五级流水的 CPU 实现

计算机体系结构大作业 Report

Author

黎金宁, 515030910592

Shanghai Jiao Tong University

lijinning@sjtu.edu.cn

2017 年 6 月 25 日

摘要

1 问题重述

- 在 FPGA 上实现五级流水线的 CPU 的一部分功能，并在模拟环境下实现完整的 CPU。使用 C++ 程序模拟内存和硬盘的读写，通过 USB-UART 在计算机和 FPGA 之间交换数据。
- 将五级流水线拆分成五个模块，每个模块单独实现并在 FPGA 上进行测试。测试方法是，通过 C++ 程序将预先生成的上一级的输出数据发送到 FPGA 上的模块，然后这个模块将这一级的输出数据发回 C++ 程序并检查正确性。（FPGA 上除了需要测试的模块，还要有处理数据收发的部分）模块之间有寄存器，在时钟信号的上升沿将数据传递到下一级，请将这个寄存器包含在它所连接的两个模块中的前一个模块。
- 在单独实现所有模块后，需要将所有模块连接成一个完整的 CPU 并在模拟环境中进行测试。
- 选做：硬盘读写的指令；缓存；在 FPGA 上实现完整的 CPU。
- 指令集：add, addi, sub, and, andi, or, ori, xor, xori, slt, slti, beq, bne, j, jr lb, lui, lw, sb, sw

2 原理简述

根据大作业的要求实现了一个五级流水的 CPU，并且尝试进行了与 FPGA 的通信。支持的指令集为 MIPS32，实现的指令为要求中的 20 个指令，如果时间允许的话还可以加入更多的指令来充实此处理器。

实现的模块包括取指令存储器 PC，译码阶段模块 ID，执行阶段模块 EX，访问内存阶段 MEM，回写阶段 WB，用来处理阶段之间数据传输的寄存器 IF/ID, ID/EX, EX/MEM, MEM/WB，

以及用来控制流水线运行和停止的模块 CTRL, 一个存储数据的 RAM 模块和一个寄存器模块 RegFile。

3 代码解读

3.1 取址阶段 PC

输入包括:

- 复位信号 reset
- 时钟信号 clk

输出包括:

- 读取指令的地址 pc
- 指令存储器是否能访问 flag

除了复位的时候，指令存储器是不可访问的，其他时刻均能访问取指令。Reset 的时候，把 pc 寄存器的值置为 0，否则随着时钟周期的增加，PC 的地址在指令存储器中增加四。如果遇到跳转，则 pc 存储器转到跳转的目标位置。

3.2 代码解析 ID

输入包括:

- 复位信号 reset
- 指令地址 pc
- 32 位的指令 inst
- 从寄存器模块中读到的操作数 1 regdata1
- 从寄存器模块中读到的操作数 2 regdata2

输出包括:

- 运算的类型 op
- 运算的操作数 1 reg1
- 运算的操作数 2 reg2
- 两个操作数在寄存器中地址的信号
- 写入的目标寄存器位置

首先是识别指令，根据指令码各位数上的值可以区分不同的操作类型。然后根据不同的指令特点分别确定需要几个读的寄存器，其地址由指令码结合寄存器模块可以得到。然后确定是否需要写寄存器，如果需要则输出目的寄存器的地址。

3.3 代码执行 EX

输入包括：

- 复位信号 reset
- 运算的类型 op
- 运算的操作数 1 reg1
- 运算的操作数 2 reg2
- 写入的目标寄存器位置

输出包括：

- 需要在 WB 模块中写入的寄存器以及需要写入的值

在 EX 执行过程中，首先根据不同的运算计算得到运算结果，并且根据运算类型的特点确定是否有需要在 MEM 和 WB 进行写入操作。

3.4 内存访问 MEM

输入包括：

- 接收需要在 WB 阶段写寄存器的地址和数据
- 需要读写的内存地址和数据

输出包括：

- 输出需要在 WB 阶段写寄存器的地址和数据
- 需要在 RAM 中读写的内存地址和数据

在 MEM 模块中，一是实现转发从 EX 阶段中得到的需要在 WB 阶段中访问的寄存器地址和数据，二是和数据存储模块 RAM 进行通信，把 EX 阶段中要求读写的内存地址和数据与 RAM 链接。

3.5 数据写回 WB

输入包括：

- 是否需要写入的信号 flag
- 写入的寄存器 addr
- 写入的数据 data

这部分需要的操作只是把对应的数据写入到对应的寄存器中。

3.6 Hazard 的处理

3.6.1 数据相关问题

参考了《自制编译器》上给出的数据前推的方案，也就是在 EX 阶段的结果传输到译码阶段以及将 MEM 阶段的结果传输到译码阶段。

- 在译码阶段中，如果读取的寄存器就是执行阶段传入的要写的寄存器，那么就直接把执行阶段的结果作为目的寄存器的值
- 如果读取的寄存器就是访问内存阶段要写的目的寄存器，那么直接把访存阶段的结果作为目的寄存器的值

3.6.2 load store

在译码阶段检查当前指令与上一条指令是否存在 load 相关，如果存在，就让 ID，IF 阶段暂停，相当于插入一个空指令。

4 与 FPGA 通信

大体思路是把写好的代码使用 vivado 直接发送到 FPGA 上，并把测试的代码嵌入到 CPU 的代码中，让 FPGA 运行并跑出结果，利用 FPGA 上的 LED 灯和显示器把过程中的答案显示出来，以验证 CPU 结果的正确性。

但是由于时间不够，CPU 程序在 FPGA 上的运行出现了很多问题，没办法调试正确。我觉得还是按照建议的方法把每一步的结果发送到 FPGA 然后再传回电脑来验证会比较好。

遇到的问题是电脑接上 FPGA 后，vivado 一直无法识别到 FPGA 的存在，网上说是接口没接好，我换了几个数据线都没用。vivado 这套理论学了很久都没弄清楚。

然后是怎么把过程中的结果转换为 LED 灯的信号也不好实现。