**实验九报告**

学号 2017K8009929044

2017K8009929025

姓名 李昊宸 李颖彦

箱子号 33

一、实验任务（10%）

1. CPU增加break例外的支持

2. 增加地址错、整数溢出、保留指令例外支持

3. 增加时钟中断支持，绑定在硬件中断5号上。

4. 增加6个硬件中断支持和2个软件中断支持，对应IP7~IP2，IP1~IP0。

二、实验设计（40%）

（一）总体设计思路

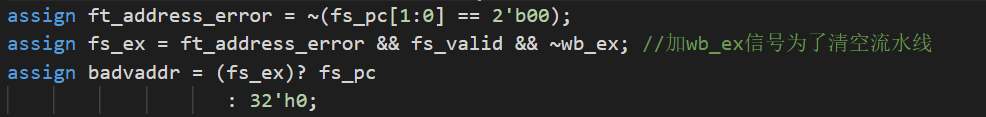
根据各个例外的要求，逐个实现例外。其中：取值错误在IF产生，BREAK指令的处理在ID阶段产生，取数和存数地址错在EXE阶段产生（主要是因为此时产生alu\_result）,MEM没有大的修改，WB阶段处理时钟中断，外部中断，以及各个例外的集中处理。

（二）重要模块1设计：IF模块/取指错误

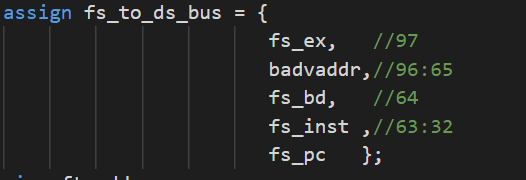
1. 工作原理

当PC末两位不为0时，触发取值错误。

1. 具体代码实现



ft\_address\_error在pc的末两位不为0，即不四字节对齐时置为1。如果出现了取指错误，代表if阶段发生例外，将此阶段的fs\_ex置为1，并存下错误地址，放入badvaddr中。将数据打包，传至下一阶段。



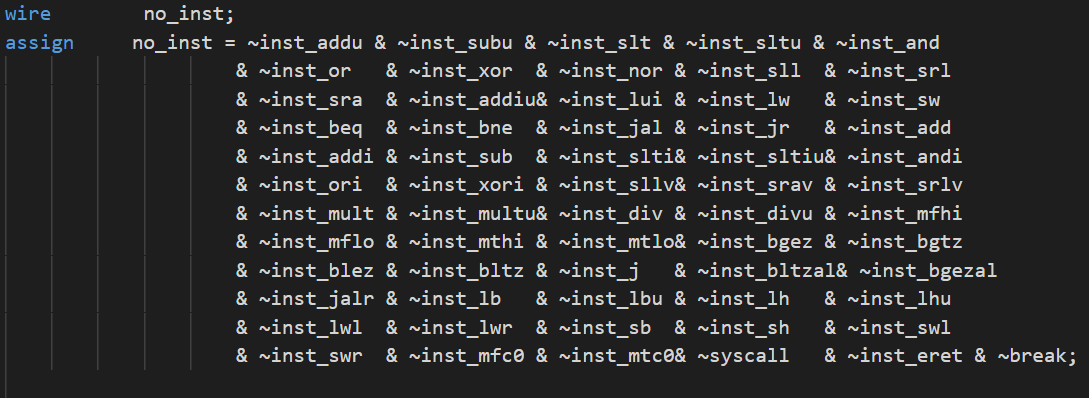
（二）重要模块2设计：ID阶段/保留指令例外/Break指令/其他

1. 工作原理

通过对所有已有指令取反来判断是否触发保留指令例外，通过对指令进行译码来确定是否是break指令，如果出现上述两种情况之一，触发ID阶段的例外。

1. 具体代码实现

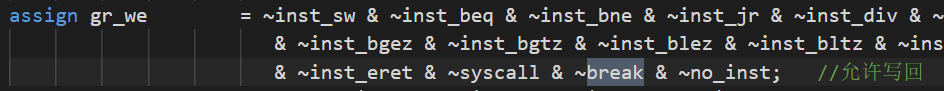
a)保留指令例外



如上图，对所有已有指令取法，若所有指令均不满足，触发保留指令例外。

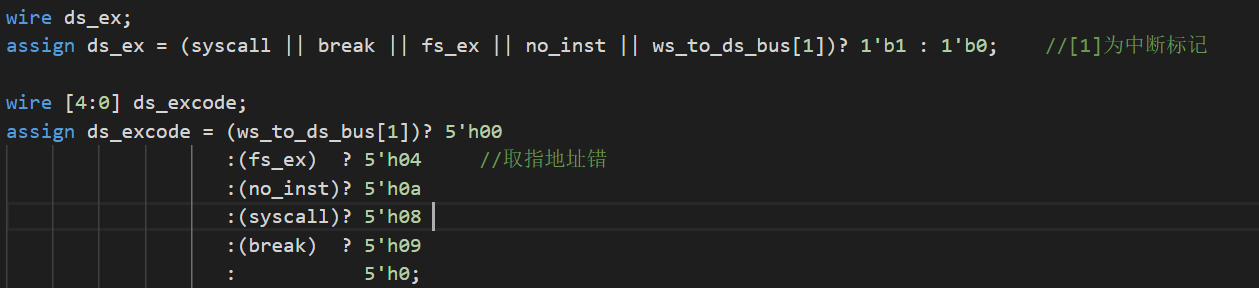
b)break指令相关



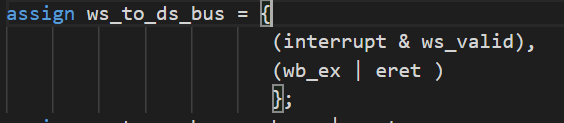


gr\_we的修改是防止数据的写入和前递。

c)id模块总体处理



如上，如果是syscall,break,no\_inst或者fs阶段发生例外（取值错）或者wb阶段总体处理时发现此时有例外，ds\_ex信号拉高，ws\_to\_ds\_bus[1]为：

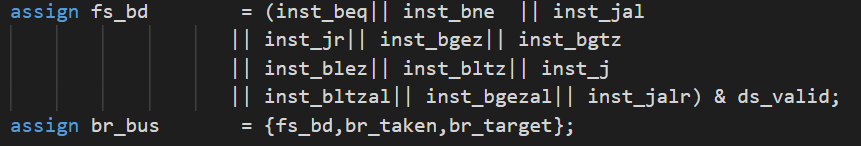


此时，根据例外处理优先级来对excode赋值，首先如果此时wb阶段有信号，则清空流水线，否则按图中顺序进行处理。

d）后续准备

1）

延时槽在此阶段可以进行处理



如果此指令是分支跳转指令，则为下一条指令（此时在if阶段的指令）打上延时槽标记。

2）



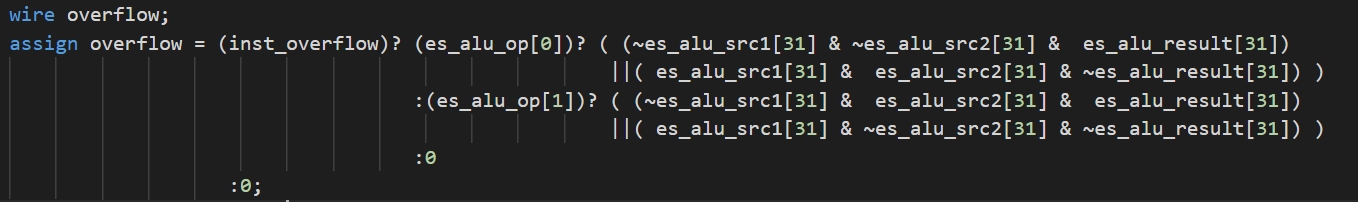
将会发生overflow的指令标记。方便exe阶段处理。

（三）重要模块3设计：EXE阶段/溢出异常/取数异常/存数异常

1. 工作原理

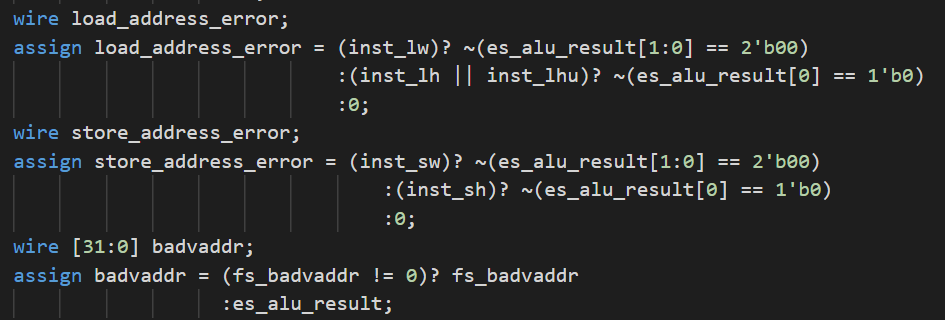
在exe阶段会产生alu\_result，而取数与存数指令的地址均为alu\_result，所以想要判断是否地址异常，只需在此时判断alu\_result即可。另外，如果是会发生溢出的指令，在产生结果后对结果进行溢出判断。

1. 具体代码实现
2. overflow



首先判断是否是会产生overflow的指令，之后判断：

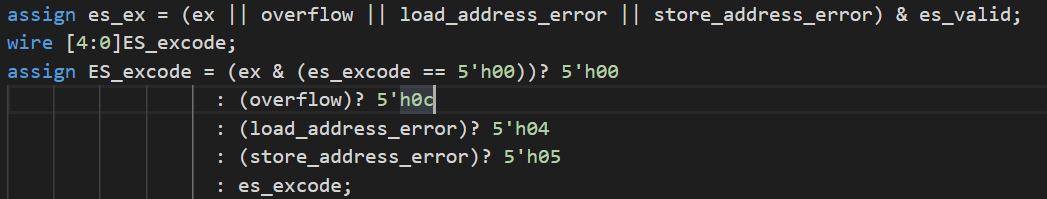
1. 如果是加法，两个正数相加产生负数，或者两个负数相加，产生例外。
2. 如果是减法，负数减正数得到正数，正数减负数得到负数，产生例外。
3. 存数与取数异常：



如上图：根据lw,lh,lhu,sw,sh的地址对齐要求，如果不满足要求报错，对于badvaddr，如果取指错误，那么取指错误优先级较高，优先处理；否则将badvaddr保存为alu\_result，即错误的地址。

c)汇总：

exe的例外信号为前面阶段传递过来的例外信号或上三个在exe阶段发现的例外，并根据优先级来决定excode，其中第一行是代表刷新流水线。



（四）重要模块4设计：WB阶段/时钟中断/例外汇总处理

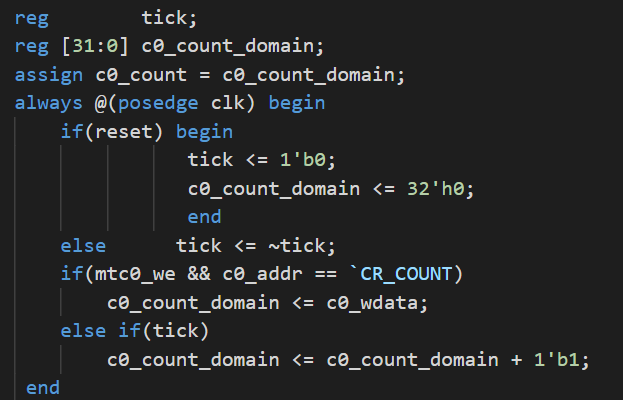
1、工作原理

在wb阶段通过count和compare寄存器来实现时钟中断，并且所有的例外均汇总到wb阶段来集中处理。

2、具体代码实现

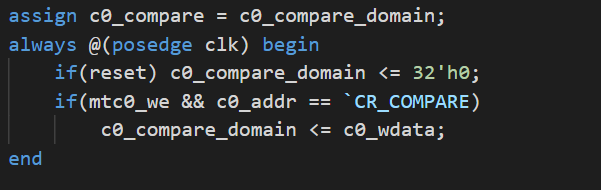
a）时钟中断：

1.count寄存器：



count寄存器每两个时钟周期加1，并且是可写的，所以可以被mtc0修改

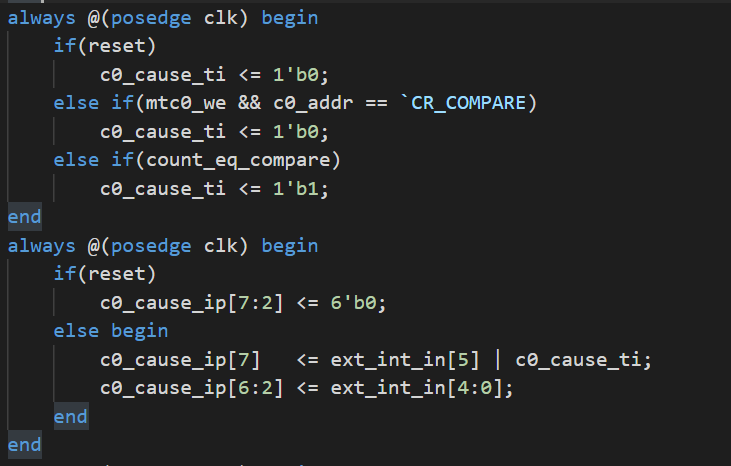
2.compare寄存器：



3.时钟中断触发条件：



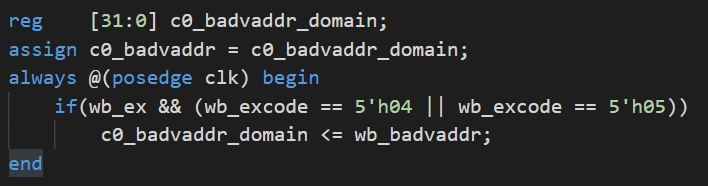
compare！=0这一条件是为了避免在compare没有被使用时仍然触发了时钟中断



在写compare寄存器时，时钟中断自动清零，当count和compare相等时，c0\_cause\_ti信号拉高。可以看到，时钟中断复用ip7即hw5.

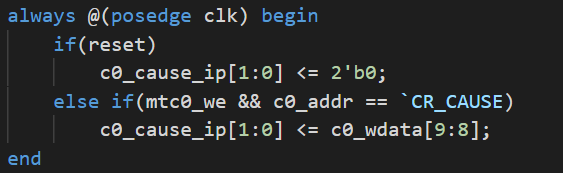
b)badvaddr

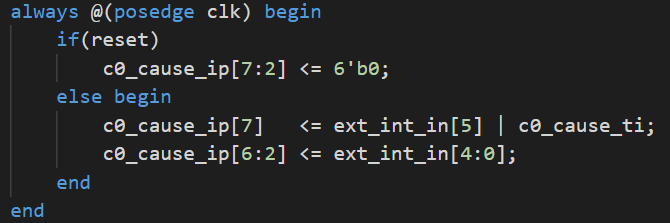
在取指，取数，读数地址发生例外后，将错误地址写入badvaddr寄存器。



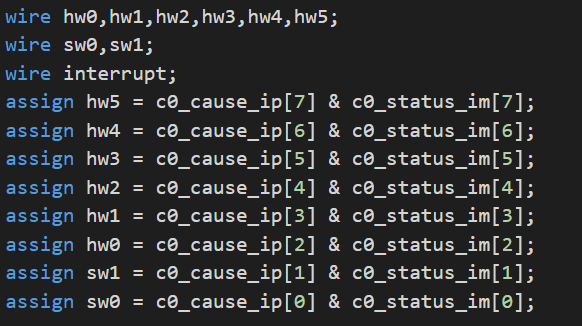
1. 中断

cause寄存器被外部中断修改，或是被软件修改：

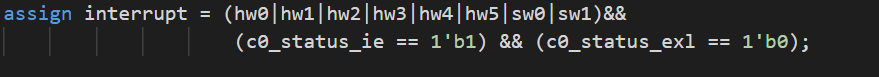




之后汇总：



当cause寄存器对应位（表示产生）和status寄存器对应位（表示允许）均为1时，这个中断准备产生。



然后当status\_ie位为1，exl位为0时，中断产生。

三、实验过程（50%）

（一）实验流水账

（二）错误记录

1、错误1：周期错位

（1）错误现象

在发生地址错错误时，例外信号总是在取指之前就报出，导致在流水线中位于例外之前的一条指令在执行时由于被标记了例外信号，会出现不定的执行错误。

（2）分析定位过程

出现例外提前标记的原因，可能是对地址错例外的判定提前于取指过程。由于指令ram取指的同步性，在时

钟上升沿之前向ram发出的读请求，读出值会在上升沿时返回。也就是说，IF流水级在上升沿之前将nextpc送至指令ram，在上升沿到来时更新fs\_pc为nextpc，此时对应返回的指令也是fs\_pc所对应的指令。于是问题可能出现在对fs\_pc还是nextpc进行地址错例外判定。

（3）错误原因

*assign ft\_address\_error = ~(nextpc[1:0] == 2'b00);*

错误出现在对地址错的例外判定使用了nextpc，而不是当前的fs\_pc。

修改方式：*assign ft\_address\_error = ~(fs\_pc[1:0] == 2'b00);*

（4）修正效果

修改后，出现了如同错误2的bug。

2、错误2：逻辑语句书写错误

（1）错误现象

在添加地址错例外语句后，原先应该发生例外，工作正常的语句如今出现了不正确的写错误。

（2）分析定位过程

在自己书写的代码结构中，决定是否进入例外处理入口的信号是例外信号。原先发生例外，修改后例外反而不触发，暗示例外信号拉高的逻辑可能修改错误。检查例外信号，信号本身并无错误。进一步检查，为什么会出现写错误。回顾之前的代码结构，发生例外时需要清空流水线，为避免新取出的指令对后续模块的影响，当写回级检测到例外信号时，原先的写法是将取指级的fs\_inst信号置零。很有可能是这里的修改错误导致问题的发生。

（3）错误原因

*assign fs\_inst = (（~wb\_ex | ~fs\_ex) & ~inst\_eret )? inst\_sram\_rdata : 32'b0;*

错误出现在对原先代码结构的理解错误。需要将fs\_inst置零的情况有两种：清空流水线和出现了取指级的地址错例外。于是，wb\_ex信号和fs\_ex信号的关系为只要有一个为高，fs\_inst就置零。这与上文写到的代码逻辑不符，错误出现在这里。

修改方式：*assign fs\_inst = (~wb\_ex & ~fs\_ex & ~inst\_eret )? inst\_sram\_rdata : 32'b0;*

（4）修正效果

修改后，成功运行。

3、错误3：延时槽错误处理

（1）错误现象

遇到处于延时槽的指令中断未正常处理

（2）分析定位过程

此时出现了奇怪的情况，bd信号并未拉高，而前一条指令bd拉高。

（3）错误原因

*assign ds\_bd = (inst\_beq|| inst\_bne || inst\_jal*

*|| inst\_jr|| inst\_bgez|| inst\_bgtz*

*|| inst\_blez|| inst\_bltz|| inst\_j*

*|| inst\_bltzal|| inst\_bgezal|| inst\_jalr) & ds\_valid;*

这里直接将当前指令视为延时槽指令，可实际上应该是下一条指令

修改方式：*assign fs\_bd = (inst\_beq|| inst\_bne || inst\_jal*

*|| inst\_jr|| inst\_bgez|| inst\_bgtz*

*|| inst\_blez|| inst\_bltz|| inst\_j*

*|| inst\_bltzal|| inst\_bgezal|| inst\_jalr) & ds\_valid;*

*assign br\_bus = {fs\_bd,br\_taken,br\_target};*

（4）修正效果

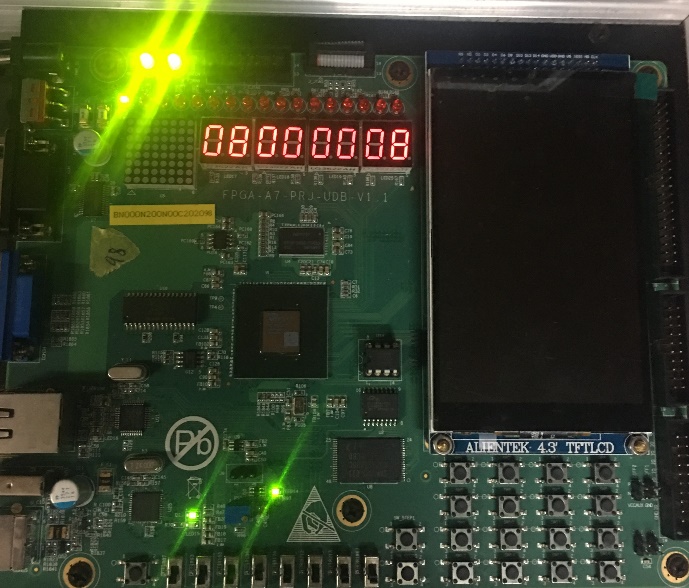
修改后，成功运行。

四、实验总结（可选）

又到了一周一度的激动人心的上板验证环节



图一 测试上板成功拍照留念



图二 记忆游戏上板成功拍照留念