**实验15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lab14的基础上加入了三种TLB例外处理，主要更改了IF（包括pre-if和IF级）流水级的逻辑和EXE级的逻辑。

二、实验设计（40%）

（一）总体设计思路

本次实验主要更新了涉及了虚实地址转换的IF级流水线和EXE级流水线，其余流水线只是添加了相关总线流水信息，分模块设计思路如下：

1. Pre-IF模块：产生tlb\_refill和tlb\_invalid信号， 将原next\_pc改为tlb\_nextpc，新next\_pc会根据mapped的情况选择是否映射。定义pre\_ex信号，当预取指出错会取消发至总线的取指请求，同时添加流水级readygo信号判断逻辑使流水级正常运行。添加了3个触发器，在pre-if向IF级流水的时候暂存pre-if级的tlb例外信息和tlb\_nextpc。
2. IF模块：根据上述三个触发器在IF模块完成excode报出和badvaddr的更新，同时根据是否发生tlb例外更新fs\_pc。
3. EXE模块：改动较为简单，处理了refill，invalid，modify三种tlb例外，不涉及流水控制信号，只需要在es\_ex和es\_excode等例外相关信号的更新。同时整合IF级和EXE级的refill例外，向下传递直到wb级报出

CPU总体设计图如下图所示：

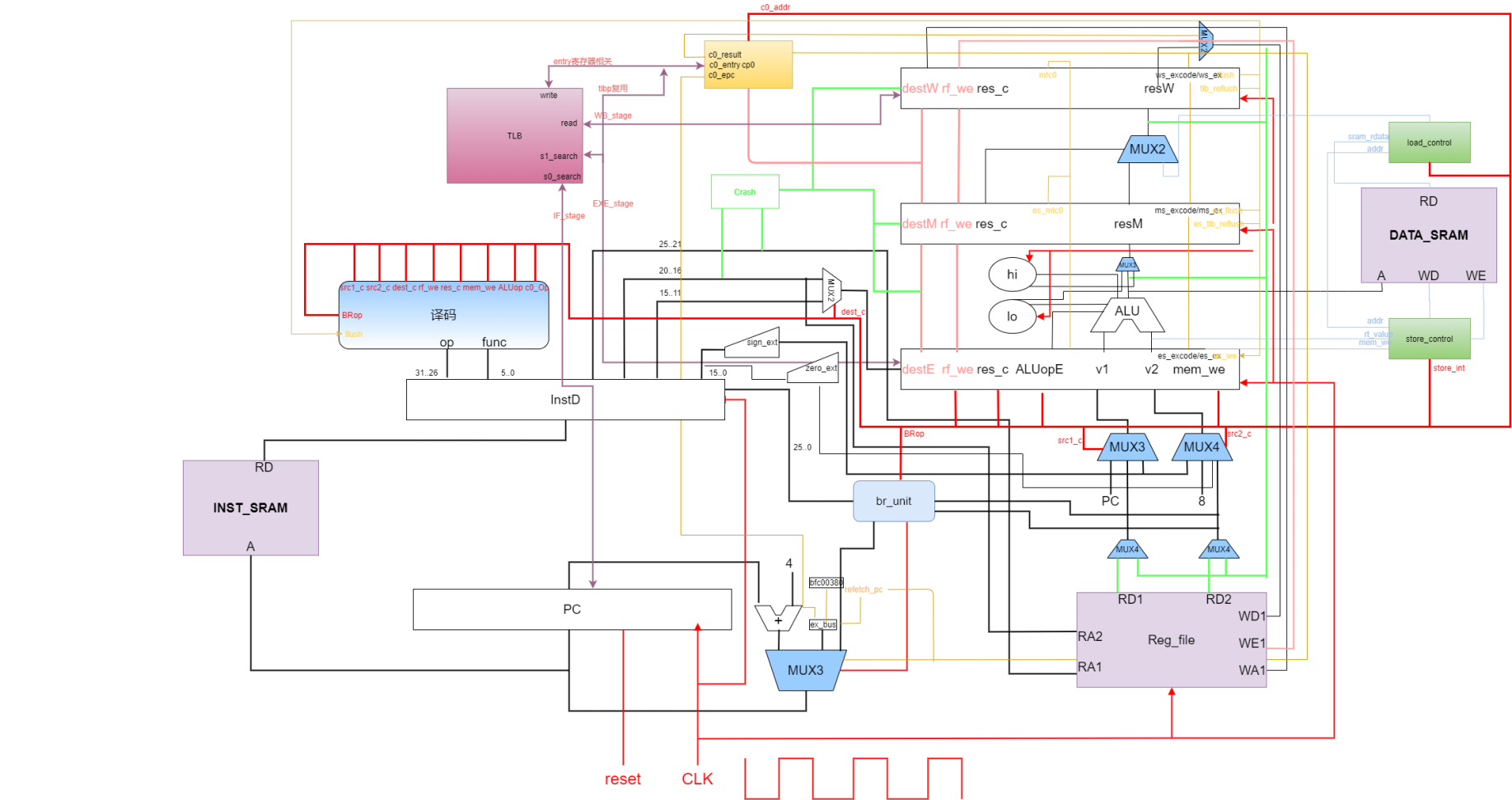


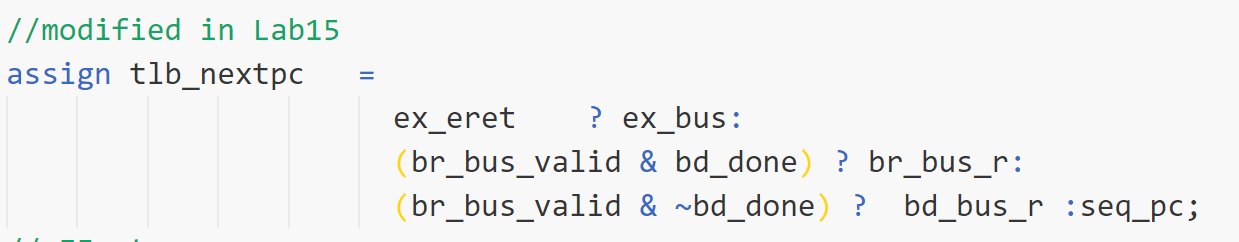
图01: 整体CPU设计

（二）重要模块1设计：IF模块（描述lab15更新）

1. 工作原理（内容描述）  
    接收ws模块报出的tlb\_refill错误，跳转到对应例外处理地址。内部的信号更新见详细内容描述
2. 接口定义（仅展示lab14更新）

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Fs\_to\_ds\_bus | OUT | 104 | 向下一级额外流水了tlb\_refill\_r信号 |
| Ws\_tlb\_refill | IN | 1 | 上一次处理器的refill信号，决定例外跳转地址 |

1. 功能描述  
   将原本next\_pc更新为tlb\_nextpc：

 图02:：tlb\_nextpc的设计

Pre-IF级：

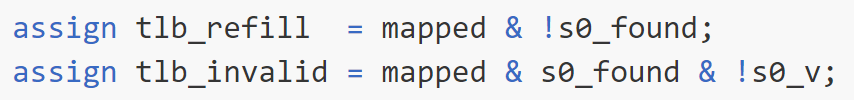
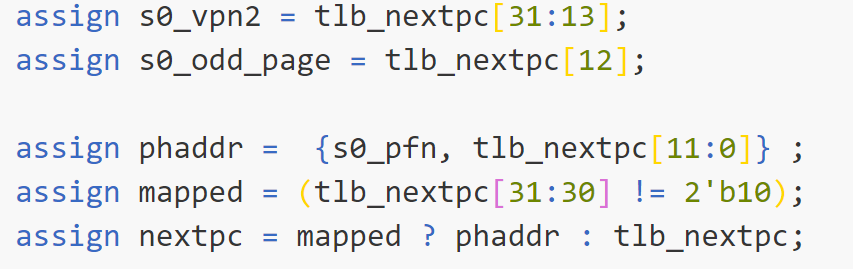
根据tlb查找信息，返回取值阶段两种tlb例外信号如下：

图03: tlb例外信号

用tlb\_nextpc作为查找的tlb映射关系如下，当pc信号高位和次高位不为10的时候表明不在可映射段，这时候需要用物理地址，物理地址的拼接方式如下图phaddr所示：

图04: next\_pc选择

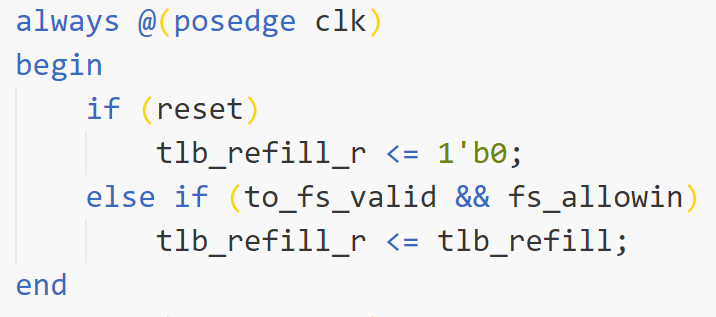
由于tlb例外在pre-if级发生，当指令流动至if级，对应的tlb信息则是下一条指令的信息，所以在向if级流水的时候需要用寄存器暂存tlb例外信息和此时的tlb\_nextpc值，这里仅展示一个触发器的逻辑（当可以向if级流水的时候进行暂存）如下所示：

图05:暂存pre-if级信息

IF级部分：

fs\_pc需要在产生tlb例外的时候选择tlb\_nextpc。

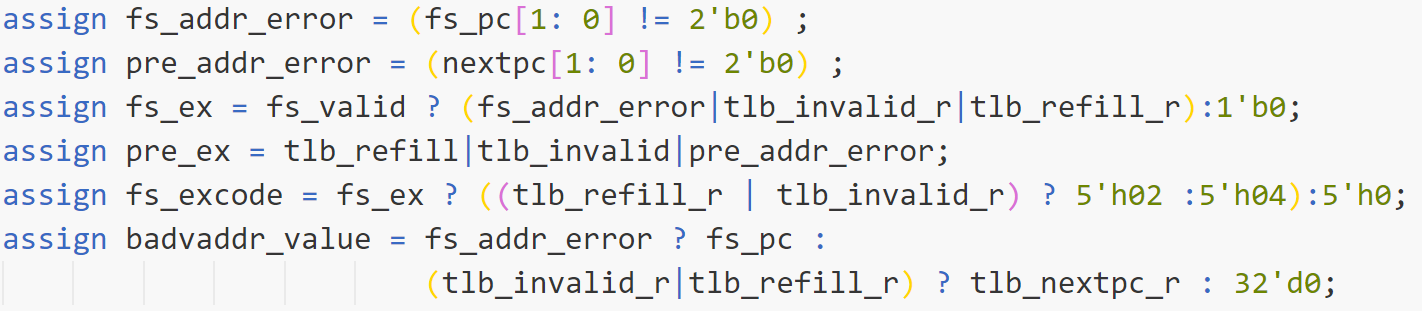
在整个IF级的错误处理分为两个大类，地址错例外和tlb例外，这两类例外其实在preif级就可以判断出来，对于badvaddr的值，当为地址错例外的时候用fs\_pc作为值，当为tlb例外的时候需要用之前暂存的tlb\_nextpc\_r值，整体代码如下图所示：

图06:整体例外处理代码设计

增加了pre\_ex错误，包含pre-if级的tlb例外和地址错例外，这个信号会用于阻塞向总线发送请求和拉低cancel信号，涉及代码如下所示：

assign to\_fs\_valid  = ~reset & ((inst\_sram\_addr\_ok & inst\_sram\_req) || pre\_ex);//lab15

assign inst\_sram\_req = !br\_stall & !reset & fs\_allowin & !request\_control & !fs\_ex & !pre\_ex;

else if((ws\_ex|eret\_flush |tlb\_reflush)&((to\_fs\_valid & !pre\_ex) | (!fs\_allowin & !fs\_ready\_go))) //valid 和 ready\_go用的同一个信号 所以需要保证为高的时候拉低cancel

在本小组设计中，to\_fs\_valid信号相当于同时作为了pre-if级的valid和ready\_go信号，这里会牵扯到需要cancel的情况的第一种，需要做特别设计，所以这里需要用pre\_ex来控制拉低cancel信号，具体的逻辑见错误记录1。

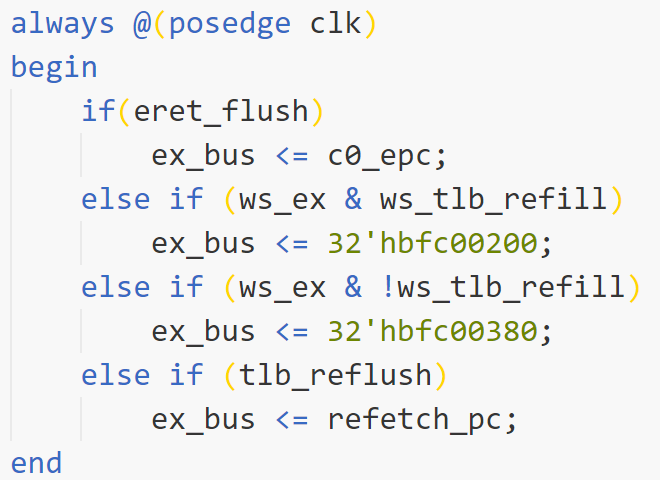
最后，由于之前使用ex\_bus寄存器存储发生例外时的跳转地址，tlb\_refill是特别的例外处理地址，所以对于添加了tlb例外的ex\_bus寄存器代码如下所示：

图07:例外跳转地址寄存器逻辑

（三）重要模块2设计：EXE模块

1. 工作原理

添加了三种tlb例外处理，同时将IF阶段和EXE阶段都可能出现的refill信号进行整合，向下一级流水传递

1. 接口定义（lab14更新）

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| Es\_to\_ms\_bus | OUT | 134 | 增添了向下一级传递的综合refill信号的总线 |
| ds\_to\_es\_bus | IN | 214 | 增添了带有IF阶段refill的总线 |

1. 功能描述

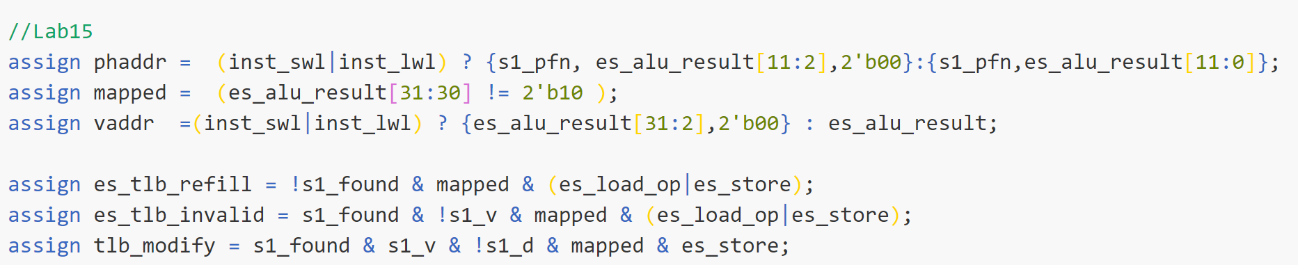
EXE阶段的例外处理基本与IF阶段类似，而且不需要对流水线控制信号进行太多处理，同时多了一个modify例外，具体的处理代码如下所示：

图08:EXE阶段tlb例外信号逻辑

在这里按照之前的讲义要求，在指令为swl和lwl指令的时候，把物理地址和虚拟地址低两位都强制抹0，refill和invalid只会在涉及到load或者store的时候触发，modify时在store时映射段对应的dirty位不为1的时候触发。

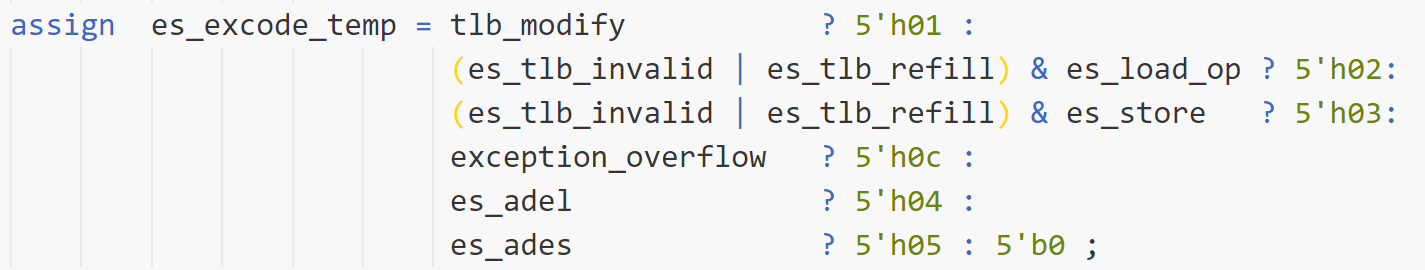
查阅指令集手册后，更新excode的逻辑如下图所示：

图09:EXE阶段excode逻辑代码

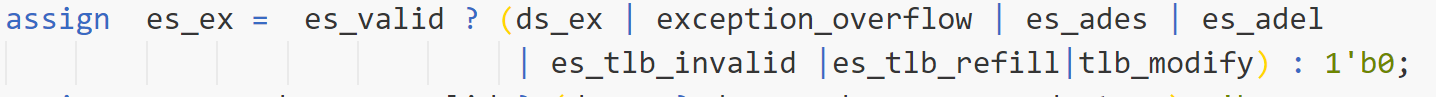
同时将新加入的三种例外填入es\_ex错误中，（在es\_we中有！es\_ex选项，从而在发生例外时阻塞请求等，这些处理之前的实验报告中都有提及，这里就不重复展示代码了）

图10:EXE阶段es\_ex逻辑代码

三、实验过程（50%）

（一）实验流水账

12.7晚上8:00-23:00完成代码的设计和编写，仿真出现错误，暂时搁置

12.8早7:30起床修改了一处代码逻辑，通过仿真。

12.21开始写实验报告，零零碎碎大约花费3小时完成。

（二）错误记录

1、错误1：例外处理的下一条指令被错误的取消

1. 错误现象

cancel错误拉高导致例外处理的第一条指令被取消掉了

（2）分析定位过程

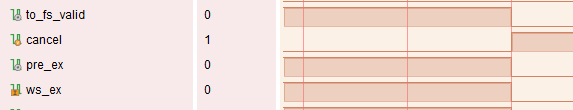
观察波形，发现是cancel信号错误的被额外拉高了几拍，导致例外处理结束后下一条指令也被取消掉，波形如下图所示：

图11:cancel错误拉高示意图

（3）错误原因  
 在之前的实验设计中，to\_fs\_valid信号与pre\_if级的ready\_go信号，并无区别，因此代码中将二者并为一个信号。

但在这个实验中，当preif级发生TLB例外时，需要将preif级的ready\_go信号一直拉高保证流水级正常运转。在一开始的代码处理中认为pre\_if级不会产生例外，所以当例外刷新if级时会发现to\_fs\_valid信号拉高，错误的判断为需要取消的情况。

（4）修正效果，

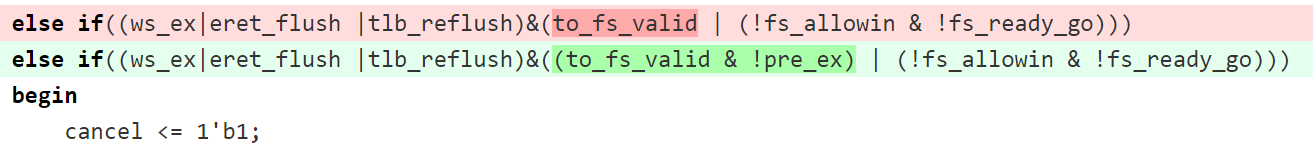
 修改前后代码变化如下图所示：

图12:gitee代码修改记录

在修改后仿真通过，上板通过。

四、实验总结（可选）

本次实验主要涉及到三种tlb例外的添加，其中exe阶段数据访问的tlb例外添加是比较简单的，因为在exe阶段的只有一条指令，所有的信息都是当前指令生成的，不需要使用额外的寄存器去存相应的信息，而在取指级，由于preif级和if级对应的是两条pc，则需要使用一些寄存器存储相关信息以保证对应的例外标志不会出错，还是比较麻烦的。

同时在这次实验中，发现讲义中要求出现地址相关例外时不能向总线发出请求，第一版的代码设计中，虽然发出了请求但也能通过测试，因为被标记例外的指令实际上是什么也做不了的。但是这与设计思想不符，因为这会导致软件设计人员无法控制的地址访问，因此又设计了发生例外取消访存的逻辑，顺便把之前的取指错例外加上，感觉实验的测试可以完善上这一部分。