实验 7-3 报告

第 28 小组
付琳晴

一、实验任务（10%）

实现三种 TLB 相关例外：Refill、Invalid、Modified。

二、实验设计（30%）

（一）线路设计

三种 TLB 例外有异曲同工之妙，产生 TLB 例外的地方一共两处：（1）从 inst_sram 取指令时（2）从 data_sram 取数据或存数据时。因此需要在原来线路的基础上添加部分线路，主要在流水线 fetch 级和 exe 级做修改。报出 TLB 例外的地方依照先前例外线路，仍集中到 exe 级报出。TLB 例外线路设计如图 1.

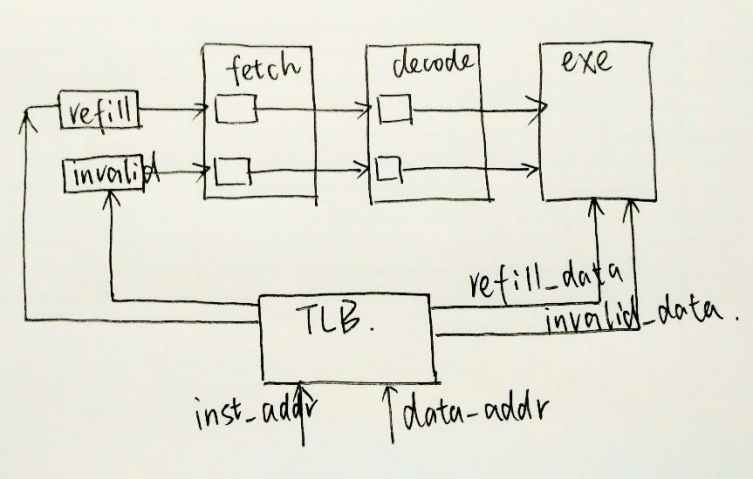

图 1. TLB 例外线路设计

在取指令时发生的例外事实上并不在 fetch 级，而是早于 fetch 级一个周期，因为 Inst_sram_addr 是提前送出的。流水线关系如图 2. 因此需要将 TLB 输出的 refill 和 invalid 信号存一个周期再加入到流水级中。

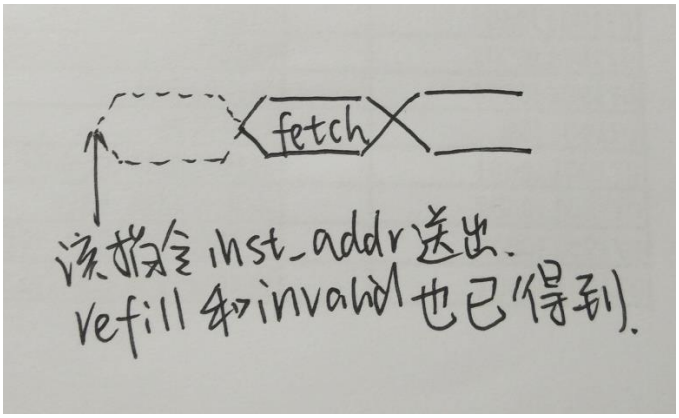

图 2. 取指时 TLB 例外流水线关系

在执行 load 和 store 指令时将在 exe 级送出地址，因此数据的 refill、invalid 和 modified 信号直接在 exe 级产生并使用。

（二）TLB 内部信号

TLB 内部查询信号有两套，因此 refill 信号两条，invalid 信号两条，modified 发生情况在 store 情况下，需要一条信号。因此一共需要添加五条信号线路。TLB refill 只需要根据 hit 信号是否全零得出，相关代码如图 3.

```
assign tlb_refill_inst  =   (vaddr_inst >= 32'h80000000)?1'b0:
                            (hit_inst == 32'd0)?1'b1:1'b0;
```

图 3. TLB refill

TLB invalid 信号需要在 hit 信号有效的情况下查看命中的项 V 域是否是 1，若为 0 则产生 TLB invalid 例外，相关代码如图 4.

```
assign tlb_invalid_inst =   (vaddr_inst >= 32'h80000000)?1'b0:
                            ((hit_inst[0] & ((vaddr_inst[12] & ~CDV1[0][0])|(~vaddr_inst[12] & ~CDV0[0][0])))|
                            (hit_inst[1] & ((vaddr_inst[12] & ~CDV1[1][0])|(~vaddr_inst[12] & ~CDV0[1][0])))|
                            (hit_inst[2] & ((vaddr_inst[12] & ~CDV1[2][0])|(~vaddr_inst[12] & ~CDV0[2][0])))|
                            (hit_inst[3] & ((vaddr_inst[12] & ~CDV1[3][0])|(~vaddr_inst[12] & ~CDV0[3][0])))|
                            (hit_inst[4] & ((vaddr_inst[12] & ~CDV1[4][0])|(~vaddr_inst[12] & ~CDV0[4][0])))|
                            (hit_inst[5] & ((vaddr_inst[12] & ~CDV1[5][0])|(~vaddr_inst[12] & ~CDV0[5][0])))|
                            (hit_inst[6] & ((vaddr_inst[12] & ~CDV1[6][0])|(~vaddr_inst[12] & ~CDV0[6][0])))|
                            (hit_inst[7] & ((vaddr_inst[12] & ~CDV1[7][0])|(~vaddr_inst[12] & ~CDV0[7][0])))|
                            (hit_inst[8] & ((vaddr_inst[12] & ~CDV1[8][0])|(~vaddr_inst[12] & ~CDV0[8][0])))|
                            (hit_inst[9] & ((vaddr_inst[12] & ~CDV1[9][0])|(~vaddr_inst[12] & ~CDV0[9][0])))|
                            (hit_inst[10] & ((vaddr_inst[12] & ~CDV1[10][0])|(~vaddr_inst[12] & ~CDV0[10][0])))|
                            (hit_inst[11] & ((vaddr_inst[12] & ~CDV1[11][0])|(~vaddr_inst[12] & ~CDV0[11][0])))|
                            (hit_inst[12] & ((vaddr_inst[12] & ~CDV1[12][0])|(~vaddr_inst[12] & ~CDV0[12][0])))|
                            (hit_inst[13] & ((vaddr_inst[12] & ~CDV1[13][0])|(~vaddr_inst[12] & ~CDV0[13][0])))|
                            (hit_inst[14] & ((vaddr_inst[12] & ~CDV1[14][0])|(~vaddr_inst[12] & ~CDV0[14][0])))|
                            (hit_inst[15] & ((vaddr_inst[12] & ~CDV1[15][0])|(~vaddr_inst[12] & ~CDV0[15][0])))|
                            (hit_inst[16] & ((vaddr_inst[12] & ~CDV1[16][0])|(~vaddr_inst[12] & ~CDV0[16][0])))|
                            (hit_inst[17] & ((vaddr_inst[12] & ~CDV1[17][0])|(~vaddr_inst[12] & ~CDV0[17][0])))|
                            (hit_inst[18] & ((vaddr_inst[12] & ~CDV1[18][0])|(~vaddr_inst[12] & ~CDV0[18][0])))|
                            (hit_inst[19] & ((vaddr_inst[12] & ~CDV1[19][0])|(~vaddr_inst[12] & ~CDV0[19][0])))|
                            (hit_inst[20] & ((vaddr_inst[12] & ~CDV1[20][0])|(~vaddr_inst[12] & ~CDV0[20][0])))|
                            (hit_inst[21] & ((vaddr_inst[12] & ~CDV1[21][0])|(~vaddr_inst[12] & ~CDV0[21][0])))|
                            (hit_inst[22] & ((vaddr_inst[12] & ~CDV1[22][0])|(~vaddr_inst[12] & ~CDV0[22][0])))|
                            (hit_inst[23] & ((vaddr_inst[12] & ~CDV1[23][0])|(~vaddr_inst[12] & ~CDV0[23][0])))|
                            (hit_inst[24] & ((vaddr_inst[12] & ~CDV1[24][0])|(~vaddr_inst[12] & ~CDV0[24][0])))|
                            (hit_inst[25] & ((vaddr_inst[12] & ~CDV1[25][0])|(~vaddr_inst[12] & ~CDV0[25][0])))|
                            (hit_inst[26] & ((vaddr_inst[12] & ~CDV1[26][0])|(~vaddr_inst[12] & ~CDV0[26][0])))|
                            (hit_inst[27] & ((vaddr_inst[12] & ~CDV1[27][0])|(~vaddr_inst[12] & ~CDV0[27][0])))|
                            (hit_inst[28] & ((vaddr_inst[12] & ~CDV1[28][0])|(~vaddr_inst[12] & ~CDV0[28][0])))|
                            (hit_inst[29] & ((vaddr_inst[12] & ~CDV1[29][0])|(~vaddr_inst[12] & ~CDV0[29][0])))|
                            (hit_inst[30] & ((vaddr_inst[12] & ~CDV1[30][0])|(~vaddr_inst[12] & ~CDV0[30][0])))|
                            (hit_inst[31] & ((vaddr_inst[12] & ~CDV1[31][0])|(~vaddr_inst[12] & ~CDV0[31][0])))
                            ;
```

图 4. TLB invalid

TLB modified 信号需要在 hit 信号和对应项 V 域都是 1 的情况下查看 D 域，若 D 域为 0，则发生 TLB modified 例外，相关代码如图 5. 注意，该例外只会由 store 指令产生，因此 TLB 模块外还要用该信号与上 store 指令判断信号。

```
assign tlb_modified     =   (vaddr_data >= 32'h80000000)?1'b0:
                            (hit_data[0] & ((vaddr_data[12] & CDV1[0][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[0][1:0] == 2'b01)))|
                            (hit_data[1] & ((vaddr_data[12] & CDV1[1][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[1][1:0] == 2'b01)))|
                            (hit_data[2] & ((vaddr_data[12] & CDV1[2][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[2][1:0] == 2'b01)))|
                            (hit_data[3] & ((vaddr_data[12] & CDV1[3][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[3][1:0] == 2'b01)))|
                            (hit_data[4] & ((vaddr_data[12] & CDV1[4][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[4][1:0] == 2'b01)))|
                            (hit_data[5] & ((vaddr_data[12] & CDV1[5][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[5][1:0] == 2'b01)))|
                            (hit_data[6] & ((vaddr_data[12] & CDV1[6][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[6][1:0] == 2'b01)))|
                            (hit_data[7] & ((vaddr_data[12] & CDV1[7][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[7][1:0] == 2'b01)))|
                            (hit_data[8] & ((vaddr_data[12] & CDV1[8][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[8][1:0] == 2'b01)))|
                            (hit_data[9] & ((vaddr_data[12] & CDV1[9][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[9][1:0] == 2'b01)))|
                            (hit_data[10] & ((vaddr_data[12] & CDV1[10][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[10][1:0] == 2'b01)))|
                            (hit_data[11] & ((vaddr_data[12] & CDV1[11][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[11][1:0] == 2'b01)))|
                            (hit_data[12] & ((vaddr_data[12] & CDV1[12][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[12][1:0] == 2'b01)))|
                            (hit_data[13] & ((vaddr_data[12] & CDV1[13][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[13][1:0] == 2'b01)))|
                            (hit_data[14] & ((vaddr_data[12] & CDV1[14][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[14][1:0] == 2'b01)))|
                            (hit_data[15] & ((vaddr_data[12] & CDV1[15][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[15][1:0] == 2'b01)))|
                            (hit_data[16] & ((vaddr_data[12] & CDV1[16][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[16][1:0] == 2'b01)))|
                            (hit_data[17] & ((vaddr_data[12] & CDV1[17][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[17][1:0] == 2'b01)))|
                            (hit_data[18] & ((vaddr_data[12] & CDV1[18][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[18][1:0] == 2'b01)))|
                            (hit_data[19] & ((vaddr_data[12] & CDV1[19][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[19][1:0] == 2'b01)))|
                            (hit_data[20] & ((vaddr_data[12] & CDV1[20][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[20][1:0] == 2'b01)))|
                            (hit_data[21] & ((vaddr_data[12] & CDV1[21][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[21][1:0] == 2'b01)))|
                            (hit_data[22] & ((vaddr_data[12] & CDV1[22][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[22][1:0] == 2'b01)))|
                            (hit_data[23] & ((vaddr_data[12] & CDV1[23][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[23][1:0] == 2'b01)))|
                            (hit_data[24] & ((vaddr_data[12] & CDV1[24][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[24][1:0] == 2'b01)))|
                            (hit_data[25] & ((vaddr_data[12] & CDV1[25][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[25][1:0] == 2'b01)))|
                            (hit_data[26] & ((vaddr_data[12] & CDV1[26][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[26][1:0] == 2'b01)))|
                            (hit_data[27] & ((vaddr_data[12] & CDV1[27][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[27][1:0] == 2'b01)))|
                            (hit_data[28] & ((vaddr_data[12] & CDV1[28][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[28][1:0] == 2'b01)))|
                            (hit_data[29] & ((vaddr_data[12] & CDV1[29][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[29][1:0] == 2'b01)))|
                            (hit_data[30] & ((vaddr_data[12] & CDV1[30][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[30][1:0] == 2'b01)))|
                            (hit_data[31] & ((vaddr_data[12] & CDV1[31][1:0] == 2'b01)|(~vaddr_data[12] & CDV0[31][1:0] == 2'b01)))
                            ;
```

图 5. TLB modified

三、实验过程（60%）

（一）实验流水账

付琳晴：

12.24 日下午：开始写代码。

12.25 日下午+晚上：课上 debug 完成。

12.26 日上午：写实验报告。

（二）错误记录

1、错误 1

（1）错误现象

PC 一直高阻态。

（2）分析定位过程

相比于之前的代码，对于 PC 取指部分只增加了一种 refill 中断导致的赋值情况，说明 refill 控制信号是高阻态，

国科大26209H计算机体系结构

才会导致 PC 取高阻态。查看 refill 信号，用一些 wire 信号 debug 看 refill 的每种赋值情况的值，意识到最初 TLB 里没有初始化，都是高阻态，故而 refill 信号是高阻态，但其实最开始用不到 TLB，因为最开始没有需要虚实地址转化的部分，因此应该添加判断虚地址大小的部分，若虚地址大于 0x8000_0000，就直接将 refill 等信号置 0，从而避免最开始 refill 信号高阻态带来的问题。

但是这样修改过后还是高阻态，那么一定还是某个判断信号是高阻态，导致 refill 信号时高阻态，从而影响 PC。继续看 refill 的各个判断情况，最终发现是用到一个信号的[193:191]位，结果写成了[193:192]，导致判断错误，影响到了继续判断的部分，导致高阻态。

（3）错误原因

Refill 信号在不需要虚实地址转换的时候应该恒等于 0，有个关键信号位数写错。

（4）修正效果

PC 有示数，且程序直接跑过 8、9 测试用例。

2、错误 2

（1）错误现象

测试用例第 10 个，在进行了一次 refill 处理程序之后，应该会触发一次 TLB Invalid，但是我的 CPU 又报了一次 TLB refill.

（2）分析定位过程

最开始 debug 时对于 TLB refill 的产生情况有点混淆了，以为只要得到的物理地址还是错的就应该还触发 TLB refill，因此以为是自己 TLB 里存入的项不对，导致虚实转换得到的物理地址错了，然后就又触发 TLB refill。对着汇编代码验算了很久，发现 TLB 里存入的项是对的，然后就不知道错哪里了，甚至怀疑用的测试用例有问题。后来和室友又交流了一下，意识到错误的地方不在这里，而且第二次查 TLB，查到了对应项算出来物理地址，这就不该发生 TLB refill，而应该因为 V 位是 0 发生 TLB invalid，因此往 TLB 里存入的正确，只是我的 CPU 在报出 refill 还是 invalid 信号的时候有问题。然后查看了这两个信号的赋值，发现居然报了 refill_inst，这个信号是从流水级第一级一直传到 exe 级的，因此肯定源头有问题，于是查看最初从 TLB 中传出来的 refill 信号，最终发现用 reg 往后传的时候赋值情况写错。错误代码如图 6. 错误属于多此一举。这样非一直赋值，会导致即使 TLB 传出的 refill 信号已经变成了 0，但 reg 信号还是会一直保持之前的 1. 应该把 if 的情况去掉，一直把 wire 信号传给 reg 就对了。

国科大UCAS2019计算机体系结构研讨课17-18秋季

```
always @(posedge clk) begin
    if (rst) begin
        tlb_refill_inst_reg <= 1'b0;
    end
    else if (inst_vaddr < 32'h80000000)begin
        tlb_refill_inst_reg <= tlb_refill_inst;
    end

end
```

图 6. 多此一举导致赋值错误

（3）错误原因

refill 信号用 reg 往后传的时候赋值条件写错。

（4）修正效果

保证了 refill 信号正确的沿流水线传下去。

（5）归纳总结

举一反三发现自己 invalid_inst 信号也像这样写错，因此也改过来，测试 10 就跑过了。

3、错误 3

（1）错误现象

连续好几条高位为 0x33333 的指令地址取址，会连续触发 fetch TLB refill。

（2）分析定位过程

回想了一下，之前写中断的代码部分应该考虑过这个问题，一定是采取过避免方法的，之后写的代码一定是忘记加了。查看了一下，发现确实，和之前写的代码相差了一个~cp0_status_exl 信号，这个信号属于关中断，抑制了其他中断发生。

（3）错误原因

忘记关中断，导致连续几条指令触发中断。

（4）修正效果

不会连续触发 fetch TLB refill。

国科大UB6200计算机体系结构研讨课17-18秋季

四、实验总结

（一）组员：付琳晴

感觉这可能是体系结构写的最后一个实验了，之后的那个实验因为自己没有写 axi 接口，可能没法完成了 emmm，所以来絮叨总结一下整个学期。

感觉这是继计算机组成原理之后最认真完成的一门实验课，而且经过这一个学期的打磨，感觉自己对体系结构的理解可能是达到了大学最高峰…大四考研时候可能水平都不如现在了…这学期体系结构的每一个实验几乎都是自己写的（流水线加阻塞的那次让前队友写了），因此对于每一行代码都了解的一清二楚，清楚到后半学期每一次遇到 bug 都有直觉是哪里错的，就会直接跑到认为错的地方查看，改一改就可以解决。这种感觉在第五次中断任务的时候达到顶峰，用自己的 CPU 跑自己写的电子表程序的时候，出错后用仿真看波形。若不是由于底层自己写的，上层也是自己写的，可能根本无法理解为什么会出错，也不会一看波形就瞬间能明白哪里的问题。自己完完全全明白硬件会自动做的操作，可能隐藏的问题，这种感觉真的太好了。（当然，当时发现应该是 CPU 的错误之后，还是毅然决然的改了软件，因为硬件真的好难改哦。）

这学期很神奇的是，虽然看似毫无关系，但是操作系统实验和体系结构实验总是能在同一个时间段，任务达到出奇的互补，比如时间中断，比如这次的 TLB 任务。操作系统那边都会很恰好的涉及到体系结构正在做的底层对应的上层部分，而且都是 MIPS 架构。比如在 TLB 任务书里，说对页表的维护是软件做的，硬件不用管。而操作系统任务书里就会说，TLB 查找是硬件做的，程序员不用管。这种互补真的对我深入理解软硬件对接有很大的帮助，而且在写底层得到的知识，可以用在操作系统实验中，知识完全是相通的。所以尽管两个实验课在同一个学期学很累很崩溃，但我还是建议下一届依旧保持这种课程安排，因为二者学到的知识是恰好互补的，在繁重的实验的压迫下才能达到知识最大的摄入度。

感觉自己现在对于汇编代码有着更深入的理解了，果然想学懂汇编代码不能只学汇编，应该学懂汇编代码下的硬件部分，才能理解汇编代码为什么要这么做。（但是我对 x86 架构还是什么都不懂…）现在回头看这学期的每一个阶段实验，其实确实不难，但是我刚开学的时候第一次实验写不出来还是急的快哭了，是的就是那次最简单的改成同步 ram 的实验。所以啊，人对困难的接受度果然是一点点被打磨的…

非常非常非常感谢邢金璋助教，对我写代码帮助很多，会给我打一大段话不厌其烦地解释一些问题，而且经常即使很晚也能及时回复我的问题。相比于其他理论课的助教，邢金璋助教付出的要多的多也更认真，这样的助教真的很良心了。汪老师也是我见过的工程经验最丰富讲课最实在的老师，介绍 verilog 编程经验和工程上经常出现的错误，还有代码规范，都是很具体实在的问题，对我写代码的整体思路有很大帮助。

非常感谢体系结构实验课，让我得以窥见底层编程工程师们的艰辛，感觉以后不好意思抱怨电脑慢了，能写成一个能正确运行的 CPU 就已经太不容易了。

国科大BS2009H计算机体系结构研讨课17-18秋季