixed,
        PosDecode,
        EntireRange,
        0x0, // Granularity
        0x0, // Min Base Address
        0xffff, // Max Base Address
        0xafff0000, // Translate
        0x10000 // Length
      )
    }) // Name(RBUF)
    Return (RBUF)
  } // Method(_CRS)
```

acpi_pci_root_add() //drivers/acpi/pci_root.c
->pci_acpi_scan_root() //arch/arm64/kernel/pci.c
  ->acpi_pci_root_create();
    ->pci_scan_child_bus(bus); //枚举设备
  ->pci_assign_unassigned_root_bus_resources(bus); //分配资源

PCI设备枚举代码分析

```
static unsigned int pci_scan_child_bus_extend(struct pci_bus *bus,
/*  */
                                              unsigned int available_buses)
{
        unsigned int used_buses, normal_bridges = 0, hotplug_bridges = 0;
        unsigned int start = bus->busn_res.start;
        unsigned int devfn, fn, cmax, max = start;
        struct pci_dev *dev;
        int nr_devs;

        dev_dbg(&bus->dev, "scanning bus\n");

        /* Go find them, Rover! */
        for (devfn = 0; devfn < 256; devfn += 8) {
                nr_devs = pci_scan_slot(bus, devfn);

                /*
                 * The Jailhouse hypervisor may pass individual functions of a
                 * multi-function device to a guest without passing function 0.
                 * Look for them as well.
                 */
                if (jailhouse_paravirt() && nr_devs == 0) {
/*
 * Scan bridges that are already configured.  We don't touch them
 * unless they are misconfigured (which will be done in the second
 * scan below).
 */
for_each_pci_bridge(dev, bus) {
        cmax = max;
        max = pci_scan_bridge_extend(bus, dev, max, 0, 0);

        /*
         * Reserve one bus for each bridge now to avoid extending
         * hotplug bridges too much during the second scan below.
         */
        used_buses++;
        if (cmax - max > 1)
                used_buses += cmax - max - 1;
}
```

```
int pci_scan_slot(struct pci_bus *bus, int devfn)
/*  */
{
        unsigned fn, nr = 0;
        struct pci_dev *dev;

        if (only_one_child(bus) && (devfn > 0))
                return 0; /* Already scanned the entire slot */

        dev = pci_scan_single_device(bus, devfn);
        if (!dev)
                return 0;
        if (!pci_dev_is_added(dev))
                nr++;

        for (fn = next_fn(bus, dev, 0); fn > 0; fn = next_fn(bus, dev, fn)) {
                dev = pci_scan_single_device(bus, devfn + fn);
                if (dev) {
                        if (!pci_dev_is_added(dev))
                                nr++;
                        dev->multifunction = 1;
                }
        }
```

openEuler

PCI设备对应的内核结构体

```
struct pci_bus {
        struct list_head node;          /* Node in list of buses */
        struct pci_bus  *parent;        /* Parent bus this bridge is on */
        struct list_head children;      /* List of child buses */
        struct list_head devices;       /* List of devices on this bus */
        struct pci_dev  *self;          /* Bridge device as seen by parent */
        struct list_head slots;         /* List of slots on this bus;
                                           protected by pci_slot_mutex */
        struct resource *resource[PCI_BRIDGE_RESOURCE_NUM];
        struct list_head resources;     /* Address space routed to this bus */
        struct resource busn_res;       /* Bus numbers routed to this bus */

        struct pci_ops  *ops;           /* Configuration access functions */
        struct pci_ops  *backup_ops;
        struct msi_controller *msi;     /* MSI controller */
        void            *sysdata;       /* Hook for sys-specific extension */
        struct proc_dir_entry *procdir; /* Directory entry in /proc/bus/pci */

        unsigned char   number;         /* Bus number */
        unsigned char   primary;        /* Number of primary bridge */
```

PCI总线对应的内核结构体

```
/* The pci_dev structure describes PCI devices */
struct pci_dev {
        struct list_head bus_list;      /* Node in per-bus list */
        struct pci_bus  *bus;           /* Bus this device is on */
        struct pci_bus  *subordinate;   /* Bus this device bridges to */

        void            *sysdata;       /* Hook for sys-specific extension */
        struct proc_dir_entry *procent; /* Device entry in /proc/bus/pci */
        struct pci_slot *slot;          /* Physical slot this device is in */

        unsigned int    devfn;          /* Encoded device & function index */
        unsigned short  vendor;
        unsigned short  device;
        unsigned short  subsystem_vendor;
        unsigned short  subsystem_device;
        unsigned int    class;          /* 3 bytes: (base,sub,prog-if) */
        u8              revision;       /* PCI revision, low byte of class word */
        u8              hdr_type;       /* PCI header type (`multi' flag masked out) */
#ifdef CONFIG_PCIEAER
        u16             aer_cap;        /* AER capability offset */
        struct aer_stats *aer_stats;    /* AER stats for this device */
#endif
        u8              pcie_cap;       /* PCIe capability offset */
        u8              msi_cap;        /* MSI capability offset */
        u8              msix_cap;       /* MSI-X capability offset */
        u8              pcie_mpss:3;    /* PCIe Max Payload Size Supported */
        u8              rom_base_reg;   /* Config register controlling ROM */
        u8              pin;            /* Interrupt pin this device uses */
        u16             pcie_flags_reg; /* Cached PCIe Capabilities Register */
        unsigned long   *dma_alias_mask;/* Mask of enabled devfn aliases */

        struct pci_driver *driver;      /* Driver bound to this device */
```

```
struct device   dev;                    /* Generic device interface */

int             cfg_size;               /* Size of config space */

/*
 * Instead of touching interrupt line and base address registers
 * directly, use the values stored here. They might be different!
 */
unsigned int    irq;
struct resource resource[DEVICE_COUNT_RESOURCE]; /* I/O and memory regions + expansion ROMs */
```

PCI设备对应的内核结构体

openEuler

PCI设备驱动

```c
int pci_setup_device(struct pci_dev *dev)
{
        u32 class;
        u16 cmd;
        u8 hdr_type;
        int pos = 0;
        struct pci_bus_region region;
        struct resource *res;

        hdr_type = pci_hdr_type(dev);

        dev->sysdata = dev->bus->sysdata;
        dev->dev.parent = dev->bus->bridge;
        dev->dev.bus = &pci_bus_type;
        dev->hdr_type = hdr_type & 0x7f;
        dev->multifunction = !!(hdr_type & 0x80);
        dev->error_state = pci_channel_io_normal;
        set_pcie_port_type(dev);
```

```c
struct bus_type pci_bus_type = {
        .name           = "pci",
        .match          = pci_bus_match,
        .uevent         = pci_uevent,
        .probe          = pci_device_probe,
        .remove         = pci_device_remove,
        .shutdown       = pci_device_shutdown,
        .dev_groups     = pci_dev_groups,
        .bus_groups     = pci_bus_groups,
        .drv_groups     = pci_drv_groups,
        .pm             = PCI_PM_OPS_PTR,
        .num_vf         = pci_bus_num_vf,
        .dma_configure  = pci_dma_configure,
};
EXPORT_SYMBOL(pci_bus_type);
```

PCI总线

PCI驱动

```c
static struct pci_driver sec_pci_driver = {
        .name = "hisi_sec2",
        .id_table = sec_dev_ids,
        .probe = sec_probe,
        .remove = sec_remove,
        .err_handler = &sec_err_handler,
        .sriov_configure = sec_sriov_configure,
        .shutdown = hisi_qm_dev_shutdown,
};
```

```c
static int sec_probe(struct pci_dev *pdev, const struct pci_device_id *id)
{
        struct sec_dev *sec;
        struct hisi_qm *qm;
        int ret;

        sec = devm_kzalloc(&pdev->dev, sizeof(*sec), GFP_KERNEL);
        if (!sec)
                return -ENOMEM;
#ifdef CONFIG_CRYPTO_QM_UACCE
        qm->phys_base = pci_resource_start(pdev, PCI_BAR_2);
        qm->size = pci_resource_len(qm->pdev, PCI_BAR_2);
#endif
        qm->io_base = devm_ioremap(dev, pci_resource_start(pdev, PCI_BAR_2),
                                   pci_resource_len(qm->pdev, PCI_BAR_2));
        if (!qm->io_base) {
                ret = -EIO;
                goto err_release_mem_regions;
        }

        ret = dma_set_mask_and_coherent(dev, DMA_BIT_MASK(64));
        if (ret < 0) {
                dev_err(dev, "Failed to set 64 bit dma mask %d", ret);
                goto err_iounmap;
        }
        pci_set_master(pdev);

        num_vec = qm->ops->get_irq_num(qm);
        ret = pci_alloc_irq_vectors(pdev, num_vec, num_vec, PCI_IRQ_MSI);
        if (ret < 0) {
                dev_err(dev, "Failed to enable MSI vectors!\n");
                goto err_iounmap;
        }
```

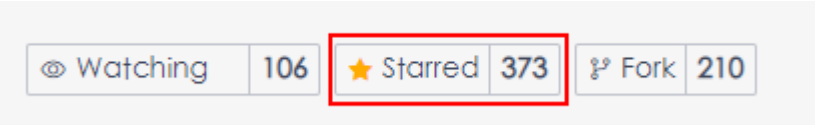
uler

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

openEuler

Thank you

openEuler