**Lab 7报告**

学号：2020K8009929034、2020K8009929019、2020K8009929037

姓名：胡康、李子恒、吕星宇

箱子号：05

一、实验任务（10%）

为设计的CPU增加AXI总线接口，更好地实现CPU与系统中的内存、外设进行交互。整体任务分为三个阶段：

1. 将原有CPU访问SRAM的接口调整为类SRAM总线接口，在原有基础上增加握手信号
2. TODO
3. TODO

二、实验设计（40%）

（一）总体设计思路

1. 设计类SRAM总线：增加CPU的端口信号，利用req、addr\_ok、data\_ok等信号实现握手机制（具体端口信号的增加见重要模块1）。同时。为了保证流水线的正常运作，更改部分ready\_go、allowin信号；最后考虑异常清空流水线、转移计算未完成等特殊情况。

2. TODO

（二）重要模块1设计：类SRAM总线接口信号

1. 工作原理

作为类SRAM总线接口信号，实现握手机制，与内存、外设进行交互。由于内存可以抽象为inst\_sram与data\_sram，故接口信号也分为inst信号与data信号。两组信号的接口定义见表1，由于整体功能较为相似，故在表1中一并展示。

CPU与内存的交互主要分为三种：取指、load类访存、store类访存。对于取指操作，CPU发出读请求（inst\_sram\_req），内存接收请求并返回接收成功信号（inst\_sram\_addr\_ok）与指令（inst\_sram\_rdata）；对于load类访存，CPU发出读请求（data\_sram\_req）并用字节数（data\_sram\_size）表示请求的字节数，内存接收请求并返回接收成功信号（data\_sram\_addr\_ok）与指令（data\_sram\_rdata）；对于store类指令，CPU发出写请求（data\_sram\_req）以及字节数（data\_sram\_size）、字节写使能（data\_sram\_wstrb）、写数据（data\_sram\_wdata），内存接收请求并返回成功信号（data\_sram\_addr\_ok）表示写入完成。

1. 接口定义

| **名称** | **位宽** | **方向** | **功能描述** |
| --- | --- | --- | --- |
| clk | 1 | input | 时钟信号 |
| req | 1 | output | 请求信号 |
| wr | 1 | output | 1表示写请求，0表示读请求 |
| size | 2 | output | 传输的字节数 |
| addr | 32 | output | 请求的地址 |
| wstrb | 4 | output | 写请求的字节写使能 |
| wdata | 32 | output | 写请求的写数据 |
| addr\_ok | 1 | input | 该次请求的地址传输ok |
| data\_ok | 1 | input | 该次请求的数据传输ok |
| rdata | 32 | input | 该次请求返回的读数据 |

1. 功能描述

在SRAM的基础上增加了握手机制，更加贴近真实情况。

1. 重要模块2设计：
2. 工作原理

2、功能描述

3、

三、实验过程（50%）

（一）实验流水账

2022.10.27 19：00——23：00 阅读讲义8.1、8.2；

2022.10.28 20：00——24：00 exp14 debug

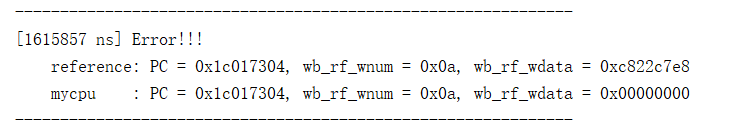
2022.10.29 19：30——23：24 完成exp14仿真及上板

2022.10.30 8：00——10：30 完成exp14的实验报告

（二）错误记录

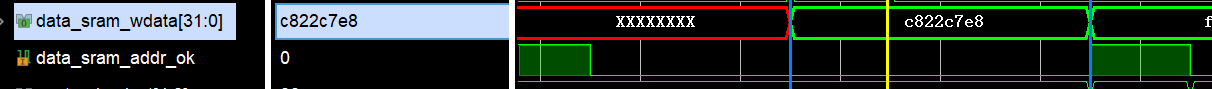
1、错误1：EXE阶段未控制addr\_ok有效时再进入下一阶段

（1）错误现象

如图x，某指令写回数据发生错误

（2）分析定位过程

查询汇编文件可以看到这是一条load指令，且波形图显示在WB阶段拿到数据时就已经出错，于是推测可能是之前的store指令存入该内存的数据错误。从该时刻向前找写入该地址的store指令，找到了不久前刚向该地址写入数据的某store指令。检查该指令EXE阶段与MEM阶段的访存行为并重点锁定data\_sram\_wdata与data\_sram\_addr\_ok两个信号后可以发现，该数据有效期间data\_sram\_addr\_ok始终为0（如图x），说明正确的数据没有写入。



（3）错误原因

未控制该store指令在EXE阶段看到addr\_ok为1才能进入MEM阶段，导致数据没有正确写入，进而导致后续load指令读出错误数据。

（4）修正效果

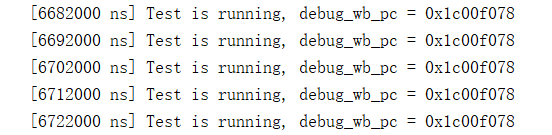
按照讲义P195的要点讲解，修改es\_ready\_go使得访存指令的req与addr均为1后才能进入下一级

（5）归纳总结

由于本阶段增加了握手机制，很多原本一拍内一定能完成的操作都变为未知，因此要让流水线在某些情况下“暂停”，而实现的方式便是通过ready\_go信号。

2、错误2：清空流水线时没有修改fs\_valid为0

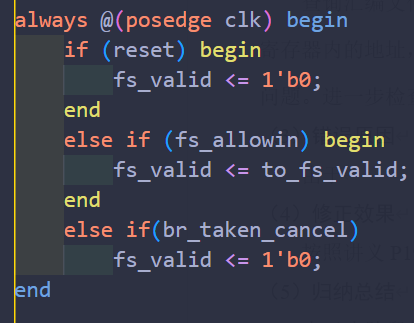
（1）错误现象

仿真跑到660000ns后在0x1c00f078处进入死循环（如图x）

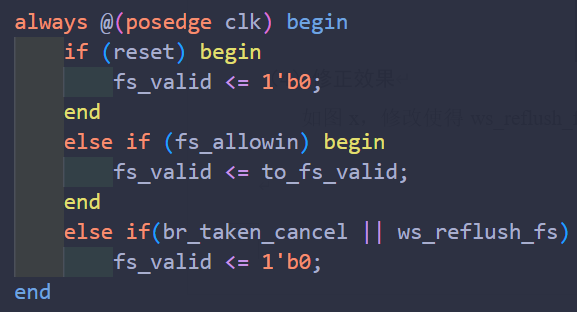
（2）分析定位过程

查询汇编文件可以看到0x1c00f078处指令是一条ertn指令，波形图显示该指令执行后，next\_pc取到了ERA寄存器内的地址，但该地址无法进入IF级，导致流水线“断流”。这说明错误原因是pre-IF级与IF级的交互存在问题。进一步检查相关信号后，发现是由于fs\_valid始终为1，导致该地址无法进入IF级。

（3）错误原因

图x展示了原有的fs\_valid的赋值逻辑。可以看到，当清空流水线（ws\_reflush\_fs为1）时，三个条件都不符合。若此时fs\_valid为1，那么接下来fs\_valid将一直为1，导致next\_pc的地址永远无法进入IF级

（4）修正效果

如图x，修改使得ws\_reflush\_fs为1时，fs\_valid变为0。这样便可以让ERA寄存器内的地址及时进入IF级，使流水线重新流起来。

（5）归纳总结

在exp13的设计中，由于IF阶段一定可以拿到ERA寄存器的地址，因此ertn指令清空流水线的操作可以只将其余阶段valid改为0，而保留fs\_valid为1；但在exp14中，由于握手机制的存在，next\_pc的地址不确定何时能进入IF阶段，因此需要将fs\_valid也及时改为0。可见随着后续功能的增加，之前实验的设计也要考虑是否相应地改变。

四、实验总结（可选）