**实验十四报告**

学号 2017K8009929044

2017K8009929025

姓名 李昊宸 李颖彦

箱子号 33

一、实验任务（10%）

1. 掌握 TLB MMU 的知识。

2. 理解 CPU中地址翻译的机制。

3. 理解 MIPS架构的 TLB miss处理过程

二、实验设计（40%）

（一）总体设计思路

本次实验要求加入CPU增加 TLB相关例外：Refill、Invalid、Modified。TLB是页表的缓存，所以当它查找不命中的时候，需要将内存中的页表项填入其中，这个过程由TLB Refill例外完成。所有的页表项只有在确切需要使用的时候操作系统才会给它分配物理页的空间，也就是说最初填入 TLB的页表项的 PFN、C、D、V都是 0值。随后 TLB查找虽然命中但发现 V位为 0，将触发一个 TLB Invalid例外。当根据虚拟页号在TLB中查找到该项，且对应物理页的V位为1，D位为0，且该访问是 store时，触发一个TLB Modified例外，说明有写行为发生。

（二）重要模块1设计：TLB重填例外

1. 工作原理

当发生下列条件时触发 TLB Refill 例外： 依据虚拟页号在 TLB中未查找到该项。

例外入口：0xbfc00200

控制寄存器 Cause的 ExcCode域：

0x02 (TLBL)：取指或读数据

0x03 (TLBS )：写数据

响应例外时的额外寄存器状态更新：

BadVAddr 记录触发例外的访问内存的虚地址。

EntryHi VPN2域更新为 VA31..13

1. 功能描述

当TLB模块返回的found信号为0，并且访问的地址处于mapped段时，发生TLB miss，也即触发TLB Refill例外。例外发生时，需要与普通例外处理不同的是，需要将BadVAddr修改为触发例外的虚拟地址，并将EntryHi的VPN2域更新为VA31..13，随后跳转到0xbfc00200。

1. 具体代码实现

取指级：

*assign tlb\_miss=!s0\_found&&mapped;*

*assign fs\_ex = (ft\_address\_error|tlb\_miss\_reg|tlb\_invalid\_reg) && fs\_valid && ~wb\_ex; //加wb\_ex信号为了清空流水线*

*assign badvaddr = (ft\_address\_error)? fs\_pc:*

*(tlb\_miss\_reg|tlb\_invalid\_reg)?BAD\_VADDR*

*: 32'h0;*

判断是否发生TLB miss事件，准备好将要写入BadVAddr寄存器的值，并且准备好例外信号。

这里需要与地址错例外进行区分的地方是，地址错例外可以通过fs\_pc的值做判断，判断完成后可以直接拉高例外信号。TLB miss的判断是在pre-IF阶段完成，所以需要将例外信号用寄存器保存下来，等到触发例外的PC更新到fs-pc时再拉高例外信号。对refill例外信号的处理见下。

*always @(posedge clk)begin*

*if(reset)*

*tlb\_miss\_reg<=0;*

*else if(to\_fs\_valid && fs\_allowin)*

*tlb\_miss\_reg<=tlb\_miss;*

*if(reset)*

*BAD\_VADDR<=0;*

*else if(to\_fs\_valid && fs\_allowin)*

*BAD\_VADDR<=true\_npc;*

*End*

tlb\_ex信号是从写回级传到取指级的tlb\_refill信号

*always@(posedge clk)begin*

*if(reset)*

*TLB\_EX<=1'b0;*

*else if(tlb\_ex)*

*TLB\_EX<=1'b1;*

*else if(inst\_addr\_ok)*

*TLB\_EX<=1'b0;*

*End*

发生refill例外时的例外入口地址与其他例外不同，这里需要补充修改。

*assign true\_npc= (WB\_EX&&TLB\_EX)?32'hbfc00200:*

*(WB\_EX&&~TLB\_EX)?32'hbfc00380:*

*// inst\_eret? cp0\_rdata :*

*INST\_ERET? cp0\_rdata :*

*WS\_REFETCH?cp0\_rdata:*

*buf\_valid ?buf\_npc:*

*br\_valid & (j\_reg | ~br\_taken)?buf\_br: //j\_reg为0时存在前递相关*

*nextpc;*

对fs\_pc的修改也需要补充如果在写回级检测到TLB miss例外的返回地址处理

*always @(posedge clk) begin*

*if (reset) begin*

*fs\_pc <= 32'hbfbffffc; //trick: to make nextpc be 0xbfc00000 during reset*

*end*

*else if(wb\_ex&&!tlb\_ex)*

*fs\_pc <= 32'hbfc00380;*

*else if(wb\_ex&&tlb\_ex)*

*fs\_pc <= 32'hbfc00200;*

*else if(inst\_eret | ws\_refetch)*

*fs\_pc <= cp0\_rdata;*

*else if (to\_fs\_valid && fs\_allowin) begin*

*if(tlb\_miss||tlb\_invalid)*

*fs\_pc<=true\_npc;*

*else*

*fs\_pc <= TRUE\_NPC;*

*end*

*end*

最后考虑对于发送到总线上和从总线返回的数据。如果触发TLB miss，即Refill例外，那么这个虚拟地址是无意义的，如果将其发到总线上去会带来意想不到的后果。所以如果pre-IF级检测到TLB miss，就强制将发送到总线上的访问地址修改为0xbfc00380。然后需要将返回的fs\_inst修改成0，这样就完成了取指令的处理。

*assign inst\_sram\_addr = (tlb\_miss||tlb\_invalid)?32'hbfc00380:TRUE\_NPC;*

*assign fs\_inst = (~WB\_EX & ~fs\_ex & ~inst\_eret & ~ws\_refetch &~tlb\_miss\_reg &~tlb\_invalid\_reg)? inst\_sram\_rdata : 32'b0;*

执行级：

*assign tlb\_miss=!s1\_found&&mapped&&load\_store;*

*assign badvaddr = (ex&&(tlb\_miss\_ds||tlb\_invalid\_ds))? fs\_badvaddr*

*:es\_alu\_result;*

执行级的TLB miss处理的判断过程与pre-IF较为类似，在这里就不过多赘述，只说明不同的几点：

1. 取指级使用的是s0通道，执行级使用的是s1通道
2. 执行级对badvaddr值的修改需要考虑是否之前已经触发过TLB例外，如果没有才能修改成当前访问的虚地址
3. 对于执行，不需要做像取指级那样向总线上发送0xbfc00380地址，而是直接将执行级的例外信号拉高，复用先前对例外发生时取消访存请求的数据通路。

（三）重要模块2设计：TLB无效例外

1. 工作原理

当发生下列条件时触发 TLB Invalid例外： 依据虚拟页号在 TLB中查找到该项，但对应物理页的 V位为 0。

例外入口：0xbfc00380

控制寄存器 Cause的 ExcCode域：

0x02 (TLBL)：取指或读数据

0x03 (TLBS )：写数据

响应例外时的额外寄存器状态更新：

BadVAddr 记录触发例外的访问内存的虚地址。

EntryHi VPN2域更新为 VA31..13

1. 功能描述

当TLB模块返回的found信号为1，并且访问的地址处于mapped段，但返回的字段V为0时，发生TLB 触发TLB Invalid例外。例外发生时，需要与普通例外处理不同的是，需要将BadVAddr修改为触发例外的虚拟地址，并将EntryHi的VPN2域更新为VA31..13，随后跳转到0xbfc00380。

1. 具体代码实现

取指级：

*assign tlb\_invalid=s0\_found&&!s0\_v&&mapped;*

*assign fs\_ex = (ft\_address\_error|tlb\_miss\_reg|tlb\_invalid\_reg) && fs\_valid && ~wb\_ex; //加wb\_ex信号为了清空流水线*

*assign badvaddr = (ft\_address\_error)? fs\_pc:*

*(tlb\_miss\_reg|tlb\_invalid\_reg)?BAD\_VADDR*

*: 32'h0;*

判断是否发生TLB Invalid事件，准备好将要写入BadVAddr寄存器的值，并且准备好例外信号。

这里需要与地址错例外进行区分的地方是，地址错例外可以通过fs\_pc的值做判断，判断完成后可以直接拉高例外信号。TLB Invalid的判断是在pre-IF阶段完成，所以需要将例外信号用寄存器保存下来，等到触发例外的PC更新到fs-pc时再拉高例外信号。对Invalid例外信号的处理见下。

*always @(posedge clk)begin*

*if(reset)*

*tlb\_invalid\_reg<=0;*

*else if( to\_fs\_valid && fs\_allowin)*

*tlb\_invalid\_reg<=tlb\_invalid;*

*if(reset)*

*BAD\_VADDR<=0;*

*else if(to\_fs\_valid && fs\_allowin)*

*BAD\_VADDR<=true\_npc;*

*End*

*assign true\_npc= (WB\_EX&&TLB\_EX)?32'hbfc00200:*

*(WB\_EX&&~TLB\_EX)?32'hbfc00380:*

*// inst\_eret? cp0\_rdata :*

*INST\_ERET? cp0\_rdata :*

*WS\_REFETCH?cp0\_rdata:*

*buf\_valid ?buf\_npc:*

*br\_valid & (j\_reg | ~br\_taken)?buf\_br: //j\_reg为0时存在前递相关*

*nextpc;*

对fs\_pc的修改也需要补充如果在写回级检测到TLB invalid例外的返回地址处理。需要补充的是，只有refill例外的入口地址是0xbfc00200，故通过写回级传回的tlb\_ex信号判断写回级的例外是refill还是其他的例外。

*always @(posedge clk) begin*

*if (reset) begin*

*fs\_pc <= 32'hbfbffffc; //trick: to make nextpc be 0xbfc00000 during reset*

*end*

*else if(wb\_ex&&!tlb\_ex)*

*fs\_pc <= 32'hbfc00380;*

*else if(wb\_ex&&tlb\_ex)*

*fs\_pc <= 32'hbfc00200;*

*else if(inst\_eret | ws\_refetch)*

*fs\_pc <= cp0\_rdata;*

*else if (to\_fs\_valid && fs\_allowin) begin*

*if(tlb\_miss||tlb\_invalid)*

*fs\_pc<=true\_npc;*

*else*

*fs\_pc <= TRUE\_NPC;*

*end*

*end*

最后考虑对于发送到总线上和从总线返回的数据。如果触发TLB invalid例外，那么这个虚拟地址是无意义的，如果将其发到总线上去会带来意想不到的后果。所以如果pre-IF级检测到TLB invalid，就强制将发送到总线上的访问地址修改为0xbfc00380。然后需要将返回的fs\_inst修改成0，这样就完成了取指令的处理。

*assign inst\_sram\_addr = (tlb\_miss||tlb\_invalid)?32'hbfc00380:TRUE\_NPC;*

*assign fs\_inst = (~WB\_EX & ~fs\_ex & ~inst\_eret & ~ws\_refetch &~tlb\_miss\_reg &~tlb\_invalid\_reg)? inst\_sram\_rdata : 32'b0;*

执行级：

*assign tlb\_invalid=s1\_found&&!s1\_v&&mapped&&load\_store;*

*assign badvaddr = (ex&&(tlb\_miss\_ds||tlb\_invalid\_ds))? fs\_badvaddr*

*:es\_alu\_result;*

总的来说，invalid例外除了触发条件与refill例外有所不同之外，其他的基本完全一致。故不再赘述。

（四）重要模块3设计：TLB修改例外

1. 工作原理

当发生下列条件时触发 TLB Modified例外：

 依据虚拟页号在TLB中查找到该项，且对应物理页的V位为1，D位为0，且该访问是 store,。

例外入口：0xbfc00380

控制寄存器 Cause的 ExcCode域：

0x01 (Mod)：写数据

响应例外时的额外寄存器状态更新：

BadVAddr 记录触发例外的访问内存的虚地址。

EntryHi VPN2域更新为 VA31..13

1. 功能描述

当TLB模块返回的found信号为1，并且访问的地址处于mapped段，返回的字段V为1，此时的指令为store型指令，但返回的字段d为0时，发生TLB 触发TLB modified例外。例外发生时，需要与普通例外处理不同的是，需要将BadVAddr修改为触发例外的虚拟地址，并将EntryHi的VPN2域更新为VA31..13，随后跳转到0xbfc00380。

1. 具体代码实现

由于modified例外仅会在store指令时触发，故本例外的修改不涉及取指模块。

执行级：

*assign tlb\_modify=s1\_found&&s1\_v&&!s1\_d&&mapped&&store;*

*assign es\_ex = (ex || overflow || load\_address\_error || store\_address\_error||tlb\_miss||tlb\_invalid||tlb\_modify);*

*assign s1\_vpn2 = (~tlbp)?es\_alu\_result[31:13]:c0\_entryhi[31:13];*

*assign s1\_odd\_page = (~tlbp)?es\_alu\_result[12]:0;*

*assign true\_addr = (es\_alu\_result[31:28] < 4'h8 | es\_alu\_result[31:28] >= 4'hc)? {s1\_pfn,{es\_alu\_result[11:0]}}*

*:es\_alu\_result;*

*assign index = s1\_index;*

*assign mapped=(es\_alu\_result[31:28] < 4'h8 | es\_alu\_result[31:28] >= 4'hc);*

*assign data\_sram\_addr  = true\_addr;*

Modified例外可以复用之前两个例外的数据通路，故不再赘述。

补充一下执行级的其他信号：

*assign ES\_excode = (ex & (es\_excode == 5'h00))? 5'h00*

*:(ex & (es\_excode == 5'h02))? 5'h02*

*:tlb\_modify? 5'h01*

*:((tlb\_miss||tlb\_invalid)&&es\_gr\_we)?5'h02*

*:((tlb\_miss||tlb\_invalid)&&!es\_gr\_we)?5'h03*

*: (overflow)? 5'h0c*

*: (load\_address\_error)? 5'h04*

*: (store\_address\_error)? 5'h05*

*: es\_excode;*

三、实验过程（50%）

（一）实验流水账

2019/12/13 22:00 – 24:00 例外添加

2019/12/14 8:00 – 12:00 debug

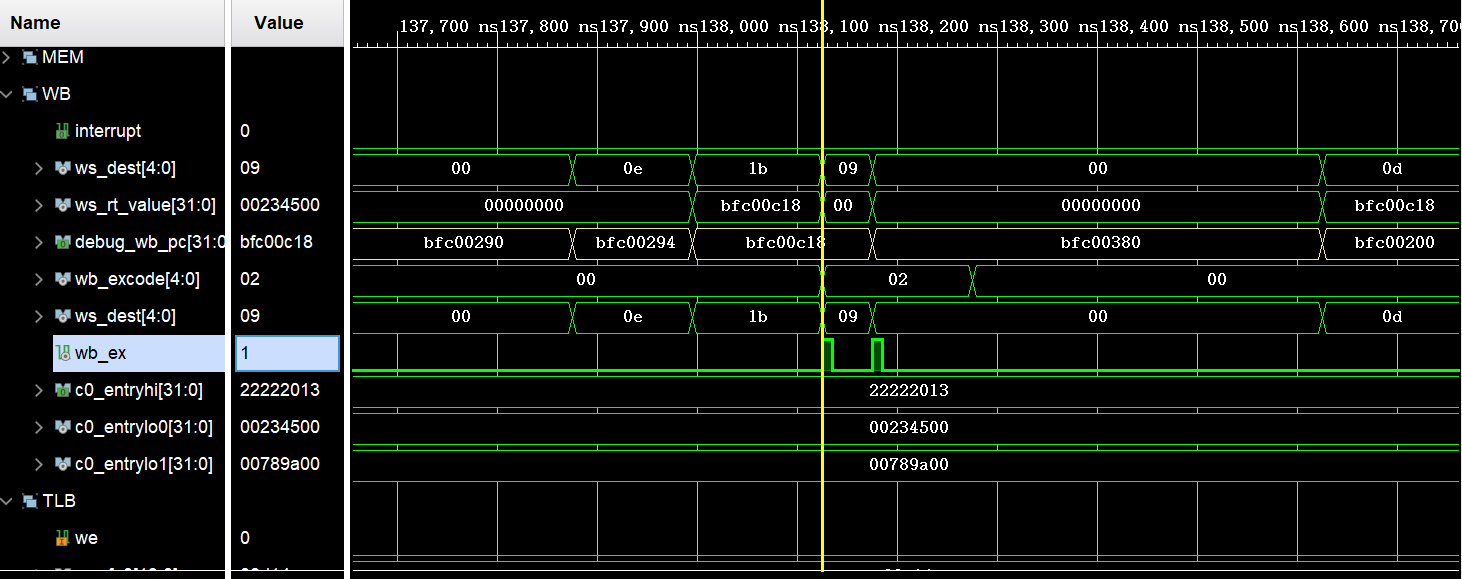
2019/12/15 6:00 – 10:00 书写实验报告

（二）错误记录

1、错误1：pc值异常

（1）错误现象

0xbfc00c18为load指令，触发了TLB例外，但是没有直接跳转到bfc00200正确的入口地址，而是先跳转到了bfc00380。



（2）分析定位过程

出现这样的问题，应该是fs\_pc的赋值逻辑出现了优先级覆盖的情况，或者是新加入的对bfc00200地址的跳转绑定的寄存器晚一拍更新了。

（3）错误原因

理论上应该需要修改fs\_pc和true\_npc的赋值逻辑，使当检测到写回级是TLB refill例外时，fs\_pc和true\_npc都修改为0xbfc00200。但是出现了只修改了true\_npc为0xbfc00200，但是fs\_pc还是沿用了之前的例外地址入口的错误。

（4）修正效果

*always @(posedge clk) begin*

*if (reset) begin*

*fs\_pc <= 32'hbfbffffc; //trick: to make nextpc be 0xbfc00000 during reset*

*end*

*else if(wb\_ex&&!tlb\_ex)*

*fs\_pc <= 32'hbfc00380;*

*else if(wb\_ex&&tlb\_ex)* 增加了这一个分支

*fs\_pc <= 32'hbfc00200;*

*else if(inst\_eret | ws\_refetch)*

*fs\_pc <= cp0\_rdata;*

*else if (to\_fs\_valid && fs\_allowin) begin //fs\_pc <= true\_npc;*

*//if(tlb\_miss\_reg||tlb\_invalid\_reg)*

*if(tlb\_miss||tlb\_invalid)*

*fs\_pc<=true\_npc;*

*else*

*fs\_pc <= TRUE\_NPC;*

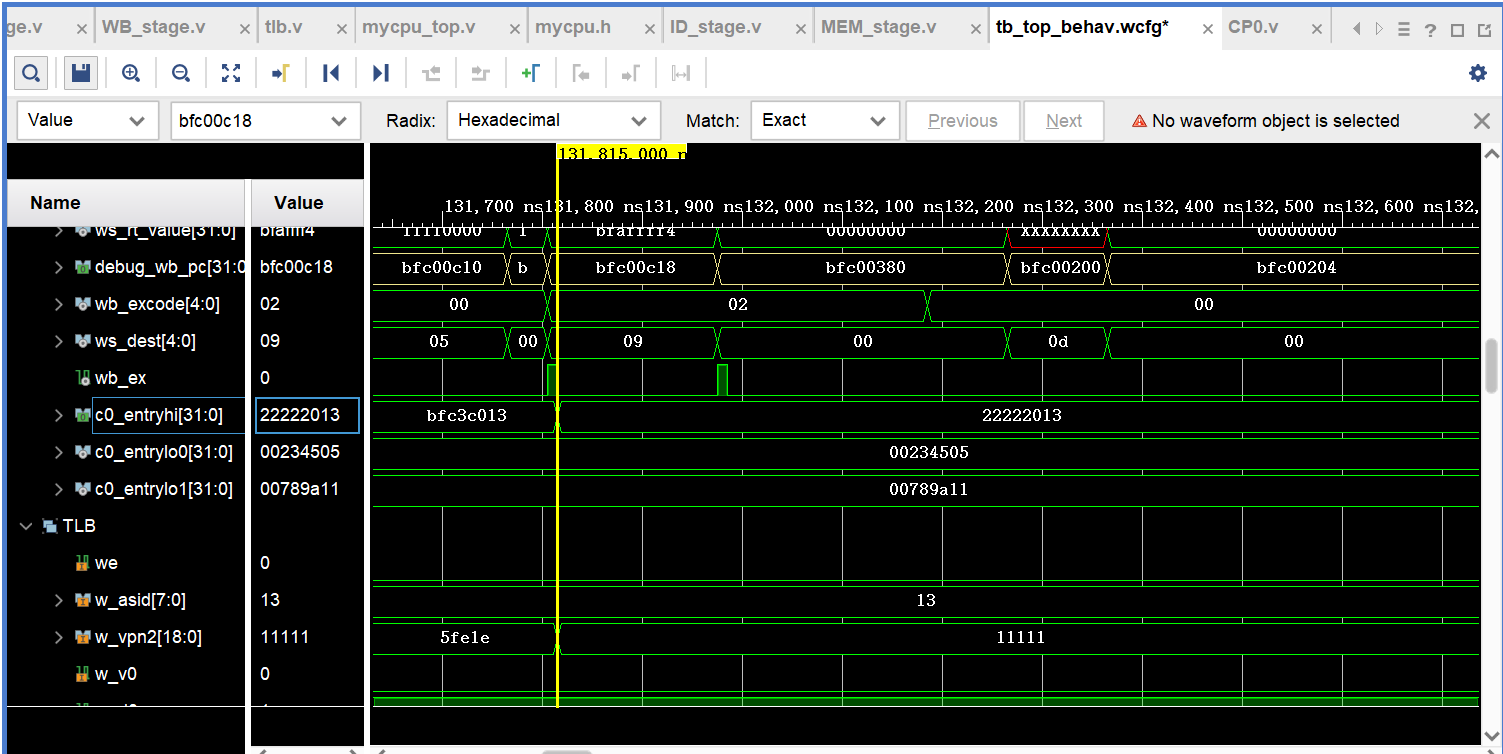
*end*

*end*

2、错误2： VPN2值异常

（1）错误现象

正确的c0\_entryhi的vpn2位应该为0x1111，但是波形上的结果是0x2222。



（2）分析定位过程

在二进制下，TLB中的VPN2是001000100010001，entryhi中的VPN2项是010001000100010，不难发现二者相差一位的偏移量。所以怀疑是修改c0\_entryhi\_vpn2时出现了错位现象。

（3）错误原因

查看CP0寄存器，发现了这样的赋值语句：

*c0\_entryhi\_vpn2<= wb\_badvaddr[31:12];*

这个问题导致的问题是，vpn2有19位，但是wb\_badvaddr[31:12]有20位，这样以来赋值语句就变成了：

*c0\_entryhi\_vpn2<= wb\_badvaddr[30:12];*

于是出现了一位的偏差。

（4）修正效果

将赋值语句修改为：*c0\_entryhi\_vpn2<= wb\_badvaddr[31:13];*

修改之后数据正常。