Project 5 Device Driver 设计文档

中国科学院大学

蔡润泽

2019 年 12 月 16 日

1. 网卡驱动

（1） 任务一实现时，你初始化了几个接收描述符（RDES），每个接收描述符中设置了哪几个域的值，所设置的域的各自含义是什么？

在任务一的实现中，本设计初始化了 64 个接收描述符。接收描述符域如下：

```
for(i=0;i<PNUM-1;addr += DESC_SIZE,i++)
{
    ((desc_t *)addr)->tdes0 = 0;
    ((desc_t *)addr)->tdes1 = 0 | (1 << 24)  | (BUFFSIZE & 0x7ff);
    ((desc_t *)addr)->tdes2 = RxBuff & 0x1fffffff;
    ((desc_t *)addr)->tdes3 = (addr + DESC_SIZE)& 0x1fffffff;
    RxBuff += BUFFSIZE;
}

((desc_t *)addr)->tdes0 = 0;
((desc_t *)addr)->tdes1 = 0 | (1 << 25)  | (BUFFSIZE & 0x7ff);
((desc_t *)addr)->tdes2 = RxBuff & 0x1fffffff;
((desc_t *)addr)->tdes3 = RECV_DESC_PTR & 0x1fffffff;
```

其中：

tdes0 的 OWN 位置 0。

tdes1 的 24 和 25 分别表示循环链表的下一个节点是下一个节点还是头结点。其 31 位也可以置起，表示完成时禁止中断，置 1 时可以加速收包效率。

tdes2 表示收包缓存的物理地址，tdes3 表示收包描述符的物理地址。

（2） 任务二实现时，检查是否有数据包到达网卡这一操作是在哪个流程中执行的？例如是时钟处理流程，还是接收线程本身，或者是其他流程中？

在任务二中，检查是否有数据到达是在时钟中断里进行处理。具体的实现方法是：在测试函数中，线程在发包后首先会检查第 64 个包有没有被收到。由于收包需要一定的时间，此时第 64 个包没有收到，因而触发了阻塞接受线程的系统调用。

而时钟中断内会进行时候是否有接收线程被阻塞的判断，若有则进行 check_recv 的操作。check_recv 函数会根据 tdes0 来检测是否收到了包，并且在收满 64 个有效数据包后，解除进程阻塞，并回到接收线程进行打印。

（3） 任务三实现时，你的设计中，每接收到几个网络包时会产生一次中断？

在任务三的实现里，本设计采取了两种设计思路。第一种思路是每 64 次进行一次中断，第二种思路是没接受一次进行一次中断。在运行效率上来看，第一种思路的效率明显更高。而在处理无效包方面，第二种思路会更加方便。因为处理无效包需要在中断

后，将 tdes0 和 DMA2 号寄存器重新置位。倘若每收 64 个包才进行一次中断，此时若在发 64 个包之前，操作系统发了一些无效的包，就会导致有效包会因描述符和 buff 不够用而丢包。

（4） DMA 接收和发送描述符采用环形链表和链型链表都是可以的，你认为使用环形链表和使用链型链表有什么区别？

采用环形链表能够更加方便的重复使用描述符。单向链表连接描述符可以实现描述符的重复利用，但对比循环链表来实现描述符，单向链表存在一个缺点。

在使用单向链表实现描述符时，一轮传输完成后，就需要重新配置相关的 dma 寄存器和 mac 寄存器，才可重复利用。但使用循环链表就不需要进行这些操作，只用重新置位描述符 own 位和向 dma 寄存器 1（或 2）写任意值即可。

（5） 设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）

在这次实验过程中，我在处理无效包的部分花费了较多的时间。原本的设计发现会导致无效包照常输出，进过排查发现这与收包的处理速度和接收描述符禁止触发中断域的置位有关。

正如（3）所言，倘若每收 64 个包才进行一次中断，此时若在发 64 个包之前，操作系统发了一些无效的包，就会导致有效包会因描述符和 buff 不够用而丢包。因此，在不涉及 Bonus 的部分，我没有每 64 次收包才进行一次中断。

2. Bonus 设计

（1） 相比较任务三而言，在 Bonus 中你是否有新增设计，以满足 Bonus 对网卡接收性能的要求？ 若有，请说明你的新增设计和用途。

实验三的设计已经可以进行描述符的重复使用，对于 Bonus 而言，需要考虑的部分则是收包的效率，因此，在接受描述符初始化时，可以置一定的禁止中断位来提升传输效率。

3. 关键函数功能

void check_recv(mac_t *test_mac) ： 在 test2 中进行是否收到 64 个包的判断。
void mac_recv_handle(mac_t *test_mac) ： 在 test3 中进行收包的处理。

参考文献

[1] project_5_guid_book_MIPS
[2] 龙芯 1C300 处理器数据手册_v1.3
[3] 龙芯 2F 处理器手册_v0.1