Project Report of RISC-V CPU

Weizhe Chen (陈伟哲)
ACM Class, Shanghai Jiao Tong University

1 Introduction

This project is a RISC-V CPU with five-stage pipeline, implemented in Verilog HDL.

2 Design

2.1 Features

Main features of this RISC-V CPU are briefly introduced in the table below.

Feature	RISC-V CPU
ISA	RISC-V(subset)
Cache	2-way set associate I-cache
Branch Prediction	static branch prediction which always predict not jump
FPGA Support	Yes
Speed	100Mhz 1.6s for pi.c*

[*]: When running 100Mhz on FPGA, it has timing failed problem , but it still run pretty good.

2.2 Specification

My CPU is a standard 5-stage pipeline CPU which has a memory controller and stall controller. It also have a I-cache and a static branch prediction for speed up.

- Considering all hazards. For structure hazard, we don't need to face it. For data hazard, I used the forwarding method to solve it. But when facing with LOAD command which I can not get the exactly value of register before MEM stage, I have to stall the pipeline until the former command pass MEM stage. For control hazard,

when I found there is a branch prediction fault, I just changed those commands into no-use commands which is actually bubbles.

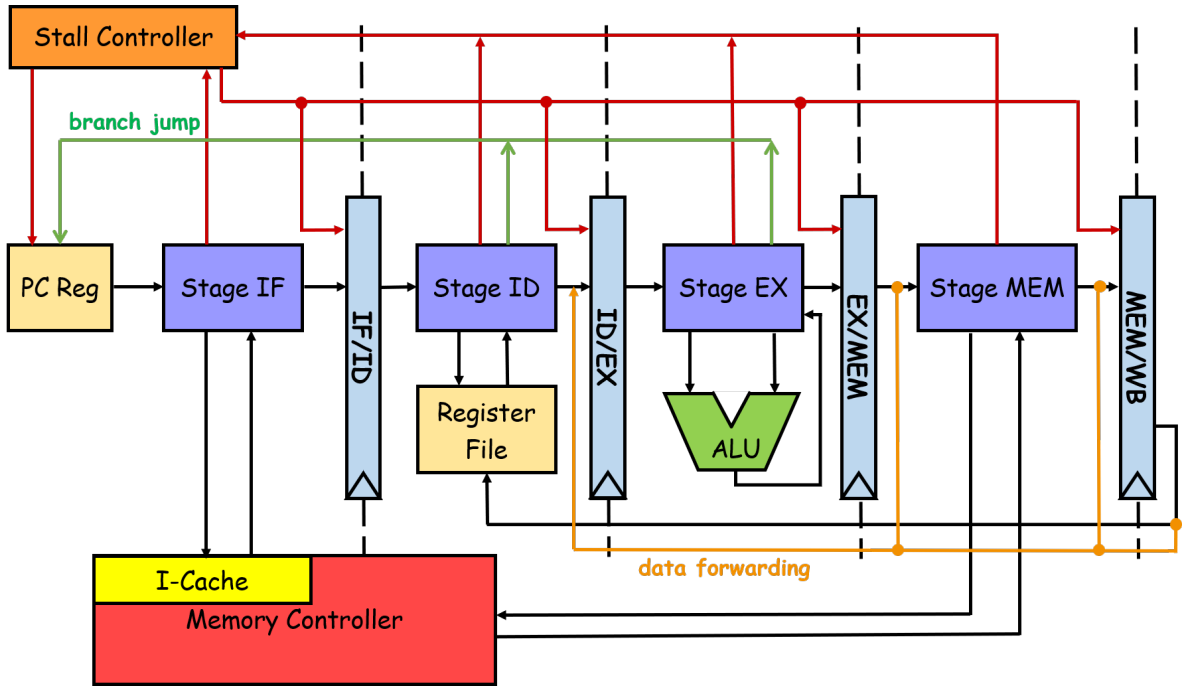

Fig. 1: Structure of my CPU Design

- Although facing with timing failed for 100Mhz in my implementation period, I can run the CPU correctly on the FPGA. The biggest reason is that almost all the timing failed is because I have a module using negedge in the CPU, which makes it has less time permitted. I use it in the part that I-cache updating and it should be done before the next posedge otherwise if the new pc is equal to the former pc, it can not hit. It can be fixed by a few code rewriting (actually special check for this case).

3 Thinkings about features

Cache Cache is urgently needed because the speed of communication with memory is really the bottleneck of this CPU. One simple instruction to be fetched need 4 cycle without cache. And with no write on instruction memory, a I-cache is a choice that is not only easy to write but also useful. My I-cache is a 2-way set associate cache, with LRU method when facing replacing problem.The cache size is 2K.

Branch Prediction A static prediction is used in my CPU. And it will always predict not jump. The reason for a prediction like that is that it don't need extra calculation

because $pc_{new} = pc_{current} + 4$ is already calculated even before we know it is a branch command.For any other branch prediction method, we have to flush the pipeline because the next pc is already readed when we do the prediction. We may add a BTB in fetch stage, but it does not work very well according to some of my classmates sayings.

4 Acknowledgements

Special thanks to Linqi Chen (陈林淇) for his guidance to me of how to use Vivado and a few suggestions of writing which truly helpes me to run on the FPGA. Also thanks many other classmates for their suggestions and help.

5 References

1. 雷思磊. *自己动手写 CPU*, 电子工业出版社, 2014.

2. Zhanghao Wu's (吴章昊) MIPS CPU project. https://github.com/Michaelvll/RISCV_CPU