Verilog 实验 lab2 实验报告

PB19071405 王昊元

2022 年 04 月 11 日

1 实验目的

1. 掌握五级流水线 CPU 的设计方法
2. 熟悉 RISC-V 的指令集和数据通路，了解其设计背后的思想
3. 进一步提高使用 verilog 编码和调试的能力，学会使用仿真（Simulation）调试

2 实验要求

1. 实现可以处理基本指令的五级流水线，需实现指令包括：SLLI、SRLI、SRAI、ADD、SUB、SLL、SLT、SLTU、XOR、SRL、SRA、OR、AND、ADDI、SLTI、SLTIU、XORI、ORI、ANDI、LUI、AUIPC
2. 实现可以处理数据相关的五级流水线，需实现指令包括：JALR、LB、LH、LW、LBU、LHU、SB、SH、SW、BEQ、BNE、BLT、BLTU、BGE、BGEU、JAL
3. 实现 CSR 数据通路及相关指令，指令包括：CSRRW、CSRRS、CSRRC、CSRRWI、CSRRSI、CSRRCI

3 实验核心实现

1. 第一阶段：实现基本指令，不考虑数据相关
   这一部分主要是补全各个模块，使得流水线能够实现基本指令，各个模块的功能也都很独立，按照其对应功能实现即可。
   需要提及的有 ALU 中的 `NAND` 计算，我在完成第一阶段时，没有认真做研究，将该运算实现为与非，但在完成第三阶段 CSR 的相关实现时，我才发现 `NAND` 的目的在于清除某些位（对应 `CSRRC` 和 `CSRRCI` 指令），也就是需要实现为 `ALU_out = ~op1 & op2;`。
   控制模块也很重要，不过由于在第一阶段不需要考虑数据相关和 CSR 相关指令，所以这一部分的控制模块只是根据指令的性质（如是否写寄存器、ALU 的数据来源、立即数的类型等）完成这一部分指令的控制。
2. 第二阶段：实现跳转指令，考虑数据相关
   这一部分主要需要完成三个部分，`BranchDecision`、`DataExtend` 和相关的控制信号。
   Br 指令只需要在模块中根据不同的指令进行判断是否跳转，输出信号即可；数据扩展模块只需要根

据不同的指令对数据进行不同的扩展即可。控制模块中，`load_type` 要根据不同的 LOAD 指令输出不同的控制信号，`br_type` 要根据不同的 Branch 指令输出不同的信号，`jal` 和 `jalr` 信号则根据指令是否为 JAL 和 JALR 判断。

此外需要注意的点有 JAL 指令为 J 类立即数，而 JALR 指令为 I 类立即数；JAL 指令和 JALR 指令都需要将 `PC+4` 写回寄存器，故 `reg_write_en` 信号为 1。

3. 第三阶段：实现 CSR 数据通路及相关指令

   完成这一部分首先需要对 CSR 的六条指令有所了解，才能正确实现指令、数据通路及控制信号。

表 1: CSR 相关指令

指令	操作内容	ALU 操作类型	数据来源
CSRRW	Atomic Read/Write CSR	OP1（将 OP1 往下一阶段传递，最后写 CSR）	寄存器数据
CSRRWI	Atomic Read/Write CSR	OP1（将 OP1 往下一阶段传递，最后写 CSR）	立即数
CSRRS	Atomic Read and Set Bits in CSR	OR（实现置位操作）	寄存器数据
CSRRSI	Atomic Read and Set Bits in CSR	OR（实现置位操作）	立即数
CSRRC	Atomic Read and Clear Bits in CSR	NAND（实现清除操作）	寄存器数据
CSRRCI	Atomic Read and Clear Bits in CSR	NAND（实现清除操作）	立即数

从表1可知各个指令的控制信号。

另外值得一提的是，因为 CSR 指令是根据指令的前 12 位（即 `inst[31:20]`）来寻址，故 CSR 寄存器设置 4096 个即可。

4 实验结果及分析

各个测试程序的仿真结果如下：

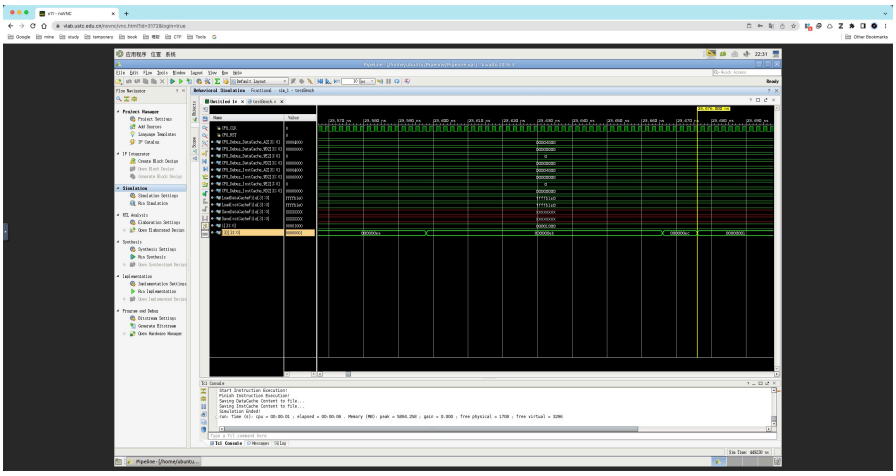

图 1: 测试 1 结果

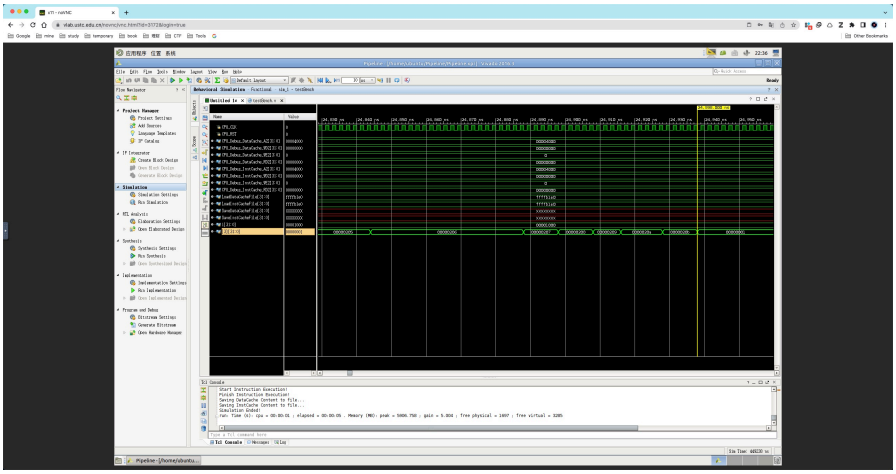

图 2: 测试 2 结果

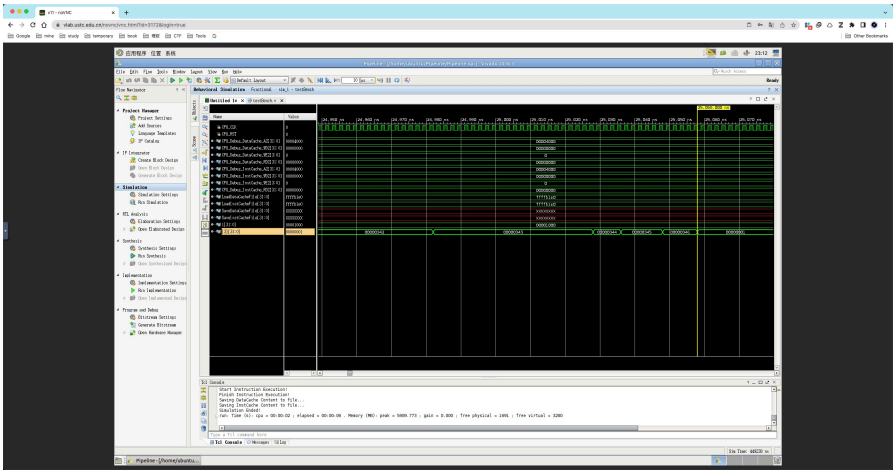

图 3: 测试 3 结果

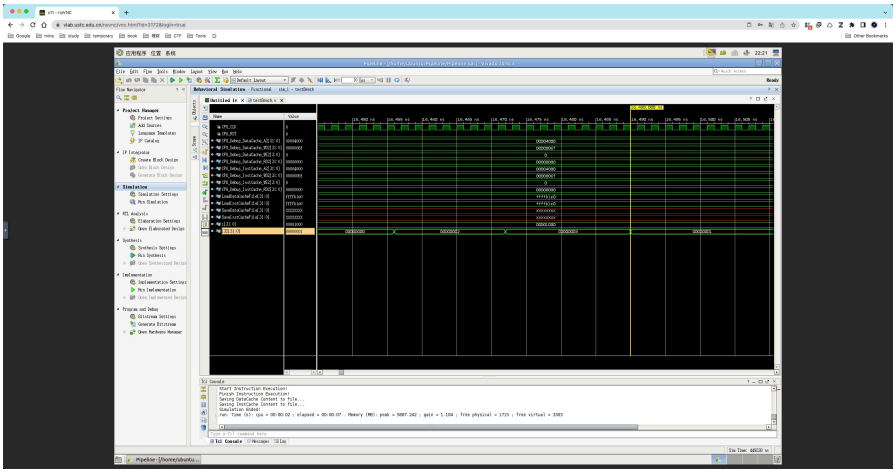

图 4: CSR 测试结果

从以上各图可以看到，3 号寄存器最后都变为 1，这意味着流水线通过了各项测试，完成了实验的各项要求。

5 实验总结

1. 通过本次实验，我对 RISCV 各种指令的熟悉程度与理解更加深刻，同时也对流水线的细节也更加了解，比如流水线的基本模式、旁路转发、数据冲突导致的流水线停顿等等。
2. 此外，我更加熟悉了 Vivado 的仿真功能，对其也有了更深入的了解，也开始尝试使用终端输出来使得 Debug 更加方便。
3. 在完成实验的过程中，我还回顾熟悉了 Verilog 语言的使用与编写，同时也可以利用 Vivado 的仿真结果进行错误分析，逐步确定 Bug 的位置，从而修复问题，完成实验。