lab1实验报告

19307130030 唐思源

一、实验思路

1. `select pc`
   - 从根据D阶段转发的`jump`信号来判断是`pc+4`还是跳转目的地。
2. `fetch`
   - 用`select pc`向指令内存请求指令
   - 将当前指令加4
3. `decode`
   - 接收指令内存传来的指令
   - 解码指令
   - 根据E、M、W阶段的要写回的寄存器`dst`和当前阶段要用的寄存器`src`进行判断然后转发
   - 根据指令`icode`进行`val1`和`val2`的选取，跳转指令直接算出结果转发给`select pc`
4. `execute`
   - 根据`icode`选择`ALU`的两个操作数，然后进行运算
   - 若指令为`SW`或`LW`，则向数据内存发出请求
5. `memory`
   - 根据指令是否为`LW`，选择`ALU`或内存的数据
   - 判断是否需要写回寄存器，结果赋给`write_enable`
6. `writeback`
   - 将数据依次接收并传递
7. `regfile`
   - 读可以立刻获得，写需要一个周期。
8. `pipeline register`
   - `bubble`：数据清零
   - `stall`：不进行操作
9. `hazard and forward`
   - `load+use`：`bubble` E阶段，`stall` F、D阶段，让在M阶段获得的内存数据可以转发给D阶段。
10. 其它
    - 虚实地址转换：一开始在`SRAMTop`里进行，但是编译报错，于是放在了`MyCore`里

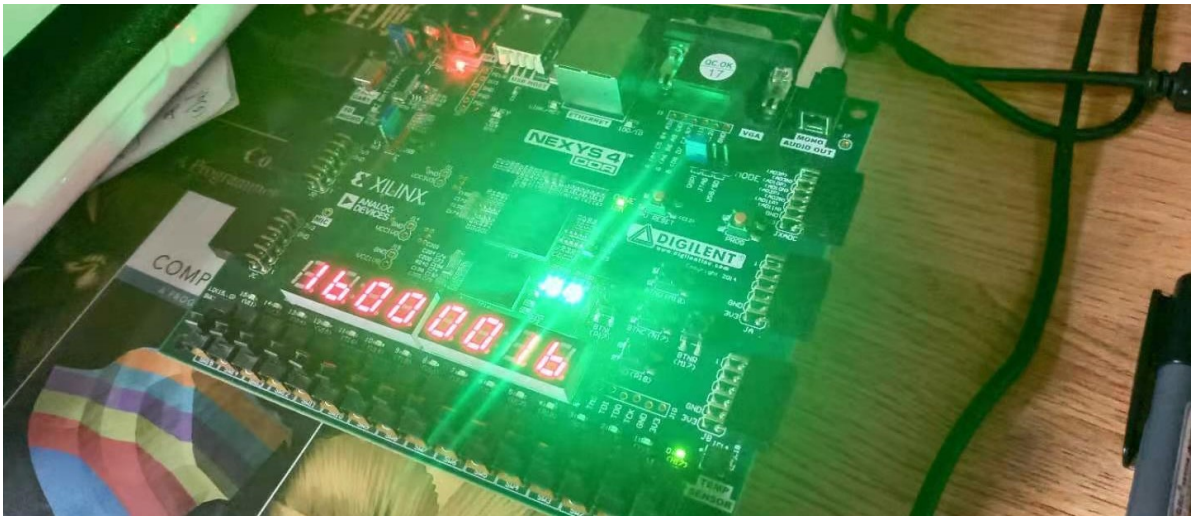
二、实验结果

仿真：

```
        [ 522000 ns] Test is running, debug_wb_pc = 0xbfc151e0
        [ 532000 ns] Test is running, debug_wb_pc = 0xbfc16180
----[ 540845 ns] Number 8'd26 Functional Test Point PASS!!!
        [ 542000 ns] Test is running, debug_wb_pc = 0xbfc04fa0
        [ 552000 ns] Test is running, debug_wb_pc = 0xbfc05f40
----[ 560175 ns] Number 8'd27 Functional Test Point PASS!!!
================================================================

Test end!
----PASS!!!
$finish called at time : 560765 ns : File "D:/Download/Github/ICS-2021Spring-FDU/vivado/test1_naive/soc_sram_func/testbench/mycpu_tb.v" Line 261
run: Time (s): cpu = 00:00:11 ; elapsed = 00:00:10 . Memory (MB): peak = 901.738 ; gain = 0.000
```

上板：



`verilator`：

```
lingnicktang@ubuntu ~/Desktop/ICS-2021Spring-FDU master ± make verilate TARGET=mycpu/SRAMTop
rilator --cc -sv --relative-includes --output-split 6000 --trace-fst --trace-structs --no-trace-params --Mdir bu
AMTop/verilated --top-module SRAMTop --prefix VModel -y source/util/ -y source/ram/ -y source/include/ -y source
/mycpu/fetch -y source/mycpu/writeback -y source/mycpu/memory -y source/mycpu/decode -y source/mycpu/utility -y
cute -Wall -Wno-IMPORTSTAR  source/mycpu/SRAMTop.sv
ingnicktang@ubuntu ~/Desktop/ICS-2021Spring-FDU master ±
```

三、实验总结

通过本次实验，我粗略了解了

- MIPS指令集的架构
- 五级流水线CPU的整体框架和部分细节
- `system verilog`的部分语法
- `verilator`的使用方式

也要感谢

- 助教们及时积极的答疑解惑
- 同学们的帮助和他们认真的态度
- 老师的指导和建议