1. About This Project

Wu Runzhe (吴润哲)

This project is a RISC-V CPU with a 5-stage pipeline, implemented in Verilog HDL, which is a course project of Computer Architecture, ACM Class @ SJTU.

2. Design

2.1 Feature

The main features of this RISC-V CPU are briefly introduced in the table below. For more details, please read the *Specification* section.

Feature	Details
ISA	RISC-V (RV32I subset)
Pipelining	5-stage pipeline
Data forwarding	to resolve data hazard
Cache	a direct mapping I-cache with a victim cache and a direct mapping D-cache
Branch prediction	tournament predictor using 1 global predictor and 1 local predictor
Branch target buffer (BTB)	to accelerate prediction and reduce stall

- Basys3 FPGA test passed

2.2 Specification

- This CPU has a standard 5-stage pipeline with complete path data forwarding. Data produced by stage EX and stage MEM can be passed directly to stage ID, which aims to resolve RAW data hazard and reduce stall.
- Stage IF is equipped with a 1024-byte direct mapping I-cache, which contains 256 instructions at most and supports consecutive hit. A victim cache is attached to it.
- Stage MEM is equipped with a 64-byte direct mapping D-cache which is implemented by the write through method.
- The tournament predictor uses 2 predictor: 1 based on global information and 1 based on local information, and combine with a selector. The global predictor has 1K entries, indexed by the history of the last 10 branches, and each entry in the global predictor is a standard 2-bit saturating counter. The local predictor has 32 entries, indexed by the address of the branch instruction, and each entry is a standard 2-bit saturating counter. The selector is a 2-bit saturating counter, too. When a prediction produced by some predictor is correct, we should attach more importance to it. Thus, the selector should be inclined to it, too. This is how the tournament predictor works.

- The branch prediction is conducted as follows: When stage IF is fetching a new instruction, it will also send the PC to BTB (branch target buffer) to quickly know whether it is a branch instruction, and if so, what the target address is. At the same time, a predictor implemented by a tournament predictor will also send the prediction (taken or not taken) to stage IF. Then it changes the PC according to the prediction. Stage EX will check whether this prediction is correct and when it is correct, it causes no stall. However, when it is incorrect, it costs 1 cycles to restore. The result and target address are sent in stage EX to update BTB and predictor.
- To resolve the structure hazard caused by memory access of both stage IF and stage MEM, a memory controller (see mem_ctrl.v) is implemented.

2.3 Rub

- When an RAW data hazard happens, forwarding is never enough. To solve a situation where the source register is to be written by a LOAD instruction in stage EX, we can only stall the pipeline.
- The FSM (finite state machine) for both stages IF and MEM is not easy to design. However, I tried several FSM design and finally choose this one.
- I use a register named "avoid_data_hazard" in stage IF to change the CPU between a 5-stage pipeline and no pipeline by adding 10 cycles after instruction so that I can debug easier.

3. Performance

The highest frequency it reaches is 130 MHz. When the frequency is not higher than 130 MHz, all test cases can be passed on FPGA. When higher than 130 MHz, some test cases fail. However, when the frequency reaches 130 MHz, the WNS (worst negative slack) has been negative.

What about the speed? Take running `pi.c` on FPGA as an example:

- 100 MHz: 1.406250 s
- 130 MHz: 0.996875 s
- 140 MHz: 0.984375 s (failed on one test case: `bulgarian.c`)

4. Acknowledgements

I am grateful to teaching assistant Li Jiasen (李嘉森) who provided instructive guidance on this project when I found nowhere to start. And I would like to express my gratitude to teaching assistant Li Zhaoyu (李照宇) for his suggestions on FPGA testing. I am also thankful for many classmates who had helped me.

References

[1] 雷思磊. *自己动手写CPU*, 电子工业出版社, 2014.

[2] John L. Hennessy, David A. Patterson, et al. *Computer Architecture: A Quantitative Approach*, Fifth Edition, 2012.