Lab2实验报告

- 焦培淇　PB17151767

实验目标

了解和熟悉流水线 CPU 的结构，以及内部数据、结构冲突的原因和处理方法，完善给定的 CPU 代码，实 现 SLLI、 SRLI、 SRAI、 ADD、 SUB、 SLL、 SLT、 SLTU、 XOR、 SRL、 SRA、 OR、 AND、 ADDI、 SLTI、 SLTIU、 XORI、 ORI、 ANDI、 LUI、 AUIPC、 JALR、 LB、 LH、 LW、 LBU、 LHU、 SB、 SH、 SW、 BEQ、 BNE、 BLT、 BLTU、 BGE、 BGEU、 JAL、 CSRRW、 CSRRS、 CSRRC、 CSRRWI、 CSRRSI、 CSRRCI 等指令。

实验环境与工具

WIN10 + Vivado 2018.2 + Visual Studio Code + ubuntu16.04

实验内容和过程

基础指令实现部分

NPCGenerator部分

添加always语句根据是否有br,jalr,jal等跳转信号，分别对NPC赋予不同的值，无特殊信号时，NPC赋为PC。

ControllerDecoder.v部分

1. 首先对于输入的inst字段进行解码，划分为三部分：opcode,func_3_bits,func_7_bits用以确定控制信号。
2. 之后对于确定起来比较简单的信号，直接采用assign语句进行简单的逻辑判断后确定其值。
3. 然后在opcode==1100011时单独处理br信号，根据func3字段确定br_type信号。
4. 最后根据opcode进行分类对于不同的指令族，分配对应的控制信号

Imm_EX.v部分

该部分为新增模块，因为在hazard里面可能需要使用最新的EX阶段的立即数，因此在此处对立即数进行存储，以便转发的时候能够使用。同时需要注意，在RV32ICore.v里面需要添加新的多路选择器，增加从该段寄存器进入ALU的数据通路。

ALU.v部分

逻辑运算单元部分实现相应的逻辑运算功能，包含AND， SLTU， ADD， SUB， XOR， OR， SLT， OP1， SLL， SRA， LUI一系列功能。

BranchDecision.v部分

通过输入的br类型，判断两个寄存器值的比较方式，然后对br信号进行赋值决定是否进行跳转。

DataExtend.v部分

根据输入的load_type对立即数进行拓展。

Hazard.v部分

1. 首先根据指令的类型决定是否对流水线进行flush和bubble；具体情况为：对于br和jalr指令，由于其在ex阶段得到有效地址，因此需要对ID和EX阶段的段寄存器进行flush；jal指令在id阶段得到有效地址，因此只进行ID阶段的flush操作；而对于LW类指令，如果其目的寄存器是执行阶段的指令的源寄存器，则需要对流水线进行bubble同时，flush EX阶段的段寄存器，将数据通过转发送入ALU。

2. 对于转发信号，根据寄存器的写使能，写地址与当前的使用地址是否相同，地址是不是0号寄存器，便可确定相应的转发源，通过赋值转发信号即可实现转发。

CSR指令实现部分

1. 首先对于ALU增加CSR功能，使其直接输出op2。
2. 对imm_extend模块，增加csr类型的立即数，完成对一个五位立即数的0拓展。
3. 在控制单元内，增加对于csr的控制信号三个：csr_read_en,csr_write_en,csr_type，分别控制对CSR寄存器的读使能，写使能和表示csr指令的类型。
4. 增加csr寄存器文件，按照riscv的文档，应该有4096个csr寄存器，此处为了简化，只设置32个也可以完成指令正确性的检验。
5. 将控制单元新产生的CSR控制信号加入EX段寄存器。
6. 将CSR的写入逻辑实现在寄存器文件中，同时增加op2_src多选器的路数，接入csr寄存器的输出数据进入alu。

完成上述模块添加，即可实现CSR指令族

实验总结

1. 工程里面的寄存器文件设置过大会导致不能生成连线图。
2. 仿真的时间要比较长才能完成指令的执行。
3. 对于ALU的op2，由于其可能用到ex阶段的立即数，需要额外添加对应的段寄存器来实现，否则会小概率的出现错误。
4. 对于时间方面，前期完成代码逻辑比较快，大概3小时就能填完所有的代码逻辑，并完成没有hazard模块的正确cpu；但是在添加hazard模块后，由于顶层模块缺少对应的imm段寄存器，导致只能通过第一个测试样例，无法通过第二和第三个，此时检查错误花费时间很长，大概花费了几天进行debug；对于最后的csr指令实现设计，参照lab1时的设计思路，花了大概2小时完成模块添加和设计验证指令；之后大概花费了5小时进行debug；至此完成实验。
5. 实验收获方面，最重要的在于通过实验加深了对于流水线的印象，能够结合课上的知识，加深自己对于cpu流水线的理解。同时复习了verilog的语法，很好的锻炼了自己的debug能力。

改进意见

我认为需要完善实验指导里面的设计图，感觉有些代码里面的信号根本和设计图的信号对不上，整个代码构建的结构和设计图表示的也不是很一样，这样容易缺少一些对于整个cpu结构的宏观的把握，最终导致debug的时候，很难找到下手的位置。（可能是代码里面的连线命名和设计图不一样导致我没太看明白）