Computer Architecture

Lab 3

R12922147 魏子翔

Modules Explanation.

Below I explain the modifications for those code from lab 2 and the main idea for branch_predictor module.

1. CPU

   First of all, I change the original branch unit module to branch_predictor module as a 2-bit predictor for beq instruction, then I initialized the state of branch predictor to zero. Also, rather than adding other new modules to handle the condition for branch instruction, I simply add some wire and assign their conditional logic value in CPU.v, which is the main program.
2. ALU_Control

   Add the condition for beq, then assign their aluctr signal same as sub instruction.
3. ALU

   Add a new signal, zero, to check whether the input rs1 data is equal to rs2 data.
4. IDEX

   In IDEX module, I first add a flush signal to set all the signals to zero, then add branch, pc_next, beq_tar, and prev_pred signals to handle the condition for beq.
5. branch_predictor

   Store a state signal as a 2-bit predictor with 00 as strongly taken, 01 weakly taken, 10 weakly not taken, and 11 strongly not taken.

Difficulties Encountered and Solutions in This Lab.

1. Since there's no datapath reference in this lab, I was confused for a long time until I decided my code structure.
2. Although I already created the datapath in my brain, I still spent a lot of time to debug, there's so many details in Verilog that are different from other languages, such as the posedge setting, the typo would not trigger the error, and so on. Also, it's harder to debug without the print function.
3. I suffered from the initialization of control signals again, same as lab 2...

Development Environment.

1. My development environment is Ubuntu with iverilog and wavetrace from vscode.