# A Case Study on Formal Equivalence Verification Between a C/C++ Model and Its RTL Design 复现报告

**摘要**

在通信系统产品的领域中，大多数数据路径数字信号处理算法最初是 MATLAB® 或 C/C++ 的高级语言中开发的。随后，设计工程师使用这些模型作为实现寄存器传输级设计的参考。验证它们等价性的传统方法涉及广泛的通用验证方法动态模拟，这可能需要数月时间并需要大量的验证工作。然而，由于使用这种方法无法探索所有输入组合，一些难以捉摸的错误可能仍然会发生。另一方面，形式等价性验证旨在验证寄存器传输级设计在所有可能的合法状态下与参考的高级 C/C++ 模型功能上等价。随着形式求解器技术的最新进展，形式等价性验证通过使用数学方法确保寄存器传输级（定时）与原始高级 C/C++ 模型（非定时）相匹配，提供了明显的好处。这大大减少了验证时间，并确保了设计状态空间的全面覆盖。本文通过采用形式等价性方法，深入探讨了具有数据路径的复杂有限状态机的验证，特别关注乘累加器、音调发生器和自动增益控制。尽管这些信号处理模块之前已经通过通用验证方法动态模拟进行了验证，但形式等价性验证能够利用新的工作流程在短短几周内识别出难以发现的错误，从而简化了验证过程。

**关键词：** 形式等价性验证；C/C++ 模型；寄存器传输级设计；形式数据路径验证

## 1 引言

在通信系统产品的开发领域，数字信号处理（DSP）算法的开发和验证是确保产品性能和可靠性的关键步骤。随着电子设备复杂性的增加，对于设计方法和验证过程的要求也在不断提高。传统的验证方法，如基于仿真的通用验证方法（UVM），虽然在一定程度上能够确保设计的正确性，但这种方法往往需要耗费大量的时间和资源，且可能无法完全覆盖所有可能的输入组合，导致难以发现所有潜在的错误 [1]。

本文的选题依据在于对现有验证方法的局限性的认识，以及对形式等价性验证技术潜力的探索。形式等价性验证作为一种新兴的验证方法，它利用数学方法来确保寄存器传输级（RTL）设计与高级 C/C++ 模型在功能上的等价性。这种方法可以显著减少验证时间，并提供对设计状态空间的全面覆盖，从而有可能发现传统方法可能遗漏的错误。

本文的选题意义在于提高验证效率：通过采用形式等价性验证技术，可以显著减少验证所需的时间和资源，这对于面临严格市场上市时间限制或资源有限的项目尤为重要；增强验证的全面性：形式等价性验证能够提供对设计状态空间的全面覆盖，有助于发现那些在传统

动态仿真中可能被忽略的边缘情况错误；推动验证技术的发展：本文的研究和实践可以为通信系统产品的设计和验证提供新的视角和工具，推动验证技术的进步，提高整个行业的设计质量和效率。

# 2 相关工作

## 2.1 寄存器传输级设计

寄存器传输级设计 [2] 是数字电路设计中的一个关键阶段，它涉及到使用硬件描述语言（HDL）如 Verilog 或 VHDL 来实现算法的硬件逻辑。在 RTL 设计中，设计师关注的是寄存器之间的数据流动和逻辑操作，而不是具体的逻辑门实现。这种设计方法允许设计师在较高的抽象层次上进行工作，同时确保设计能够被综合工具转换为门级电路。

RTL 设计的过程需要精细的编码和对硬件行为的深入理解。设计师需要考虑时钟域、流水线结构、以及寄存器和组合逻辑的相互作用。此外，RTL 设计还需要考虑性能、功耗和面积（PPA）的优化，以满足特定应用的需求。

## 2.2 通用验证方法的应用

UVM [3] 是一种用于系统级芯片（SoC）验证的框架，它提供了一套标准化的验证方法和组件。UVM 通过动态仿真来验证 RTL 设计的正确性，这种方法允许设计师在模拟环境中测试设计的行为，以确保它符合预期的功能和性能要求。UVM 的组件化和面向对象的特性使得验证环境的构建更加模块化和可重用。

然而，UVM 方法可能无法覆盖所有可能的输入组合，这可能导致一些错误难以被发现。为了解决这个问题，设计师可能会采用随机测试和约束随机测试来增加测试的覆盖率。此外，UVM 的灵活性和适应性使其能够适应各种复杂的验证场景。

## 2.3 形式等价性验证的研究

形式等价性验证 [4] 是一种基于数学证明的验证方法，它能够验证 RTL 设计与高级模型在功能上的等价性。FEV 在验证效率和全面性方面具有优势，因为它可以系统地检查所有可能的输入组合，从而提高验证的彻底性。与 UVM 相比，形式等价性验证提供了一种更为严格和全面的验证手段，尤其是在处理复杂的设计和验证场景时。

# 3 本文方法

## 3.1 本文方法概述

本次复现的工作主要包括：

实现 RISC-V 规范中运算指令的 C 模型：研究 RISC-V 指令集，深入理解 RISC-V 规范，特别是运算指令的功能和行为；开发 C 模型，使用 C 语言实现 RISC-V 运算指令的模型，确保模型能够准确反映指令的预期行为；验证 C 模型，通过测试和比较预期输出与实际输出，验证 C 模型的正确性。

熟悉“香山”处理器 [5] 对运算指令的执行：研究“香山”处理器架构，了解“香山”处理器的设计和实现，特别是其对 RISC-V 运算指令的支持；分析执行流程，研究“香山”处理器执行运算指令的具体流程和方法；理解优化和特性，探索“香山”处理器可能对运算指令执行进行的任何特定优化或添加的特性。

提取“香山”中各运算指令的约束：识别约束条件，从“香山”处理器的 RTL 设计中提取运算指令的约束条件；记录约束，详细记录这些约束，为后续的等价性验证提供参考；分析约束影响，评估这些约束对运算指令行为的影响，确保 C 模型能够模拟这些约束。

实现 C 模型与“香山”RTL 设计的等价性验证：选择验证工具，选择适合的等价性验证工具；准备验证环境，设置验证环境，包括 C 模型和“香山”RTL 设计的集成；定义验证策略，制定验证策略，确定如何比较 C 模型和 RTL 设计的行为；执行等价性验证，运行等价性验证，检查 C 模型和 RTL 设计在所有可能的合法状态下是否功能等价；分析验证结果，对验证结果进行分析，识别任何不匹配的地方，并进行调试；优化和迭代，根据验证结果对 C 模型或 RTL 设计进行必要的优化和调整，然后重复验证过程直到两者行为一致。

# 4 复现细节

## 4.1 与已有开源代码对比

在复现“香山”处理器的形式等价性验证环境中，其中对于部分 C 模型的编写采用了部分开源代码，以确保 C 模型的正确性，同时在此基础上进行创新和改进。以下是复现工作的具体概述：开源代码的引用与使用：使用了 RISC-V Spike 中的标量整型运算指令的 C 模型，它是一个开源的 RISC-V 指令集模拟器。使用了 SoftFloat-3e 中的标量浮点运算指令的 C 模型，这是一个开源的软件浮点运算库，它提供了对 IEEE 754 标准的完整支持，通过使用 SoftFloat-3e，确保了浮点运算的准确性和一致性。此外严格遵循 RISC-V 规范，编写了 RISC—V 的向量运算指令的 C 模型，以确保对运算指令的准确实现。这些模型经过精心设计，以模拟“香山”处理器的行为。根据“香山”处理器的特性，对 C 模型进行了定制化修改，以更好地模拟其行为，包括了对指令执行流程的优化，以及对特定指令集的支持。并从“香山”处理器的 RTL 设计中提取了运算指令的约束，并将其应用于 C 模型，确保了 C 模型与 RTL 设计在行为上的一致性。实现了 C 模型与“香山”RTL 设计的等价性验证，通过使用形式验证工具，快速识别并解决模型与 RTL 设计之间的差异。

## 4.2 实验环境搭建

如图 1所示，此次验证的主要过程可以分为以下几个步骤：

从 RISC-V 指令集模拟器 (riscv-isa-sim) 提取 C 模型：使用 RISC-V 官方的指令集模拟器 riscv-isa-sim 作为参考，该模拟器提供了 RISC-V 指令集的准确实现。从 riscv-isa-sim 中提取运算指令的实现逻辑，转换为 C 语言模型，以便在不依赖于特定硬件模拟器的情况下进行算法级别的验证。

编译 Chisel 代码生成 RTL 设计：将 Chisel 语言编写的香山处理器设计编译成 RTL 代码，这个过程将高级的 Chisel 代码转换为可以在 FPGA 或 ASIC 上实现的低级硬件描述。

应用约束到 C 模型和 RTL 设计：从 Chisel 编译生成的 RTL 设计中提取约束条件，将

这些约束应用到 C 模型中，确保 C 模型在模拟时能够遵守与 RTL 相同的限制。

进行形式等价性验证 (formal)：使用形式验证工具，对 C 模型和 RTL 设计进行等价性验证。验证工具将检查 C 模型和 RTL 设计在所有可能的合法状态下是否功能等价，即它们是否对相同的输入产生相同的输出。

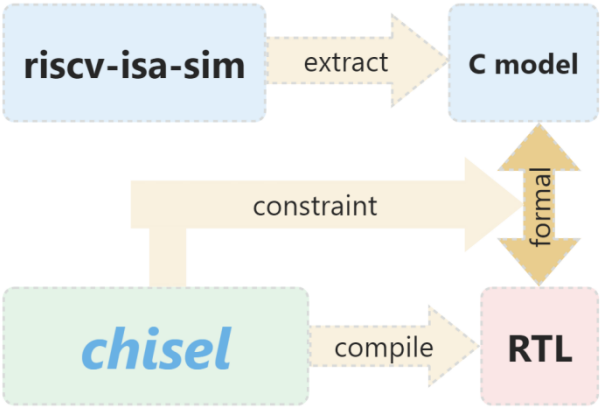


图 1. 验证框架图

### 4.3 验证流程

如图 2所示，形式等价性验证的流程可以概括为以下几个步骤：

开始：启动形式等价性验证流程。

配置 VCFormal：设置形式验证工具的配置，包括指定要验证的设计模块、选择验证策略、设置工具参数等。

运行：启动形式验证工具，让其根据配置对设计进行验证。

相等性检查：验证工具检查 C 模型和 RTL 设计在功能上是否相等。比较两个模型在所有可能的输入条件下的输出是否一致。

不相等处理：如果验证工具发现 C 模型和 RTL 设计在某些情况下不相等，流程进入 debug 阶段。如果验证工具确认两个模型在所有情况下都相等，流程进入验证完成阶段。

Debug：在不相等的情况下，验证工具会提供 debug 信息，帮助定位问题所在。

有效 Bug 检查：分析 debug 信息，判断发现的问题是否是有效的 bug，即是否影响了设计的正确性。

解决 Bug：如果确认是有效 bug，需要对设计进行修正，解决发现的问题。

如果分析后认为不是有效 bug，可能是验证工具的误报，那么可以修正约束或调整验证策略。

修正约束：如果问题是由于约束条件设置不当引起的，需要修正这些约束条件。

验证完成：在解决 bug 并重新运行验证工具后，如果所有检查都通过，即认为验证完成。

结束：形式等价性验证流程结束。

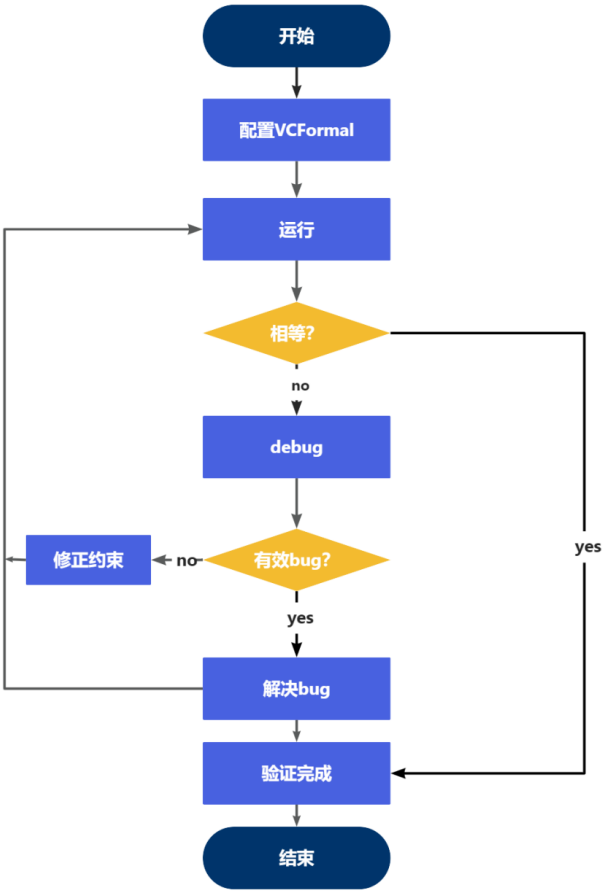


图 2. 验证流程图

### 4.4 创新点

形式等价性验证的应用：文章提出了使用形式等价性验证来验证 RTL 设计与高级 C/C++ 模型之间的功能等价性，这是一种基于数学证明的方法，可以全面覆盖设计状态空间，减少验证时间，并提高验证的全面性。

处理复杂数据路径和 FSM 行为：文章详细描述了等价性验证过程中检查功能等价性，特别是在处理类似 FSM 的数据路径行为时，如何在 C/C++ 模型中管理状态转换。

设计缩放和案例分割策略：为了加速形式验证的收敛，文章提出了设计缩放和案例分割策略，通过减少运算的位宽和隔离最具挑战性的案例来简化状态空间和计算复杂性。

中间等价点的利用：在处理复杂的影响锥 (COI) 时，文章提出了在 C/C++ 和 RTL 模型之间插入额外的断言来利用中间等价点，这有助于形式工具验证更复杂的自动生成的端到端属性。

## 5 实验结果分析

如图 3所示，这是本次形式等价性验证中对 Fcvt 模块测试的 log 文件，显示该模块执行的运算指令均已通过等价性验证。

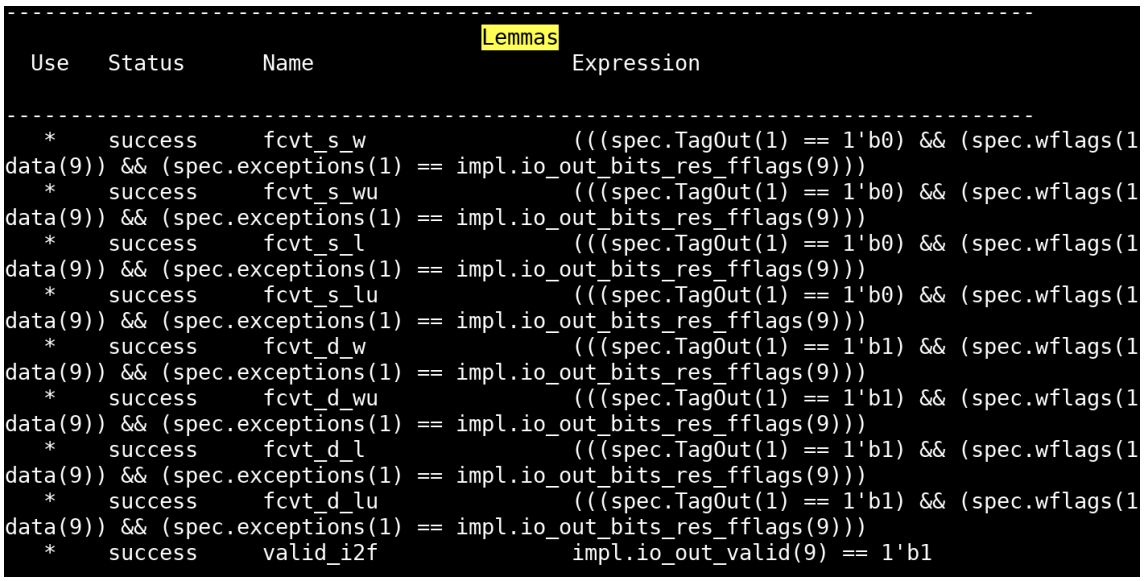


图 3. Fcvt 模块测试 log

| 类型 | 序号 | 验证模块/组件 | 结果 | 备注 |
|---|---|---|---|---|
| 标量 | 1 | bosc_ExeUnit_4 | Success | Alu |
| | 2 | bosc_ExeUnit | Success | Bku |
| | 3 | bosc_ExeUnit_1 | Success | Brh |
| | 4 | bosc_MulUnit | Success | Imul |
| | 5 | bosc_SRT16Divint_6 | Success | Idiv |
| | 6 | bosc_IntToFP_2 | Success | I2f |
| | 7 | bosc_IntFPToVec | Success | I2v |
| | 8 | bosc_IntFPToVec_1 | Success | F2v |
| | 8 | bosc_FAlu | Success | Falu |
| | 9 | bosc_FCVT | Success | Fcvt |
| | 10 | bosc_FloatFMA | Success | fma |
| | 11 | bosc_FloatDividerR64 | Success | fdiv |
| | 12 | bosc_fpsqrt_r16 | Success | fsqrt |
| 向量 | 13 | bosc_VIAluFix | Success | Vialuf |
| | 14 | bosc_VIPU | Success | Vpiu |
| | 15 | bosc_VPPU | Success | Vppu |
| | 16 | bosc_VFAlu | Success | Vfalu |
| | 17 | bosc_VCVT_8 | Success | Vfcvt |
| | 18 | bosc_VFMA | Success | Vfma |
| | 19 | bosc_VFDivSqrt | Success | Vfdivsqrt |
| | 20 | bosc_VIMacU | Success | Vimac |
| | 21 | bosc_VIDiv | Success | Vidiv |

表 1. 验证模块/组件结果表

在本次验证中，所有模块的验证结果均显示成功，如表1所示。

此次复现结果表明，“香山”处理器的各个关键模块在 RTL 设计与 C 模型之间实现了高度的等价性。这不仅验证了“香山”处理器设计的正确性，也展示了形式等价性验证在确保处理器设计质量方面的重要作用。通过这一过程，确保了处理器的各个部分都能按照预期工作，为后续的集成和系统级验证打下了坚实的基础。

## 6 总结与展望

形式等价性验证能够直接完成乘除法外的标量算子的形式化验证，同时可以完成向量算子单 uop 的乘除法以及归约算子之外其他算子的直接验证，多 uop 对于实现过程简单的算子也能够直接验证出。乘除法可以通过形式验证另外一种方法——定理证明，使用形式等价性验证对算子外处理的测试达到验证，归约等复杂操作的算子能够通过形式等价性验证高阶方法（例如 hdps，case split，assume guarantee 等）达到收敛。算子的形式等价性验证在验证流程中，启动快，上手容易，当存在反例时能迅速发现 bug。

形式等价性验证非常适合对算子进行验证，但是也会存在一些问题：无法测试到指令被冲刷的场景；部分算子的实现不全在 FUBlock；处理多 uop 场景下的乏力，往往一条向量指令是无法通过一个 uop 执行完毕的，比如 compress 指令，根据香山的 uop 拆解，直到 uopIdx 为 35 的时候，才能得到第一个的 vd0。

## 参考文献


[1] Gaetano Raia, Gianluca Rigano, David Vincenzoni, and Maurizio Martina. A case study on formal equivalence verification between a c/c++ model and its rtl design. In *International Symposium on Formal Methods*, pages 373–389. Springer, 2024.

[2] Wenji Fang, Yao Lu, Shang Liu, Qijun Zhang, Ceyu Xu, Lisa Wu Wills, Hongce Zhang, and Zhiyao Xie. Masterrtl: A pre-synthesis ppa estimation framework for any rtl design. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.

[3] N Susithra, V Yaswant, K Rajalakshmi, et al. Analyzing aes verification: A comparative study of uvm and cocotb approaches. In *2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC)*, pages 478–482. IEEE, 2024.

[4] Louis-Noël Pouchet, Emily Tucker, Niansong Zhang, Hongzheng Chen, Debjit Pal, Gabriel Rodríguez, and Zhiru Zhang. Formal verification of source-to-source transformations for hls. In *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 97–107, 2024.

[5] Yinan Xu, Zihao Yu, Dan Tang, Guokai Chen, Lu Chen, Lingrui Gou, Yue Jin, Qianruo Li, Xin Li, Zuojun Li, et al. Towards developing high performance risc-v processors using

agile methodology. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1178–1199. IEEE, 2022.