

实验 5-3 报告

学号：2016K8009908007

姓名：薛峰

一、实验任务

实验目的：

完成随机延迟的功能测试。

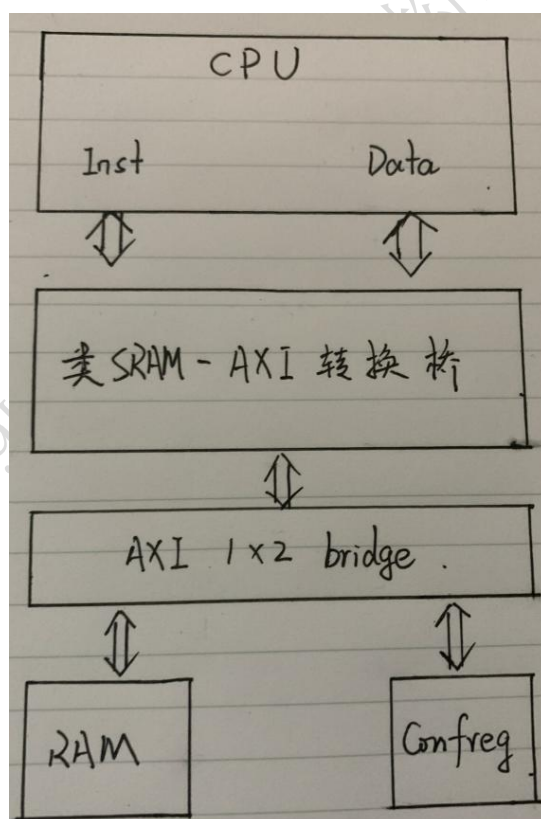
检验方法：

仿真和上板运行测试程序通过。

二、实验设计

因为在 Lab5-2 中已经支持随机延迟，因此该部分与 Lab5-2 相同。

CPU 通过 AXI 转换桥，将类 SRAM 接口转换为 AXI 总线接口，其总体的框架如下：



该项目主要修改的部分在指令和数据接口的转变，即 IF 阶段和 MEM 阶段和 WB 阶段。

对于取指，需要增加几个状态寄存器：`inst_addr_arrived`，`inst_data_arrived`。其中 `inst_addr_arrived` 用来表示取指的地址已经被转换桥接收，即已经接收到了 `inst_addr_ok`，当新的 PC 到来之后，该寄存器清零；对于状态寄存器 `inst_data_arrived`，其用来表示指令是否到来，当新的 PC 到来之后，该寄存器清零。

这两个寄存器用来控制 IF 状态的操作。当 `inst_addr_arrived` 为 0 时，表明地址还没有被接收，因此 `inst_req` 拉高，请求指令，当 `inst_addr_arrived` 为 1 时，表明地址已经被接收，因此 `inst_req` 为低。当 `inst_data_arrived` 为 1 或者 `inst_data_ok` 为 1 时，则 `IF_ready_go` 为 1，表明 IF 阶段的工作已经完成，可以进入到 ID 阶段。其中需要注意的一点是，此时有可能 ID 阶段被阻塞，因此读来的指令就会丢失，因此，需要一个寄存器来保存 `inst_rdata`，并用一个状态寄存器表示信号是否丢失，如果丢失则从寄存器中取值，否则直接读取 `inst_rdata`。

对于数据的存取，将发请求和相应等待分别分配给 MEM 阶段和 WB 阶段，即 MEM 阶段发出读/写请求，一旦 `data_addr_ok` 为 1，则进入 WB 阶段，因为 WB 阶段不会被阻塞，因此不需要状态寄存器；在 WB 阶段等在数据相应，即等待 `data_data_ok` 为 1，则表明写成功/读数据传来，此时指令才可以退出流水线。

三、实验过程

（一）实验流水账

时间	记录
12 月 16 日	一、将例外处理部分的代码模块化；
17: 30~18: 10	二、在新环境中仿真通过，上板验证通过。

（二）错误记录

（因为在 Lab5-2 中已经支持随机延迟，因此该部分仍然记录调试 Lab5-2 过程中遇到的问题）

1、错误 1

（1）错误现象

仿真时，波形不停止，并且打印出来的 PC 都是 X。

（2）分析定位过程

根据打印出来的结果判定，肯定是最开始取值的逻辑有问题。

（3）错误原因

之前 `next_PC` 是根据 `PC_ID+4` 得到，然而初始化中并没有初始化 `PC_ID`，因此，`next_PC` 为 x，导致取值的地址不对，从而 `inst_addr_ok` 一直不拉高，因此发生阻塞。

（4）修正效果

将 `PC_ID` 进行初始化从而解决此问题。

（5）归纳总结

该错误由没考虑好时序关系导致。

2、错误 2

（1）错误现象

仿真时，控制台报错。

（2）分析定位过程

根据 `test.s`，发现程序跑到了 `inst_error` 中，并且是通过一条 `bne` 指令跳转到 `inst_error` 中。这条 `bne` 指令存在数据相关，并且其需要查看的寄存器是上一条指令要 `lw` 的指令，因此需要阻塞，等到 `lw` 指令执行到 WB 阶段取到数据后，才能执行 `bne` 指令。

(3) 错误原因

对于之前的 CPU 来说，从 data_ram 取的值一拍就能返回，因此一条指令执行到 WB 阶段就能够取得有效的值。而对于本次实验来说，data_ram 取值的延迟不是一拍，因此这条 bne 指令需要阻塞到取到指令的时刻。

(4) 修正效果

修改 load_to_use 阻塞的逻辑，增加已经达到 WB 阶段，但没有取到值的情况。

(6) 归纳总结

该错误由没考虑完全数据相关的情况导致。

3、错误 3

(1) 错误现象

仿真时，跑到检测延迟槽的程序时报错。

(2) 分析定位过程

根据 test.s 发现是一条跳转指令后的延迟槽指令出错，观察波形发现这条延迟槽指令并没有进入到流水线当中，即当前的 CPU 不支持延迟槽。

(3) 错误原因

之前 next_PC 是根据 PC_ID+4 得到，也就是说每条指令完之后才判断 next_PC，因此每次都能跳转到正确的地址，而延迟槽之后的一条指令则进不到流水线当中。

(4) 修正效果

将对 next_PC 的赋值从 PC_ID+4 改为 PC_IF+4。

(7) 归纳总结

该错误由没考虑好时序关系导致。

4、错误 4

(1) 错误现象

仿真时，跑到检测 swr 指令的程序时报错。

(2) 分析定位过程

从控制台打印出来的信息，发现是一条 lw 指令取值出错，然而检查这条指令的逻辑之后发现，并没有错误。于是推测应该是之前向这个地址写数据出错了。根据这个地址的值向前搜索，发现是一条 swr 指令出错。

(3) 错误原因

之前的 AXI 接口中关于对 wstrb 的赋值是按照讲义的方式进行的。然而其中写的字节数仅为 1、2 或 4。然而 swr 和 swl 指令可能写三个字节，因此需要对 AXI 转换桥这一部分逻辑进行修改。

(4) 修正效果

将 data_size 扩充一位，用最高位表示 swl 指令，并根据 data_size 和 data_addr 的低两位对 wstrb 进行赋值。修改过后这个功能点通过。

(8) 归纳总结

该错误由没考虑到 swr 和 swl 两条指令导致。

5、错误 5

(1) 错误现象

仿真时，跑到检测 syscall 的程序时报错。

(2) 分析定位过程

根据 test.s 发现，执行到了一条 syscall 指令之后没有跳转到 bfc00380。

(3) 错误原因

查看波形后发现，当这条 syscall 指令执行到 EX 阶段时，发出例外提交，此时虽然 ID 阶段的指令检测到了例外提交，但是此时 ID 阶段的指令是无效的，并且例外提交只持续一个周期，因此例外提交信号被丢失了。

(4) 修正效果

增加 next_pc_miss 机制，用于表示对 next_pc_miss 的赋值是否有丢失。例如，当检测到例外提交时，就将 next_miss 信号拉高，用于表示丢失了一个对 next_pc 的赋值，并将此时的 next_pc 的值存入到 miss_PC 当中；因此对 next_PC 赋值时，首先判断 next_miss 是否为 1，若为，则将 miss_PC 的值赋给 next_PC，从而解决了信号丢失的问题。修改过之后，验证点通过。

(9) 归纳总结

该错误由没考虑信号丢失这一问题导致。

6、错误 6

(1) 错误现象

仿真时，跑到检测地址错例外的程序时报错。

(2) 分析定位过程

根据打印出来的信息，一个没有四字节对齐的 PC，向寄存器堆写了一个数据，因此判断是没有清空流水线造成的。

(3) 错误原因

根据 test.s 发现，要跳转的 PC 没有 4 字节对齐，但是仍然去到了指令，并且译码的时候判断 rf_wen 不为 0。因此当这条指令流到 WB 阶段时，会写寄存器堆。

(4) 修正效果

当检测到例外提交信号时，就将 ID 阶段的指令清空。修改过之后，所有测试点仿真通过。

(10) 归纳总结

由于对 ID_instruction 的赋值由 wire 信号改为了 reg 信号，因此这一部分的赋值也应该进行修改，但是之前没有考虑到这一点，所以会出错。

四、实验总结

无。