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

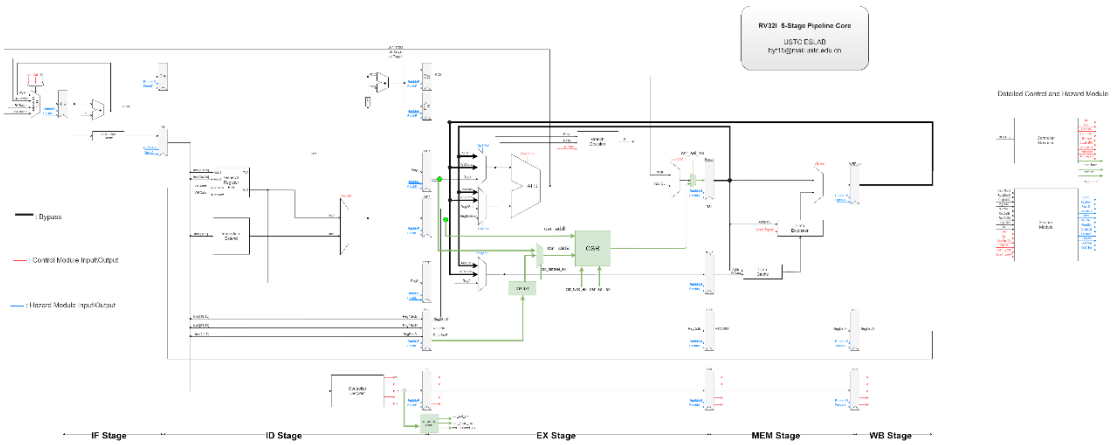
1. NPC 模块： 选择下一条指令的 pc。
2. Control 模块：根据当前指令发出各种控制信号。
3. Imm_extend 模块：根据控制信号处理立即数
4. ALU 模块 ：根据控制信号对输入的两个数据做运算
5. BranchDecision 模块： 根据控制信号与输入的两个数据决定是否跳转。
6. DataExtend 模块： 根据控制信号处理从 cache 中读出的数据

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。

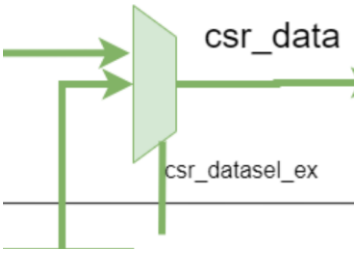
阶段三：该阶段主要是设计数据通路，实现 CSR

数据通路设计如下：

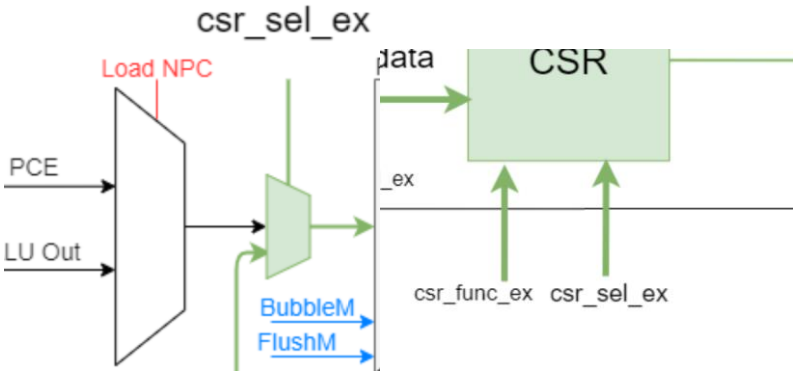


Control 里新增三个有关 CSR 的控制信号：

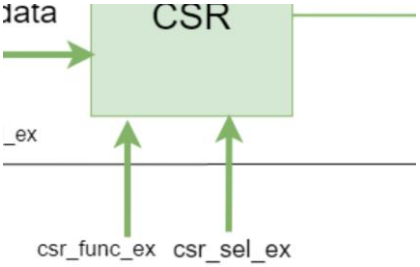
csr_datasel_ex:选择输入 CSR 的数据为 reg1 还是立即数



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。