流水线 MIPS 处理器加分申请与设计说明

郝千越 2018011153 无 85

目录

1 性能指标

表 1 处理器性能指标

性能指标	测试值
设计名称	基于 FPGA 的五级流水线 32 位 MIPS 处理器
指令集	MIPS 核心指令集（31 条）
主频	118.5MHz
CPI（基于冒泡排序程序测试）	1.187
每秒指令数	99.8Million
数据存储器容量	1KB
支持异常类型	未定义指令
支持中断类型	定时器中断
支持 forwarding 类型	MEM/WB→EX EX/MEM→EX
支持外设	定时器 4*七段数码管 8*LED 系统时钟计数器

2 主要设计优化

2.1 分支指令控制模块

本设计中开始时将 beq/bne/blez/bgtz/bltz 全部集成在 ALU 模块中，同时扩展 ALU 的两级控制信号，实现对这五条指令分支条件是否成立的判断。综合、实现后发现该路径逻辑过于复杂，延时较长，阻碍时钟频率的提升。

故将分支跳转控制的功能分离，形成单独的 Branch 模块，由 Control 模块在 ID 阶段直接生成控制信号 BranchOp 简化了相应的控制逻辑，加快执行速度。

同时，ALU 模块的输入还可能来自立即数扩展或指令中的偏移量字段，而 Branch 模块无此来源。因此，分离 Branch 模块还可以简化输入选择逻辑，进一步缩短路径。

2.2 转发控制单元

初始设计中，Forwarding 模块按照理论课知识设计在 EX 阶段，但通过分析时序报告发现涉及转发操作时，EX 阶段需要执行如下步骤：根据各寄存器由组合逻辑生成转发控制信号；由 MEM 或 WB 阶段转发数据；ALU 或 Branch 模块对来的数据进行运算处理。这一系列操作耗时很长，极大限制了处理器的时钟频率。

因此在此处做优化设计：将 Forwarding 模块提前至 ID 阶段，其输入各信号均对应提前一个流水级；将 Forwarding 模块生成的控制信号存入 ID/EX 寄存器，在 EX 阶段控制转发。由此使得 ID 阶段分担 EX 阶段的工作负担，提高流水线时钟频率。

2.3 RegisterFile 写入

一般流水线设计中，在 WB 阶段首先由 MemtoReg 信号控制选出要回写的数据，再于时钟下降沿将数据写回寄存器中。这种方式存在弊端，由于寄存器堆需要支持先写后读，时钟下降沿写入数据后，留给数据读取操作的时间小于 1/2 个时钟周期，不利于提高时钟频率。

本设计中针对此问题进行优化：在 MEM 阶段提前由 MemtoReg 信号选取要回写的数据，选出要回写的指令后，本设计在 MEM 阶段增加三个寄存器，分别存储寄存器写使能信号 RegWrite、回写寄存器编号、回写数据。由此，下一个时钟周期开始时，指令进入 WB 阶

段，回写所需数据已经全部准备完毕，可以在时钟上升沿直接写入寄存器堆。这一设计对提升时钟频率很有帮助。

3 设计辅助工具

为方便流水线以及相关汇编代码的设计和编写，实验中特别编写了两个 Python 脚本，用于生成和转换相关代码：

- randomint.py：生成随机数，并输出为对应个 sw 指令，便于写入汇编语言代码之中，而后将随机数存入流水线处理器的数据存储器中；
- convert.py：将 MARS 导出的十六进制机器码转换为对应的 Verilog 代码，便于写入指令存储器的 ROM 之中。

利用这两个 Python 脚本工具，可以较好地在汇编语言、硬件描述语言之间进行转换，极大便利了流水线处理器的设计。