Finding More Bugs in Your RISC-V CPUs with DiffTest and XFUZZ

Yinan Xu
Institute of Computing Technology, Chinese Academy of Sciences

Abstract

Ensuring the functional correctness of RISC-V processors is increasingly challenging due to complex designs and non-deterministic behaviors. Traditional design verification (DV) methods require extensive manual effort and often fail to explore the vast state space of modern CPUs. We present an automated verification framework that integrates DiffTest for co-simulation and XFUZZ for coverage-guided fuzzing.

*DiffTest employs the Diff-Rule-Based Agile Verification (DRAV) method and standardized information probes to refine the reference model (REF) dynamically, ensuring robust discrepancy detection. XFUZZ enhances test generation through footprint memory-based mutation and reuses the LibAFL fuzzing engine to systematically explore execution paths. Our framework achieves **95.3% coverage in less than 10 hours** using SPEC CPU2006 seeds and uncovers **four long-standing functional bugs**, including one that persisted for nearly **10 years**. These results validate our approach for improving RISC-V processor verification.*

Introduction

The rapid evolution of RISC-V processors and the rise of open-source hardware have significantly increased CPU design complexity. Ensuring functional correctness is challenging due to intricate microarchitectural optimizations and non-deterministic behaviors. Traditional design verification (DV) methods, reliant on manual test development and directed testing, often fail to achieve sufficient coverage, leaving critical bugs undetected. While fuzzing is widely used in software testing to explore execution states systematically, its application to hardware verification remains limited by inefficiencies in seed selection and mutation strategies.

In this work, we present an automated verification framework that augments existing DV methodologies by integrating two complementary techniques: **DiffTest**, a co-simulation framework that systematically refines the reference model (REF) using the Diff-Rule-Based Agile Verification (DRAV) methodology, and **XFUZZ**, a coverage-guided hardware fuzzer that leverages footprint memory-based mutations along with the open-source LibAFL engine. This approach enables broader test-case generation and systematic execution path exploration without requiring additional manual intervention.

Experimental evaluation on open-source RISC-V processors demonstrates the efficacy of this approach. Using SPEC CPU2006 benchmarks as seeds, XFUZZ achieves **95.3% total coverage in less than 10 hours**, significantly improving state space exploration. Additionally, in terms of bug detection, our framework uncovers **four long-standing functional bugs** in two widely used RISC-V CPUs (Rocket Chip and Spike), with one persisting for nearly **10 years**, while also identifying functional issues in XiangShan. These results validate the effectiveness of our framework in improving the efficiency and completeness of functional verification for modern RISC-V processors.

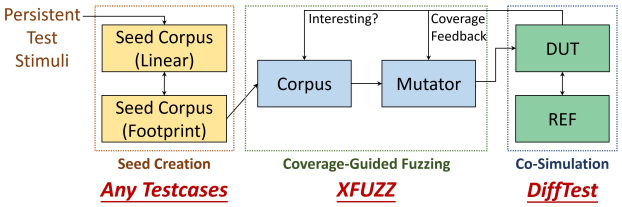

Figure 1: *System overview of DiffTest and XFUZZ.*

Methodologies

DiffTest and XFUZZ are the default co-simulation and fuzzing frameworks for the open-source XiangShan project. In 2024, they helped identify **102 potential bugs** through over **1700 hours** of nightly regression testing. By integrating systematic co-simulation with coverage-guided fuzzing, our framework enhances verification completeness while minimizing manual effort. This section details our approach.

Figure 1 presents an overview of our verification framework. DiffTest performs co-simulation between the design under test (DUT) and the reference model (REF), identifying execution discrepancies in real time. XFUZZ introduces footprint memory-based fuzzing to systematically mutate test inputs, guided by structural and functional coverage feedback. Together, these components automate verification, improve test efficiency, and uncover hard-to-detect processor bugs.

DiffTest: A Systematic Co-Simulation Framework

DRAV Method: DiffTest employs the Diff-Rule-Based Agile Verification (DRAV) method to systematically handle non-deterministic behaviors in RISC-V processors. Traditional co-simulation often encounters false positives due to permissible variations in execution order and microarchitectural optimizations. DRAV dynamically refines the reference model (REF)

using runtime execution feedback from the DUT, enabling precise identification of actual design discrepancies rather than expected variations.

Information Probes: To enable efficient and structured verification, DiffTest integrates standardized information probes that extract critical execution data from the DUT. These probes serve as a bridge between low-level HDL implementations and high-level HCL designs, ensuring consistency across different verification environments. By providing well-structured observation points, they allow DiffTest to capture deviations in architectural states, register updates, and memory interactions.

Standard Co-Simulation Interfaces: Leveraging DRAV and information probes, DiffTest establishes a uniform set of APIs for co-simulation, facilitating interoperability between diverse RISC-V implementations and simulation backends such as Verilator, VCS, and Palladium. These standardized interfaces simplify the process of integrating new DUTs and REFs, reducing the verification overhead and improving scalability.

XFUZZ: Coverage-Guided Hardware Fuzzing

Footprint Memory-Based Fuzzing: Traditional CPU fuzzers often rely on linear memory representations that include irrelevant or unused data, limiting mutation efficiency. XFUZZ introduces footprint memory, which captures only the memory regions actively accessed during execution. By focusing on execution-relevant bytes, footprint memory enhances mutation effectiveness, ensuring that generated test cases have a higher probability of reaching unexplored execution states.

LibAFL Integration: To leverage state-of-the-art software fuzzing methodologies, XFUZZ reuses the open-source LibAFL engine, a widely adopted modular fuzzing framework. This integration enables XFUZZ to inherit robust coverage feedback mechanisms and advanced mutation strategies, benefiting from community-driven enhancements and ongoing performance optimizations.

Evaluation

We assess the effectiveness of DiffTest and XFUZZ by evaluating co-simulation interfaces, functional coverage, and bug detection.

Standardized DiffTest Interfaces

DiffTest provides a unified co-simulation framework for verifying RISC-V processors. It enables dynamic REF adaptation, structured execution data extraction, and standardized APIs across diverse simulation platforms, ensuring efficient and scalable verification.

While DiffTest offers off-the-shelf support for DUTs such as Rocket, XiangShan, and NutShell and REFs such as Spike and NEMU, its design is not limited to these. As a framework, it is theoretically compatible with any RISC-V DUT and can be easily ported to new designs. Additionally, it supports multiple simulation platforms, including Verilator, VCS, Palladium, and FPGAs, providing flexibility across different verification environments.

Coverage Improvement with XFUZZ

We evaluate XFUZZ's impact on functional coverage using real-world SPEC CPU2006 benchmarks as fuzzing seeds. Unlike traditional CPU fuzzers that rely on handcrafted instruction sequences or synthetic test cases, XFUZZ benefits from footprint memory to focus mutations on execution-relevant memory regions. Experimental results show that XFUZZ achieves **95.3% total coverage** in **less than 10 hours**, significantly outperforming prior formal-assisted fuzzing approaches. The improvement is attributed to: a) footprint memory's ability to prioritize execution-relevant data, eliminating redundant mutations; b) LibAFL's adaptive mutation scheduling, optimizing test input prioritization; c) continuous feedback from DiffTest, refining seed selection dynamically.

Bug Detection Efficiency

XFUZZ's integration with DiffTest significantly enhances bug detection. During extensive testing, our framework uncovered **four long-standing functional bugs** in widely used RISC-V processors, including Rocket Chip and Spike—the oldest persisting for nearly 10 years. Additionally, in large-scale regression testing on XiangShan, XFUZZ revealed several functional issues, demonstrating its ability to detect complex processor bugs.

Conclusions

We present an automated verification framework for RISC-V processors that integrates co-simulation and fuzzing to enhance functional verification. DiffTest enables precise discrepancy detection, while XFUZZ improves test generation and execution path exploration. Together, they automate verification, reduce manual effort, and uncover subtle bugs missed by traditional methods. As open-source RISC-V verification tools, DiffTest and XFUZZ welcome contributions from both the research and industrial communities to further advance RISC-V processor validation.