# Lab3

## 实验目的

1. 根据给出的代码实现一个能处理数据相关的RISCV的cpu core
2. 自己设计数据通路，实现CSR

## 实验环境和工具

Windows10下使用vivado的仿真工具

## 实验内容与过程

阶段一：该阶段主要完善除Hazard以外的所有模块。完善的顺序为流水线上的先后顺序

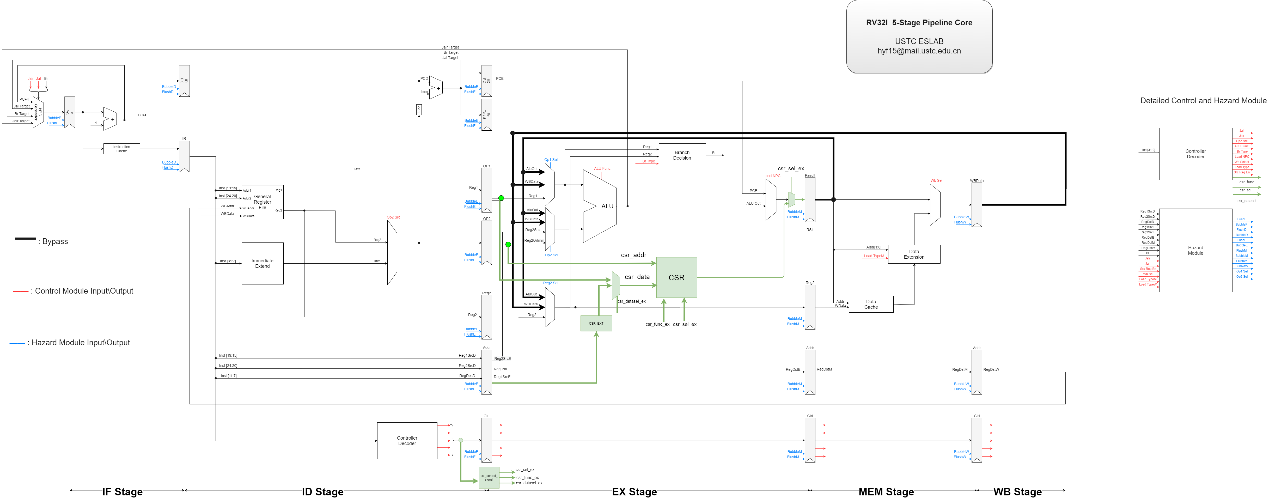
1. NPC 模块： 选择下一条指令的pc。
2. Control模块：根据当前指令发出各种控制信号。
3. Imm\_extend模块：根据控制信号处理立即数
4. ALU模块 ：根据控制信号对输入的两个数据做运算
5. BranchDecision模块： 根据控制信号与输入的两个数据决定是否跳转。
6. DataExtend模块： 根据控制信号处理从cache中读出的数据

阶段二：该阶段主要完成Hazard模块

1. 开始时初始化为五个阶段全部bubble
2. Rst时五个阶段全部flush
3. 当遇到Jalr或Br跳转时，flush ID段和EX段
4. 当遇到Jal时，flush ID段
5. 当上一条指令为load且其目的寄存器为当前指令的源寄存器，此时出现RAW相关，需要bubble IF段和ID段，flush EX段
6. 当MEM段结果和WB结果为EX段的源寄存器时，设置op1\_sel和op2\_sel用于转发。

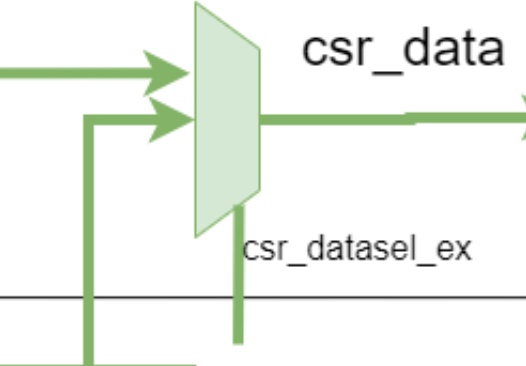
阶段三：该阶段主要是设计数据通路，实现CSR

数据通路设计如下：

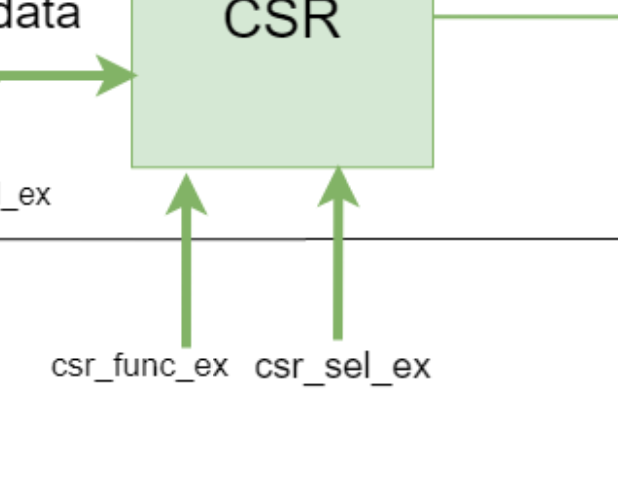
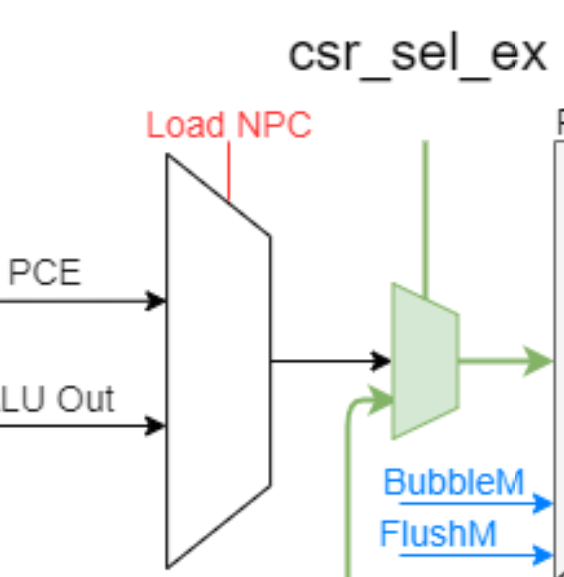


Control里新增三个有关CSR的控制信号：

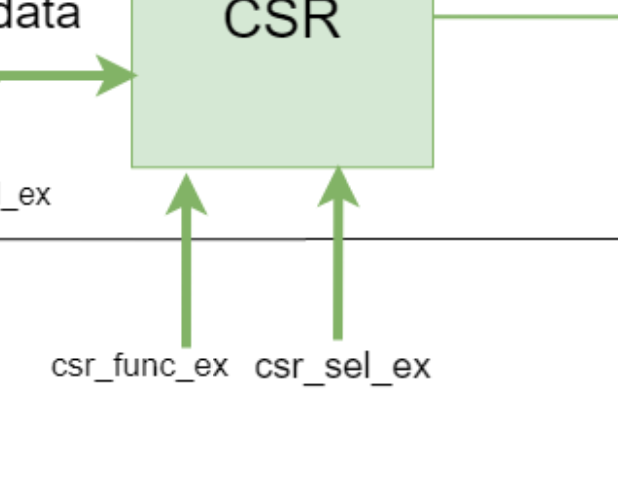
csr\_datasel\_ex:选择输入CSR的数据为reg1还是立即数



csr\_sel\_ex:选择EX段的结果为CSR寄存器还是LU\_OUT/PCE，在CSR中控制寄存器可写信号。



csr\_func: CSR指令的类型，用于控制CSR单元内的计算



CSR寄存器内部采用同步写，异步读。

## 实验总结

下面是实验中碰到的几个bug：

1. 在控制器中对op2\_src的赋值做了两次，导致仿真时出现冲突，大部分时间显示为x
2. 在控制器中branch部分的immtype赋值使用了阻塞赋值（只有这一处笔误），导致某个测试用例immtype赋值错误
3. Jalr和Br的优先级要高于Jal
4. DataExtend中一开始使用的是addr \* offset，LH的地址不一定是2的倍数，实际测试时出现错误，将其地址设为2的倍数时正确。
5. Branch Decision中BGE对应的是小于等于而不是小于
6. Hazard中reg2\_sel与op2\_sel不同，不需要判断alu\_src2
7. CSR中开始有时序问题，需要用一个寄存器保存开始时对应寄存器的值，否则输出的可能为修改后的值。

这个实验我花的时间大约在30个小时左右，其中大概20~24个小时阶段一，阶段二，大约6~7个小时用于CSR。

在总共30个小时的时间中，我花在阅读代码，编写代码，设计CSR的时间大概为6~8个小时，剩下二十多个小时的时间全部用来debug。上述bug中 1. 用了大约8个小时，2. 用了大约6个小时，3 4 5 6 7花的时间都小于一个小时，CSR时序问题花了4个小时左右。

这里debug的难点在于与错误相关的信号非常多，debug的时候需要看很多信号,比较繁琐。而且有些bug与逻辑无关，非常难找。我花的时间最长的两个bug都与代码逻辑无关，只能靠一遍遍阅读代码来找。

## 意见

希望能设置一个共享文档，每个人把自己遇到的bug写进去，可以节省时间。