# Comprehensive Lockstep Verification for NaxRiscv SoC integrating RISCV-DV, RVLS, and Questa/UVM

Billal Ighilahriz and Olivier Savry

Univ. Grenoble Alpes, CEA, LETI, F-38000 Grenoble, France [1]
Email: firstname.name@cea.fr

## Abstract

*This paper presents a robust lockstep verification framework for the NaxRiscv System-on-Chip (SoC) —a flexible, open-source, out-of-order, and superscalar RISC-V core. Our methodology integrates cycle-accurate RTL simulation using Verilator, functional co-simulation with the Spike golden model, and constrained-random test generation via RISCV-DV to achieve a comprehensive verification framework. Functional coverage metrics are collected through a Questa/UVM-based verification environment, enabling early detection of subtle behavioral mismatches and corner-case failures. Our methodology rigorously validates ISA compliance but also significantly enhancing the overall verification process.*

## Introduction

Processor verification remains one of the most challenging phases in modern hardware design, especially given the growing complexity of contemporary architectures. To ensure robust performance, advanced verification techniques and specialized tools are employed to rigorously test designs and uncover hidden bugs. In our framework for verifying RISC-V architectures, randomized execution scenarios are generated and detailed Register-Transfer Level (RTL) simulation outputs are meticulously compared against those from a trusted reference model.

Our approach is built around a lockstep co-simulation technique, where the Design Under Test (DUT) and a golden Instruction Set Simulator (ISS) perform identical operations in parallel, enabling real-time discrepancy detection and efficient debugging. This methodology has been applied to NaxRiscv [1]—a highly configurable, out-of-order, and superscalar core supporting multiple ISA extensions (including RV32/64IMAFDC), it supports operating systems like Linux and FreeRTOS and facilitates rapid on-FPGA prototyping when integrated with the LiTeX [2] framework. Developed using SpinalHDL [3]—modern hardware design language uses the Scala Build Tool (SBT) to build the project and simulate the execution of the core. Moreover,

## Multi-level verification strategy

The verification methodology is structured as a multi-layered approach that combines directed testing, real-world benchmarking, and constrained-random testing to ensure a comprehensive evaluation of both architectural and microarchitectural aspects of the design. The process begins with an automated Jenkins Continuous Integration (CI) pipeline that executes standard compliance test suites —specifically, riscv-tests and riscv-arch-tests—which serve as established benchmarks for verifying that the Instruction Set Architecture (ISA) and its extensions conform to the specifications. These industry-recognized suites provide an essential baseline by checking both basic ISA and microarchitectural conformance.

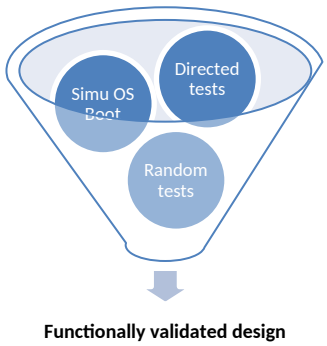


**Figure 1** : Multi-level verification strategy

Following these directed tests, the methodology incorporates the execution of real-world benchmarks to assess performance and software–hardware integration. This phase involves running applications and performance[1] metrics such as CoreMark and Dhrystone, as well as executing operating system boot sequences ranging from FreeRTOS to the Linux boot stage.

To further stress the design and explore its functional boundaries, the strategy leverages RISCV-DV [4], a tool that automatically generates constrained-random assembly programs. These tests deliberately target edge cases—such as misaligned instructions, register dependencies (RD/RS), pipeline hazards, memory access violations, and unhandled interrupts—thereby probing complex execution paths. In parallel, a robust functional coverage model, based on Universal Verification Methodology (UVM) and embedded

[1] This work was funded thanks to the French national program ”Programme Investissement d’Avenir IRT Nanoelec” ANR-10-AIRT-05

within RISCV-DV, ensures exhaustive validation of all ISA features, covering various opcodes, extensions (such as F, D, and C), and the proper interactions with system registers (CSRs). This integrated approach guarantees that every aspect of the design is rigorously validated, from basic compliance to resilience under real-world conditions.

## Lockstep Verification Workflow

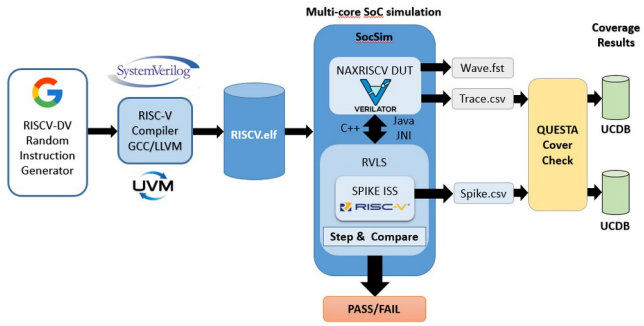


**Figure 2** : Lockstep RTL & ISS Verification Platform

We present a comprehensive lockstep verification framework that integrates several advanced tools to ensure both functional and architectural compliance for the NaxRiscv SoC. At its core, RISCV-DV—a robust, open-source framework developed by Google, coded in SystemVerilog/UVM, this generator of randomized instruction sequences and exhaustive test programs for the RV32/64IMAFDC ISAs across multiple privilege modes (M, S, and U), deliberately targeting critical edge cases to maximize functional coverage.

These test programs are executed in SocSim, a Scala-based multi-core SoC simulator that leverages Verilator for cycle-accurate RTL simulation and golden Instruction Set Simulator (ISS). The RTL simulation environment provides precise clock-cycle modeling and deep visibility into internal signals (such as the program counter, Control and Status Registers (CSRs), and memory transactions), with detailed Value Change Dump (VCD) outputs available for further analysis. The simulator's outputs are rigorously compared against the functional emulation provided by Spike—an open-source ISS that serves as our golden reference by accurately managing CSR operations, general and floating-point registers, and critical extensions like Compressed, Atomics, and the Floating-Point Unit

To achieve real-time trace alignment, SocSim incorporates the **RISC-V Lock-Step (RVLS)** framework via a Java Native Interface (JNI). This integration meticulously synchronizes instruction execution and asynchronous events (e.g., interrupts and exceptions) on a cycle-by-cycle basis by comparing the DUT's state with that of Spike (Step & Compare). In parallel, comprehensive coverage metrics—including instruction, CSR, and branch —are extracted from the execution traces. The DUT's tracer.log and Spike's spike.log are first converted to CSV format; then, using the Questa RTL simulator, the coverage data is collected into a binary Unified Coverage Database (UCDB) file and finally transformed into a human-readable report via the "vcover report" command. This detailed coverage analysis provides metrics for each instruction along with various coverpoints such as operand registers, branch outcomes, and memory alignment, ensuring that the verification process remains both exhaustive and precise

A Jenkins-based CI pipeline automates regression testing and dynamically scales RISCV-DV test generation until predefined coverage targets are met, providing rapid feedback on design modifications. By seamlessly integrating RISCV-DV's comprehensive test generation, RVLS's rigorous lockstep enforcement, and Questa/UVM's detailed coverage analysis, our framework effectively eliminates blind spots, guarantees strict ISA compliance, and facilitates early bug detection—ultimately reinforcing the reliability of our NaxRiscv SoC implementations.

## Results Overview

Our experimental results clearly demonstrate that our SoC-level integrated lockstep verification framework can detect critical issues that conventional core-level simulations fail to detect. Initially, discrepancies in the IO/memory mapping triggered *˝trap_store_access_fault˝* events. In another instance, a misaligned instruction generated by RISCV-DV resulted in a malformed opcode, introduce a mismatch between DUT and the ISS due to a lack of verification of the Rd/Rs registers of certain compressed instructions (e.g., C.LWSP) in the DUT causing a *˝trap_load_page_fault˝* events. Additionally, mismatches in the CSR configurations between the DUT and the golden reference Spike, led to observable execution anomalies (e.g., *DUT missed a trap*). These findings confirm that real-time cycle-by-cycle comparison and random generation of instruction sequences is essential to uncover subtle design issues, thereby enhancing overall processor reliability.

## Conclusion

This work presents a robust locking verification framework that bridges the critical gap between architectural specifications and real-world operational integrity in the NaxRiscv SoC. By integrating cycle-accurate simulation (via Verilator), functional co-simulation (using Spike), RISCV-DV-driven constrained random test generation, and Questa/UVM-powered coverage analysis—all synchronized through RVLS's real-time trace alignment—the framework significantly enhances verification efficacy. This cohesive approach not only ensures strict architectural compliance but also enables early detection of even the most elusive corner-case bugs, including memory access integrity, interrupt handling, and exception management. With its rigorous and scalable methodology, this framework strengthens confidence in the SoC's functional correctness and resilience across diverse operational scenarios.

## Références


[1] NaxRiscv—An Open-Source OoO Superscalar Softcore, Suisse: Available online: https://github.com/SpinalHDL/NaxRiscv, 2022.

[2] F. Kermarrec, S. Bourdeauducq, H. Badier et J. Le Lann, «LiteX—An open-source SoC builder framework based on Migen Python DSL.,» *arXiv,* n° %12005.02506, 2020.

[3] «SpinalHDL,» An Open-Source high-level HDL, 2014. [En ligne]. Available: https://github.com/SpinalHDL/SpinalHDL. [Accès le 2024].

[4] Google, «RISC-V DV,» 2019. [En ligne]. Available: https://github.com/google/riscv-dv. [Accès le 2024].