# 实验2 — 暂停与Forwarding

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名： | 汤尧 | | 学号： | 3200106252 | | | 学院： | 计算机学院 | | |
| 课程名称： | | 计算机系统Ⅱ | | | 同组学生姓名： | 无 | | | |
| 实验时间： | | 周四3，4，5节课 | 实验地点： | | 紫金港机房 | | 指导老师： | | 卢立、申文博 | |

# 实验目的和要求

1. 深入理解流水线冒险，掌握利用stall 处理流水线冒险，保证流水功能正确性的基本技术
2. 完善流水线的基本功能，实现 Forwarding 机制
3. 尝试思考改进的方案

# 实验内容和原理

## 实验内容

* 1. 在 lab1-2 的基础上加入 stall 机制。
  2. 在 lab2-1 的基础上加入 Forwarding 机制。

## 设计模块

2.2.1stall实现分析

以数据冲突为例进行分析，常见的数据相关有 WAW, WAR, RAW ，由于我们的指令都是顺序执行的，这里只考虑 RAW 一种情况即可。当指令流入 ID 阶段时，我们对指令进行译码，根据译码的结果读取对应寄存器的值，此时如果发现待读取的寄存器发生了 RAW 冲突，就要触发 stall 机制进行处理。

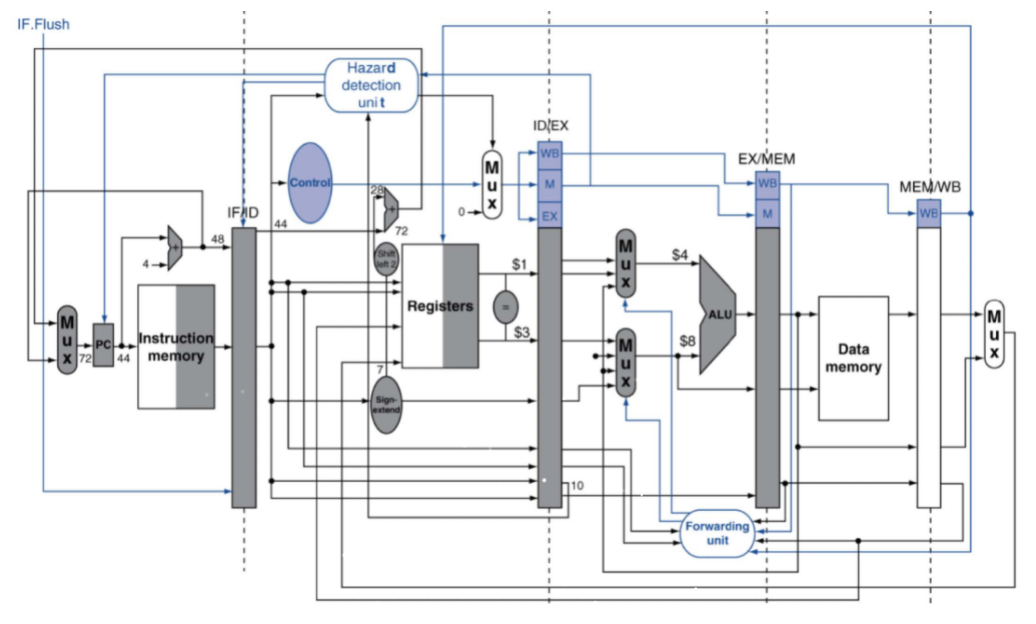
我们不妨假设 ID 段要读取的寄存器设与 EX 段将要写入的寄存器发生了冲突。（与 MEM, WB 段冲突的情况请同学们自行考虑）那么从逻辑上我们的 stall 机制应当使得 ID 段的指令被锁住， 直到寄存器的值可以被正确读取前都不应当向后流入 EX 段，否则读取的寄存器值错误，会影响指令执行的正确性。例如相邻的两条指令为 addi x1, x0, 1; add x2, x1, x0 第二条指令执行时 x1 寄存器的值应该为1，而在流水线中上一条指令还并没有执行完，如果不添加 stall 机制，读取到的值为第一条指令尚未执行时 x1 寄存器的值。

也就是说接下来的一个周期，该指令仍然锁住在 ID 阶段，直到与之冲突的 EX 阶段指令流入到 WB 段完成写回，使得该指令可以读取到正确的寄存器的值时，该指令才可以流入到 EX 段。那么我们以 ID 阶段来看，向前看 IF, ID 两阶段都应当锁住当前的指令和状态，等待若干周期后继续执行，向后看 EX, MEM, WB 阶段应当继续执行段内的指令，正常的让指令流入下一阶。

再考虑分支跳转指令的情况，当执行到分支跳转指令时，由于可能发生跳转，在 pc 置位前已经流入 IF 段的指令可能并不是我们将要执行的指令， stall 机制需要考虑流水线内指令的锁存和刷新两方面的问题，对于分支跳转指令，根据同学们实现的流水线不同，也需要 stall 不同的周期数。

2.2.2 forwarding 介绍

在流水线的运行过程中，数据冲突是拖累流水线效率的一个重要的因素。在顺序单发射处理器中，数据冒险只需要考虑RAW一种情况，本质上就是由读写同一个寄存器所导致的，所以数据冒险本身的检测只需对流水线中相应流水段中指令的寄存器编号进行比较，即可对数据冒险的发生进行判断。



实现 Forwarding 的基本功能需要在流水线的基础上添加冲突检测逻辑和相应的数据通路，这样就能够在WB阶段之前，将最新的寄存器的数据回传给后续的指令，减少 stall 的周期数。

# 主要仪器设备

• HDL：Verilog、SystemVerilog

• IDE：Vivado

• 开发板：NEXYS A7

# 操作方法与实验步骤

## 操作方法

完成Stall机制和Forwarding机制设计，进行仿真测试检验CPU基本功能。

## 实验步骤

1. 使用开发工具建立工程，推荐在 Vivado 2019.2 以上版本完成。
2. 实现 stall 机制和Fowarding机制。
3. 进行仿真测试，以检验 CPU 基本功能。
4. 进行上板测试，以检验 CPU 设计规范，上板测试调试工具相关内容请参考lab0。

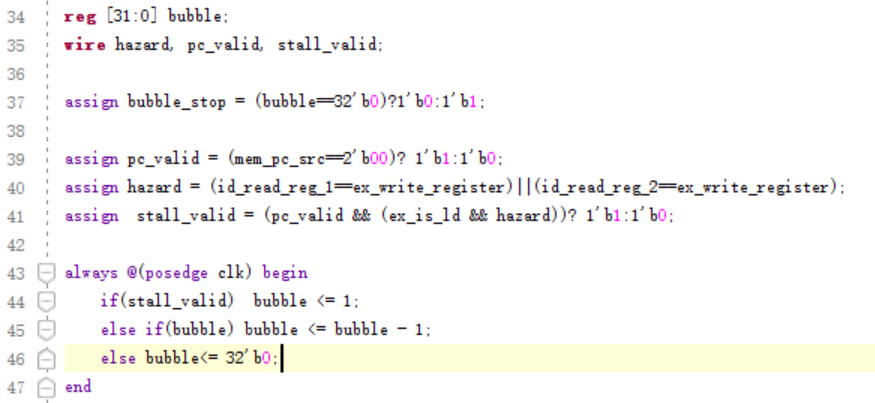
# 实验结果与分析

5.1 主要代码片段

Stall模块

需要判断以下几点：

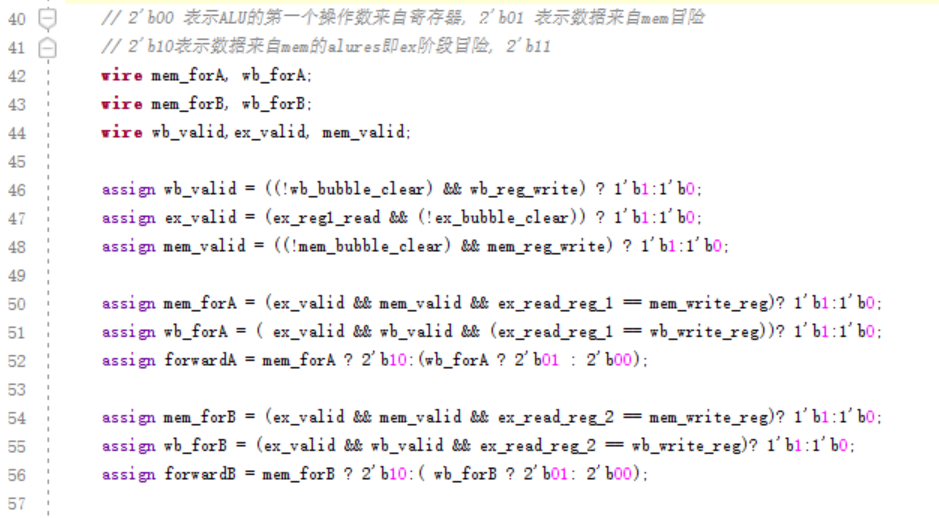
1. 是否存在hazard冲突。
2. 是否为id指令，因为只有ld指令的hazard冲突通过前递无法解决
3. 是否会因为跳转预测失败。在实际代码中，如果ld指令在跳转指令后，跳转指令需要在mem阶段写回，则先满足跳转指令，因为跳转指令如果执行那么ld指令将会被flush清空。（这里未判断指令是否跳转，实际应当判断bne和beq的执行不成功问题）并且插入bubble会干扰跳转指令下一条pc的地址。



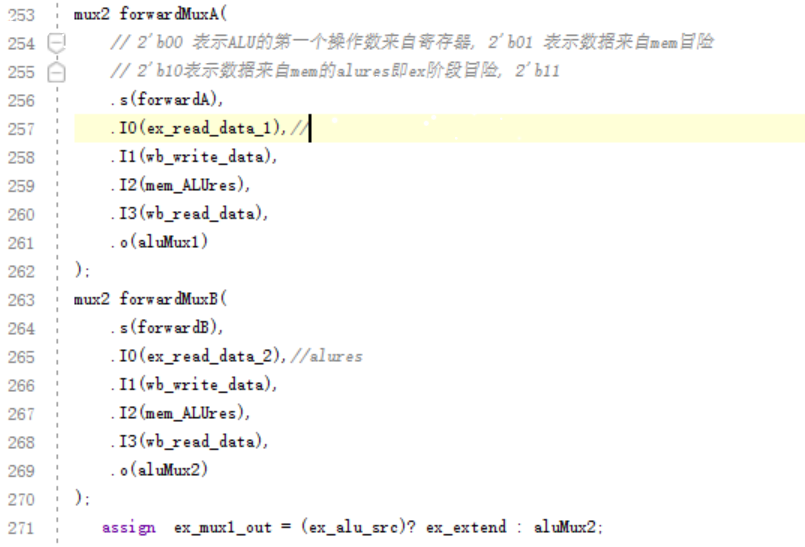
Forwarding模块

需要满足以下几点：

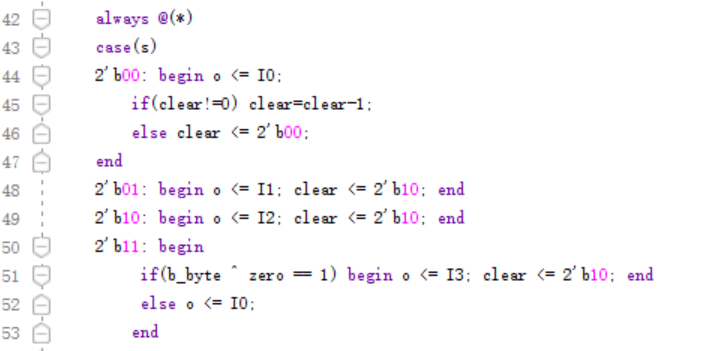
1. 产生数据冲突，即mem/wb段要写入的寄存器和ex段需要使用的寄存器（id阶段读取的寄存器）相同。
2. wb/mem阶段是否需要为插入bubble后的无效指令，若为bubble则不执行。



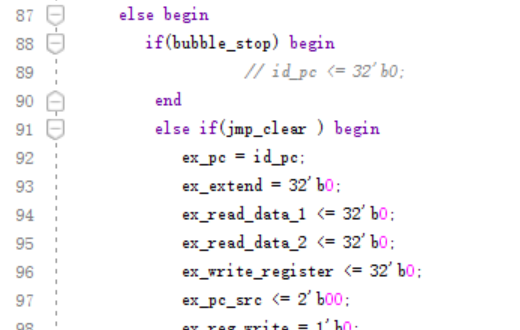
SCPU中forwarding选择器模块



PCmux中如果跳转则发送清空ex，mem寄存器中信号值。

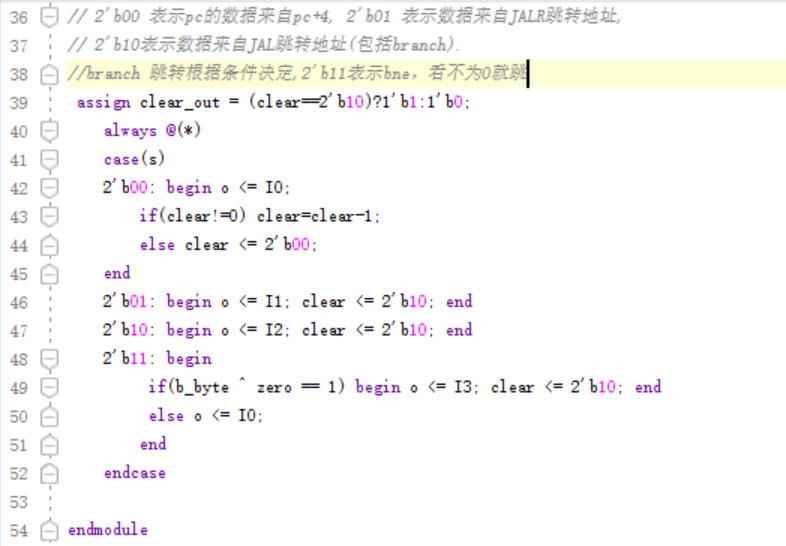


状态寄存器收到bubble信号时什么都不做，收到clear信号时清空信号。

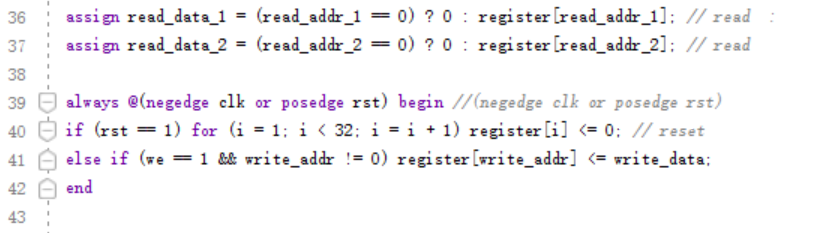


跳转模块

pc控制模块，遇到跳转指令时先预测不跳转，pc顺序读之后的指令。如果跳转指令成立，需要跳转，则发送清空指令，将后续信号全部清空。

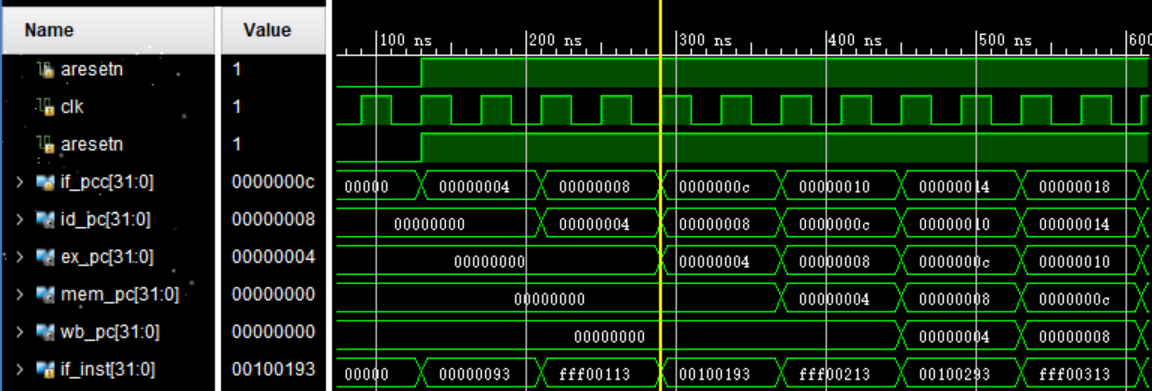


Register模块采用double dump，上升沿写入下降沿读出，这样就不需要解决wb冲突。

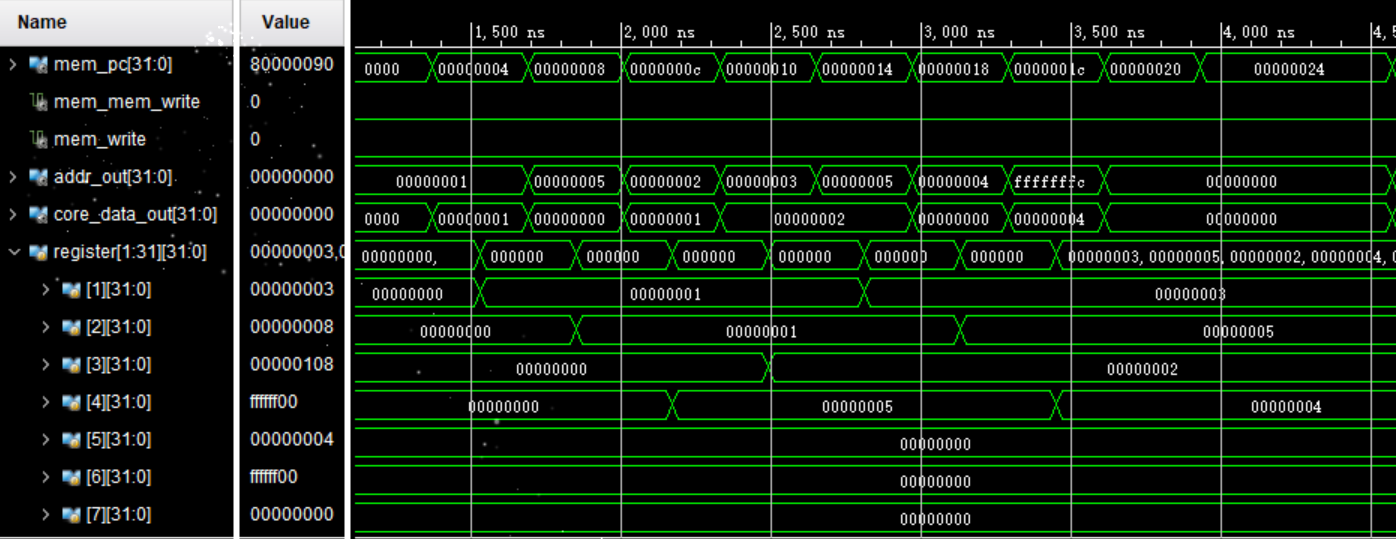


5.2 仿真模拟

不同模块的pc随时钟周期依次传递。



使用测试代码，寄存器变化和预想结果一致。



# 思考题

1-1 Stall 暂停

1. 请你对数据冲突情况进行分析归纳，试着将他们分类列出。
   1. 上一条指令要修改寄存器，下一条指令要读取寄存器，下一条指令读取时上一条指令还未修改成功。
   2. 上一条指令需要从内存中读取数据放入寄存器，下一条指令需要读取该寄存器的值，此时上一条指令还未读出。
2. 如果 EX, MEM, WB 段中不止一个段的写寄存器与 ID 段的读寄存器发生了冲突，该如何处理？

forward是有优先级的，先就近判断mem段是否需要写寄存器寄存器，再判断wb段。因为有可能在wb段写回的寄存器在mem段继续被改写。

* 1. Forwarding 前递

1. 在引入 Forwarding 机制后，是否意味着 stall 机制就不再需要了？为什么？

不是。数据前递不能解决的冲突：load指令，必须通过stall机制插入一个bubble，在mem阶段后半周期读出之后，进入wb阶段，通过数据前递，将要写回的值给ex阶段。

1. 你认为 Forwarding 机制在实际的电路设计中是否存在一定的弊端？如果存在，请给出你的理由。

不存在。