Lab5 PB19030800 陈磊

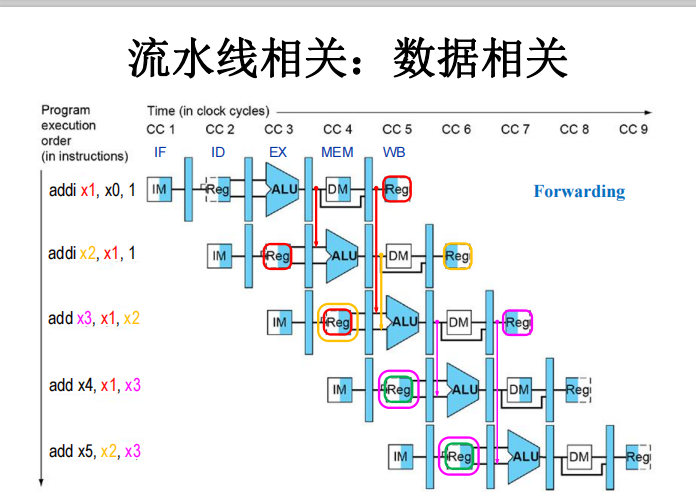
实验步骤：

1. 添加流水线寄存器if\_id,id\_ex,ex-mem，mem\_wb,在时钟上沿更新，
2. 添加forward与hazard\_branch模块，
3. 调整模块端口，reg\_file变为写优先（PS:本实验相当部分的工作量在原单周期pcu与新增的pipeline寄存器的端口分配与逻辑变更）

模块分析:

注：Regsrc在没有信号的时候=2’bxx（2‘b00为有效信号），所以后面有效ctrl中会出现X

1.module forward



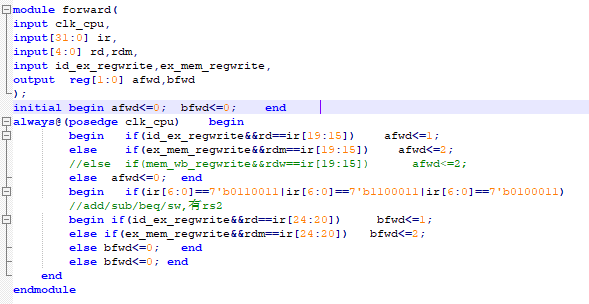
这里采用时序逻辑，就图中实例分析，

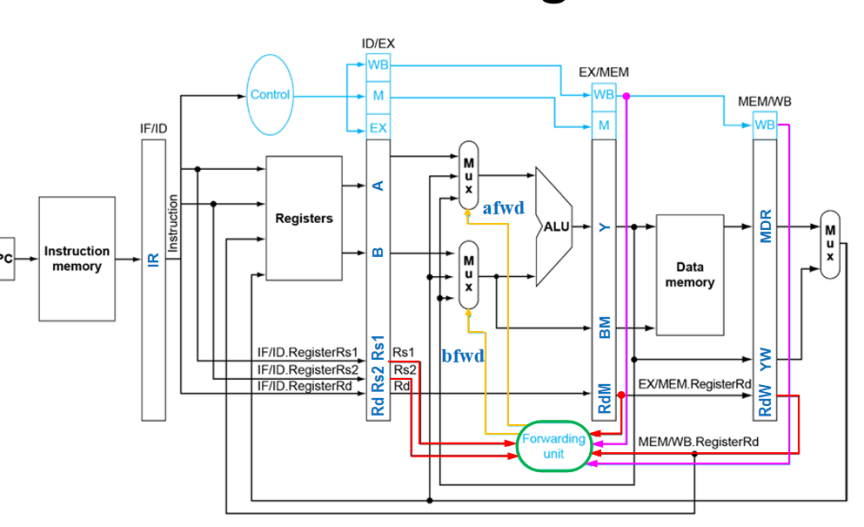
在第三条指令执行到EX阶段时钟上沿(此时ir仍为第三条指令取出的代码，rd，rdm仍为第二、一条指令目标寄存器，id\_ex\_regsrc、ex\_mem\_regsrc属于第二、一条指令），

此时判断是否需要前递，输出afwd,bfwd,

采用多选器根据afwd,bfwd选择rs1,rs2,alu采用组合逻辑，当前周期输出正确的aluresult

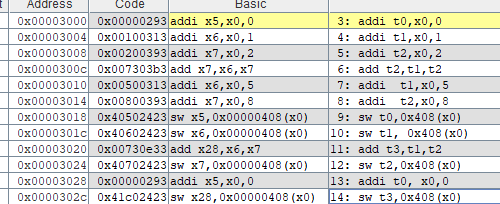
代码与示意图如下：

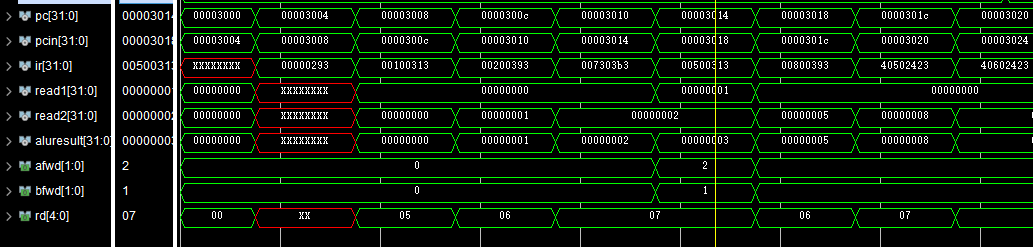




PS:该图为组合逻辑，使用的是时序逻辑，将forward unit各输入信号改为提前一个周期的信号即可，

汇编代码与仿真结果如下：





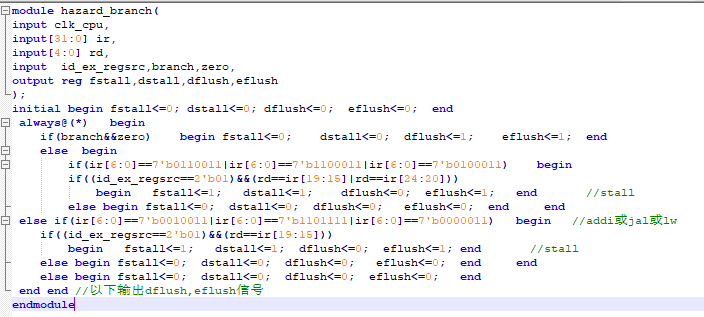
对0x0000300c：add x7,x6,x7,

可以看到0x00003014处，该指令执行到EX阶段，rs1=x6,afwd<=2(rs1==提前两个周期的rd)

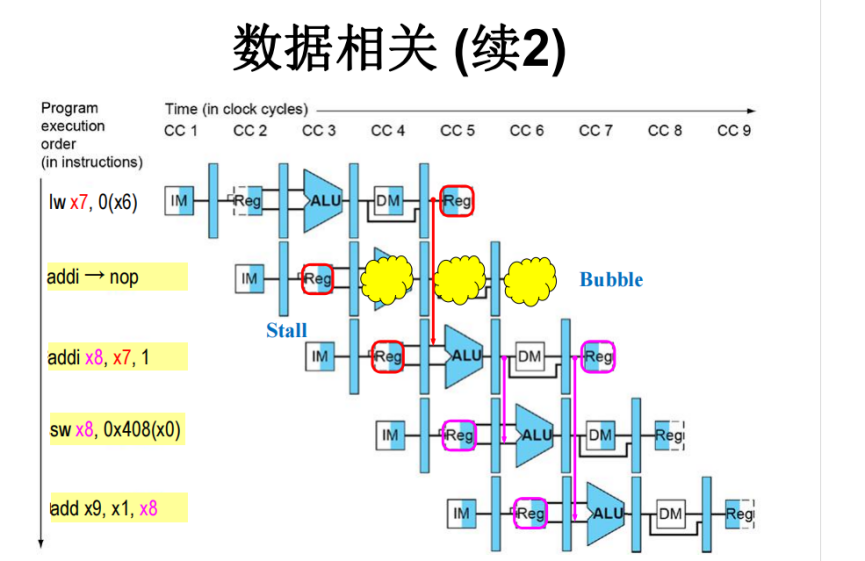
rs2=x7,bfwd<=1(rs2==前一周期的rd),所以read1=x6=1,read2=x7=2,aluresult=3,rd=7,写入x7

各寄存器的详细值仿真结果：

1. module hazard\_branch



**hazard**



由此图可见，若load指令与use指令至少相差2位置，必然可以前递，

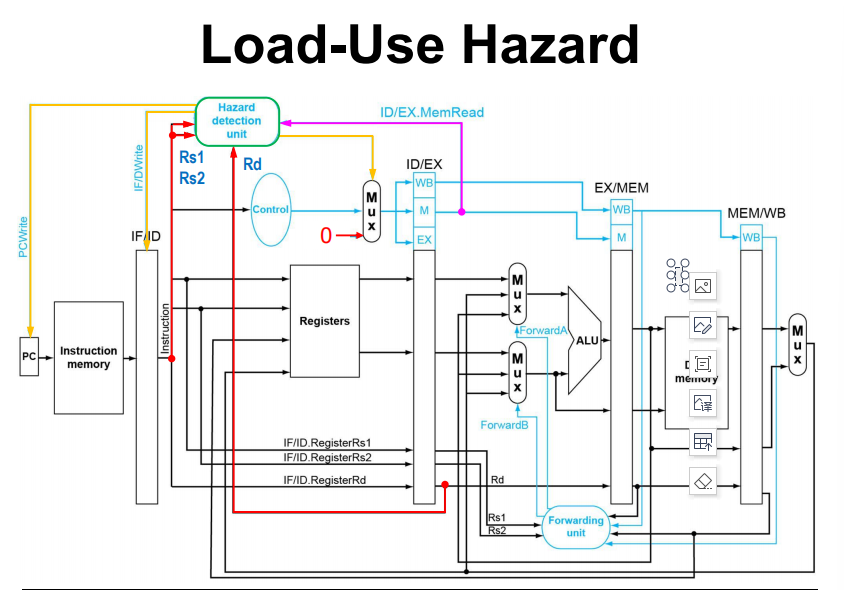
只有相邻指令会出现load-use停顿，

采用组合逻辑，在use指令的id阶段检测，

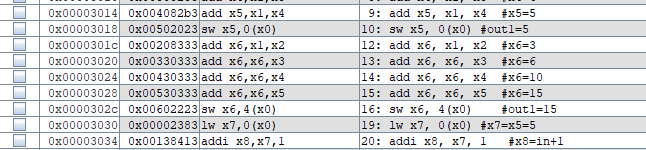
if id\_ex\_regsrc==2’b1&&(rd==rs1|rd==rs2),停顿use指令与其后的指令，

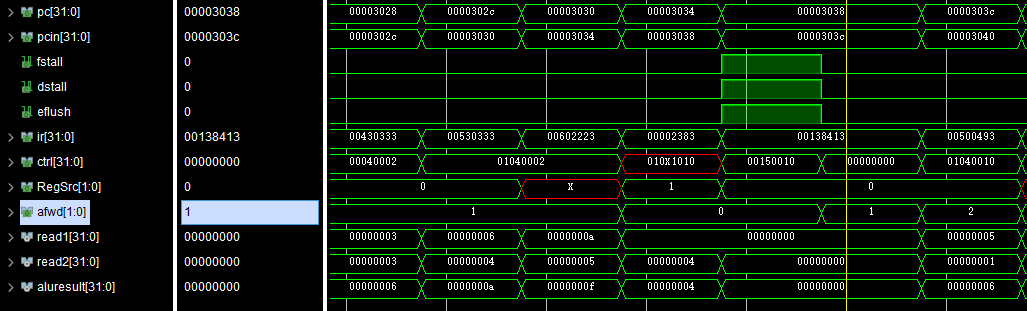
EX阶段插入一条nop指令，

下一阶段上沿fstall,dstall,eflush作为条件判断变量停顿pc,if\_id寄存器，置0id\_ex寄存器



汇编代码与仿真结果：





0x00003038前半段，use指令0x00003034:addi x8,x7,1处于id阶段，

hazard部分检测到load\_use冒险，fstall,dstall,eflush<=0，

0x00003038后半段pc,ir取原有值，ctrl置0,

之后进入use指令ex阶段,x8=x7+1

值得一提——

3038后半段afwd<=1,because前一周期 use id阶段,load处于ex阶段,检测到rs1=rd

303c:afwd<=2, because 前一周期use由于停顿仍id段，load继续到mem段，rs1=rdm

writedata=6前递给x7,x8=x7+1=7; 完成控制冒险

**branch:**

分支不发生时就是pipeline 正常执行，branch module只要处理分支发生时善后工作

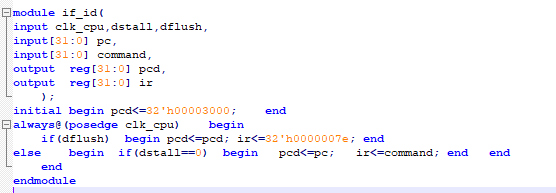
分支发生，组合逻辑module branch立即把dflush，eflush置1，

下一周期作为条件判断变量置0if\_id\_reg与id\_ex\_reg，

提一下dflush的刷新想法：目标是使下一周期判定控制信号为0；

可以直接在dflush判定为1的周期（也即if\_id\_reg发生的周期）将cu各控制信号置为0，

此时可由if\_id\_reg向cu输出特定ir（如0x0000007e)，cu接收到该ir后输出全为0信号，



cu将该指令译码，各控制信号输出为0，

cu译码段添加下述段：

7'b1111111: begin //若为dflush==1时输出指令，则将控制信号全部置为0

jal=0;

Branch=0;

RegSrc=2'bxx;

MemWrite=0;

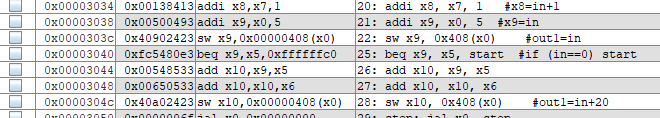
Alusrc=0;

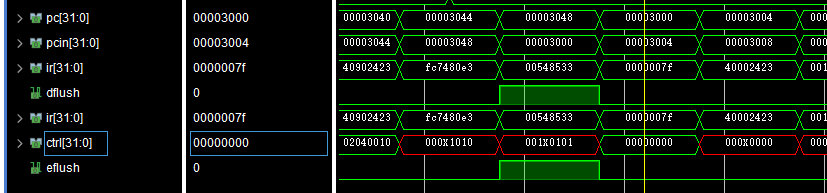
Regwrite=0;

ALUOp=2'b00;

End

汇编代码与仿真结果：





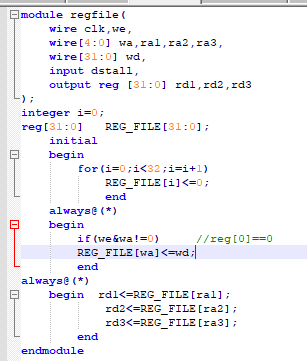
3048 beq demand进入ex阶段，dflush,eflush<=0,并在下一周期跳转到3000;

3000处,3044指令进入ex,ctrl<=0,3048进入id,ir=eflush专用指令0000007f;

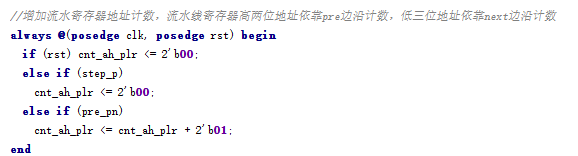
3004处3048进入ex，ctrl控制信号置0或X(Regsrc==2’bxx)

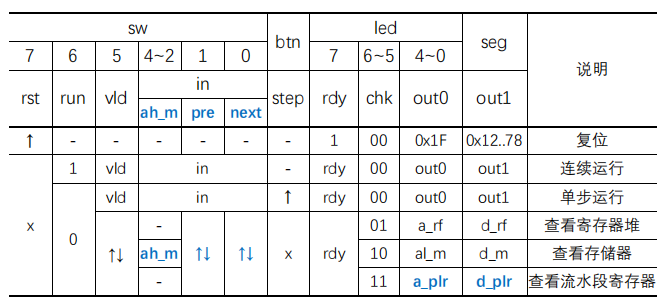
1. 写优先regfile

改为组合逻辑即可近似看成写优先，反正终究会读成写入的值

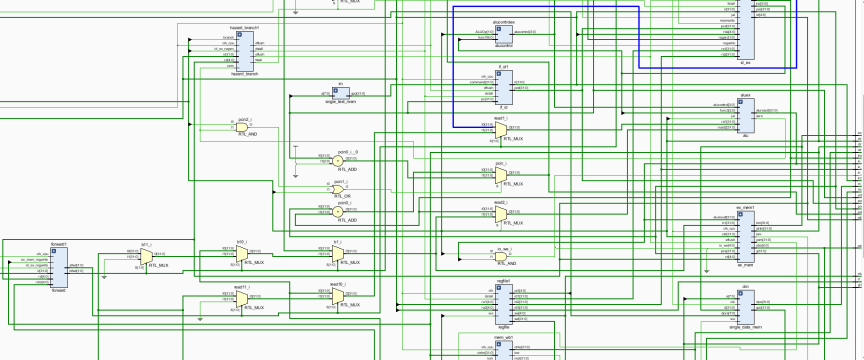


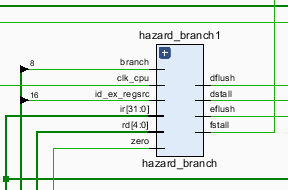
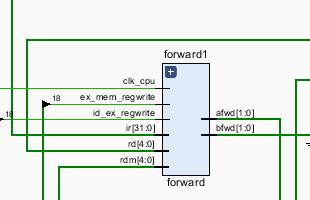
4.pdu

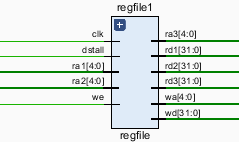
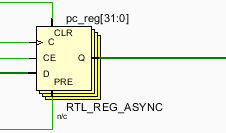
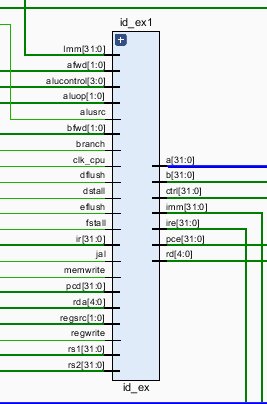
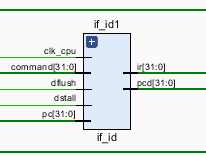




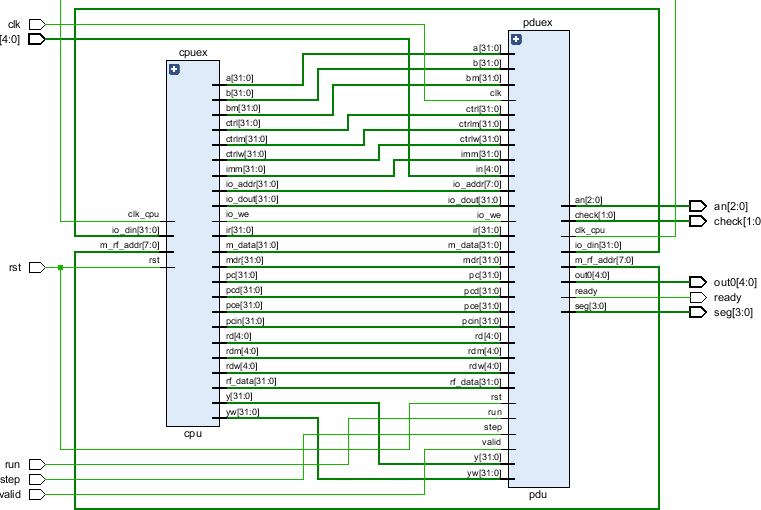
三、RTL:







(module regfile 的dstall端口一开始设计错了，后来没用上也就没删了）



1. 实验收获与问题：
2. 了解cpu处理数据冒险，控制冒险与分支（不预测）的过程；
3. 学习使用分析错误信号来源，仿真中加入对应信号，一步步溯源的debug的方法；
4. 完成一部分工作后就要及时备份，避免debug错误且又回忆不起来原有写法的惨剧；

修改.v文件，也可把要改代码段赋值并注释，再修改，若错误也可方便找回原有写法；