RISC-V 指令集 CPU 设计报告 F1603024 516030910582 刘啸远

任务描述

实现一个支持 RV32I Base Integer Instruction Set V2.0 中 2.1-2.7 指令的 CPU，并烧在 FPGA(Basys3)上，使用 UART 协议与电脑通信，使用电脑内存作为其内存。

模块划分

该 CPU 结构类似 TOMASULO，支持多个 ALU 超标量，但与 TOMASULO 在预约栈、ROB 的实现上有所不同，以下为其模块简图。

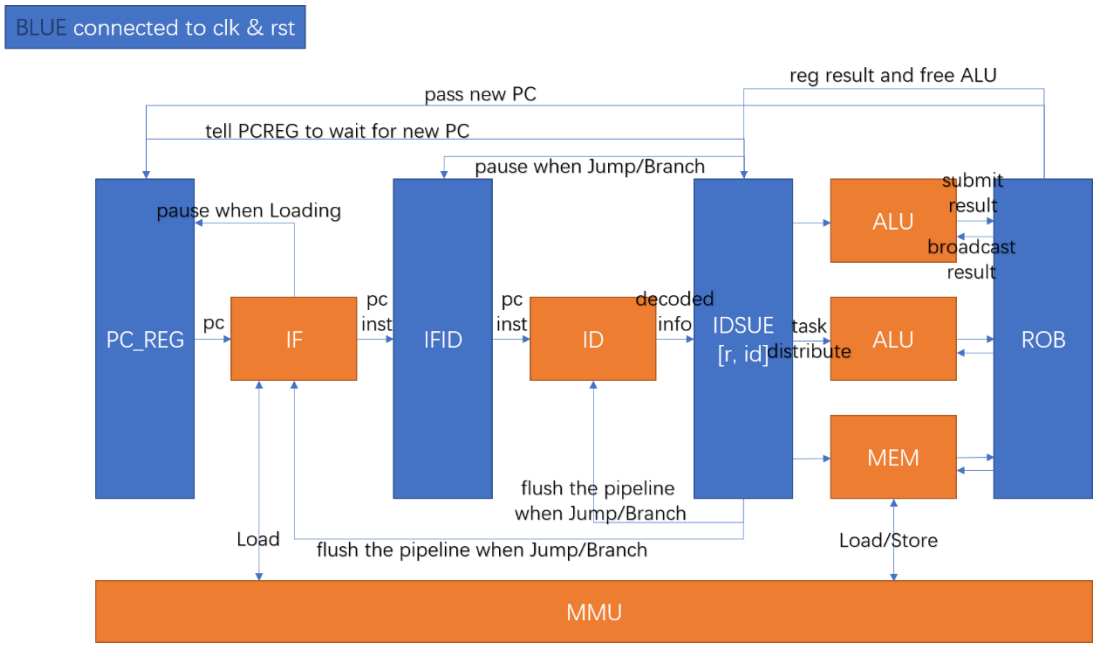

图 1：CPU 模块简图

模块名	clk?	功能
PC_REG	√	生成新指令地址，跳转时接受新地址
IF	×	接收 pc 值并访存取指令，在读取失败时暂停 PC_REG
IFID	√	传递 pc、指令内容
ID	×	将指令分解为具体运算类型、寄存器、立即数等具体部分
IDSUE	√	接收、解析指令（寄存器改名），并选择空闲的处理单元发射出去
ALU	×	接收运算任务，运算，提交至 ROB
MEM	×	接收存储/读取任务，结束后提交至 ROB
ROB	√	依次向 IDSUE 提交运算结果修改寄存器，并对结果广播
MMU	×	作为 CPUCORE 和外层的中继，修改访问格式

设计特点

1. 类似 Tomasulo，允许超标量

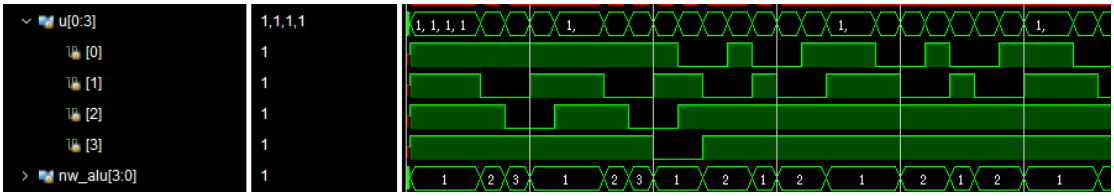

图 2：连接 3 个 ALU（图中[1][2][3]）和 1 个 MEM（图中[0]）时工作情况（低电平为工作）

2. 预约栈合并进入 ALU，ALU 监听 ROB 广播来获取未知操作数

3. “ROB”不再是所谓 Buffer，而是通过循环编号（1-f）来与 IDSUE 同步

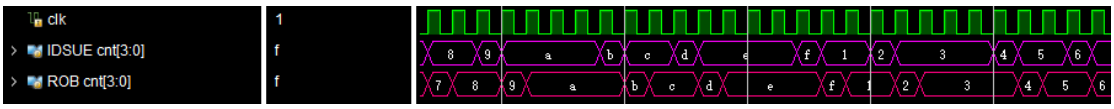

图 3：两阶段内置计数器进行同步，舍弃 Buffer 表项

4. 跳转通过 IDSUE 暂停流水线和向 ROB 报 Exception 完成，由 ROB 向 PC_REG 提交新 pc

心得体会

1. **技能生疏**：verilog 语言不熟，不熟悉硬件设计思路，烧板子之前没做过

2. **环境配置**：linux 下板子不识别，VIVADO 装了好几遍

3. **结构不够简化**：控制线思路上会乱，总会遗漏某些情况

4. **没有自己实现 UART**：由于通信不是很了解，所以直接套了助教代码的 CPU_core

5. **缺少设计经验**：经历了板子 LUT 不够（25000+>20800）以至于砍设计；经历了不满足时间要求（WNS -3ns 左右）以至于降频到 50MHz。

Name	Slice LUTs
cpu	25498
UART (uart_comm)	205
MEM_CTRL	91
d7s (display_7seg)	73
CORE (CPUCORE)	24606
_rob (ROB)	924
_pcreg (PCREG)	66
_mmu (MMU)	104
_mem (MEM)	216
_ifid (IFID)	1544
_if (IF)	11
_idsue (IDSUE)	18971

_id (ID)	693
_alu_2 (ALU_2)	659
_alu_1 (ALU_1)	781
_alu_0 (ALU)	677
COMM (multchan_comm)	521
clk (clk_wiz_0)	0

图 4：初稿使用 LUT 超出板子 20800 的限制

6. **时间不是很够：**基本上把整个结构思路理清的时候已经考试周了，没有足够的时间进行仔细的调试和出数据，也没有时间完整的学习助教的通信方式导致踩了不少坑。板子上最后也没有调对，十分遗憾。

参考资料以及感谢

感谢范舟同学和吴章昊同学提供的测试数据；感谢张凯羿同学在二进制文件方面提供的帮助；感谢孙雪晖同学在烧板子方面提供的协助；感谢其他同学在我完成该项作业时的支持

参考资料如下：

RISC-V 官方文档 https://riscv.org/specifications/

烧板子相关 https://wenku.baidu.com/view/c9486ce6a45177232f60a2f6.html

Zzk 助教的 mipsCPU https://github.com/sxtyzhangzk/mips-cpu

《自己动手写 CPU》 雷思磊

《自己设计制作 CPU 与单片机》 姜咏江

《深入理解 OpenRISC 体系结构》 甄建勇

《圈圈教你玩 USB》 刘荣