自研处理器的实践：SWIFT 开发技术报告

刘志成

1. **引言**

什么是 SWIFT 处理器？

SWIFT 的全称是 Simple, Well-integrated, Intelligent, Fast, Tiny（简单、集成良好、智能、高速、小巧）。它采用了六级流水线架构，并实现了支持 rv32im\_zicsr\_zifencei 扩展的 RISC-V 指令集。处理器配备了独立的 L1 指令缓存（ICache）和数据缓存（DCache），以确保高效的内存访问。

研究背景与意义

随着信息技术的快速发展，处理器性能的提升和能效优化成为了嵌入式系统和高性能计算领域的重要研究方向。RISC-V 指令集架构（ISA）因其开放性、模块化设计和灵活的扩展能力，在学术界和工业界得到了广泛关注。与传统的封闭式架构相比，RISC-V 为开发者提供了更加自由的设计空间，特别适合用于教育、研究以及定制化应用场景。

在嵌入式系统领域，对高效、小型化处理器的需求尤为迫切。这类处理器需要兼具高性能、低功耗和资源节约的特点，以满足物联网（IoT）、边缘计算和实时控制等领域的应用需求。而流水线架构和高速缓存（如 ICache 和 DCache）的引入，是提升处理器性能和资源利用率的重要手段。

1. **系统架构设计**

SWIFT 处理器的整体架构

整体采用了 6 级流水线的架构，包括 IF1 IF2 ID EX LS WB。

基于 RV32M 的指令集扩展

实现了整数乘/除法的扩展

模块划分

流水线寄存器模块

取指、译码、执行、访存、写回模块

冒险处理模块

ICache DCache 缓存模块

AXI 互联模块

技术选型：工具链与开发环境

verilator 仿真器

香山开源工具链 difftest NEMU

Riscv 交叉编译器

Xilinx vivado 工具链

1. **核心技术实现**

流水线设计



ALU实现细节

目前用的乘除号，后续改成华莱士树乘法器和SRT16 定点除法器

寄存器与存储系统设计

包含分离的 L1 的 ICache 和 DCache

分支预测与跳转逻辑

分支默认不跳转

高效流水线设计与数据冒险处理

数据冒险使用前馈，load-use 冒险使用气泡，控制冒险flush+气泡

1. **仿真与功能验证**

总体仿真框架



源代码编译成的 .elf 可执行文件，通过 Verilator 仿真 CPU 和 NEMU 的模拟 CPU 运行，在单步执行的过程中进行逐步对比验证，确保实现结果正确。

提供了 3 个验证方式：

Picotest 验证基本指令集实现是否正确

Functest 验证基本程序功能实现是否正确

Device-test 验证外设访问实现是否正确

除此以外，在此基础上运行了南大 nanos OS 进行操作系统层面的验证

Nanos 实现了一个简单的文件系统，系统中断和自陷操作，IO输入输出，并在此基础上运行了 bird nterm 等等小程序

1. **SoC 板上系统设计**

总体框架



使用 AXI4 总线进行互联，FFT UART DDR 使用 Xilinx 官方 IP 核

开发板使用 ZCU102

1. **优化与创新**

使用华莱士树和SRT进行乘除法，提升流水线吞吐率

添加一个 FFT IP，实现专用领域的计算加速验证

在我的 SoC 的系统上运行操作系统

1. **实现成果与分析**

SWIFT 关键性能指标

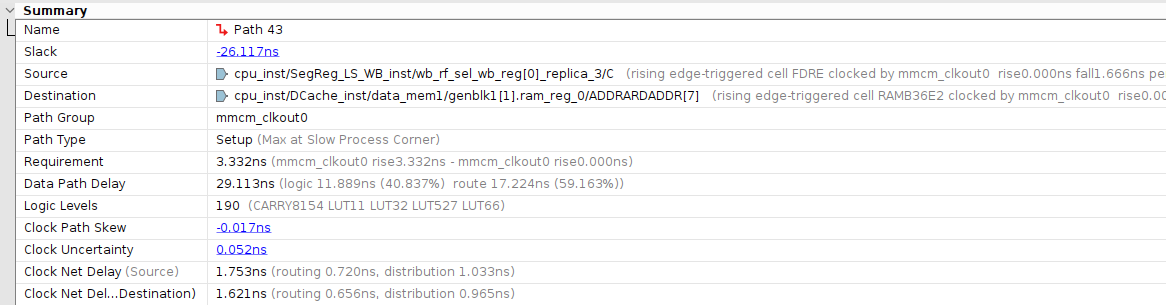
时钟频率

1. **开发经验与挑战**

跨工具链的协同开发经验

基于香山开源工具链，囊括了 riscv 交叉编译器，verilator 仿真器，vivado 工具链

第一次尝试发现了时序违例问题，通过将 ALU 优化予以解决



第二次尝试

1. **展望与未来规划**

在此基础上继续学习香山处理器有关乱序超标量处理器的设计方式

学习 GPGPU 的设计方式

1. **总结**