RISC-V CPU Project Report

张志成 518030910439

2020 年 1 月 4 日

1 Design & Architecture

1.1 Main features

Features	Description
ISA	RISCV32I subset
Pipeline	5-stage pipeline with data-forwarding
Cache	1KB 2-way set associative Instruction Cache & 256Byte direct-mapped Data Cache
Branch Prediction	512 Byte Branch Target Buffer with 1-bit counter

1.2 Design & Datapath graph

The design is a standard 5-stage pipeline with forwarding from EX/MEM latch and MEM/WB latch to ID stage.

The CPU is mainly controlled by a centralized controller which deals with the ready signal(rdy) and controls the stalling of each stage if necessary. PC Reg component determines the next PC based on the prediction made by the IF stage using BTB and the correct target address determined in the EX stage, or simply PC+4 if the current instruction is not a branch.

The datapath graph of the CPU is illustrated in **Figure 1**.

2 Some remarks & thoughts

(1) Performance

The performance is listed as follows(at 250MHz):

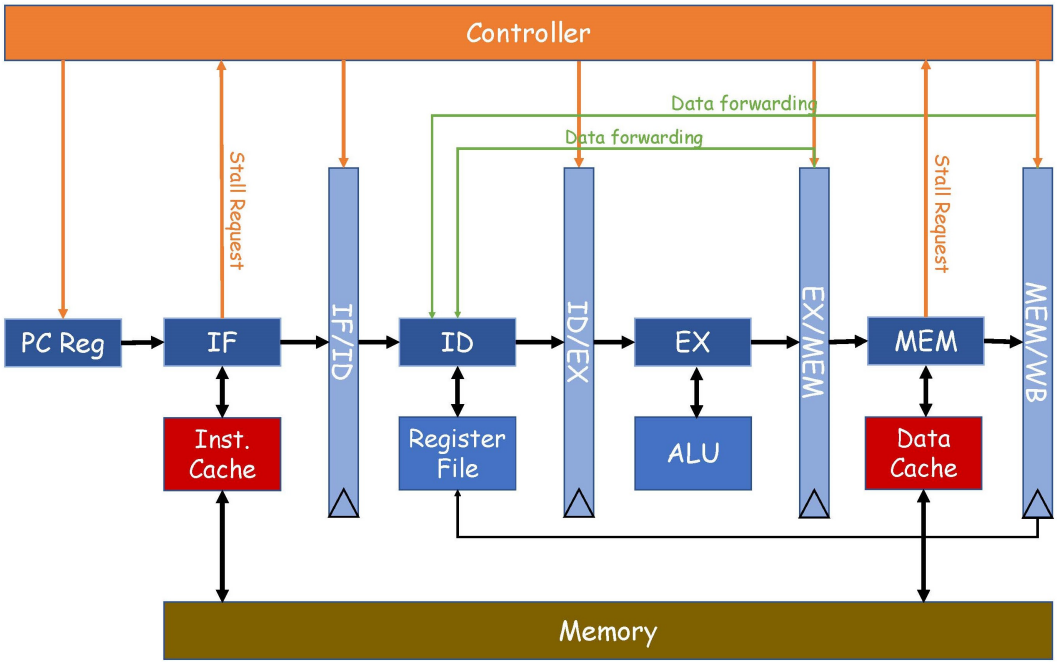

Figure 1: Datapath Graph of the CPU

Program	Runtime
pi	0.5625s
heart(by Yu Zhen)	157.109s

(2) Branch Prediction

Branch Prediction requires calculating the target address in IF stage. However, a significant amount of delay will be introduced by implementing an adder in IF stage.

Adding a **Branch Target Buffer(BTB)** resolves this problem. BTB records the target address for all *taken* branches. Thus, the IF stage uses the current PC to index the BTB, if an entry is found, then the branch is predicted taken and the program uses the corresponding address as the next PC.

(3) Overclocking

Since I wrote the code with the principle of minimizing delay in mind, there is a positive 2ns WNS at 100MHz.

Overclocking to 250MHz still yields stable performance.

(4) Debugging

Debugging is hard, especially on the FPGA, but not impossible. Setting up debug probe allows me to find problem with the rdy signal.
Also, don't just test *pi*! The code could still be incorrect even if *pi* runs correctly.

3 Conclusions

(1) Smaller is faster.
This is especially true when dealing with a not-so-good FPGA board. I tried

- adding an additional adder in the ID stage
- making the cache bigger to hold more instruction
- more sophisticated branch prediction schemes such as tournament predictor, predictor based on global history, etc.

Without any exception, the performance is undermined significantly. My final version is small(using only ˜20% of the board), but can be overclocked to achieve a pretty good performance.

4 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.