**Lab 4报告**

学号：2021K8009910001

姓名：徐畅杰

箱子号：34

一、实验任务（10%）

1. 完成5级流水线的拆分，通过在中间增加寄存器与控制逻辑，让处理器流起水。

二、实验设计（40%）

（一）总体设计思路

本次实验设计主要分为三步：

1.增加寄存器用于储存不同流水级的信号，从而将处理器切分为5级流水，分别为取指（fetch），译码（trans），执行（exe），访存（mem），写回（wrtieback）。为了命名的简洁与同意，在本设计中，所有寄存器的后缀名都为其接受的流水级的名词，例如pc\_mem，就表示上一个时钟周期从mem流水级中流出的pc值，也就是将要送入writeback的时钟值。

2.通过增加检测机制用于检测指令写后读相关，具体实现上就是把在执行，访存，写回级的写回寄存器号与译码流水级需要读出的两个寄存器进行比较，若判等电路发现相等而且发生冲突的流水级的值确实会写回（gr为1且valid为1），而且发生冲突的寄存器确实会在该指令中使用，那么就会检测出冲突。

而在处理该冲突时，会将该指令阻塞在译码流水级，但是为了简化相关逻辑，这里选择其余部分的传递仍然正常，但是会将其对应的valid信号抹为0，也就相当于传递了一条空指令。

为了处理这里阻塞流水级之前的流水级，需要增加控制逻辑将指令阻塞在之前的流水级，具体方法为给每个流水级增加了allowin信号，一个流水级的allowin为1，当且仅当该流水级没有冲突而且该流水级下一级allowin为1或者该流水级无效。但是由于指令ram为同步ram，其在取出指令后会有一个流水级的延迟，当发生写后读相关时下一条指令的地址已经送入了指令ram，这也就导致了不能再想当然地认为指令ram送出的就是下一个送入译码流水级的指令，为此增加了一个寄存器用于暂存指令ram的指令，当发现这一级被阻塞时译码级会使用寄存器中的指令而非指令ram取出的指令，当阻塞结束后继续使用指令ram取出的指令，相应的当发生阻塞时，送入指令ram的pc值也不再更新。

而在检测控制相关时，只需要查看是否有分支指令，若有则将分支的流水级之前前至最后取出指令的流水级的valid都抹为0即可。

3.前递的设计：

当发生写后读冲突时，其实也可以复用其检测信号用于进行前递工作。当发生写后读冲突时，若冲突的流水级内容已经和将要写回的值一致（比如执行完的alu结果且已经确定写回内容不来自内存），则用该值代替从寄存器堆中取出的内容，并传递给下一流水级。

本设计中并未将beq与bne指令提前到译码级进行，因为这样若发生前递，会产生很大的时间开销，有一条从执行级触发器的Q端到ALU到前递路劲，到译码级四选一，到转移指令的判定，到gennextpc的四选一到取值请求的虚实转换，到指令ram的输入的关键路径，会导致性能的下降。在这里我们选择将结果前递回到译码级但是不进行跳转的判断，而将跳转的执行推迟到执行流水级进行，虽然这样会多取消一条指令，而且延迟了一个周期，但是时钟频率相比增加前递之前不会有显著的降低。

（二）重要模块1设计： 冲突检测与前递逻辑

1. 工作原理

通过比对译码阶段时的有效的源寄存器与执行、访存、写回阶段的需要写回的寄存器值是否相同，与能否进行前递，得知是否发生冲突以及是否可以通过前递解决。

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| rf\_raddr1、rf\_addr2 | IN | 10 | 两个源寄存器 |
| dest\_exe、dest\_mem、  dest\_trans | IN | 15 | 可能导致冲突的之后流水级的写回寄存器 |
| valid\_trans、  valid\_mem、  valid\_exe | IN | 3 | 表示是否为有效判断会冲突，同时用于进行前递逻辑判断 |
| gr\_we\_trans、  gr\_we\_exe、  gr\_we\_mem | IN | 3 | 判断是否会写回用于进行前递与写回的判断 |
| res\_from\_mem\_trans、  res\_from\_mem\_exe | IN | 2 | 判断写回内容是否来自内存得知是否可以通过前递解决 |
| nconf\_trans | OUT | 1 | 得知是否在trans阶段发生冲突 |

1. 功能描述

用于得知是否会在译码流流水级发生冲突以及能否通过前递解决。

重要模块2设计：流水线阻塞逻辑

1. 工作原理

通过查看各个流水线是否发生冲突与阻塞得知流水线是否需要进行阻塞

1. 接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| nconf\_fc  nconf\_trans  nconf\_exe  nconf\_mem  nconf\_exe | IN | 5 | 检测各个流水线是否与之后的流水线存在冲突 |
| allowin\_fc  allowin\_trans  allowin\_exe  allowin\_mem  allowin\_wb | OUT | 5 | 告诉各级流水线是否allowin |
| Valid\_fc  Valid\_trans  Valid\_exe  Valid\_mem | IN | 4 | 得知各级别的流水线是否为有效 |

1. 功能描述

一级流水线为allowin的当且仅当其下一级无效或者下一级为allowin且无冲突的。用于进行流水线是否可以正常流水的判断。

重要模块2设计：流水寄存器

1. 工作原理

在allowin等信号的控制下，写入需要写入的信息。

1. 接口定义

太多了，略，每一级流水线都有输入输出。

1. 功能描述

暂存需要流到下一流水级的信息，使得处理器可以流起水。

三、实验过程（50%）

（一）实验流水账

9.25 13：00 – 14：00 完成lab7的设计工作。

9.25 14：30 – 17：00 完成lab7的代码工作。

9.25 18：30 – 20：00 完成lab7的调试工作。

10.1 13：00 – 14：00 完成lab8的设计工作。

10.1 14：30 – 17：00 完成lab8的代码工作。

10.1 18：30 – 20：00 完成lab8的调试工作。

10.5 13：00 – 14：00 完成lab7的设计工作。

10.5 14：30 – 17：00 完成lab7的代码工作。

10.5 18：30 – 20：00 完成lab7的调试工作。

10.6 13：30 – 17：00 完成该实验报告。

（二）错误记录

1、错误1：除法指令未完成就消失

（1）错误现象

除法没有做完，下一条指令就进入，导致计算错误。

（2）分析定位过程

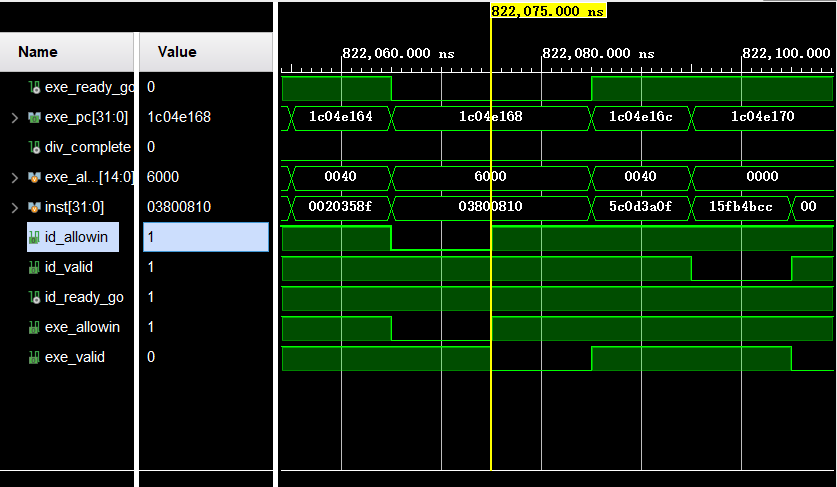


图1.1

可以发现除法指令没有做完valid就被抹去，导致下一条指令进入。

（3）错误原因

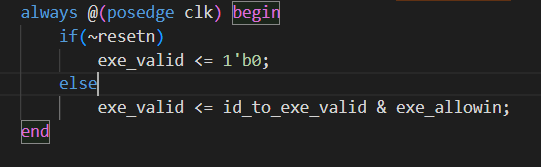
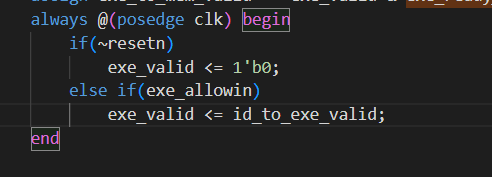


图1.2 流入条件打错了

执行级valid的进入条件错误。

（4）修正效果



按照原因修改，之后一切正常。

2、错误2： 如之前描述，因为指令ram为同步ram带来的错误。

（1）错误现象

发生因为写后读导致的阻塞时，指令ram需要取的下一条地址实际上已经送入了，所以译码流水级的指令也就错了

（2）分析定位过程



图2.1指令ram中取出的内容没有正常阻塞，导致错误

发生因为写后读导致的流水线阻塞时，从指令ram中取出的指令没有被正常阻塞，导致错误。

（3）错误原因

指令实际上在阻塞时送入指令ram中的是下一条要取出的指令的地址，但是由于被阻塞了实际上我们希望的是其保持之前送入的指令，由于该cpu实现上都是非阻塞赋值，所以在进行预取指时并不能得知是否需要阻塞。所以我们选择增加一个32位的寄存器用于存储被阻塞在译码流水级之外的指令。

（4）修正效果

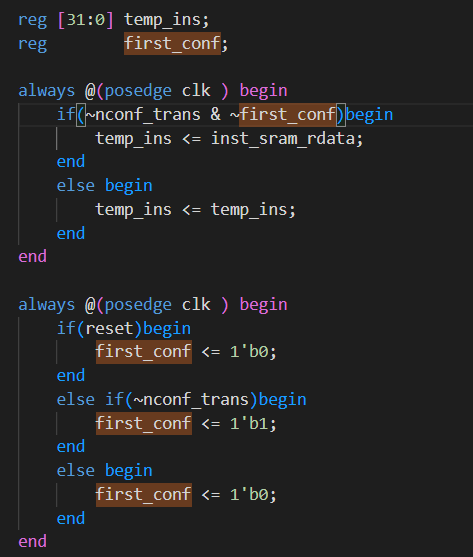




图2.2-2.3 增加的指令寄存器与控制逻辑

修改后可以正常运行，无功能性bug。

四、实验总结（可选）