实验 14 报告

学号：2017K8009922027 2017K8009929011
姓名：张磊、郭豪
箱子号：39

一、实验任务（10%）

任务：a) CPU 增加 TLB 相关例外：Refill、Invalid、Modified；

b) 运行专用测试 tlb_func，通过全部 9 项测试；

二、实验设计（40%）

（一）总体设计思路

1. 首先，增加发生 TLB_refill、TLB_invalid、TLB_modified 三种例外的例外跳转入口；

2. 在取指级和执行级，分别增加判断 pc 和访存地址是否位于 maped 空间的判断；

3. 取指级使用 TLB 的 s0 查询端口，执行级使用 TLB 的 s1 查询端口；

4. 如果查询成功，因为 TLB 查询是组合逻辑，所以当拍就可以返回需要转换的 pc 和访存地址；

5. 否则，如果出现 TLB 例外，则将例外信息传回，并修改 excode，随流水级向后传递，到写回级后触发例外处理；

6. CP0 模块中增加触发 TLB 例外时，对 CP0_Badvaddr 和 CP0_Entryhi 寄存器的更新；

（三）重要模块 1 设计：取指级的 PC 转换；

1、工作原理

通过判断 PC 的最高两位是否位 2’b10 得出，PC 是否位于 maped 空间，如果 PC 位于 maped 空间，则需要将其转换为经 TLB 查询转换后的 PC；

如果查询成功，则将其复制给 real_fs_pc，否则，报相关的 TLB 例外，逐级传递；

```
assign fs_tlb_refill = tlb_refill_ex;
assign pc_unmaped = (fs_pc[31:30] == 2'b10);          // Unmaped Space
assign fs_to_tlb_pc = fs_pc;
assign real_fs_pc = (!pc_unmaped & !tlb_refill_ex & !tlb_invalid_ex) ? tlb_to_fs_pc : fs_pc;
```

2、功能描述

将取指级的 Maped 空间的 PC 值转换为 TLB 转换后的 PC；

（四）重要模块 2 设计：执行级的访存地址转换；

1、工作原理

通过判断访存地址的最高两位是否为 2’b10 判断，访存地址是否位于 Maped 空间，若访存地址位于 Maped 地址空间，且当前指令是访存指令，则查询 TLB，进行地址转换；

若查询成功，则使用转换后的访存地址进行访存，否则报相应的 TLB 例外，例外随流水级传递，到写回级统一处理；

```
// lab14
assign addr_unmaped = (data_sram_vaddr[31:30] == 2'b10);            // Unmaped Space
assign addr_maped = !addr_unmaped & (es_mem_we | es_load_op);

assign maped_addr = data_sram_vaddr;
assign data_sram_addr  = (!addr_unmaped & !tlb_to_es_refill_ex) ? tlb_to_es_addr : data_sram_vaddr;
assign data_sram_vaddr = es_alu_result;
```

2、功能描述

完成访存级的访存地址的 TLB 转换；

三、实验过程（50%）

（一）实验流水账

1. 2019 年 12 月 12 日 12:00 - 16:00 完成 rtl 代码编写；
2. 2019 年 12 月 12 日 18:00-2019 年 12 月 13 日 04:00 完成前 2 个 bug 的修改；
3. 2019 年 12 月 13 日 13:00-2019 年 12 月 13 日 18:00 完成后 2 个 bug 的修改；

（二）错误记录

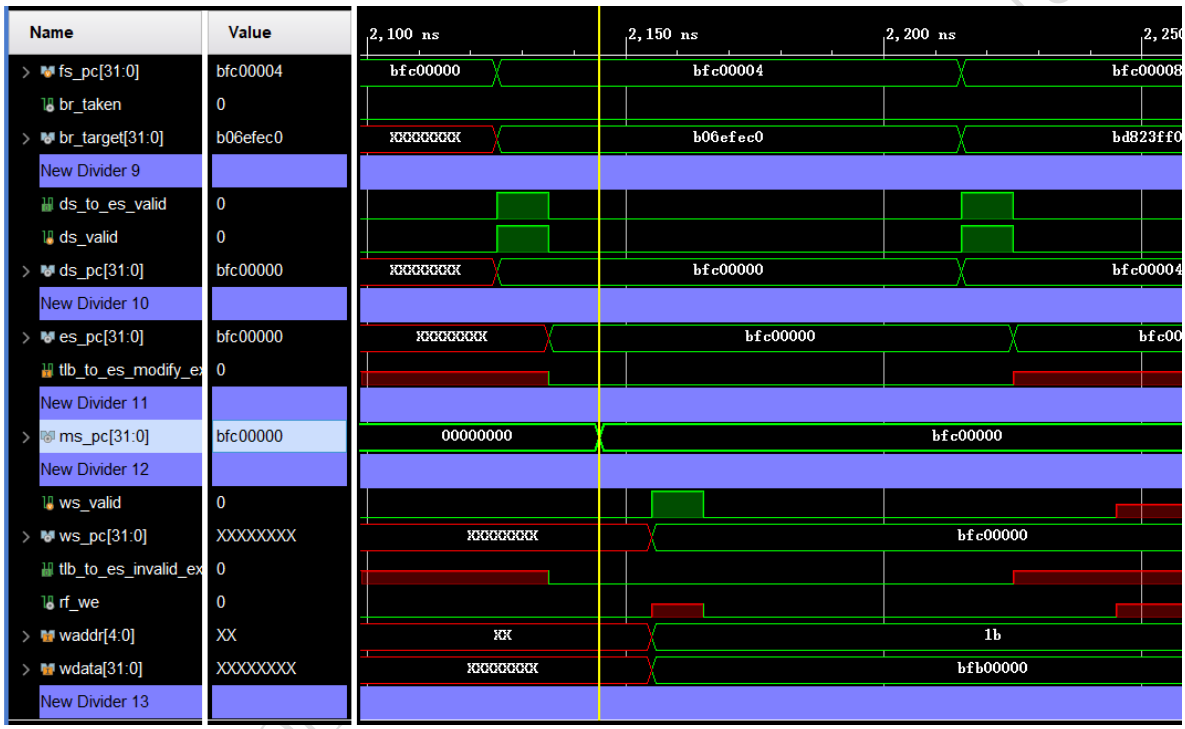
1、错误 1：开始仿真 PC 就卡在 0xbfc00000；

（1）错误现象

```
Test begin!
[  22000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  32000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  42000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  52000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  62000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  72000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  82000 ns] Test is running, debug_wb_pc = 0xbfc00000
[  92000 ns] Test is running, debug_wb_pc = 0xbfc00000
```

（2）分析定位过程

查看波形中的各阶段 PC 值：



发现 PC 卡在了访存级；于是查看访存级控制流水的相关变量的值；

（3）错误原因

发现 es 到 ms 级的 valid 信号为 X，于是向前查找，一直查找到取指级；

```
always @(posedge clk) begin
    if (reset) begin
        ms_valid <= 1'b0;
        es_to_ms_bus_r <= 32'b0;
    end
    else if (ms_allowin) begin
        ms_valid <= es_to_ms_valid;
    end
    else if(eret_flush || flush_pipe || ws_ex_in) begin
        ms_valid <= 1'b0;
    end
    if (es_to_ms_valid && ms_allowin) begin
        es_to_ms_bus_r  <= es_to_ms_bus;
    end
end
```

```
input          ds_tlbwir_in,
input          es_tlbwir_in,
input          ms_tlbwir_in,
input          ws_tlbwir_in
input          flush_pipe,
input [31:0]   tlbwi_flush_pc,
```

找到，原来是 ws 级的 ws_tlbwir_in 赋值错误；

```
assign ws_tlbwir_out = ws_tlbwir_ex;
```

此处应该合取上 ws_valid 信号；

```
assign ws_tlbwir_out = ws_tlbwir_ex & ws_valid;
```

（4）修正效果

修正后，顺利开始仿真；

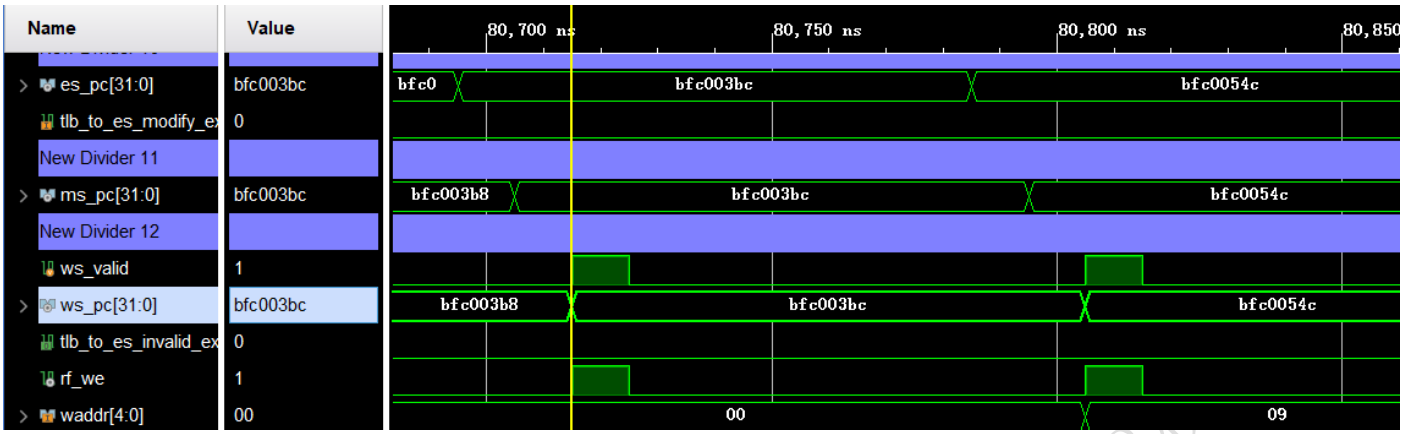
2、错误 2：Entrylo0 和 Entrylo1 错误

（1）错误现象

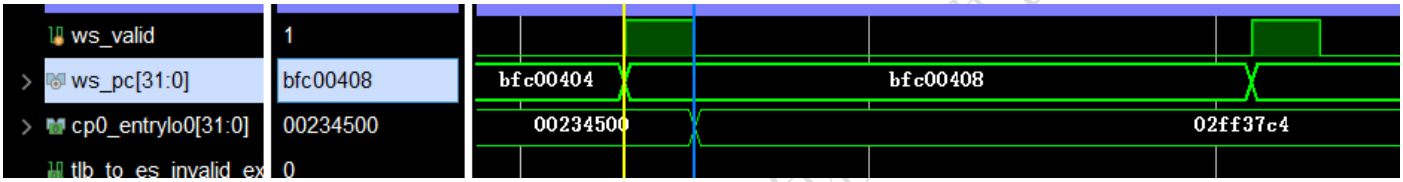
```
        [  52000 ns] Test is running, debug_wb_pc = 0xbfc00a44
        [  62000 ns] Test is running, debug_wb_pc = 0xbfc00b50
----[  64395 ns] Number 8'd05 Functional Test Point PASS!!!
----[  70185 ns] Number 8'd06 Functional Test Point PASS!!!
        [  72000 ns] Test is running, debug_wb_pc = 0xbfc006e4
------------------------------------------------------------
[  81165 ns] Error(  0)!!! Occurred in number 8'd07 Functional Test Point!
------------------------------------------------------------
        [  82000 ns] Test is running, debug_wb_pc = 0xbfc006c4
------------------------------------------------------------
```

（2）分析定位过程

对比波形，查找 test,S 文件中的汇编代码；



波形显示，从 0xbfc003bc 跳转到了 0xbfc0054c，说明触发了例外，所以例外跳转应该没有问题，继续向前查找；发现当 0xbfc00408 对应的指令 mtc0 执行时出现了错误；



查看 bfc00408 对应的指令：

```
/mnt/shared/UCAS-Computer-Architecture/lab14/tlb_func/start.S:162
bfc00400: 3c1a02ff  lui k0,0x2ff
bfc00404: 375a37c2  ori k0,k0,0x37c2
/mnt/shared/UCAS-Computer-Architecture/lab14/tlb_func/start.S:163
bfc00408: 409a1000  mtc0  k0,c0_entrylo
```

此时，写书 entrylo0 的值应该是 0x02ff37c2，但是波形却显示，写入值为 0x02ff37c4；

于是查看 cp0 代码中的 entrylo0 模块；

（3）错误原因

发现在将 entrylo0 组成整体时，v 位与 d 位的位置写反了；（相同的错误出现在 entrylo1 中）

```
assign cp0_entrylo0 =
{
        cp0_entrylo0_zero,
        cp0_entrylo0_pfn,
        cp0_entrylo0_c,
        cp0_entrylo0_v,
        cp0_entrylo0_d,
        cp0_entrylo0_g
};
```

（4）修正效果

修正后，顺利通过该测试点。

3、错误 3：PC 卡在 0xbfc006e4

（1）错误现象

```
============================================================
Test begin!
        [  22000 ns] Test is running, debug_wb_pc = 0xbfc006e4
        [  32000 ns] Test is running, debug_wb_pc = 0xbfc006e8
        [  42000 ns] Test is running, debug_wb_pc = 0xbfc006e8
        [  52000 ns] Test is running, debug_wb_pc = 0xbfc006e8
        [  62000 ns] Test is running, debug_wb_pc = 0xbfc006e8
        [  72000 ns] Test is running, debug_wb_pc = 0xbfc006e8
        [  82000 ns] Test is running, debug_wb_pc = 0xbfc006e8
```

（2）分析定位过程

由于上次仿真都还正常，所以一定是我最近添加的代码出了问题，所以查看我刚才在 es 级添加修改的代码；

（3）错误原因

发现，在执行级发起访存地址转换的前提，还需要判断出执行级当前的指令是 Load 型或者 Store 型；

尝试添加上这个条件，看看能不能通过仿真；

```
assign addr_unmaped = (data_sram_vaddr[31:30] == 2'b10);          // Unmaped Space
assign addr_maped = !addr_unmaped;
```

（4）修正效果

修正后，解决了 PC 卡住的问题，继续开始仿真。

```
// fa814
assign addr_unmaped = (data_sram_vaddr[31:30] == 2'b10);          // Unmaped Space
assign addr_maped = !addr_unmaped & (es_mem_we | es_load_op);
```

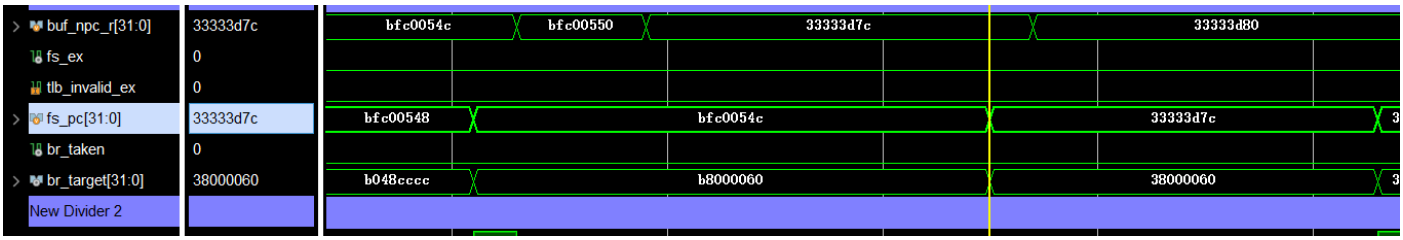
4、错误 4：取指级的 TLB 转换错误

（1）错误现象

```
        [  82000 ns] Test is running, debug_wb_pc = 0xbfc006e4
        [  92000 ns] Test is running, debug_wb_pc = 0xbfc004b0
----[  94005 ns] Number 8'd08 Functional Test Point PASS!!!
        [ 102000 ns] Test is running, debug_wb_pc = 0xbfc00394
        [ 112000 ns] Test is running, debug_wb_pc = 0x33333e9c
------------------------------------------------------------
[ 121995 ns] Error(  0)!!! Occurred in number 8'd09 Functional Test Point!
------------------------------------------------------------
        [ 122000 ns] Test is running, debug_wb_pc = 0xbfc00554
============================================================
Test end!
Fail!!!Total   1 errors!
```

（2）分析定位过程

顺利通过了前 8 个测试点，在第 9 个测试点发生错误；于是查看波形，找到发生错误的 PC，查看对应的 test.S 文件中的汇编指令；



发现，取指级的 PC 并没有转换为经 TLB 转换后的 PC；于是查看取指级的 fs_pc 的生成代码；

（3）错误原因

发现，经 TLB 转换后的 pc 没有替换之前的 pc，所以新增加一个 real_fs_pc，用来在需要查询 TLB 时选择 TLB 转换后的 PC；

```
assign fs_to_tlb_pc = fs_pc;
assign real_fs_pc = (!pc_unmaped & !tlb_refill_ex & !tlb_invalid_ex) ? tlb_to_fs_pc : fs_pc;
```

（4）修正效果

修正后，顺利通过全部 9 项测试。

四、实验总结（可选）

1. 写代码的之前思考的还是不够全面，忽视了很多问题，直到 debug 的时候才发现，下次写代码之前，还是要再仔细思考一下；

2. 大部分的 bug 还是由于自己太过粗心导致的，写代码的时候还是要细心，虽然我也不知道要怎么更加细心，感觉已经很注意了，好难受；

3. 尽量不要熬夜写代码吧，虽然 bug 找不出来很难受，睡不着觉，但还是要早些睡觉，休息好了再开始 debug 效率也会高一些；