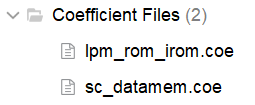
**Lab 3 实验报告**

潘文峥（520030910232）

1. **实验目的**
2. 掌握不同类型指令在数据通路中的执行路径；
3. 掌握Vivado仿真方式。
4. **实验平台**
5. 计算机1台(尽可能达到 8G 及以上内存)；
6. Xilinx Vivado开发套件(2020.2版本)。
7. **任务步骤**
8. 从前序实验中导入已完成的各个模块本次实验中所需完成的模块，并完成指令rom和数据ram的例化。最终形成的文件树如图1所示。





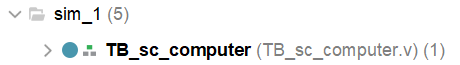


图1：文件树

1. 完成sc\_cpu.v文件的代码实现。这一部分是本实验的关键，需要根据不同指令类型构建数据通路，将前序实验中已经完成的立即数产生模块、alu模块、控制指令生成模块等子模块连接起来，使cpu可以正常连续工作。

以下展示各部分数据传递关系和相应的代码实现：（已提供的部分略去）

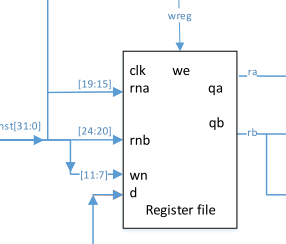
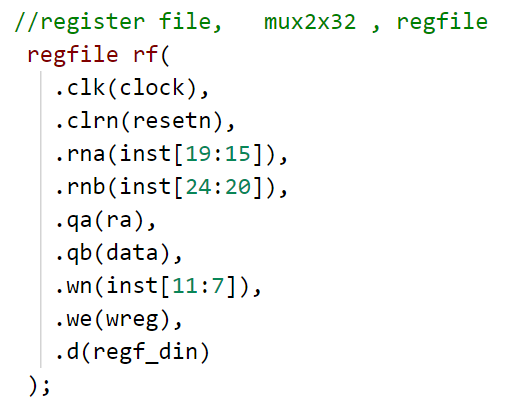
 

图2：寄存器堆的实现

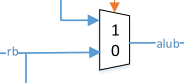
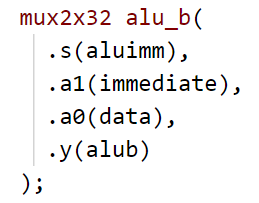
 

图3：根据aluimm指令决定alub的取值

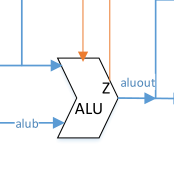
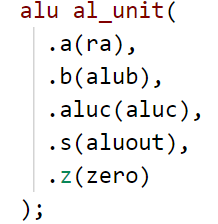
 

图4：ALU模块的实现

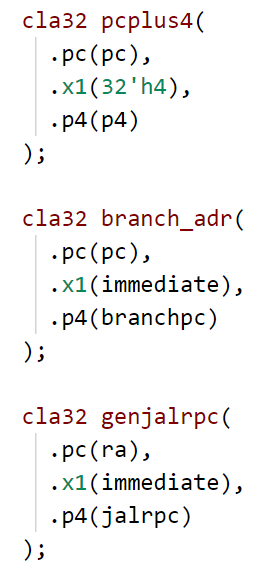
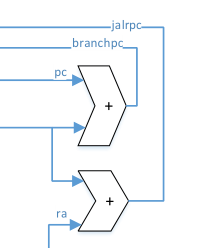
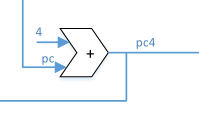


图5：三个加法器：分别计算分支/跳转/顺序情况下的下一条指令地址

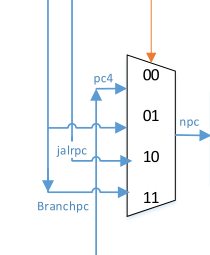
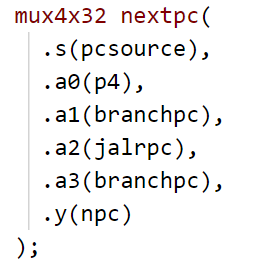
 

图6：四路选择器：根据pcsource给出下一条指令地址

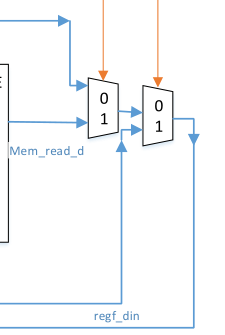
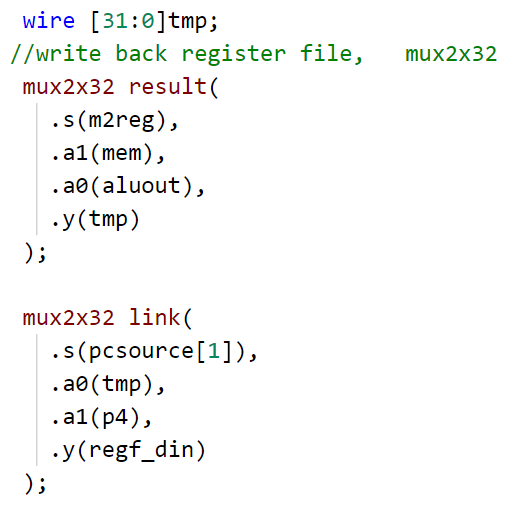
 

图7：两个二路选择器：根据m2reg和pcsource[1]决定写回寄存器的内容

1. **出现的问题与解决方案**

根据仿真结果中出现的不符合预期的情况，反复调试，找到实验1、2中部分问题代码加以修改。

（1）

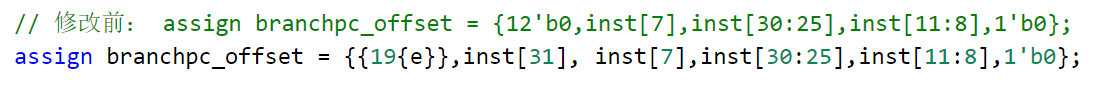


图8：immext.v中，分支指令地址偏移应作符号拓展

（2）

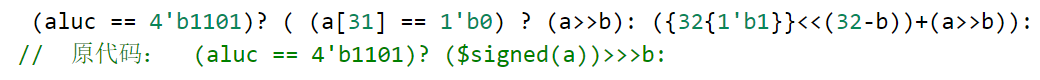


图9：alu模块中算数右移按原有方式会出错，故手动拼接实现

（3）

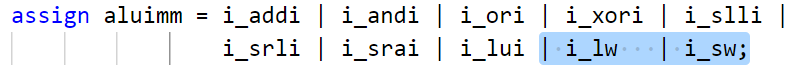


图10：原sc\_cu模块中的aluimm控制信号漏掉了lw和sw

（4）应特别注意拼写问题：vivado平台对变量名拼写错误导致的问题不会显式报错，只能通过仿真中出现的问题进行排查分析找到错误，耗费大量时间。

（5）例化ram过程中读写位宽选择错误会导致sw和lw的执行不能得到正确结果，经检查更正为32bits即可。

1. **仿真结果演示**

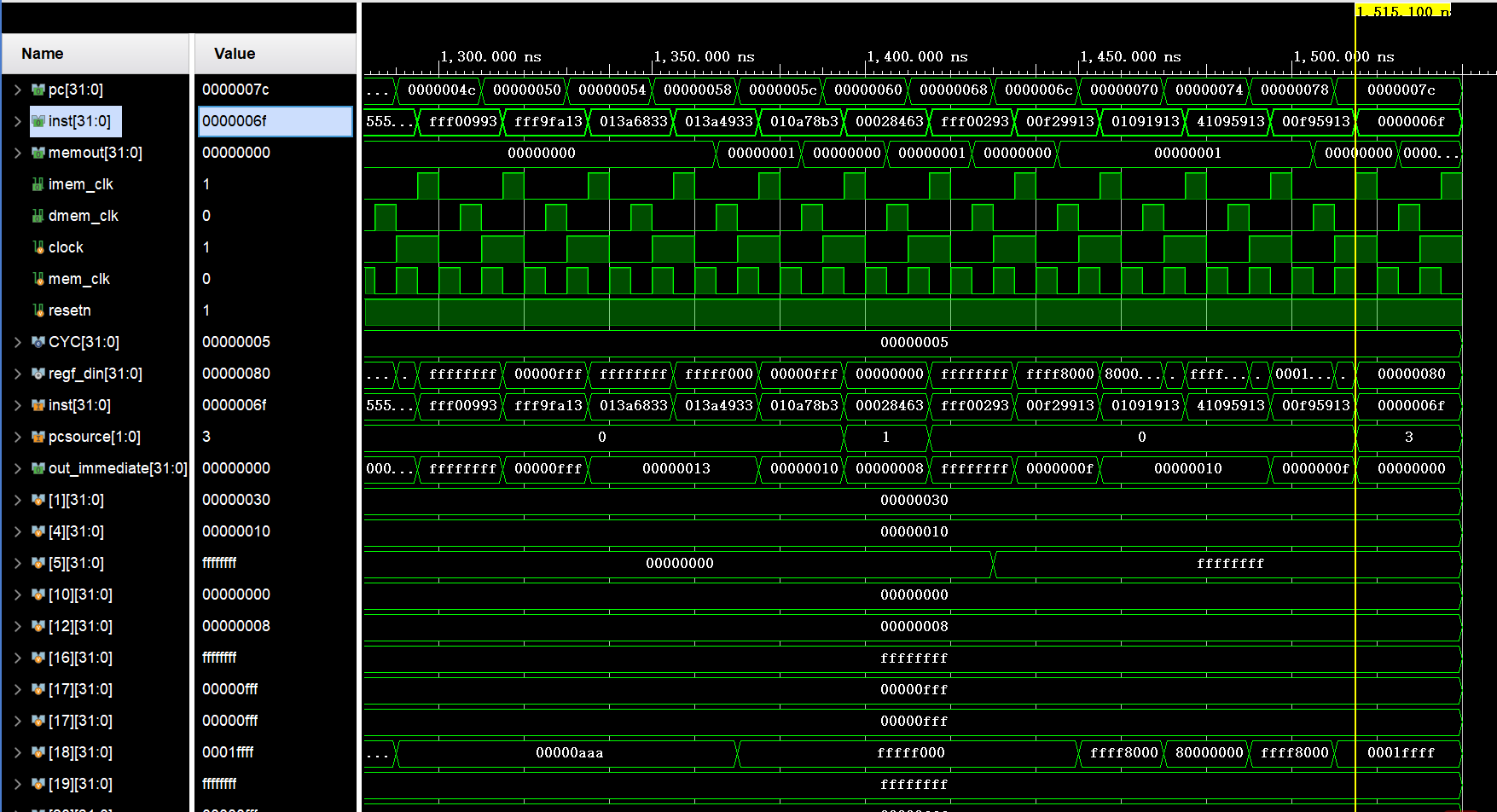


图11：仿真结果（1）

从仿真结果（1）可以看出，1515ns后，指令正确执行完毕。最后的指令地址为0000007c；最后一条指令为0000006f；此时x18寄存器的值为0001ffff。均与实验2.1中仿真表给出的结果一致，达到了实验目的。



图12：仿真结果（2）

从仿真结果（2）可以看到，790ns后，寄存器x19的值为0000000a，说明处理器正确执行了sw、lw指令，并在四次循环过程中正确得到了1+2+3+4的结果。