实验 14 报告

学号： 2017K8009929008 2017K80099290013

姓名：

箱子号：72

一、实验任务（10%）

掌握 TLB MMU 的知识，实现 CPU 中虚实地址转换和经过 TLB 访存的过程，实现 TLB miss 处理。

二、实验设计（40%）

（一）总体设计思路

1、**地址转换**：访存地址发出前检查，若位于 unmapped 段则直接抹去高 3 位，若位于 mapped 段，用 TLB 返回的物理页号拼接后得到物理地址

2、**TLB 例外判断**：TLB 返回的 s_found， s_v 和 s_d 用于判断 refill，invalid，modify 三种例外

3、**TLB 例外处理**：

* 随例外总线传至 WB 阶段

* 在 WB 阶段把 c0_EntryHi_vpn2 更新为 badvaddr 中相应的位，并控制 IF 段跳入 TLB 例外的处理入口

（二）重要模块设计：TLB 例外判断

1、 部分代码

```
//pre IF
assign tlb_refill  = ~judge_addr0 && ~s0_found && fs_valid;
assign tlb_invalid = ~judge_addr0 && s0_found && ~s0_v && fs_valid;

//EXE
assign tlb_refill_e  = ~judge_addr1 && ~s1_found
                       && es_valid && (es_load_op || es_store_op);

assign tlb_refill    = tlb_refill_d || tlb_refill_e;

assign tlb_invalid_e = ~judge_addr1 && s1_found && ~s1_v
                         && es_valid && (es_load_op || es_store_op);
assign tlb_invalid   = tlb_invalid_d || tlb_invalid_e;

assign tlb_modifies  = ~judge_addr1 && s1_found && s1_v && ~s1_d
                       && es_valid && es_store_op;
```

图 1：TLB 例外判断

```
assign es_excode = id_ex_op?id_excode
                   :(tlb_refill_e&&es_load_op)? 5'h02
                   :(tlb_refill_e&&es_store_op)? 5'h03
                   :(tlb_invalid_e&&es_load_op)? 5'h02
                   :(tlb_invalid_e&&es_store_op)? 5'h03
                   :(tlb_modifies_e&&es_store_op)? 5'h01
                   :(overflow?5'h0c
                   :(data_addr_error?(es_load_op?5'h04:5'h05)
                   :id_excode));
```

图 2：EXE 阶段 es_excode 逻辑

2、 功能描述：

* judge_addr 用于标识访存地址是否位于 unmapped 段

* EXE 阶段先判断自己的 TLB 例外，然后和 pre IF 阶段的例外信息合并后一起往后传

（三）重要模块设计：TLB 例外处理

1 、部分代码：

```
//WB阶段
    else if((tlb_refill || tlb_invalid || tlb_modifies) && c0_status_exl==0)begin
        c0_EntryHi_vpn2 <=badvaddr[31:13];
    end

//pre IF阶段

    assign nextpc = (ws_valid && fs_ex_op && ~ws_tlb_refill) ? 32'hbfc00380 :
                    (ws_valid && fs_ex_op && ws_tlb_refill)  ? 32'hbfc00200 :
                    (ws_valid && (fs_eret||ws_re_fe) )  ? ws_badvaddr[31:0] :
                    br_taken ? br_target : seq_pc;
```

图 3：TLB 例外处理

2、功能描述：

* 若有 tlb 例外，在 WB 阶段把 c0_EntryHi_vpn2 更新为 badvaddr 中相应的位

* 若 WB 传来 tlb refill 例外，控制 pre IF 段跳入 TLB 例外的处理入口

三、实验过程（50%）

（一）实验流水账

周一下午：阅读任务书，更新环境，整理实验思路；

周二到周六：逻辑设计，代码实现，仿真调试，上板运行；

周六晚上：小组交流代码实现和报告内容；

周一下午：写报告。

（二）错误记录

1、错误 1：端口位数写错

错误现象：仿真打印错误：

```
================================================================
Test begin!
        [ 22000 ns] Test is running, debug_wb_pc = 0xbfc0085c
        [ 32000 ns] Test is running, debug_wb_pc = 0xbfc00914
        [ 42000 ns] Test is running, debug_wb_pc = 0xbfc003f8
        [ 52000 ns] Test is running, debug_wb_pc = 0xbfc00af8
================================================================
Test end!
----PASS!!!
```

图 4：仿真报错 1

错误查找过程：

仔细查找新加入的变量的波形，发现其中定义的一个变量应该为 32 位但写成了 1 位导致后续判断出现问题；

修改方法：

正确定义变量 true_data_addr 的位数；

```
wire [31:0]true_data_addr;
assign true_data_addr = judge_addr1? (es_result & 32'h1fffffff) : {pfn_addr1,es_result[11:0]};
```

图 5：修正方法 1

2、错误 2：写入 EntryHi 寄存器的判断条件错误

错误现象：仿真出现指令循环：

```
[ 122000 ns] Test is running, debug_wb_pc = 0xbfc00200
[ 132000 ns] Test is running, debug_wb_pc = 0xbfc00304
[ 142000 ns] Test is running, debug_wb_pc = 0xbfc00350
[ 152000 ns] Test is running, debug_wb_pc = 0xbfc002f0
[ 162000 ns] Test is running, debug_wb_pc = 0xbfc00340
[ 172000 ns] Test is running, debug_wb_pc = 0xbfc00228
[ 182000 ns] Test is running, debug_wb_pc = 0xbfc0032c
[ 192000 ns] Test is running, debug_wb_pc = 0xbfc00214
[ 202000 ns] Test is running, debug_wb_pc = 0xbfc00318
[ 212000 ns] Test is running, debug_wb_pc = 0xbfc00200
[ 222000 ns] Test is running, debug_wb_pc = 0xbfc00304
[ 232000 ns] Test is running, debug_wb_pc = 0xbfc00350
[ 242000 ns] Test is running, debug_wb_pc = 0xbfc002f0
[ 252000 ns] Test is running, debug_wb_pc = 0xbfc00340
```

图 6：仿真报错

错误查找过程:

发现例外处理时，tlbwi 指令写入的数据错误，不断引发 tlb_refill 例外，如下图所示，正确的值为 1999，但 EntryHi 寄存器发生改变，tlbwi 时将 5fe00 写入：

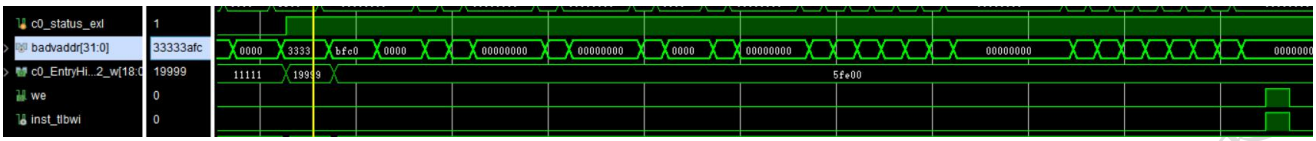

图 7：错误波形

修改方法:

保证只有当之前没有例外的时候（即 c0_status_exl==0）才将 badvaddr 写入 c0_EntryHi_vpn2 寄存器，具体内容如下：

```
else if((tlb_refill || tlb_invalid || tlb_modifies) && c0_status_exl==0)begin
    c0_EntryHi_vpn2 <=badvaddr[31:13];
end
```

图 8：修正方法 2

3、错误 3：上板错误

错误现象：晶体管显示为 09000008

错误查找过程:

通过查看报错信息，发现 c_invalid 变量存在多驱动的错误：

```
always @(posedge clk)begin
    if(reset)begin
        c_refill <= 1'b0;
        c_invalid <= 1'b0;
    end
    else if(tlb_refill == 1'b1)begin
        c_refill <= 1'b1;
    end
    else if(to_fs_valid && fs_allowin)begin
        c_refill <= 1'b0;
    end
end
always @(posedge clk)begin
    if(reset)begin
        c_invalid <= 1'b0;
    end
```

图 9：错误代码 3

修改方法:

保证 reset 的时候只有一处给 c_invalid 赋值。