National Tsing Hua University
Department of Electrical Engineering
EE429200 IC Design Laboratory, Fall 2021

Lab 05: RTL Simulation and Debugging

Assigned on Oct 14, 2021
Due day in Oct 21, 2021

Objective

In this lab, you will learn:

1. The IF stage and caches' operations in a 3-stage CPU.
2. A tool, *Spyglass*, to check coding style and synthesizability.
3. The operation of single port SRAM.
4. Add a finite state machine (FSM) as control unit.

Demo checklist:

- ☐ Show TA your simulation result of new version CPU (including all instructions).
- ☐ Answer the following questions
    1. Describe how you can modify the codes to comply with the *Spyglass*'s requirements.
    2. What does instruction_2 do (describes or in pseudo code)?

Environment Setup

Copy lab file packages from ee4292. Decompress the package and enter it. You can check the file list in Appendix.

```
$ cp ~ee4292/iclab2021/lab05.zip .
$ unzip lab05.zip
$ cd lab05/
```

Note: please run simulation in the ***sim*** directory to maintain a fine data management.

Description

In Lab4 we have finished most parts of ID and EXE stages of CPU, left IF stage for manual, and ignored caches. In this lab, we will construct the instruction fetch function and cache blocks for our CPU, and complete the overall system.

Action Items

I. Complete CPU sub modules.

1. Add the I-cache and D-cache into the CPU, both of them should be constructed with ***SRAM256x32s_m.v***.
2. Please modify the ***controller.v*** according to control signal table. (Hint: You should add the LW and SW control signal.)
3. Simulate CPU with testbench ***test_top.v*** provided in *sim*, and verify CPU function. It will generate register.txt to help you check answer.

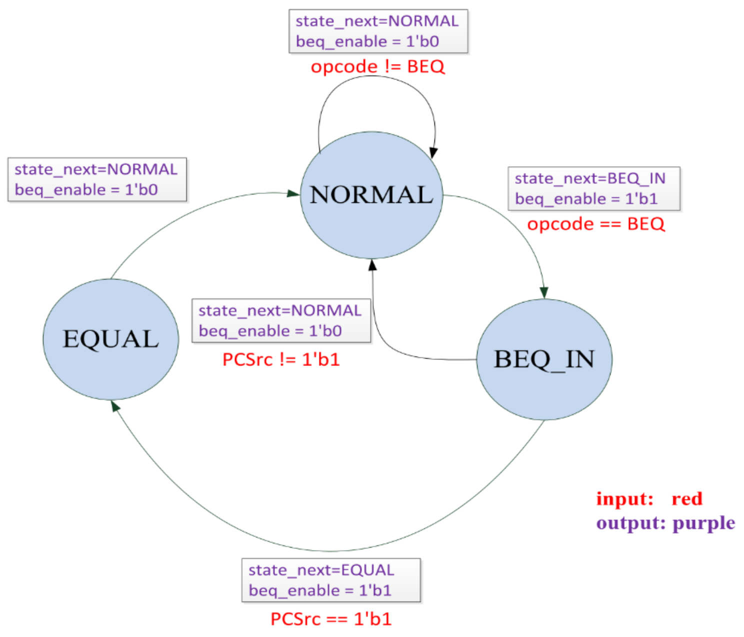
II. Add a Controller unit for branch control.

1. Please refer to the pdf **cpu_support_material** part2.
2. The finite state machine for branch control is shown below. Please reference the previous lab exercise (controller.v) and design a controller unit with the following features:
   - All operations initiated at the positive edge trigger.
   - Control signals:

beq_en	When beq_en=0, fetch instructions from I-cache into the controller, register file and prepare them to be ready for decode.
PCSrc	When PCSrc=1, PC counter will be the BEQ address.

   - There are three different operating states for the controller, **NORMAL** operation, **BEQ_IN** and **EQUAL**.
   - You need to add **BEQ** operation in opcode decoder as you did in Action I.
3. Understand how testbench and instructions work, then verify your CPU with two provided instruction lists.
4. After functional check, we should check the synthesizability. Use Spyglass to analyze your code.

State Diagram of the Controller



Appendix

Directory	Filename	Description
sim/action_I	presim.f	Simulation files script
sim/action_I	test_top.v	Test bench for this CPU
sim/action_I	instruction.v	Instruction to test CPU
sim/action_I	golden_register.v	golden pattern for this CPU
sim/action_II	presim.f	Simulation files script
sim/action_II	test_top.v	Test bench to this CPU
sim/action_II	instruction1/2.txt	Instructions to test CPU
sim/action_II	golden_register.v	golden pattern for this CPU
source	top.v	Top module for this CPU
source	ID_stage.v	RTL for ID stage
source	IF_stage.v	RTL for IF stage
source	controller.v	RTL for control signal
source	regfile.v	RTL for register file
source	ID_EXE.v	RTL for ID to EXE flip flop
source	EXE_stage.v	RTL for EXE stage
source	alu.v	RTL for alu module
source	CPU_define.v	Definition file
source	PC.v	RTL for program counter
source	IF_ID.v	RTL for IF to ID flip flop
source	SRAM256x32s_m.v	SRAM model