Design of High-Speed Floating-Point Unit for RISC-V Processor*

CHANG Longxin, YU Zhiguo, ZHONG Xiaoyu, GU Xiaofeng*
(*Engineering Research Centre of IoT Technology Applications (Ministry of Education), Jiangnan University, Wuxi Jiangsu* 214122, *China*)

Abstract: Floating-point unit is one of the speed bottlenecks of high-performance processor. Based on the widely used open-source RISC-V floating-point unit prototype, a high-speed floating-point unit for RISC-V processor is designed. Static timing analyses are performed for the submodules which have worst timing in prototype, including the floating-point fused multiply-add, integer-to-floating-point, floating-point divider and sqrt submodules. Next, the critical modules most in need of optimization are located. Then, the design methods are proposed based on algorithm optimization and pipeline optimization. A radix-4 Booth-Wallace multiplication module is designed to replace the original multi-bit wide multiplication algorithm. A parallel leading zero detection algorithm based on binary tree is designed to replace the original serial leading zero detection algorithm, and the pipeline stages of some submodules are increased. Based on the SMIC 55 nm technology, the performance of RISC-V floating-point unit prototype before and after the optimization are evaluated. The operating frequency reaches 820 MHz after optimization, increased by 39.46%, and the area is increased by 15.14%.

Key words: RISC-V; floating-point unit; radix-4 Booth-Wallace multiplier; parallel leading zero detection
EEACC: 1265A **doi**: 10.3969/j.issn.1005-9490.2022.06.003

面向 RISC-V 处理器的高速浮点单元设计*

常龙鑫, 虞致国*, 钟啸宇, 顾晓峰
(江南大学物联网技术应用教育部工程研究中心, 江苏 无锡 214122)

摘 要: 浮点单元是高性能处理器的速度瓶颈之一, 基于广泛应用的开源 RISC-V 浮点单元原型, 设计了一种面向 RISC-V 处理器的高速浮点单元。对该原型中时序最差的浮点融合乘加、整数转浮点、除法开方子模块分别进行静态时序分析, 并定位其中需要优化的关键模块。针对该浮点单元原型中存在的问题, 提出基于算法优化和流水线优化的设计思路, 设计基 4 Booth-Wallace 乘法模块替代原有多位宽乘法模块, 设计基于二叉树的并行前导零检测模块替代原有串行前导零检测模块, 增加了部分子模块的流水线级数。基于 SMIC 55 nm 工艺对优化设计前后的 RISC-V 浮点单元原型进行了性能评估, 优化后的工作频率达到 820 MHz, 提升了 39.46%, 而面积开销增加了 15.14%。

关键词: RISC-V; 浮点单元; 基 4 Booth-Wallace 乘法; 并行前导零检测
中图分类号: TP302.2 **文献标识码**: A **文章编号**: 1005-9490(2022)06-1289-07

近年来, 由加州大学伯克利分校设计的 RISC-V 指令集获得了学界和业界的广泛关注。目前已有多款 RISC-V 处理器被研发和应用, 如 Rocket、BOOM、XuanTie C910、PULP 系列处理器等[1-3], 面向 RISC-V 的软件生态环境也越来越完善。由于具有精度高、表示范围广等优势, 浮点运算已成为图像和视频处理、导航定位、科学模拟等高性能计算领域的重要组成部分[4-5], 浮点运算的性能成为了高性能处理器的速度瓶颈之一。

在面向 RISC-V 处理器的浮点单元(Float Point Unit, FPU)研究中, 潘树朋等[6]设计了一款符合 IEEE 754 标准的低功耗 RISC-V FPU, 并将其集成于开源处理器蜂鸟 E203 上, 该 FPU 在 FPGA 上的工作频率可达 100 MHz[7]。Tiwari 等[8]设计了一款基于 Posit 编码的 RISC-V FPU, 在 FPGA 上能够达到 100 MHz。PULP 是一款面向低功耗领域的开源

项目来源: 江苏省重点研发计划(BE2019003-2); 中央高校基本科研业务费专项资金资助(JUSRP51510)
收稿日期: 2021-07-06 **修改日期**: 2021-12-10

RISC-V 处理器,在 TSMC 55 nm 工艺下,其 FPU 的工作频率可达 140 MHz[9]。U540 是一款 RISC-V SoC,在 TSMC 28 nm 工艺下,其 FPU 的工作频率可达 1.5 GHz[10]。现有研究中的 RISC-V FPU 主要面向高能效、低功耗、高精度等领域,工作频率较低,对高性能 RISC-V FPU 依然缺乏深入研究。

开源的 hardfloat 仓库是一个被广泛应用的开源 RISC-V FPU 原型[11],由 UCB 基于 Chisel 语言编写而成[12],Rocket、BOOM 等 RISC-V 处理器均采用该原型作为其 FPU。本文将对 hardfloat 仓库提供的 RISC-V FPU 原型进行研究,分析其性能瓶颈,并从算法和流水线两个方面,提出优化设计思路,以实现一种面向 RISC-V 处理器的高速浮点单元。

1 RISC-V FPU 原型研究

对 RISC-V FPU 原型中时序最差的子模块进行静态时序分析,以研究该原型当前存在的问题。

1.1 架构解析

RISC-V FPU 原型来自 hardfloat 仓库,该原型实现了 RISC-V 标准扩展指令子集 F 和 D,共含有 8 个子模块。HDL 代码中各个子模块的实例名、含义及其实现的主要功能如表 1 所示。

表 1 RISC-V FPU 原型的子模块

实例名	含义	实现的主要功能
dfma	双精度浮点融合乘加模块	双精度浮点加法、乘加
sfma	单精度浮点融合乘加模块	单精度浮点加法、乘加
ifpu	整数转浮点模块	整数转浮点
divSqrt_1	双精度浮点除法开方模块	双精度浮点除法、开方
divSqrt	单精度浮点除法开方模块	单精度浮点除法、开方
fpiu	浮点转整数模块	浮点分类、浮点比较、浮点转整数
fpmu	浮点转换模块	浮点数符号注入、取最大最小值、浮点精度转换

1.2 问题分析

RISC-V FPU 原型具有众多子模块。为了有针对性地进行优化设计,首先使用 Design Compiler 工具对 RISC-V FPU 原型进行逻辑综合,以筛选出时序最差的子模块。

基于 SMIC 55 nm 工艺和(TT,1.2 V,25 ℃)工艺角,在使用 BRT(Behavioral Retiming)编译策略的情况下,对 RISC-V FPU 原型进行逻辑综合,其输入输出延时(input delay 和 output delay)均设置为时钟周期的 25%,时钟不确定性设置为时钟周期的 10%。此时,时钟频率可达 588 MHz,对应的时钟周期为 1.7 ns。为了更好地识别设计中存在的问题,在不采用 BRT 编译策略的情况下,再次以 1.7 ns 作为时钟约束进行逻辑综合。此时,时序最差的路径主要分布于 5 个子模块中:dfma、sfma、ifpu、divSqrt_1、divSqrt。

1.2.1 浮点融合乘加模块

图 1 展示了在 1.7 ns 时钟约束下,dfma 模块中 3 条代表性的关键路径,其中逻辑电路下方的数字是路径在该逻辑电路中经过的延时。

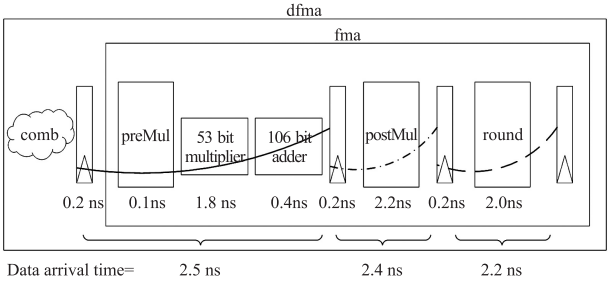


图 1 dfma 模块中的 3 条代表性关键路径

①关键路径 1

根据图 1 的时序分析可知,关键路径 1 在 53 位乘法器中的延时占据了总数据到达时间(Data Arrival Time)的 72%,可见该 53 位乘法器的时序需要被改善。

②关键路径 2

根据图 1 的时序分析可知,关键路径 2 在 postMul 中的延时占据了总数据到达时间的 92%,可见 postMul 模块的时序需要被改善。

③关键路径 3

根据图 1 的时序分析可知,关键路径 3 在 dfma 舍入模块中的延时占据了总数据到达时间的 91%,可见舍入模块的时序需要被改善。

sfma 模块的分析过程与 dfma 模块类似。基于上述分析,浮点融合乘加模块中需要改善时序的关键模块如表 2 所示。

表 2 浮点融合乘加模块中的关键模块

浮点融合乘加模块	关键模块
dfma	53 位乘法模块 postMul 模块 舍入模块
sfma	24 位乘法模块 postMul 模块 舍入模块

1.2.2 整数转浮点模块

图 2 展示了在 1.7 ns 时钟约束下,穿过 ifpu 模

块的 1 条代表性关键路径。根据时序分析可知,该时序路径在 INToRecFN_1 模块中的延时占据了总数据到达时间的 92%,而 INToRecFN_1 中时序路径主要经过一个前导零检测模块和一个舍入模块,可见前导零检测模块和舍入模块的时序需要被改善。

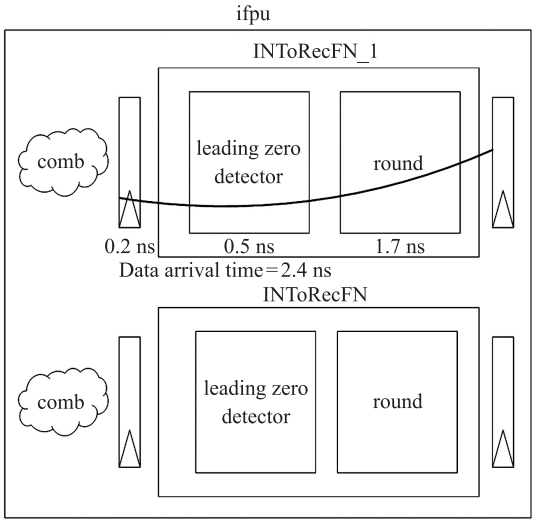


图 2　ifpu 模块中的代表性关键路径

INToRecFN 模块的分析过程与 INToRecFN_1 模块类似。基于上述分析,整数转浮点模块中需要改善时序的关键模块如表 3 所示。

表 3　整数转浮点模块中的关键模块

整数转浮点模块		关键模块
ifpu	INToRecFN_1	前导零检测模块
		舍入模块
	INToRecFN	前导零检测模块
		舍入模块

1.2.3　浮点除法开方模块

图 3 展示了在 1.7 ns 时钟约束下,穿过 divSqrt_1 模块的 1 条代表性关键路径。根据时序分析可知,该时序路径在 divSqrt_1 的舍入模块中的延时占据了总数据到达时间的 92%,可见该舍入模块的时序需要被改善。

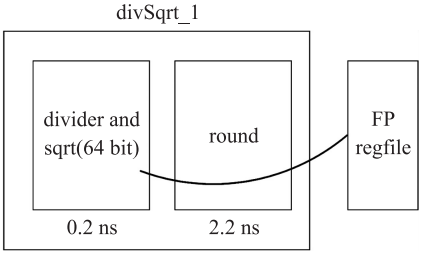


图 3　divSqrt_1 模块中的代表性关键路径

divSqrt 模块的分析过程与 divSqrt_1 模块类似。基于上述分析,浮点除法开方模块中需要改善时序的关键模块如表 4 所示。

表 4　除法开方模块中的关键模块

除法开方模块	关键模块
divSqrt_1	舍入模块
divSqrt	舍入模块

2　关键模块设计

2.1　浮点融合乘加模块

乘法器尤其是多位宽的乘法器通常具有较长的时序延时。RISC-V FPU 原型的 Chisel 代码中没有对 dfma 中 53 位的乘法器进行底层描述,而是仅使用了一个“*”号来实现,将具体实现方法交给逻辑综合工具来完成,这种做法难以保证该乘法器能够达到最优性能。为了提升该乘法器的性能,本文将采用“基 4 Booth-Wallace”乘法器(B-W 结构乘法器)[13-14]替代原有乘法器,并在 53 位乘法器中增加 2 级寄存器,构成 3 周期的 53 位乘法器。

与 dfma 不同的是,sfma 需要优化的乘法器模块是 24 位的,即需要设计 24 位的 B-W 结构乘法模块替换当前乘法模块。另外,单精度运算相对于双精度运算来说较为简单,sfma 模块的时序比 dfma 的时序宽松,因此 sfma 中的 24 位乘法器仅需增加 1 级流水线,构成 2 周期的 24 位乘法器。

由于 dfma 和 sfma 的 postMul 模块内部除了一个时延并不突出的加法器外,均没有其他的规整的子模块,决定采用增加 1 级流水线的方式优化 postMul 模块的时序;由于 dfma 和 sfma 的舍入模块内部均没有规整的子模块,决定采用增加 1 级流水线的方式优化舍入模块的时序。表 5 汇总了浮点融合乘加模块的关键模块及其设计方法。

表 5　浮点融合乘加模块的关键模块及其设计方法

关键模块		设计方法
dfma	53 位乘法模块	替换为基 4 B-W 乘法模块,增加 2 级流水线
	postMul 模块	增加 1 级流水线
	舍入模块	增加 1 级流水线
sfma	24 位乘法模块	替换为基 4 B-W 乘法模块,增加 1 级流水线
	postMul 模块	增加 1 级流水线
	舍入模块	增加 1 级流水线

2.1.1　整数转浮点模块

对于 dfma 中的 53 位乘法,基 4 Booth 算法能够将原本 53 个部分积转换为 27 个部分积,另外还需要增加额外的位来确保符号位的正确性,最终 Booth 变换将得到 28 个部分积;得到 28 个部分积后,还需要应用 7 层 Wallace 树型压缩;最后,根据优化方

案,再在 Wallace 树内增加 2 层寄存器,构成 3 级流水线的乘法器。最终编写的 53 位乘法器的结构如图 4 所示,右侧的数字代表每层部分积的个数,每条虚线均代表新增的 1 层寄存器。

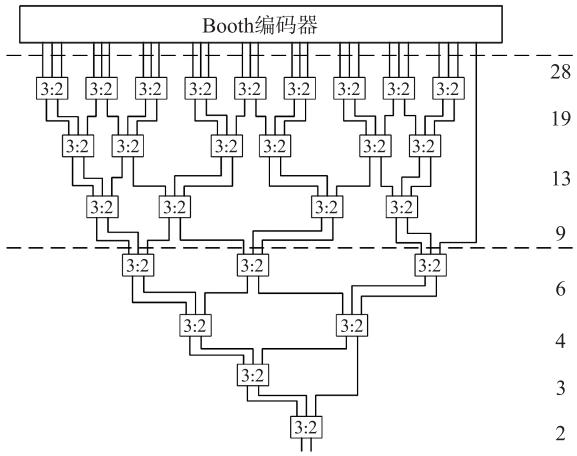


图 4 3 级流水线的 53 位乘法器结构

同理,对于 sfma 中的 24 位乘法,最终 Booth 变换将得到 13 个部分积;得到 13 个部分积后,还需要应用 5 层 Wallace 树型压缩;最后,根据优化方案,再在 Wallace 树内增加 1 层寄存器,构成 2 级流水线的乘法器。最终编写的 24 位乘法器的结构如图 5 所示,每条虚线均代表新增的 1 层寄存器。

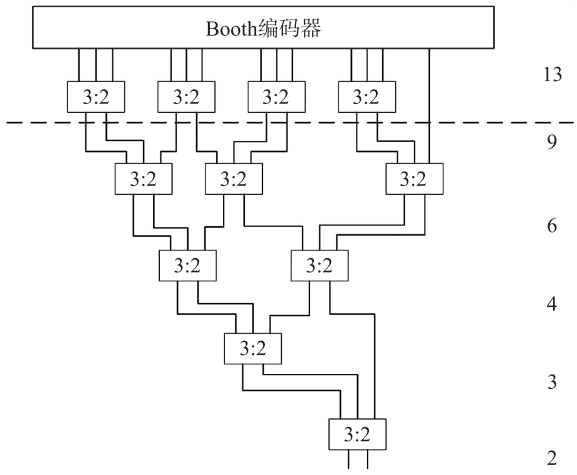


图 5 2 级流水线的 24 位乘法器结构

2.1.2 postMul 模块

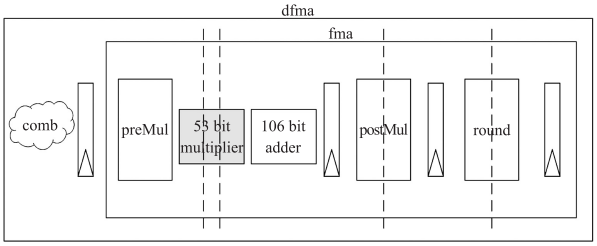
在 postMul 模块的 Chisel 代码中,为所有时序相同的信号插入 1 级寄存器,dfma 与 sfma 中针对 postMul 模块的时序优化方式相同。

2.1.3 舍入模块

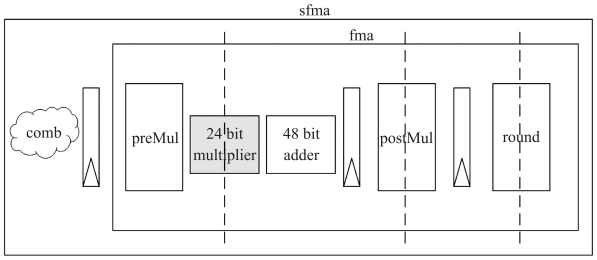
在舍入模块的 Chisel 代码中,为所有时序相同的信号插入 1 级寄存器,sfma 中针对舍入模块的时序优化方式相同。

优化后的 dfma 与 sfma 模块的示意图分别如图 6(a)、图 6(b)所示,其中灰色模块是进行了算法优化的模块,每条虚线均代表新增的 1 层寄存器。

性能优化后,dfma 和 sfma 分别增加了 4 级和 3 级流水线,分别增加了 4 个和 3 个周期的延时。



(a) 优化后的dfma模块



(b) 优化后的sfma模块

图 6 优化后的 dfma 和 sfma 模块

2.2 整数转浮点模块

整数转浮点模块中的前导零检测模块的功能是检测一个二进制序列第一个有效位之前有多少个零。图 2 中 ifpu 的关键路径经过了 64 位前导零检测模块以及舍入模块。在原有的 ifpu 中,前导零检测模块是通过二进制序列翻转器和优先编码器共同实现的,其中,二进制序列翻转器能够输出二进制序列的逆序二进制序列,优先编码器能够定位一个二进制序列中最低有效位的位置。这种方法实现的前导零检测模块具有串行的结构,会带来较大的延时,而且不规整的结构使得同时截取该模块中时序相同的所有信号线变得困难。

鉴于此,本文决定采用基于二叉树的并行前导零检测算法[15]代替现有的前导零检测算法。将原有的串行前导零检测模块替换为基于二叉树的并行前导零检测模块后,该模块的结构将变得规整,有利于同时捕捉同一时序的所有信号线。另外,为了进一步提高工作频率,决定在前导零检测模块与舍入模块之间增加 1 级流水线,在舍入模块内也增加 1 级流水线。表 6 汇总了整数转浮点模块的关键模块及其设计方法。

2.2.1 前导零检测模块

针对整数转浮点模块中的前导零检测模块,采用基于二叉树的前导零检测算法替代现有前导零检测算法,以便获得并行性更高、信号线更规整的前导

零检测模块。最终编写的基于二叉树的 64 位前导零检测模块的结构如图 7 所示。

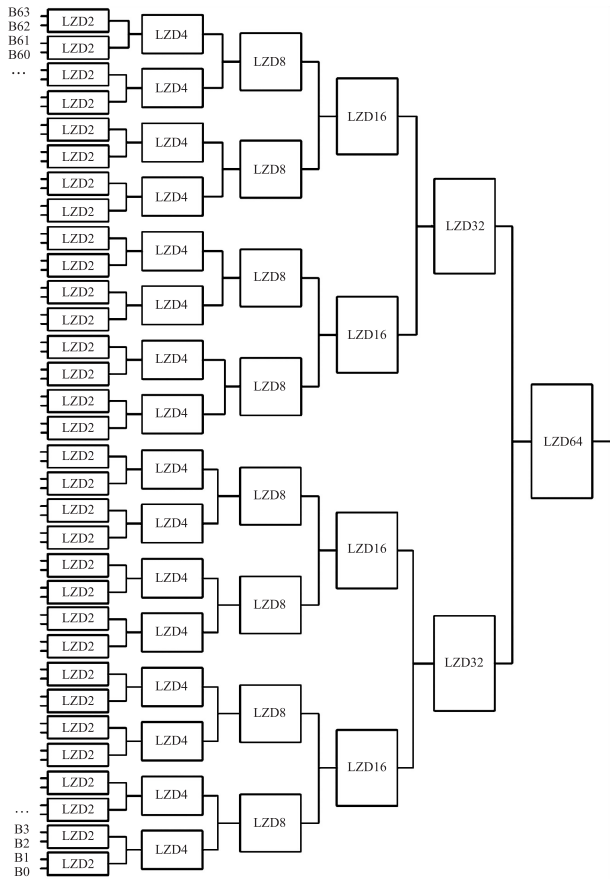


图 7　基于二叉树的 64 位前导零检测模块

优化后的前导零检测模块具有并行结构,而且信号线清晰规整,为在前导零检测模块与舍入模块之间插入 1 级寄存器提供了极大的方便。

表 6　整数转浮点模块的关键模块及其设计方法

关键模块		设计方法
INToRecFN_1	前导零检测模块	采用二叉树前导零检测模块,前导零检测模块与舍入模块之间增加 1 级流水线
	舍入模块	增加 1 级流水线
INToRecFN	前导零检测模块	采用二叉树前导零检测模块,前导零检测模块与舍入模块之间增加 1 级流水线
	舍入模块	增加 1 级流水线

2.2.2　舍入模块

在舍入模块的 Chisel 代码中,为所有时序相同的信号插入 1 级寄存器。

至此,ifpu 模块的关键模块设计已完成,优化后的 ifpu 模块如图 8 所示,其中灰色模块是进行了算法优化的模块,每条虚线均代表新增的 1 层寄存器。

性能优化后,ifpu 模块增加了 2 级流水线,增加了 2 个周期的延时。

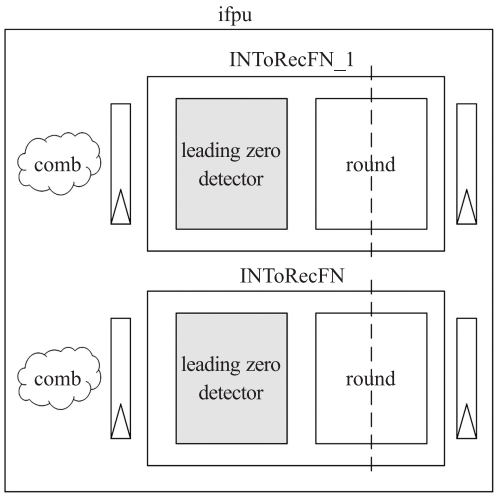


图 8　优化后的 ifpu 模块

2.3　浮点除法开方模块

由于 divSqrt_1 和 divSqrt 的舍入模块内部均没有规整的子模块,本文决定采用增加 1 级流水线的方式优化浮点除法开方模块的时序。

在浮点除法开方的舍入模块 Chisel 代码中,为所有时序相同的信号插入 1 级寄存器,divSqrt_1 与 divSqrt 中针对舍入模块的时序优化方式相同。优化后的 divSqrt_1 模块如图 9 所示,其中,虚线代表新增的 1 层寄存器。

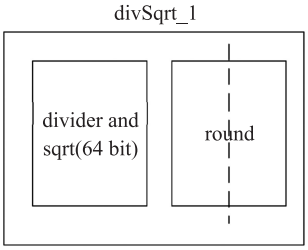


图 9　优化后的 divSqrt_1 模块

性能优化后,divSqrt_1 和 divSqrt 模块均增加了 1 级流水线和 1 个周期的延时。

3　功能验证

Rocket、BOOM 等处理器集成了开源的 RISC-V FPU 原型[2-3],本文将基于 Rocket 处理器验证优化后 FPU 原型的功能。

Rocket 处理器主要由 Chisel 语言编写的,本文执行的优化设计过程中,除了 53 位 B-W 乘法器和 24 位 B-W 乘法器采用 Verilog 编写以外,对其他模块所做的优化设计均基于 Chisel 语言实现。借助编译工具,处理器的 Chisel 代码最终能转化为 Verilog 代码,基于处理器的 Verilog 代码和 VCS(Verilog Compile Simulator)工具,能为处理器构建出一个周

期精确的 C/C++的模拟器。

RISC-V 指令测试工具 riscv-tests[16] 是 RISC-V 基金会维护的开源项目,包含大量面向 RISC-V 标准扩展指令的自测试程序。自测试程序含预设结果,如果指令执行结果与预设结果相等,则说明指令的功能是正确的,否则说明指令功能存在问题。

基于 VCS 模拟器和 riscv-tests 仓库,可以对 RISC-V 处理器进行指令级别的功能测试,测试过程如图 10 所示。基于此方法对本设计进行指令测试,测试结果如表 7 所示,结果均符合预期。

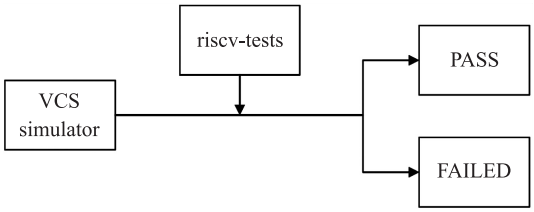


图 10 RISC-V 处理器的指令测试过程

表 7 指令测试结果

测试目录	测试内容	结果
rv64mi	机器模式整数指令	通过
rv64si	管理员模式整数指令	通过
rv64ua	用户模式原子指令	通过
rv64uc	用户模式压缩指令	通过
rv64ud	用户模式双精度浮点指令	通过
rv64uf	用户模式单精度浮点指令	通过
rv64ui	用户模式整数指令	通过
rv64um	用户模式乘除指令	通过

4 性能评估

本文采用 Design Compiler 工具评估基于 ASIC 的性能,采用的工艺库为 SMIC 55 nm RVT 标准单元库,工艺角为(TT,1.2 V,25 ℃),输入输出延时(input delay 和 output delay)均设置为时钟周期的 25%,时钟不确定性设置为时钟周期的 10%。

基于 ASIC 对本设计与现有技术的性能进行评估,其工作频率对比如表 8 所示。RISC-V FPU 原型在 ASIC 设计中的工作频率为 588 MHz,对关键模块进行优化设计后,工作频率能达到 820 MHz,性能提升了 39.46%。表 8 中,U540 FPU[10] 采用的是 28 nm 工艺,本设计采用的是 55 nm 工艺,根据按比例缩放的原理,本设计的性能不低于 U540 FPU。

表 8 还列出了完成一次浮点乘法运算所需要的延时,以展示增加流的水线级数对运算延时的影响。其中,U540 FPU[10] 具有 5 级整数流水线,但其浮点流水线的级数尚未被公开。

表 8 RISC-V FPU 的 ASIC 性能对比

	文献[9]	原型[2]	本文	文献[10]
工艺	TSMC 55 nm	SMIC 55 nm	SMIC 55 nm	TSMC 28 nm
工作频率	140 MHz	588 MHz (typical)	820 MHz (typical)	1.5 GHz (typical)
浮点乘法运算的延时	2(single)	2(single) 3(double)	6(single) 7(double)	N/A

值得指出的是,对关键模块进行优化设计之前,RISC-V FPU 原型面积是 126 456.70 μm^2;优化设计之后,面积增至 145 605.60 μm^2,增加了 15.14%。

5 结束语

基于对开源 RISC-V FPU 原型的优化设计,实现了一种高速 RISC-V 浮点单元,能够用于对浮点运算性能要求更高的场景。基于静态时序分析,对原型中时序最差的 5 个子模块进行了静态时序分析,研究了其性能瓶颈,并针对关键模块提出了基于算法优化和流水线优化的设计方法。ASIC 的综合结果表明,对关键模块设计优化后,最大工作频率提高到 820 MHz,性能提升了 39.46%,面积增加了 15.14%。

参考文献:

[1] Asanovic K,Avizienis R,Bachrach J. The Rocket Chip Generator,UCB/EECS - 2016 - 17 [R]. Berkeley: EECS Department,University of California,Berkeley,2016.

[2] Chipsalliance. Rocket Chip Generator[EB/OL]. (2021)[2021-6-8]. https://github.com/chipsalliance/rocket-chip.

[3] Celio C,David A P,Asanovic K. The Berkeley Out-of-Order Machine(BOOM): An Industry-Competitive,Synthesizable,Parameterized RISC-V Processor,UCB/EECS-2015-167[R]. Berkeley: EECS Department,University of California,Berkeley,2015.

[4] 黄兆伟,王连明. 基于 FPGA 的可配置浮点向量乘法单元设计实现[J]. 计算机应用研究,2020,37(9):2762-2765,2771.

[5] 栗学磊,朱效民,魏彦杰,等. 神威太湖之光加速计算在脑神经网络模拟中的应用[J]. 计算机学报,2020,43(6):1025-1037.

[6] 潘树朋,刘有耀,焦继业,等. 基于 RISC-V 浮点指令集 FPU 的研究与设计[J]. 计算机工程与应用,2021,57(3):80-86.

[7] SI-RISC-V. Hummingbirdv2 e203 Core and SoC[EB/OL]. 2020[2021-6-8]. https://github.com/riscv-mcu/e203_hbirdv2.

[8] Tiwari S,Gala N,Rebeiro C,et al. PERI: A Configurable Posit Enabled RISC-V Core[J]. ACM Transactions on Architecture and Code Optimization. 2021,18(3):1-26.

[9] Pulp-Platform. FPU[EB/OL]. 2017[2021-4-29]. https://github.com/pulp-platform/fpu.

[10] Sifive. Freedom unleashed FU540[EB/OL]. 2018[2021-6-8]. https://www.sifive.com/chip-designer#fu540.

[11] UC Berkeley Architecture Research. Berkeley Hardware Floating-Point Units[EB/OL]. 2020[2021-6-8]. https://github.com/

ucb-bar/berkeley-hardfloat.

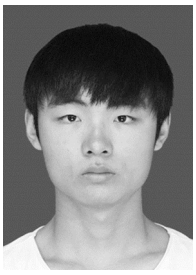
[12] Bachrach J, Vo H, Richards B, et al. Chisel: Constructing Hardware in a Scala Embedded Language[C]//DAC Design Automation Conference, San Francisco, CA, USA, IEEE, 2012: 1212-1221.

[13] Macsorley O L. High-Speed Arithmetic in Binary Computers[J]. Proceedings of the IRE, 1961, 49(1): 67-91.

[14] Wallace C S. A Suggestion for a Fast Multiplier[J]. IEEE Transactios on Electronic Computers, 1964, 13(1): 14-17.

[15] Oklobdzija V G. An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis[J]. IEEE Transactios on Very Large Scale Integration(VLSI) Systems, 1994, 2(1): 124-128.

[16] RISCV. Riscv-tests[EB/OL]. 2020[2021-6-8]. https://github.com/riscv/riscv-tests.



常龙鑫(1997—),男,汉族,河南新乡人,硕士生,主要研究方向为集成电路设计与应用;

虞致国(1979—),男,汉族,江西万年人,博士,副教授,主要研究方向为集成电路设计与测试,yuzhiguo@jiangnan.edu.cn;

钟啸宇(1996—),男,汉族,江苏徐州人,硕士,主要研究方向为微处理器的设计与优化;

顾晓峰(1971—),男,汉族,江苏无锡人,博士,教授,主要研究方向为新型半导体材料与器件,电子系统设计与应用。