**实验14报告**

学号: 2018K8009929043 2018K8009929035

姓名: 曾冕 张翔雨

箱子号： 06

一、实验任务（10%）

本实验主要在lab13的基础上将TLB模块集成到lab12完成的 AXI总线的CPU中。主要涉及CP0寄存器的更新以及相关流水级的处理

二、实验设计（40%）

（一）总体设计思路

本次实验的原理并不复杂，把实验所需要的工作分为两部分，一部分是将现有的tlb模块接入之前完成的CPU中。第二部分是将tlb指令添加，这个部分耗费了我较长的时间去进行设计，tlbp指令的设计比较简单，直接阻塞即可，对于tlbr和tlbwi指令，并没有采取讲义上设计的重取方式，而是为了尽可能的复用例外处理的逻辑，将写回级报出重取信号的指令定为tlbwi和tlbr指令本身而不是他们的下一条指令（这两条指令本身一拍就可以执行完成），在这种情况下，只需要在译码级发现当前指令是tlbwi或者tlbr时向取指级传递一个重取信号，if阶段用寄存器存储下此时的nextpc，当wb阶段报出tlb指令的刷新信号时，复用例外处理的bus和bus有效信号，重取之前存储的pc即可完成重取。

本次实验将tlb块作为与五个流水阶段并列的板块，在sram\_cpu顶层文件中实现例化。其次在cp0寄存器中完成了新的cp0寄存器的设计。Tlb与cpu的交互部分如下：s0查找端口用于指令的地址映射，s1查找端口用于数据的地址映射，主要在IF，EXE段进行相关更新。Cp0寄存器的写入和读取功能在WB阶段实现。具体分功能描述如下：

1. IF模块：利用s0查找端口，输出虚地址信息，接受tlb返回的查找信息，响应来自WB模块的tlb\_reflush信号，同时保存指令重取情况出现的pc。
2. EXE模块：利用s1查找端口，输出虚地址信息，接受tlb返回的查找信息，同时接受entryhi寄存器的信息以实现查找和tlbp指令复用s1查找端口的设计要求。
3. WB&CP0模块：cp0模块中增添了entryhi，entrylo0，entrylo1，index等和tlb项有关的寄存器，在WB模块中完成了例化，同时在tlbwi和tlbr指令到达时发出tlb\_reflush信号
4. 在ID阶段利用br\_bus传递refetch信号，当重取信号到达时保存当时的nextpc，以保证下一拍取得的是正确的PC，还有一些流水向下的数据，模块中均有修改，见后面分模块详细设计部分。

CPU总体设计图如下图所示：

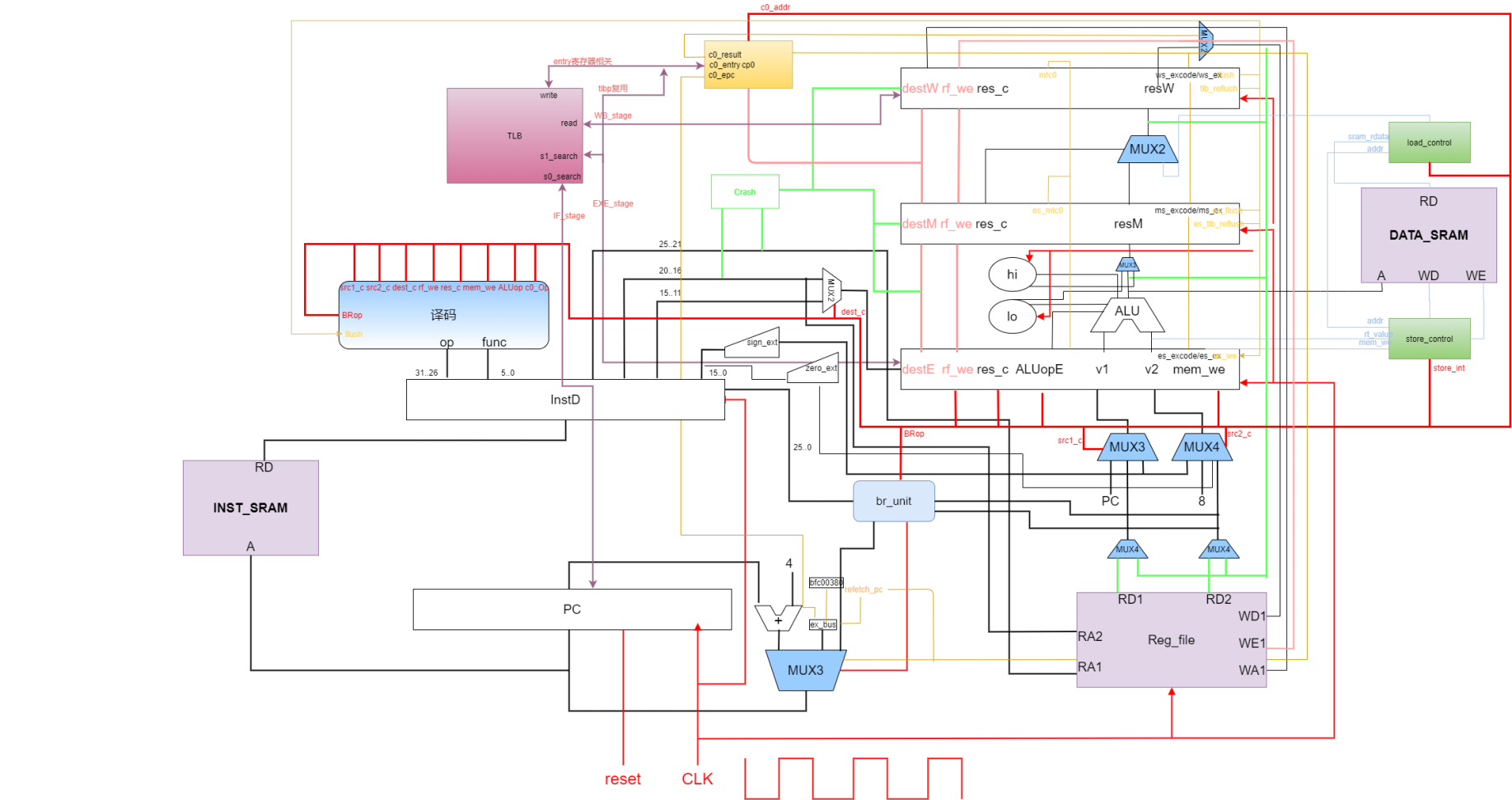


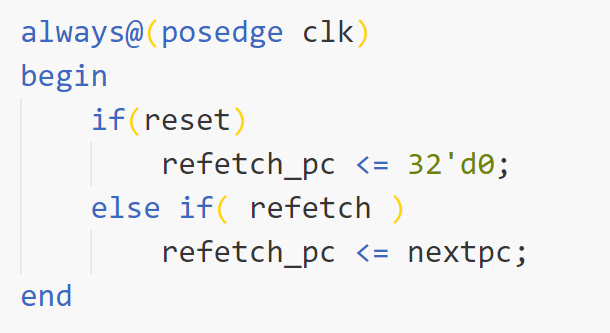
图01: 整体CPU设计

（二）重要模块1设计：IF模块（描述lab14更新）

1. 工作原理（内容描述）  
    利用TLB表项的s0查找端口作为指令的虚实地址查找。 同时根据其余模块来的信号进行流水线刷新/重新取指令等。
2. 接口定义（仅展示lab14更新）

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Tlb\_reflush | IN | 1 | TLBWI和TLBR的冲突消除信号 |
| S0\_vpn2 | OUT | 19 | 输入虚拟地址 |
| S0\_odd\_page | OUT | 1 | 是否为奇数页 |
| S0\_found | IN | 1 | 输出是否找到 |
| S0\_index | IN | 4 | 命中时TLB项数 |
| S0\_pfn | IN | 20 | 物理页框号 |
| S0\_c | IN | 3 | Cache属性 |
| S0\_d | IN | 1 | Dirty位 |
| S0\_v | IN | 1 | Valid(有效)位 |
| Br\_bus | IN | 35 | 添加了refetch信号的br总线 |

1. 功能描述  
   增添了refetch\_pc寄存器，当出现需要refetch的情况下，利用该寄存器暂存nextpc的值，如下图02所示：

图02: 时序refetch\_pc的设计

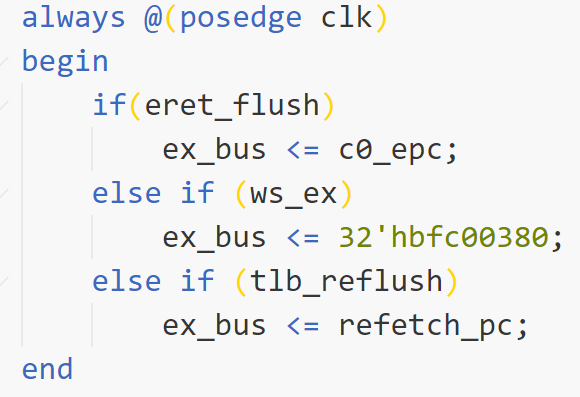
当tlbwi或者tlbr指令到达写回级时刷新流水线的情况下，例外地址寄存器会选用refetch\_pc作为下一拍跳转地址，如下图03所示：

图03: 新增例外地址寄存器的逻辑

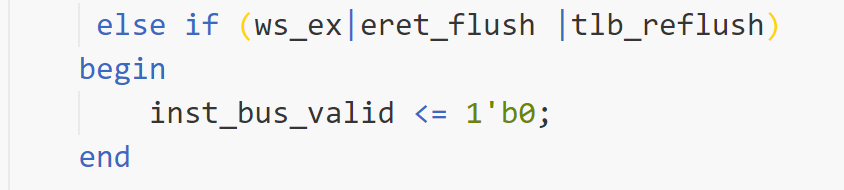
 对于tlb\_reflush的响应处理与之前的ws级例外和eret\_flush信号同级别处理，当出现即无效化下一拍的取值，由于涉及改动较多，这里仅贴出一个always块中的代码设计，表明如何进行无效化的：

图04: 刷新掉整个流水的处理示例

相同的处理同样使用于fs\_valid, request\_control, cancel， ex\_eret, br\_bus\_valid等一切和流水线控制有关的寄存器。

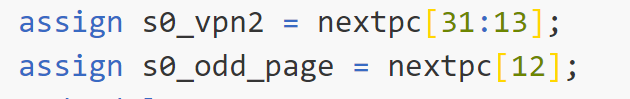
 有关指令虚实映射发出的部分较为简单，就是使用nextpc作为地址查询，这部分设计将在Lab15中更改，以使得CPU能正确进行虚实地址转换，如下图05所示:

图05: 取值阶段指令虚地址转换发出信号

（三）重要模块2设计：ID模块

1. 工作原理

（lab14新增）更新br\_bus，多了一位信号用于判断是否可能发生tlb冲突。向下一级流水的信号增添了tlb的三个指令

1. 接口定义（lab14更新）

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Br\_bus | OUT | 35 | 添加了refetch信号的br总线 |
| ds\_to\_es\_bus | OUT | 213 | 添加了3个tlb指令的流水总线 |

1. 功能描述

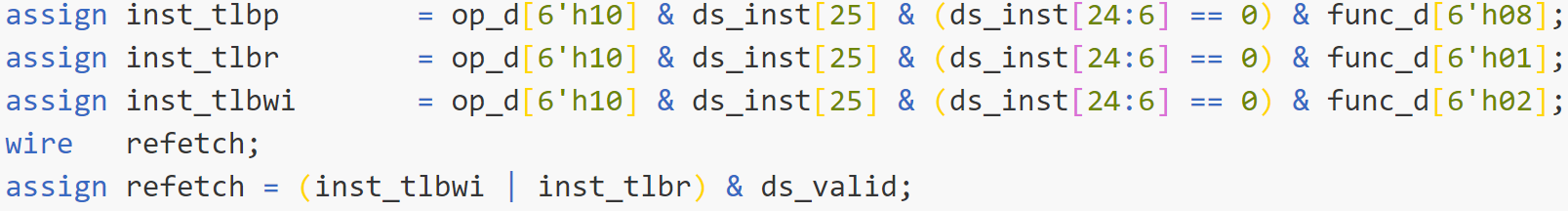
这部分设计较为简单，主要是巧妙的利用了br\_bus来传输一位判断信号，减小了设计开销，译码和refetch信号如下所示：

图06:ID阶段主要更新

部其余例行更新的inst\_valid，gr\_we信号等在此略去（注意三个tlb指令都不需要修改寄存器，所以gr\_we需要额外取非），包括需要将新指令添加至保留指令中。

（四）重要模块3设计：EXE&MEM模块（仅描述lab14更新）

1. 工作原理

Lab14中，exe 完成数据部分的虚实转换，同时该阶段执行tlbp操作。以及接受前面两级流水线传递的mtc0有效信号和reflush信号等。

MEM模块仅多添加了ms\_mtc0和ms\_tlb\_reflesh两个信号，较为简单，所以并在一起描述（也不在接口中重复叙述了）

1. 接口定义（略去了简单的总线流水位宽信号改变）

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| S1\_vpn2 | IN | 19 | 输入虚拟地址 |
| S1\_odd\_page | IN | 1 | 是否为奇数页 |
| S1\_asid | IN | 8 | 进程地址空间标志位 |
| S1\_found | OUT | 1 | 输出是否找到 |
| S1\_index | OUT | 4 | 命中时TLB项数 |
| S1\_pfn | OUT | 20 | 物理页框号 |
| S1\_c | OUT | 3 | Cache属性 |
| S1\_d | OUT | 1 | Dirty位 |
| S1\_v | OUT | 1 | Valid(有效)位 |
| S1\_vpn2 | IN | 19 | 输入虚拟地址 |
| Es\_tlbp | OUT | 1 | TLBP使能信号 |
| Ms\_mtc0 | IN | 1 | 上次流水ms级 |
| Ws\_mtc0 | IN | 1 | 上次流水ws级mtc0信号 |
| Ms\_tlb\_reflesh | IN | 1 | 上次流水ms级tlb标志刷新信号 |
| Tlb\_reflesh | IN | 1 | 上次流水ws级mtc0信号 |

3、功能描述

首先是es级的we信号需要在满足下两级流水的tlb\_reflesh信号没报出的情况，这里也是相当于将tlb指令对流水级的刷新并入了例外处理中，即exe阶段能够写入内存的条件必须是mem和wb阶段不发生tlb\_reflush。

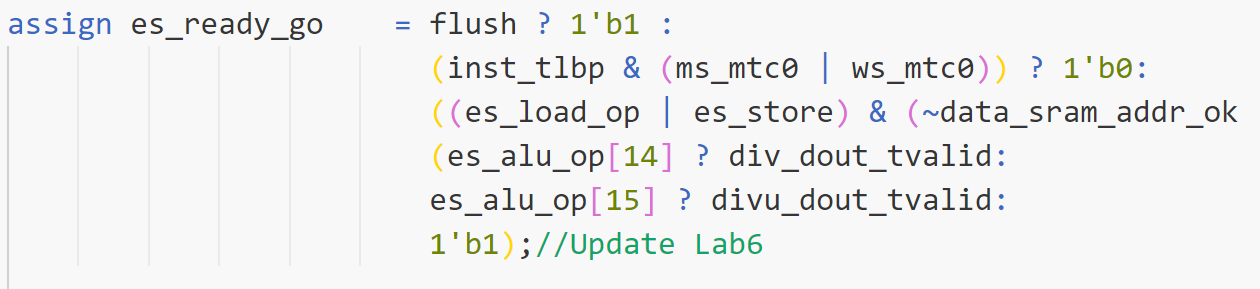
其次es\_ready\_go信号需要添加额外的阻塞判断，保证mtc0写入entryhi寄存器的情况和tlbp同时发生的情况下被阻塞在es阶段不向下一阶段流水，设计代码如下图所示：

图07:es\_ready\_go信号的阻塞逻辑

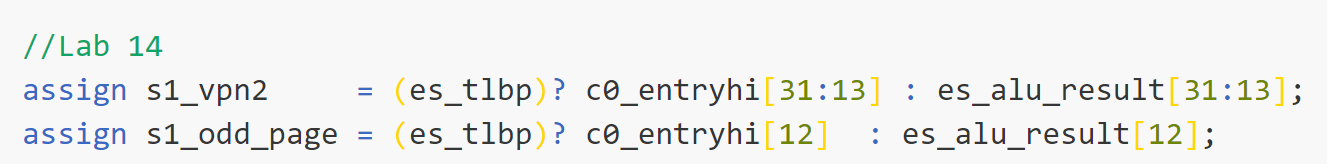
最后是数据部分的查找虚实地址转换，如下图所示：

图08:数据利用s1端口进行虚实地址转换

这里当出现tlbp指令（且本级流水有效）的情况下会复用查找端口，但是利用的是c0\_entryhi寄存器的值

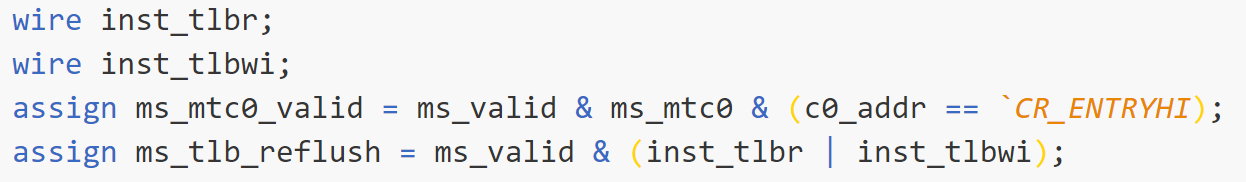
 Mem级涉及的改动很少，主要就是输出两个信号便于EXE级的阻塞，同时wb阶段也会输出类似的两个信号，逻辑如下图所示：

图09:MEM级需要向外传递的两个信号

（五）重要模块4设计：cp0模块（仅描述lab14更新）

1. 工作原理

Lab14中对 cp0模块的改动较大，且和tlb的实现有较大关联，所以单独列出作为一个重要模块介绍。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Tlbp | IN | 4 | TLBP指令 |
| Tlbp\_found | IN | 19 | 数据寻找到的标志（s1） |
| tlbr | IN | 8 | Tlbr指令 |
| r\_g | IN | 1 | 读全局标志位 |
| r\_index | IN | 4 | 读TLB项 |
| r\_vpn2 | IN | 19 | 读虚拟地址 |
| r\_asid | IN | 8 | 读进程号 |
| r\_g | IN | 1 | 读全局标志位 |
| r\_c0 | IN | 3 | 偶数Cache属性 |
| r\_d0 | IN | 1 | 偶数Dirty位 |
| r\_v0 | IN | 1 | 偶数Valid(有效)位 |
| r\_c1 | IN | 3 | 奇数Cache属性 |
| r\_d1 | IN | 1 | 奇数Dirty位 |
| r\_v1 | IN | 1 | 奇数Valid(有效)位 |
| r\_pfn0 | IN | 20 | 奇数页虚拟地址 |
| r\_pfn1 | IN | 20 | 偶数页虚拟地址 |
| C0\_entryhi | OUT | 32 | ENTRYhi寄存器内容 |
| C0\_entrylo0 | OUT | 32 | Lo0寄存器内容 |
| C0\_entrylo1 | OUT | 32 | Lo1寄存器内容 |
| C0\_index | OUT | 32 | Index寄存器内容 |

3、功能描述

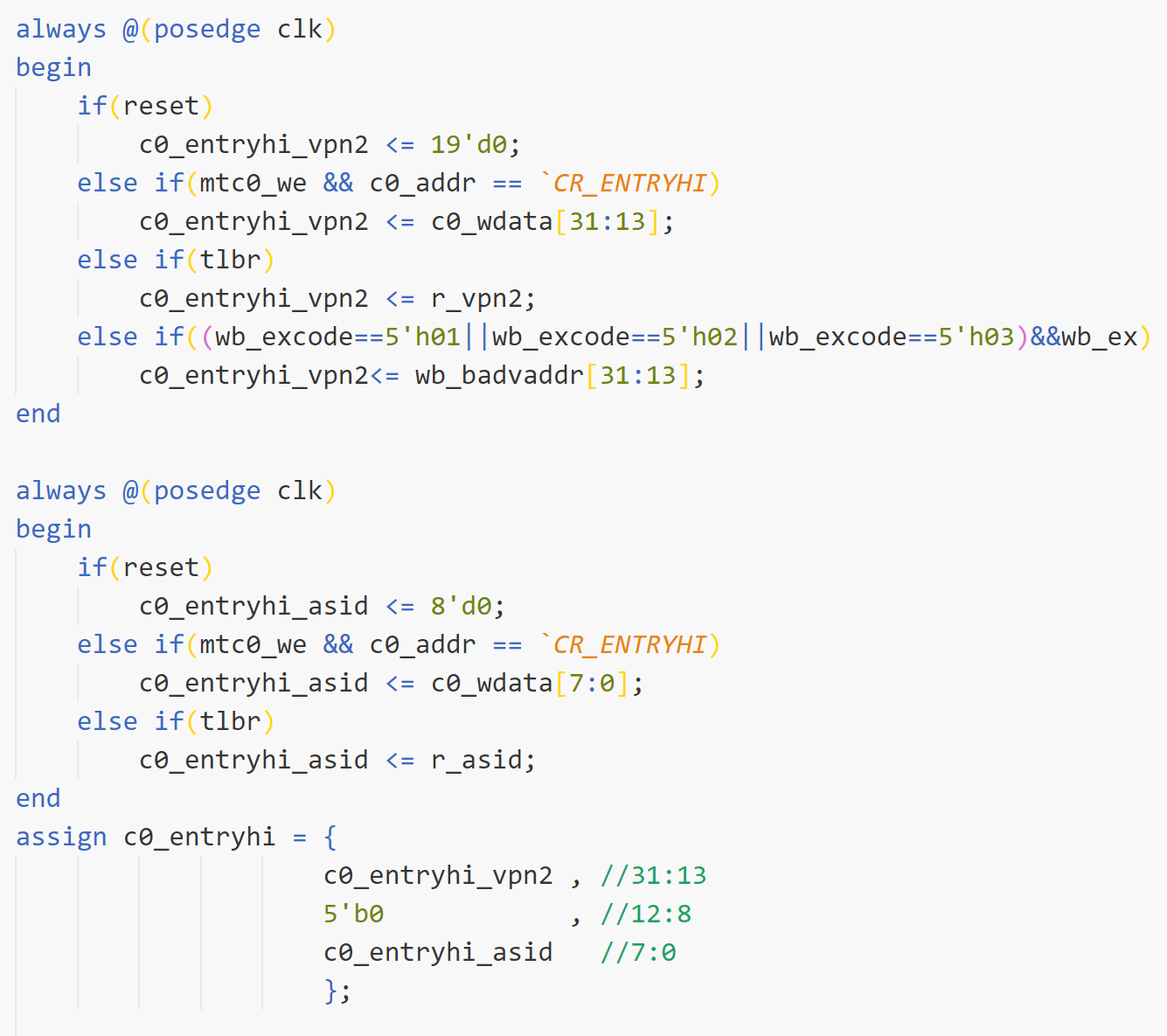
首先是extryhi寄存器的处理,主要分为两个域，然后拼接后处理：

图10:entry\_hi寄存器

对于vpn域，当指令为mtc0的时候写入wdata的31~13位，当为读tlb指令的时候将对应的虚拟地址写入，当为wb阶段发生例外的时候将对应例外虚地址写入。

对于asid域，逻辑更简单，不需要考虑发生例外写入例外虚地址情况，只是当为tlbr指令的时候写入的是r\_asid的内容。

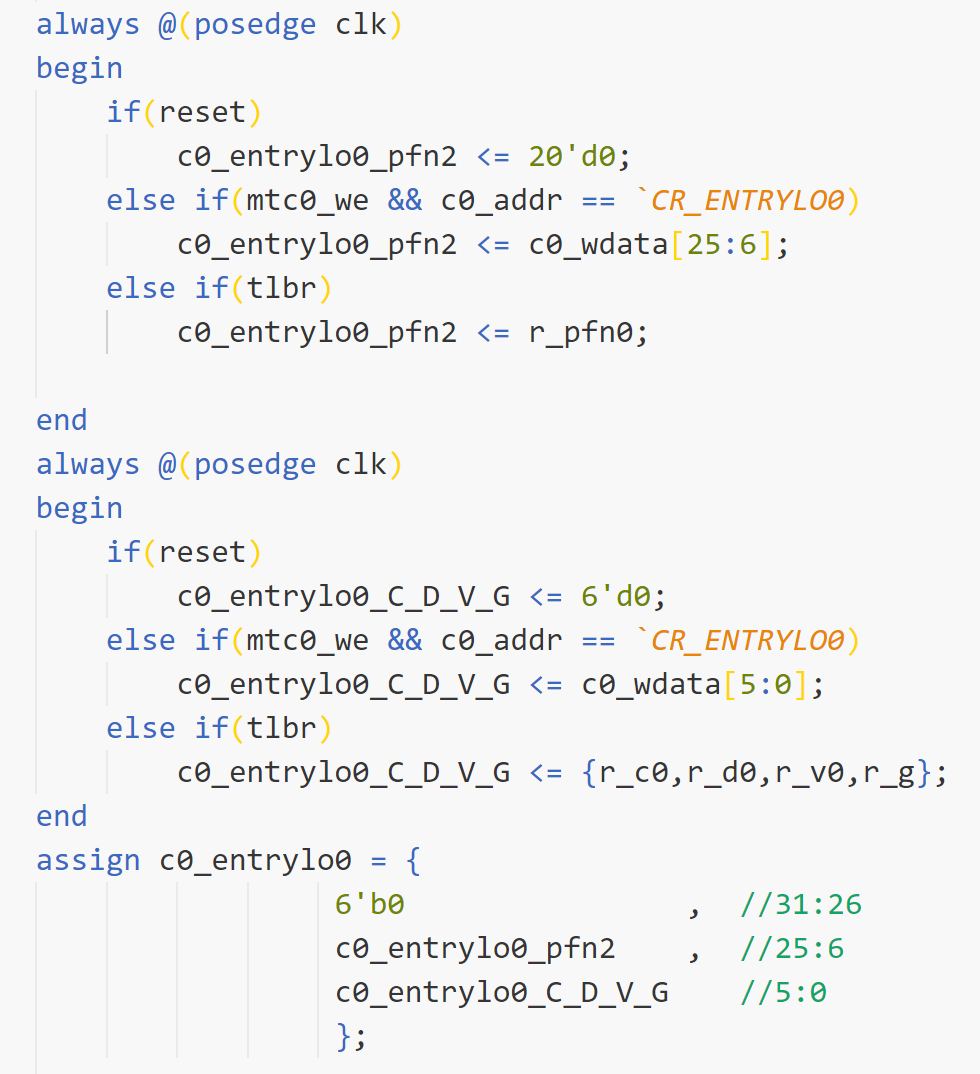
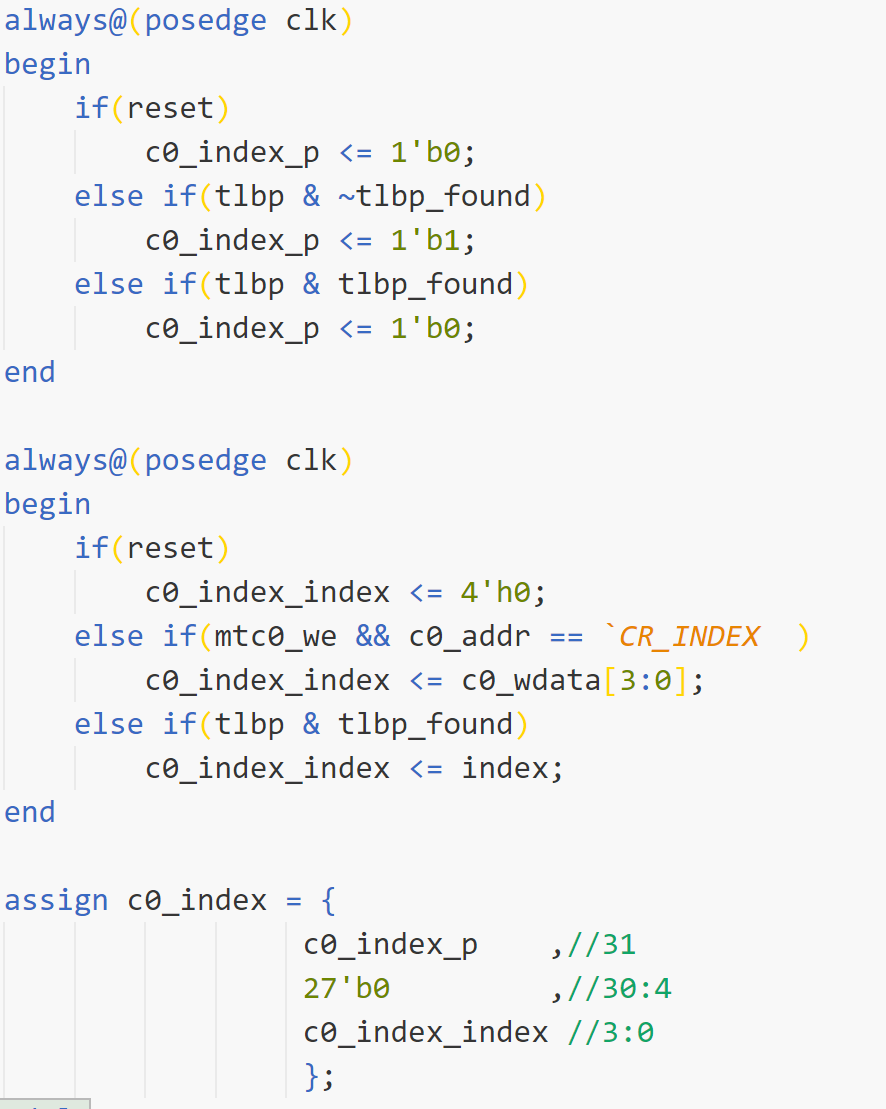
其次是entrylo0和entrylo1两个寄存器，由于除了奇偶以外都相同，这里仅展示lo0的涉及代码，如下图所示：

图11:entry\_lo0寄存器

基本设计思路相同，当为mtc0且写寄存器号匹配的时候写入wdata，tlbr的时候利用tlb读出信息更新寄存器，再将两个域拼接起来。  
 最后是index寄存器，如下图12所示：

（节省页面空间直接在图上面进行叙述）按照设计要求，当数据找到的情况下会将index的高位p位置0，否则会置1，剩余域的处理方式同上两个寄存器，mtc0写，tlbr的时候进行更新，最后域拼接。



（六）重要模块5设计：WB模块（仅描述lab14更新）

1. 工作原理

Lab14中WB主要完成了例化新cp0，完成tlbp和cp0的更新，向外发出tlb标志刷新信号等功能

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| r\_g | IN | 1 | 读全局标志位 |
| r\_index | IN | 4 | 读TLB项 |
| r\_vpn2 | IN | 19 | 读虚拟地址 |
| r\_asid | IN | 8 | 读进程号 |
| r\_g | IN | 1 | 读全局标志位 |
| r\_c0 | IN | 3 | 偶数Cache属性 |
| r\_d0 | IN | 1 | 偶数Dirty位 |
| r\_v0 | IN | 1 | 偶数Valid(有效)位 |
| r\_c1 | IN | 3 | 奇数Cache属性 |
| r\_d1 | IN | 1 | 奇数Dirty位 |
| r\_v1 | IN | 1 | 奇数Valid(有效)位 |
| r\_pfn0 | IN | 20 | 奇数页虚拟地址 |
| r\_pfn1 | IN | 20 | 偶数页虚拟地址 |
| Es\_tlbp | IN | 1 | Tlbp使能信号（建立在exe阶段流水有效情况下） |
| Tlbp\_found | IN | 1 | 实际为s1\_found信号 |
| C0\_entryhi | OUT | 32 | ENTRYhi寄存器内容 |
| C0\_entrylo0 | OUT | 32 | Lo0寄存器内容 |
| C0\_entrylo1 | OUT | 32 | Lo1寄存器内容 |
| C0\_index | OUT | 32 | Index寄存器内容 |
| Tlb\_reflush | OUT | 1 | 标志刷新信号 |
| Ws\_mtc0 | OUT | 1 | 传至exe阻塞相关信号 |
| Inst\_tlbwi | OUT | 1 | 传至tlb的we信号 |

3、功能描述

向外传递的flush信号需要加入是否为标志刷新的判断，代码略去。

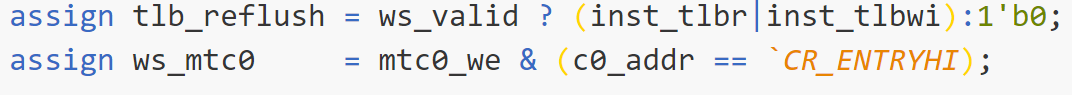
更新的两个向外传递的信号逻辑如下图所示：

图13:向外传递信号

其中mtc0冲突只会发生在是对entryhi寄存器修改的情况，所以需要额外判断。

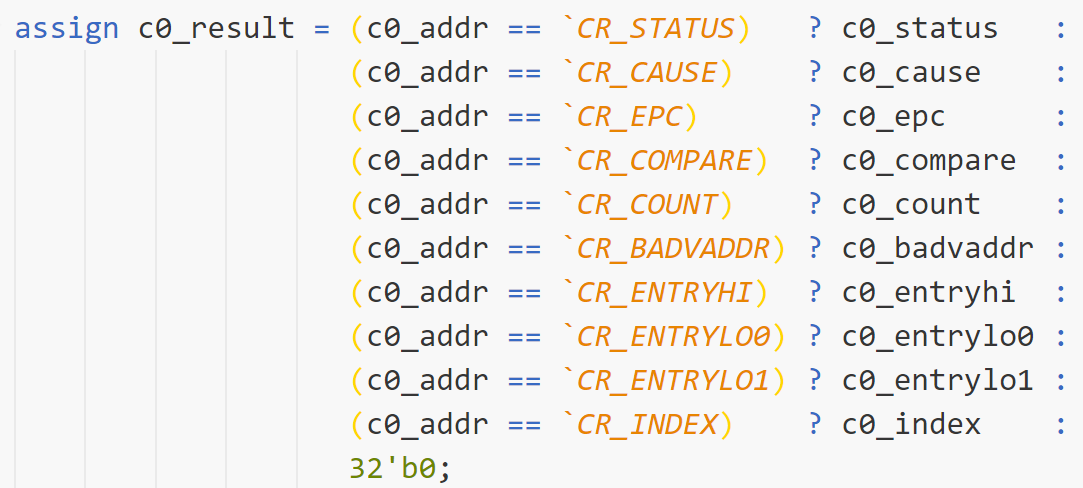
例化cp0部分略去，同时对于mfc0的result进行更新如下所示：

图14:c0\_result的额外选择器

三、实验过程（50%）

（一）实验流水账

12.3 19:00-22:00 12.4 16:00-16:30构思设计进行代码编写并debug成功

12.14开始写实验报告，零零碎碎大约花费4小时完成。

（二）错误记录

1、错误1：通过前4个测试点，后两个错误

1. 错误现象

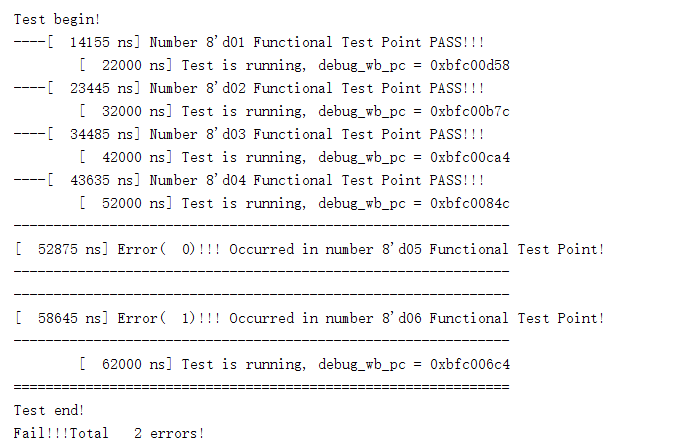
通过前四个测试，但无法通过后两个测试点：

图15:错误1的报错图

（2）分析定位过程

前四个测试点只测了新加的cp0寄存器，没有测试新指令，猜测是新指令出错，查找波形发现，应该存的重取的pc多变化了几次，根据波形判断本应该只拉高一拍的译码级重取信号拉高了很多拍，如下图所示。

图16:refetch拉高无法降低的波形图

（3）错误原因  
 refetch信号没有与ID阶段的有效信号相与，导致流水级有效信号拉低的的情况下还会重取错误的pc

（4）修正效果，

修改后代码见图06，修改后成功通过测试点。

四、实验总结（可选）

这次实验并不困难，第一部分代码我认为是体力活，需要大量重复的接口连接，是非常细致琐碎的工作，也是在我代码编写中花费时间最多的一部分，却没有什么理解上的难度。第二部分对于讲义设计的更改则是基于我偷懒和提高处理器性能的想法而实现的，代码构思的时间大部分花在这上面。在和助教交流这种设计思路的时候，也知道了讲义上设计的重取标志信号在指令系统特权态设计上的意义，而不是仅仅服务于这一次实验，由于本学期实验还不涉及到特权态的区分，因此我还是继续采用了这样可以提高性能的设计。