

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机体系结构 |
| 姓 名： | 刘思锐 |
| 学 院： | 计算机科学与技术学院 |
| 系： |  |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3200102708 |
| 指导教师： | 陈文智 |

2022年 10月 10日

**浙江大学实验报告**

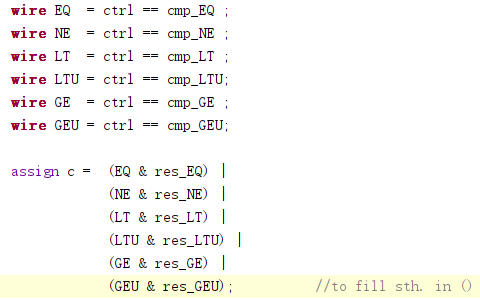
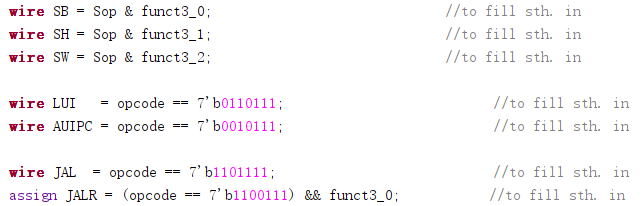
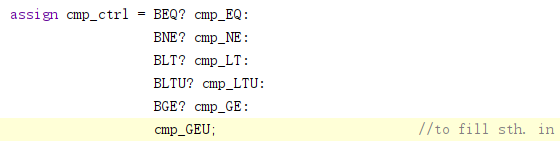
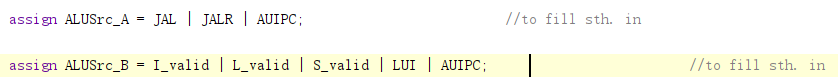
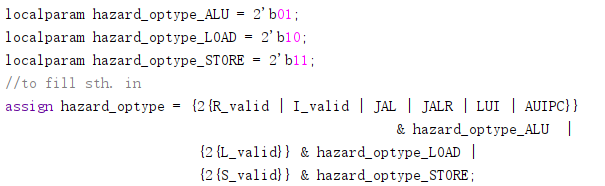
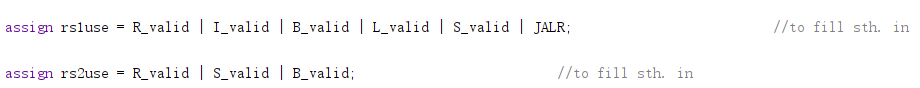
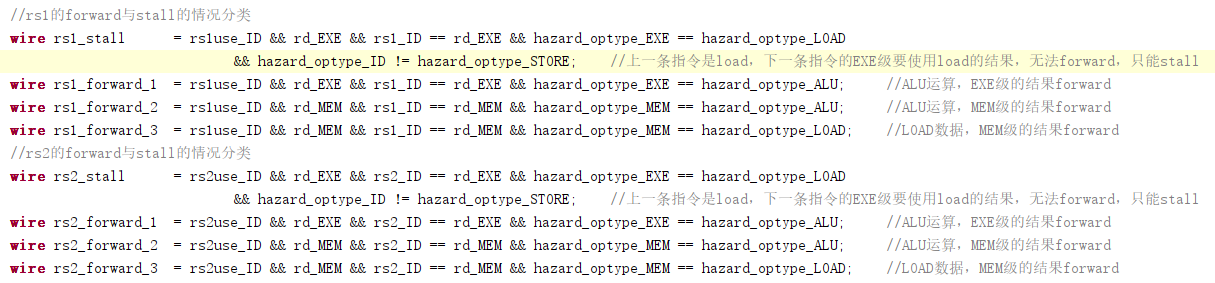
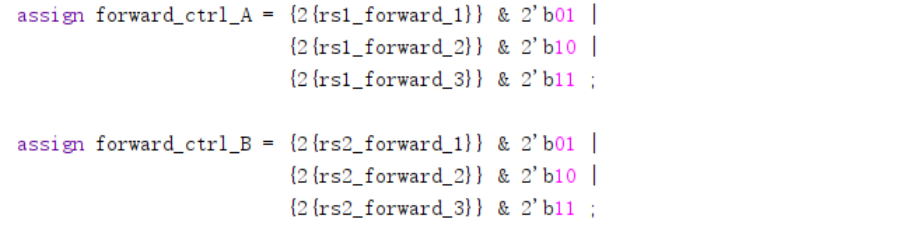
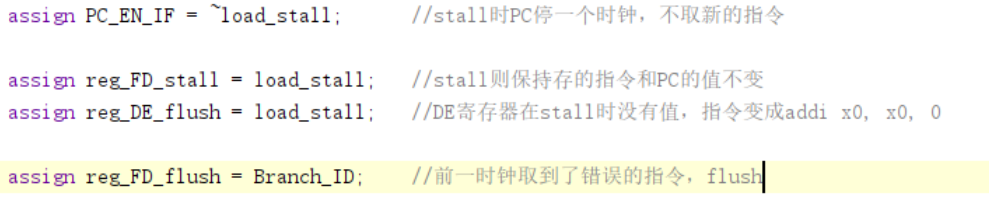
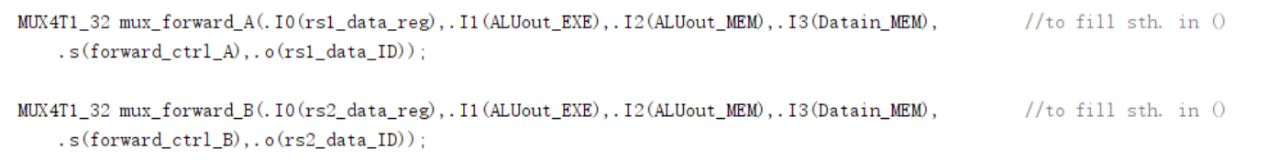
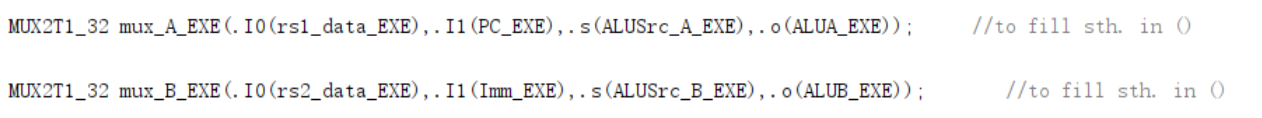
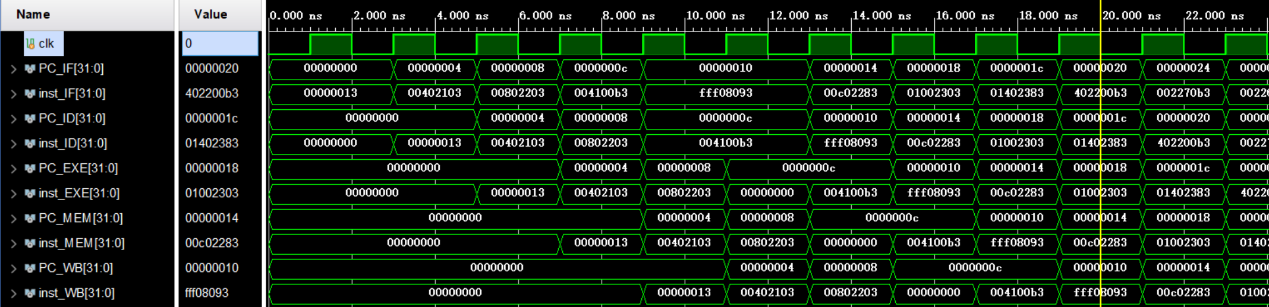
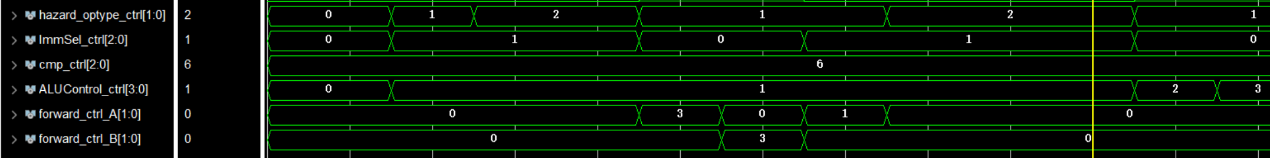
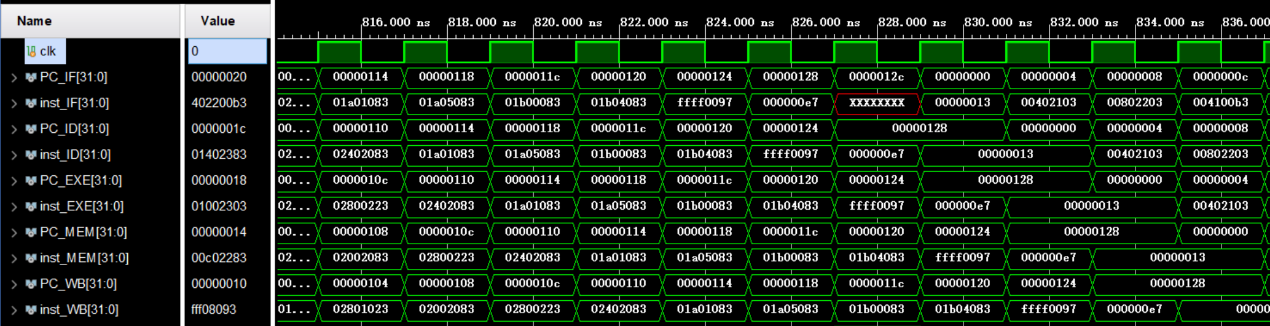
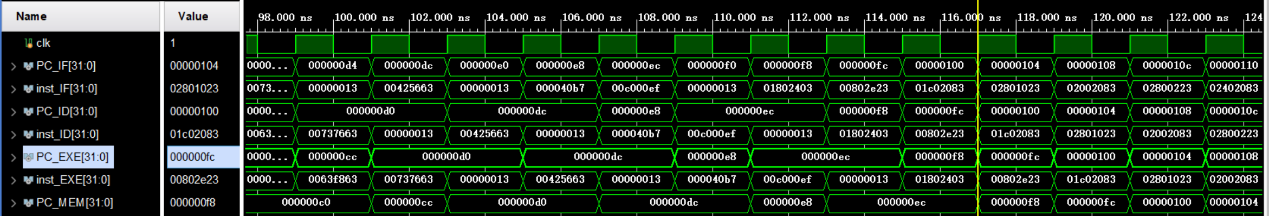
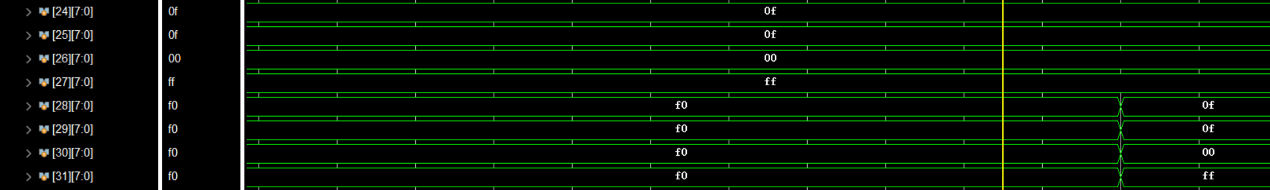
课程名称： 计算机体系结构 实验类型： 综合

实验项目名称： Pipelined CPU supporting RV32I instructions

学生姓名： 刘思锐 专业： 计算机科学与技术 学号： 3200102708

同组学生姓名： 陈镛屹 指导老师： 陈文智

实验地点： 曹西301 实验日期： 2022 年 10 月 10 日

1. 实验目的和要求
   1. Understand RISC-V RV32I instructions
   2. Master the design methods of pipelined CPU executing RV32I instructions
   3. Master the method of Pipeline Forwarding Detection and bypass unit design
   4. Master the methods of 1-cycle stall of Predict-not-taken branch design
   5. master methods of program verification of Pipelined CPU executing RV32I instructions
2. 实验内容和原理
   1. 实验内容
      1. Design of Pipelined CPU executing RV32I instructions.
         1. Design datapath
         2. Design Bypass Unit
         3. Design CPU Controller
      2. Verify the Pipelined CPU with program and observe the execution of program
   2. 实验原理
      1. 标准5级流水线中的forward情况有以下4种
         1. 前一指令EXE的结果立即forward到下一条指令的EXE操作数。
         2. 前一指令EXE的结果在MEM级才forward到下两条指令的EXE操作数。
         3. 前一指令从MEM读出的结果forward到下一条指令的MEM写数据。
         4. 前一指令从MEM读出的结果forward到下两条指令的EXE操作数。
      2. 标准5级流水线中必须stall的情况是：
         1. 前一指令是load，后一指令在EXE级需要使用前一指令读出的数据。
         2. 无条件跳转指令（包括JAL、JALR）
      3. 遇到分支指令时可行的做法有：
         1. Stall一个时钟等待分支条件判断
         2. Predict taken默认跳转条件成立，如果发现条件不成立则把已经读入的指令flush
         3. Predict not-taken默认跳转条件不成立，如果发现条件成立则把已经读入的指令flush
3. 实验过程和数据记录
   1. 补全cmp\_32.v文件
      1. cmp\_32.v是一个多功能的比较单元，关键是要先输入比较的项目，才能返回相应结果的bool值。
   2. 补全CtrlUnit.v文件
      1. 补全解码部分，根据指令中的opcode与func3确定指令的类型。（图中仅展示部分代码）
      2. 连接CtrlUnit和cmp\_32
      3. 根据DataPath图补全控制信号
      4. 分析当前指令类型并生成相应信号，用于竞争的判断。
   3. 补全HazardDetectionUnit.v文件
      1. 根据指令类型和寄存器使用情况判断当前是否需要forward和stall。
      2. 根据forward情况生成mux的控制信号
      3. 根据分支跳转情况和stall情况判断各级流水线之间的寄存器是否需要flush、PC应该如何取值。**这里对跳转指令使用predict not-taken策略。**
   4. 补全RV32core.v文件，主要是各路mux的控制信号与实际接线相对应。
      1. 跳转相关的mux
      2. Forward相关的mux 
      3. EXE相关的mux
   5. 进行仿真实验
4. 实验结果分析
   1. 第一种stall情况，load后立刻取做操作数，同时也有第一、第三种forward情况。
   2. 第二种stall情况，JAL指令
   3. 第二种forward情况（ALU运算的结果从MEM级forward给下两条指令的EXE级）在rom里似乎没有出现。
   4. 第四种forward情况，Load的结果立即store。
5. 讨论与心得

第一次实验的内容并没有超出上个学期学过的计算机组成，但是复杂的连线和HazardUnit中多种多样的条件判断对细心和耐心的要求很高，加之Vivado重新生成一次bit文件要花将近十分钟，为了完成这次实验还是在实验室蹲了相当长时间。