CA Lab3 report

B10902105 何珮瑄

Modules

Besides modules implemented in lab1, I add new modules below:

branch_check

There are 5 inputs `Branch_i` `branch_result_i` `predict_i` `Imm_i` `PC_i` `Flush_o` `PC_o` and two outputs `Flush_o` `PC_o`.

This module is to check if the prediction is correct. First initialize `Flush_o` to zero, and if `branch_reslut_i` is not equal to `predict_i` which indicates making a wrong predict, then set `Flush_o` to 1.

Besides, if `branch_result_i` is true, than calculate the target address `PC_o = PC_i + (Imm_i << 1)` , on the other hand if `branch_result_i` is false, `PC_o = PC_i + 4`.

branch_predictor

there are 5 inputs `clk_i` `rst_i` `Branch_i` `update_i` `result_i` and a output `predict_o`

this module is the implement of the state diagram in spec, which updates the history based on the actual branch outcome (`update_i`) and the predicted outcome (`result_i`).

- If the prediction is correct, the history is reinforced in the corresponding direction (strongly taken or strongly not taken).
- If the prediction is incorrect, the history is weakened in the corresponding direction.

branch_result

this module performs a logical AND operation on the inputs `Zero_i` and `Branch_i`. The output `result_o` will be true if both inputs are true, meaning that there should be a branch.

branch

there are 4 inputs `Branch_i` `predict_i` `Imm_i` `PC_i` and 2 output `Flush_o` `PC_o`.

If both `Branch_i` and `predict_i` are true, it indicates that the branch is predicted and taken, triggering a pipeline flush (`Flush_o` set to 1).

Flush

there are 4 inputs `IFID_Flush_i` `IDEX_Flush_i` `IFID_Branch_PC_i` `IDEX_Branch_PC_i` and two outputs `IFID_Flush_o` and `branch_PC_o`

The `IFID_Flush_o` is initially set to 0. If a flush is required in the IDEX stage (`IDEX_Flush_i` is 1):

- Set `branch_PC_o` to the program counter associated with the branch instruction in the IDEX stage (`IDEX_branch_PC_i`).
- Set `IFID_Flush_o` to 1.
- If no flush is required in the IDEX stage but a flush is required in the IFID stage (`IFID_Flush_i` is 1):
    - Set `branch_PC_o` to the program counter associated with the branch instruction in the IFID stage (`IFID_branch_PC_i`).
    - Set `IFID_Flush_o` to 1.
- If neither flush condition is met, set `IFID_Flush_o` to 0.

CPU

Finally, the CPU connects these components together as the final data path given in the spec.

Difficulty

I've encountered a twofold challenge. First, distinguishing between two types of flushes has proven tricky. One type is triggered when the branch predictor anticipates a taken branch, necessitating a pipeline flush. The second type arises from incorrect predictions, demanding a correction in the pipeline state. As I strive to illustrate this process in a single-cycle diagram, deciding which pipeline stages to flush under different conditions has proven to be a daunting task. Adding to the complexity is the consideration of consecutive branches, where the flow of instructions through the pipeline becomes intricate. Balancing the program counter and prediction outcomes becomes crucial in ensuring a seamless and correct execution of instructions.

Development Environment:

macOS & iverilog