温州大學

WENZHOU UNIVERSITY

本科毕业设计 (论文)

基于 RISC-V 的处理器仿真设计

教学单位: 计算机与人工智能学院

专业名称: 网络工程

学　　号: 21211835111

学生姓名: 卢宇鑫

指导教师: 金可仲

指导单位: 计算机与人工智能学院

完成时间: 2025 年 4 月 13 日

摘要

本文提出了一种基于 RISC-V 架构的处理器仿真设计方法，专注于 RV32E 指令集的高效实现。研究从处理器核心的内部结构出发，系统地介绍了取指单元、译码单元、计算单元、访存单元和写回单元的设计细节。通过合理连接这些关键单元，构建了一个简单的单周期处理器架构。在此基础上，引入握手信号以优化时序逻辑，进一步将单周期处理器升级为多周期处理器，显著提升了处理器的时序灵活性与性能表现。

进一步地，本文采用 ARM 的 AXI4 总线协议，实现了处理器与外部设备的高效连接，并成功将其集成到片上系统（SoC）中。为了进一步优化处理器性能，处理器引入了流水线技术和高速缓存技术，显著提升了处理器的并行处理能力和内存访问速度，使其能够更好地满足现代计算任务的需求。

在验证环节，本文详细探讨了处理器的模拟仿真过程。通过使用开源工具 Verilator 将 RTL 代码编译为 C++ 程序并进行仿真运行，确保了仿真的高效性和灵活性。同时，研究使用一种高效的差分测试方法（Differential testing），通过与行为模拟器的状态比对，实现了测试的高效性和准确性，从而确保了设计的高效性和可靠性。

综上所述，本文的研究为 RISC-V 架构处理器的设计与验证提供了一种系统化、高效化的方法，具有重要的理论意义和实践价值，为后续的处理器设计与优化工作奠定了坚实的基础。

关键词： RISC-V；体系结构；性能优化；设计方法

Abstract

This paper presents a processor simulation design method based on the RISC-V architecture, focusing on the efficient implementation of the RV32E instruction set. Starting from the internal structure of the processor core, the study systematically introduces the design details of the fetch unit, decode unit, execution unit, memory access unit, and write-back unit. By properly connecting these key units, a simple single-cycle processor architecture is constructed. On this basis, handshake signals are introduced to optimize timing logic, further upgrading the single-cycle processor to a multi-cycle processor, which significantly enhances the processor's timing flexibility and performance.

Furthermore, this paper employs ARM's AXI4 bus protocol to achieve efficient connections between the processor and external devices, successfully integrating it into a System on Chip (SoC). To further optimize the processor's performance, pipeline and high-speed cache technologies are introduced, which significantly improve the processor's parallel processing capabilities and memory access speed, enabling it to better meet the needs of modern computing tasks.

In the verification section, this paper thoroughly explores the simulation process of the processor. By using the open-source tool Verilator to compile RTL code into C++ programs and run simulations, the efficiency and flexibility of the simulation are ensured. Additionally, an efficient differential testing method is employed, comparing the states with those of a behavioral simulator to achieve high efficiency and accuracy in testing, thereby ensuring the efficiency and reliability of the design. In summary, this study provides a systematic and efficient method for the design and verification of RISC-V architecture processors, which holds significant theoretical and practical value and lays a solid foundation for subsequent processor design and optimization work.

Key Words: RISC-V; Architecture; Performance Optimization; Design Approach

目录

第 1 章 绪论

1.1 研究背景

芯片被誉为现代工业的掌上明珠，是信息时代的基石。自 1959 年世界上第一颗芯片诞生以来[1]，芯片技术以惊人的速度发展，推动了从个人计算机到智能手机、从数据中心到物联网设备的全面革新。半导体行业也因此成为当今世界最具战略意义和经济价值的行业之一。经过半个多世纪的发展，全球处理器架构市场逐渐形成了两大主流阵营：面向高性能计算的 X86 架构和面向嵌入式系统的 ARM 架构。然而，随着技术的不断演进和应用场景的多样化，现有的体系结构逐渐暴露出诸多问题：

1) **复杂指令集架构（CISC）的效率问题**：以 X86 为代表的复杂指令集架构虽然功能强大，但其指令集冗长且复杂，导致指令执行效率较低。为了实现复杂的指令功能，处理器需要集成更多的晶体管和电路，这不仅增加了芯片的设计难度和制造成本，还显著提高了功耗和发热量，限制了其在低功耗场景中的应用。

2) **闭源与授权限制**：X86 和 ARM 架构均属于闭源架构，其核心技术和指令集受到严格的版权保护。使用这些架构需要获得相应公司的授权许可，这不仅增加了研发成本，还限制了中小企业和新兴市场的准入，阻碍了技术的普及和创新。

3) **市场垄断与供应商锁定**：由于 X86 和 ARM 架构在各自领域的主导地位，相关技术的使用高度依赖于特定供应商的支持。这种市场垄断和供应商锁定的局面导致技术更新缓慢，用户选择受限，进一步抑制了行业的发展活力。

为了解决现有体系结构存在的问题，并顺应现代计算机体系结构设计的发展趋势，RISC-V（开源精简指令集架构）应运而生。RISC-V 以其开源、免费、开放和自由的特性，迅速成为全球学术界和工业界关注的焦点。任何个人或组织都可以自由使用、修改和分发 RISC-V 的设计，这为处理器架构的创新和普及提供了前所未有的机会。

RISC-V 的起源可以追溯到 20 世纪 80 年代初，当时加州大学伯克利分校的 David Patterson 教授和斯坦福大学的 John Hennessy 教授分别提出了精简指令集计算（RISC）理念[2]。RISC 的核心思想是通过简化指令集，使处理器设计更加高效、易于实现和优化。基于这一理念，伯克利分校开发了 RISC-I 和 RISC-II 原型机，成功验证了 RISC 架构的可行性和优越性。2010 年，加州大学伯克利分校的 Krste Asanović 教授及其团队启动了 RISC-V 项目，旨在设计一种全新的、开放的指令集架构，以满足现代计算的多样化需求[3]。2014 年，伯克利团队发布了 RISC-V 的初始规范，包括 32 位和 64 位的基本指令集[4]，这一规范的发布标志着 RISC-V 正式进入公众视野。

2015 年，RISC-V 基金会（RISC-V Foundation）正式成立，吸引了包括谷歌、英特尔、英伟达等全球顶尖科技公司以及众多学术机构和研究组织的加入。2017 年，首批基于 RISC-V 的商用芯片发布，展现了 RISC-V 在实际应用中的巨大潜力。2020 年，为了避免潜在的政治和法律风险，RISC-V 基金会迁至瑞士，并更名为 RISC-V International，进一步提升了 RISC-V 的国际化水平和开放性。

如今，RISC-V 已成为全球范围内最具活力的开源指令集架构之一，其应用领域涵盖

嵌入式系统、物联网、高性能计算、人工智能、数据中心等多个领域。RISC-V 的崛起不仅为处理器架构的设计和实现提供了新的思路，也为全球半导体行业注入了新的活力，推动了技术的民主化和创新生态的繁荣发展。

更重要的是，在中国的芯片产业中，主流 CPU 架构一直受西方国家的制约，整体的处理器行业生态与国际先进水平依旧存在不小的差距，RISC-V 架构的出现为中国开源芯片的发展提供了新的机遇。2021 年，开源首次被写入《中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要》[5]，全国各界都投入到对 RISC-V 开源生态的建设，对 RISC-V 的支持大幅度上升。例如，由多家 RISC-V 领域重点企业、研究机构、行业协会发起成立的 “中国 RISC-V 产业联盟”；由网信办、中科院等多个国家部委支持成立的 “中国开放指令生态 RISC-V 联盟”；由加州伯克利大学和清华大学合作发起成立的 “RISC-V 国际开源实验室”；由中科院、多家行业龙头企业和顶尖科研单位发起成立的 “北京开源芯片研究院” 等。以上例子说明在国家的政策支持下，各行各业的积极参与下，中国的 RISC-V 开源生态环境正在以不可阻挡的速度发展。

1.2 国内外研究现状

近年来，RISC-V 技术以其开源、模块化和可扩展的特性，在全球范围内引发了广泛关注，并取得了令人瞩目的研究成果。从高性能计算到嵌入式系统，从学术研究到商业应用，RISC-V 正在迅速崛起，成为处理器架构领域的重要力量。

2019 年，西部数据公司（Western Digital）推出了 SweRV 核心，这是一款高性能的 RISC-V 处理器核心，专为数据中心和存储应用设计，SweRV 的发布不仅展示了 RISC-V 在高性能计算领域的潜力，还标志着 RISC-V 从学术研究向工业应用的重大跨越[6]。2021 年，阿里巴巴旗下的平头哥半导体发布了玄铁 907 处理器，这是一款基于 RISC-V 架构的高性能处理器，已成功授权给多家企业使用，进一步推动了 RISC-V 的商业化进程[7]。2024 年，中国科学院计算技术研究所推出了 “香山” 昆明湖架构 V2，这是一款开源的高性能 RISC-V 处理器，其卓越的性能和创新的设计再次证明了 RISC-V 在技术创新上的巨大潜力[8]。

在系统级集成方面，Vedran Dakić 等人提出了一种异构 RISC-V SoC 设计，该设计集成了高性能的乱序核心、高能效的顺序核心以及专用加速器，充分展现了 RISC-V 在灵活性和可扩展性方面的优势[9]。此外，Koch 等人开发了针对 RISC-V 的 FPGA 框架 FABulous，为 RISC-V 处理器的快速原型设计和验证提供了高效的工具支持[10]。在设计方法学方面，钟等人提出了一种软硬件联合验证的设计方法，通过结合 Verilator 软件仿真和 FPGA 硬件验证，显著提高了 RISC-V 处理器的开发效率和可靠性[11]。在总线设计方面，郝等人采用 ARM 公司提出的 AHB 总线协议，成功实现了系统总线的设计，不仅保持了处理器的高性能，还显著减小了芯片的流片面积[12]。针对浮点数运算的挑战，潘等人提出了一种优化的浮点运算单元（FPU）设计，通过改进算法和硬件结构，显著提升了浮点数运算的效率和精度[13]。

1.3 技术对比

在现代处理器架构中，RISC（精简指令集计算）和 CISC（复杂指令集计算）是两种主要的设计理念。以 RISC-V、X86 和 ARM 为例，RISC-V 作为 RISC 架构的代表，展现出显著的优势，尤其是在开源性、模块化设计、可扩展性、功耗和性能等方面。

1) 在开源性方面，RISC-V 的完全开源特性使其在灵活性和可扩展性方面远超 X86 和 ARM。这种开源性不仅降低了开发成本，还允许开发者根据具体需求进行定制和优化，从而推动了创新和多样化应用的开发。相比之下，X86 和 ARM 均为非开源架构，需要授权才能使用，这在一定程度上限制了其在特定领域的应用。

2) 在处理器设计方面，RISC-V 的模块化设计和高可扩展性是其显著优势。它允许开发者根据不同的应用场景灵活选择和扩展指令集，从而更好地适应从嵌入式系统到高性能计算的多样化需求。而 X86 架构由于其复杂指令集和非模块化设计，在可扩展性方面表现较差，难以满足新兴应用的快速变化需求。

3) 在功耗方面，RISC-V 的设计理念使其在低功耗场景中表现出色。其简洁的指令集和高效的执行效率使得处理器能够在较低的功耗下运行，这对于移动设备和物联网应用尤为重要。相比之下，X86 架构由于其复杂的设计和较高的功耗，在移动和嵌入式领域存在明显劣势，尽管其在高性能计算中仍具有一定的优势。

4) 在性能方面，RISC-V 通过高效的指令执行和流水线设计，能够实现高性能处理。虽然 X86 和 ARM 在某些应用场景中也能提供高性能，但 RISC-V 的高效设计使其在功耗和性能的平衡上更具优势。此外，RISC-V 的开源性和模块化设计进一步增强了其在性能优化方面的潜力。

5) 在指令数量方面，与 X86 架构的指令集相比，RISC-V 的指令集格式更为简洁且高效。RISC-V 仅定义了六种指令格式，且每条指令长度固定为 32 位或 64 位，这极大地降低了指令译码时的复杂度和开销。相比之下，X86 架构由于其 CISC 的特性，指令长度不定，每次取指需按照最大指令字长读取，并在译码阶段进行分割，这无疑增加了指令处理的复杂度。此外，RV32I 的基础指令数量仅为 47 条，即使加上乘除法扩展（RV32M），指令总数也不超过 60 条。如图1-1所示，X86 架构在发布初期有 80 条指令，到 2015 年，其指令数量已增长至 1338 条，增加了 16 倍。

综上所述，RISC-V 作为 RISC 架构的代表，在开源性、模块化设计、可扩展性、功耗和性能等方面展现出显著的优势，使其在现代处理器架构中具有广阔的应用前景。

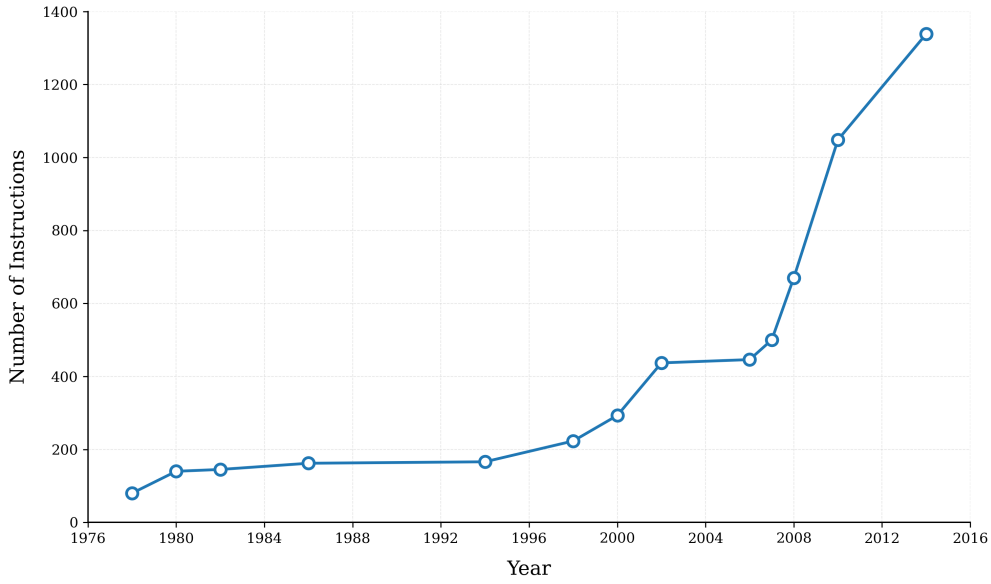

图 1-1 x86 指令集发展历程（1978-2014）

表 1-1 指令集对比

特性	RISC-V	X86	ARM
类型	RISC	CISC	RISC
开源性	开源	闭源	闭源
模块化	支持	不支持	不支持
扩展性	高	低	低
功耗	低	高	低
性能	高	高	高

1.4 本章小结

本章从研究背景、国内外研究现状以及技术对比三个方面对 RISC-V 架构进行了全面介绍。首先，通过对现有处理器架构（如 X86 和 ARM）的分析，指出了复杂指令集架构（CISC）的效率问题、闭源与授权限制以及市场垄断与供应商锁定等问题。这些问题的存在促使了开源精简指令集架构（RISC-V）的诞生。RISC-V 以其开源、免费、开放和自由的特性，迅速成为全球学术界和工业界关注的焦点，为处理器架构的创新和普及提供了前所未有的机会。

在国内外研究现状方面，近年来 RISC-V 技术取得了显著的进展。从嵌入式系统到高性能计算，从学术研究到商业应用，RISC-V 正在迅速崛起，成为处理器架构领域的重要力量。众多企业和研究机构纷纷推出了基于 RISC-V 的处理器和相关技术，进一步推动了 RISC-V 的商业化进程和技术创新。

在技术对比部分，通过对 RISC-V、X86 和 ARM 三种架构的详细对比，展示了 RISC-V 在开源性、模块化设计、可扩展性、功耗和性能等方面的优势。RISC-V 的完全开源特性降

低了开发成本，其模块化设计和高可扩展性使其能够灵活适应多样化的应用场景。在功耗方面，RISC-V 的设计理念使其在低功耗场景中表现出色，而其高效的指令执行和流水线设计则使其在性能上具有显著优势。

总的来说，RISC-V 具有极高的发展潜力，且逐渐成为 X86 和 ARM 双足鼎立之后的“第三极”。

第 2 章 RISC-V 处理器介绍

2.1 RISC-V 处理器指令集

(1) 指令集格式

RISC-V 基础指令集（Base ISA）作为 RISC-V 架构的核心，为处理器提供了基本的指令集框架。如表2-1和图2-1所示，RV32I 包含了六种基本指令类型：

(a) **R 型**：两个操作数来自寄存器，用于寄存器间操作；

(b) **I 型**：两个操作数来自立即数和寄存器，用于实现立即数和取数操作；

(c) **S 型**：两个操作数来自寄存器，一个操作数来自立即数，用于实现访存操作；

(d) **B 型**：两个操作数来自寄存器，一个操作数来自立即数，用于实现条件分支操作；

(e) **U 型**：一个操作数来自立即数，用于实现长立即数操作；

(f) **J 型**：一个操作数来自立即数，用于实现无条件跳转操作。

从指令格式就可以看出 RISC-V 指令集最大的优点——精简。首先，指令的长度都为 32 位，同一种类型的指令格式单一，大幅度减小了译码器的开销以及实现难度。其次，R 型指令提供了三个寄存器，这对于需要三个操作数的指令不需要额外的访存，避免了访存带来的开销。然后，寄存器的编码都在指令的固定位置（rs1 和 rs2），在译码之前就可以先读取寄存器的值读。最后，立即数的符号位被编码至指令的最高位，所以，立即数的符号扩展操作可以与译码操作并行处理。

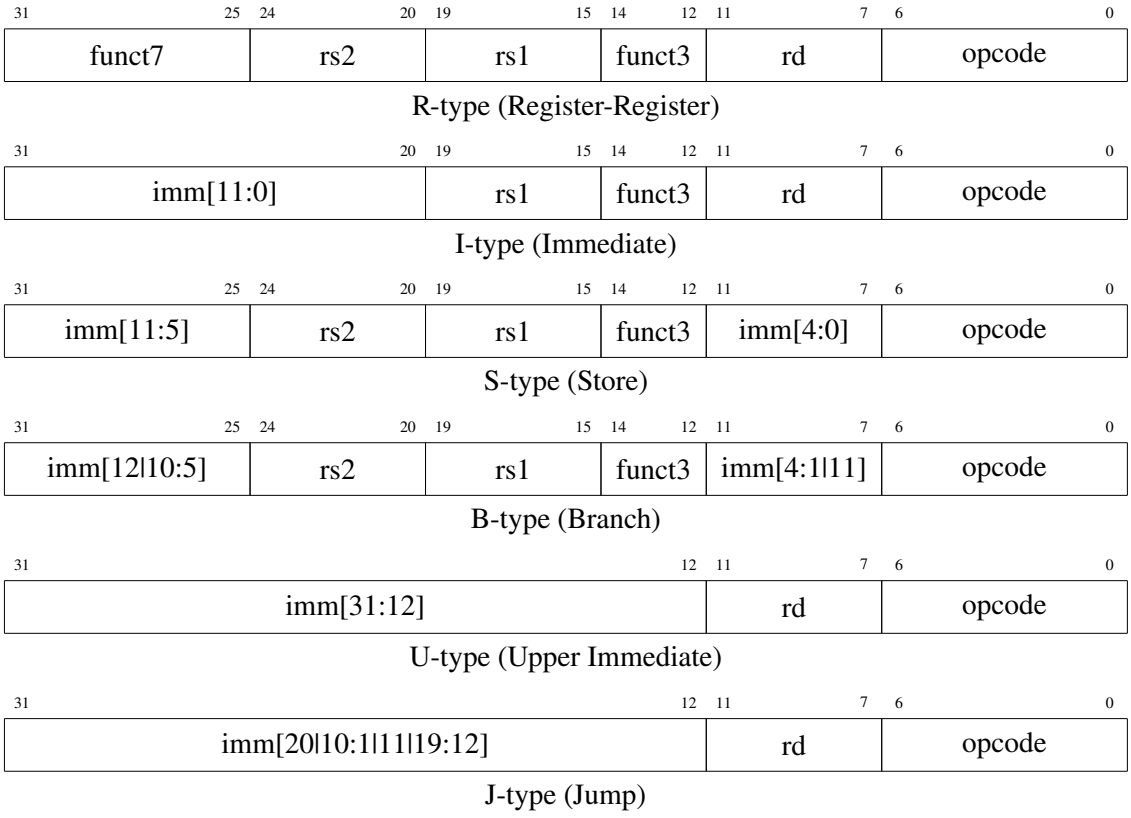

图 2-1 RV32I 指令集格式

表 2-1 RISC-V 指令格式介绍

类型	用途	示例指令
R 型（寄存器-寄存器操作）	算术和逻辑运算	add x1, x2, x3
I 型（立即数操作）	加载、立即数操作和跳转	addi x1, x2, 10
S 型（存储操作）	数据从寄存器存储到内存	sw x1, 12(x2)
B 型（条件分支）	条件分支跳转	beq x1, x2, label
U 型（高位立即数操作）	加载高位立即数	lui x1, 0x12345
J 型（无条件跳转）	函数调用或长跳转	jal x1, label

(2) 通用寄存器

如图2-2表示, RISC-V 有 32 个寄存器, 特殊的是, x0 寄存器硬连线为 0, 可替代约 15% 的指令操作，而 X86 需要显式 XOR 清零。为了提升处理器的性能，数据应该尽量存储在寄存器中，但是频繁的恢复和保存寄存器需要不断地访问内存，会带来不小的开销。为了避免这种情况，RISC-V 的处理方案是设置临时寄存器（t0-6）和保存寄存器（s0-11）。临时寄存器的值不需要保存至内存，而保存寄存器的值需要保存至内存。

相比 X86 的 16 个寄存器，RISC-V 架构的寄存器数量多一倍，适当地提升寄存器的数量，处理器可以充分地调度更多的寄存器，以至于加快程序编译和运行的速度。但是，寄存器并不是越多越好，由于其制造工艺和其特殊性导致了寄存器的成本昂贵，所以更多的寄存器会导致更高的成本和硬件复杂度，指令集也需要更多的比特位来对寄存器进行编码，一定程度上压缩了其余字段的编码空间，会提升编码和译码的复杂度。

表 2-2 RISC-V 寄存器介绍

寄存器	名称	功能
x0	zero	始终为 0
x1	ra	返回地址
x2	sp	栈指针
x3	gp	全局指针
x4	tp	线程指针
x5	t0	临时寄存器/链接寄存器
x6-7	t1-2	临时寄存器
x8	fp / s0	帧指针/保存寄存器
x9	s1	保存寄存器
x10-11	a0-1	函数参数/返回值
x12-17	a2-7	函数参数
x18-27	s2-11	保存寄存器
x28-31	t3-6	临时寄存器

(3) 指令集扩展

RISC-V 指令集的另一个显著特点是模块化设计。RV32I 作为基础 ISA，虽然只定义了 47 条指令，但是足以支持运行基本的软件，其稳定性为开发者提供了可靠的指令集基础。模块化设计允许开发者根据具体需求选择特定的扩展，例如：

(a) **RV32M**：支持乘除法扩展，从基本指令集分离出来的一个单独标准，需要设计对应的乘除法单元，适用于嵌入式系统、低功耗微控制器、高效处理整数运算的场景。

(b) **RV32A**：支持原子指令扩展，对共享内存的数据进行操作的一种方式，能够保证多线程并发执行的一致性，适用于多核处理器、操作系统内核、实时系统的场景。

(c) **RV32F**：支持单精度浮点扩展，新增了浮点寄存器，支持单精度浮点的传输、比较、转换、分类，适用于图形处理、传感器数据处理、轻量级机器学习推理的场景。

(d) **RV32D**：支持双精度浮点扩展，将浮点寄存器扩展至 64 位，新增了支持双精度浮点数相关的运算，适用于科学计算、高精度工程仿真、复杂模型训练的场景。

这种灵活的扩展机制使得 RISC-V 能够适应从嵌入式系统到高性能计算的多样化应用场景，如：物联网设备只需 RV32M + RV32F，而科学计算需要 RV32F + RV32D。这种设计理念为芯片设计者、软件开发者和终端用户带来了多方面的好处。

2.2 RISC-V 处理器相关技术

第 3 章 RISC-V 处理器设计

3.1 取指单元（IFU）

3.2 译码单元（IDU）

3.3 计算单元（EXU）

3.4 访存单元（LSU）

3.5 写回单元（WBU）

3.6 异常处理

第 4 章 处理器相关技术设计

4.1 AXI4 总线

4.2 片上系统（SoC）

4.3 系统优化

4.3.1 存储优化——高速缓存

4.3.2 并行优化——流水线

第 5 章 仿真测试

5.1 verilator 仿真

5.2 软硬件差分测试

第 6 章 总结和展望

6.1 本文总结

6.2 未来展望

参考文献

[1] Faggin F, Hoff M E, Mazor S, et al. The history of the 4004[J]. Ieee Micro, 1996, 16(6):10-20.

[2] Cocke J, Markstein V. The evolution of risc technology at ibm[J]. IBM Journal of research and development, 1990, 34(1):4-11.

[3] Waterman A, Lee Y, Patterson D A, et al. The risc-v instruction set manual, volume i: Base user-level isa [J]. EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62, 2011, 116:1-32.

[4] Asanović K, Patterson D A. Instruction sets should be free: The case for risc-v[J]. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, 2014.

[5] 黄鑫. “十四五”软件业开源生态加快构建[R]. 经济日报, 2021.

[6] Marena T. Risc-v: high performance embedded swerv™ core microarchitecture, performance and chips alliance[J]. Western Digital Corporation, 2019.

[7] 平头哥发布全新 RISC-V 处理器[J]. 中国集成电路, 2021, 30(06):21.

[8] 王凯帆, 徐易难, 余子濠, 等. 香山开源高性能 RISC-V 处理器设计与实现[J]. 计算机研究与发展, 2023, 60(03):476-493.

[9] Daki0107 V, Mr0161i0107 L, Kuni0107 Z, et al. Evaluating arm and risc-v architectures for high-performance computing with docker and kubernetes[J/OL]. Electronics, 2024, 13(17). https://www.mdpi.com/2079-9292/13/17/3494. DOI: 10.3390/electronics13173494.

[10] Fpga '21: The 2021 acm/sigda international symposium on field-programmable gate arrays[C]. New York, NY, USA: Association for Computing Machinery, 2021.

[11] 钟戴元, 曾庆立, 周佳凯, 等. 基于 RISC-V 处理器的软硬件联合验证平台设计与实现[J]. 信息技术与信息化, 2024(11):23-26.

[12] 郝振和, 焦继业, 李雨倩. 基于 AHB 总线的 RISC-V 微处理器设计与实现[J]. 计算机工程与应用, 2020, 56(20):52-58.

[13] 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(03):80-86.

致谢

谢谢!