Project 5 Device Driver 设计文档

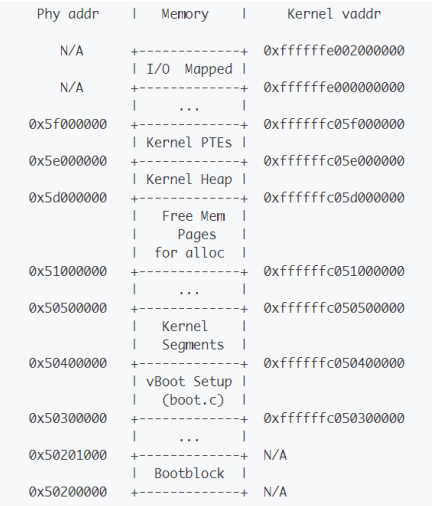
中国科学院大学
王嵩岳
2020 年 12 月 28 日

1. 网卡驱动

本次实验需要驱动的是 Xilinx PYNQ Z2 开发板 PS 端的千兆以太网控制器（GEM）。我们的软核配置在 PL 端，这个过程需要 PS 和 PL 端构成的系统联合配置以实现驱动功能。

（1） 进行 I/O 及控制寄存器的映射

我们需要映射系统级控制寄存器（System Level Control Registers）、网卡与 DMA 相关的控制寄存器（GEM）和外部中断控制器（Platform-Level Interrupt Controller）。RISCV 采用 I/O 统一编址，我将它们映射在了内核页表的 0xffffffe000000000 处。



由于 RISCV 页表项无法设置 uncache 属性，所以我们写这些寄存器后需要手动刷 D-cache 以便让设备看到。

（2） 初始化 PLIC 和网卡实例

我们需要填写的过程是初始化 RX、TX 描述符环。

以 RXBD 为例，首先我们从数组分配一段连续的空间给描述符环。

```
/* Allocate Rx and Tx BD space each */
RxBdSpacePtr = &(bd_space[0]);
TxBdSpacePtr = &(bd_space[0x10000]);
```

然后清空这个描述符，以它为基址设置描述符环（注意参数中有实地址）。

```
XEmacPs_BdClear(&BdTemplate);
/*
 * Create the RxBD ring
 */
Status = XEmacPs_BdRingCreate(&(XEmacPs_GetRxRing
                              (EmacPsInstancePtr)),
                              (UINTPTR) kva2pa(RxBdSpacePtr),
                              (UINTPTR) RxBdSpacePtr,
                              XEMACPS_BD_ALIGNMENT,
                              RXBD_CNT,
                              NULL);
if (Status != XST_SUCCESS) {
    EmacPsUtilErrorTrap
        ("Error setting up RxBD space, BdRingCreate");
    return XST_FAILURE;
}
```

然后通过 clone 方法做 32 个描述符加入环中。

```
Status = XEmacPs_BdRingClone(&(XEmacPs_GetRxRing(EmacPsInstancePtr)),
                             &BdTemplate, XEMACPS_RECV);

if (Status != XST_SUCCESS) {
    EmacPsUtilErrorTrap
        ("Error setting up RxBD space, BdRingClone");
    return XST_FAILURE;
}
```

轮询收发设计

发送设计：

（1） 申请一个 BD：由于我们的设计中，一次发送只会发送一个以太网帧，所以发送过程只需一个描述符。

```
Status = XEmacPs_BdRingAlloc(&(XEmacPs_GetTxRing(EmacPsInstancePtr)),
                             1, &Bd1Ptr);

if (Status != XST_SUCCESS) {
    EmacPsUtilErrorTrap("Error allocating TxBD");
    return XST_FAILURE;
}
```

（2） 设置 BD 属性：我们需要设置这个描述符的 Last 位并清除 Used 位，然后将以太网帧的物理地址和长度写入 BD 中。

```
XEmacPs_BdSetAddressTx(Bd1Ptr, (UINTPTR)TxFrame);
XEmacPs_BdSetLength(Bd1Ptr, length);
XEmacPs_BdClearTxUsed(Bd1Ptr);
XEmacPs_BdSetLast(Bd1Ptr);
```

（3） 将这个 BD 写入硬件并刷新 D-cache。

```
Status = XEmacPs_BdRingToHw(&(XEmacPs_GetTxRing(EmacPsInstancePtr)),
                            1, Bd1Ptr);
if (Status != XST_SUCCESS) {
    EmacPsUtilErrorTrap("Error committing TxBD to HW");
    return XST_FAILURE;
}

// remember to flush dcache
Xil_DCacheFlushRange(0, 64);
```

（4） 设置 TX BD ring 的指针并开始数据传输

```
// set tx queue base
XEmacPs_SetQueuePtr(EmacPsInstancePtr, EmacPsInstancePtr->TxBdRing.BaseBdAddr, 0, XEMACPS_SEND);
```

发送等待设计：

（1） 当没有开启网卡中断模式时，我们进行轮询 TXSR 的 complete 位以确定传输是否完成，传输完成后清除此位。

```
while (1)
{
    txsr = XEmacPs_ReadReg(EmacPsInstancePtr->Config.BaseAddress,
                XEMACPS_TXSR_OFFSET);
    if (txsr & XEMACPS_TXSR_TXCOMPL_MASK){
        XEmacPs_WriteReg(EmacPsInstancePtr->Config.BaseAddress,
                  XEMACPS_TXSR_OFFSET, txsr | XEMACPS_TXSR_TXCOMPL_MASK);
        break;
    }
}
```

当开启网卡中断模式后，首先会把进程阻塞。直到网卡中断到来时再进行下面的操作。

（2） 释放 BD 并重建 ring，这里 BD 从硬件读出

```
Status = XEmacPs_BdRingFree(&(XEmacPs_GetTxRing(EmacPsInstancePtr)),
                            1, Bd1Ptr);

if (Status != XST_SUCCESS) {
    EmacPsUtilErrorTrap("Error freeing up TxBDs");
    return XST_FAILURE;
}
EmacPsSetupTxBD(EmacPsInstancePtr);
```

（3） 如果是网卡中断模式，此时发送过程处理完毕，可以再次开启 TX 端的网卡中断。

```
if (irq_mode)
XEmacPs_IntEnable(
    EmacPsInstancePtr,
    (XEMACPS_IXR_TXCOMPL_MASK | XEMACPS_IXR_TX_ERR_MASK));
```

接收设计：

由于我们支持一次接收多个以太网帧，所以我们需要申请多个描述符。对各个描述符写入接收信息应该写入的物理内存地址，并在最后一个描述符设置 Last 位。

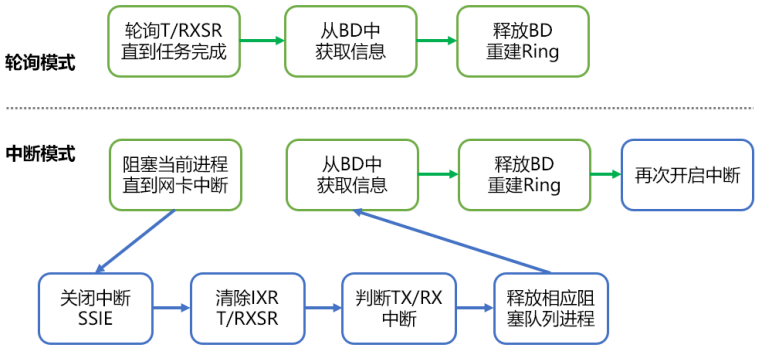
```
Status = XEmacPs_BdRingAlloc(&(XEmacPs_GetRxRing(EmacPsInstancePtr)),
                             num_packet, &BdRxPtr);
CurBd = BdRxPtr;
for (int i = 0; i < num_packet; i++){
    XEmacPs_BdSetAddressRx(CurBd, (UINTPTR)(RxFrame + i));
    XEmacPs_BdClearRxNew(CurBd);
    if (i == num_packet - 1)
        XEmacPs_BdSetRxWrap(CurBd);
    CurBd = XEmacPs_BdRingNext(&(XEmacPs_GetRxRing(EmacPsInstancePtr)), CurBd);
}

Status = XEmacPs_BdRingToHw(&(XEmacPs_GetRxRing(EmacPsInstancePtr)),
                            num_packet, BdRxPtr);
```

接收等待与发送等待不同的是，接收可能需要等多个描述符，这些任务不会一起完成。所以我们在循环中轮询 RXSR，或阻塞当前进程，当接收到的 BD 总数对上之后再返回，完成接收传输。

2. C-Core 设计

网卡中断的设计：

对于接收和发送进程，与轮询唯一的不同之处在于，在 Wait 时跳过轮询并直接阻塞。当网卡中断到来时，再去从硬件获取 BD 并处理信息。



对于外部中断，首先需要进入 PLIC 的处理函数，然后中断处理完毕后再调用 eoi 通知 PLIC 结束中断处理。

参考文献

[1] Xilinx embeddedsw official driver: emacps

[2] ug585-Zynq-7000-TRM

[3] RISC-V Spec Vol.2 Privileged Arch v1.10