RISC-V CPU Project Report

卢禹杰

2020 年 1 月 3 日

1 简介

本次大作业主要目的是实现一个基于 RISCV 架构, rv32ia 指令集的 CPU。本项目一共实现了以下内容:

- 取指-译码-执行-访存-回写的五级流水架构
- FPGA 150MHz 测试通过
- 512Byte 的指令缓存，可以容纳 128 条 instruction，pi 测试点时间为 3s

本项目主要使用 verilog 硬件设计语言编写，使用 iverilog 进行模拟，Xilinx Vivado 进行仿真综合。(在写完大作业之后我才发现 iverilog 是一个很好的测试正确性的工具。我还写了一个脚本来进行快速测试正确性。)

2 基本架构

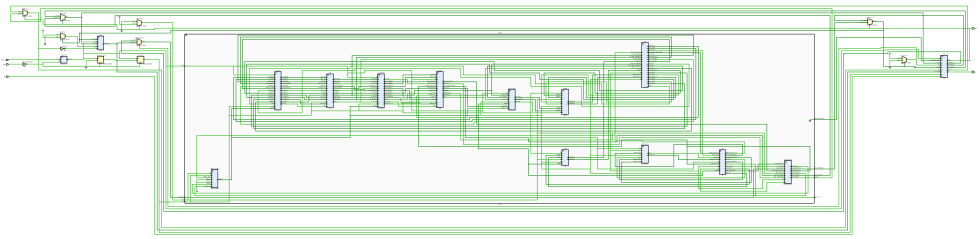

图 1: 基本架构

本项目采用的架构为经典的五级流水架构，cpu.v 中各模块功能如下

1. if: 取指令模块，与 MemoryController 以及 cache 交互
2. InstCache: 一个 512 bytes 的指令寄存器，可以做到 hit 两周期取值

3. regfile: 寄存器模块，与 wb 和 id 交互
4. if_id: 流水线寄存器，将指令传给 id
5. id: 指令译码以及判断分支以及跳转与否
6. id_ex: 流水线寄存器，将指令以及译码信息传给 ex
7. ex: 指令执行，完成所有的计算工作
8. ex_mem: 流水线寄存器，将指令以及内存访问信息传给 mem
9. mem: 内存访问模块，与 MemoryController 交互
10. mem_wb: 流水线寄存器，将指令以及寄存器读写信息传给 wb
11. MemController: 调度取指令和 mem 阶段的内存访问，与真正的内存交互
12. StallController: 控制流水线的暂停，解决 hazard

3 遇到的问题

在作业过程中，我主要遇到了一下问题

- 安装 RiscV Toolchain 失败，之后我在官方社区提了 issue 并得到了解答，发现是大作业仓库上的的安装教程有问题。采用社区提供的安装教程我成功的安装上了，并且我写了一个更方便的脚本来生成.data 文件。
- Linux 上的 Vivado 有比较严重的 bug，最开始我根本安装不上之后向社区提问之后我才知道是有两个动态链接库缺失了，手动创建软链接之后可以安装。之后 Simulate 的时候又出现了奇怪的报错，google 之后也解决了这个问题。
- 最开始的时候不会取指令，感觉一个周期只能读一个字节实在是太麻烦了，之后看了学长的代码才想到用一个有限状态机来解决这个问题。
- 上板的时候我遇到了许多的 critical warning。但是当我解决其中的两个 warning 之后，我发现自己的正确性又出大问题了，因为我有一个变量 memdone 必需在两个 always 语句中使用，尝试了很多方法都没解决。最后换了一个思路解决了这个问题。

4 大作业心得

本次大作业我学到了很多：

- 对五级流水以及教材中的一些设计思想更加熟悉了。
- 学习了新的硬件设计语言，提升了自己的编程能力与设计能力。
- 对数字电路更加了解了。

5 关于大作业的建议

- riscv-toolchain 的安装教程需要更新，因为版本的错误浪费了很多时间在上面。
- 希望能够提供更多上板的教程和资料，很多时候错了不知道为什么，问了同学才知道。以及 Vivado 其实有一些很强大的功能，但我也是最后一周才从同学那里了解到的，希望以后能有更多的这方面的资料。