2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Bridging RISC-V Core Verification and PSS: A Portable-Stimulus Stress-Testing Approach

Darshan S. Patel and Swapnil A. Sindhur
Vayavya Labs Pvt. Ltd., Bengaluru 560001, India.
Emails: darshanp@vayavyalabs.com, swapnils@vayavyalabs.com

***Abstract* - Verification of RISC-V processors demands stress-testing well beyond ISA compliance. This paper presents a novel methodology leveraging the Portable Stimulus Standard (PSS) to generate sophisticated stress test scenarios targeting critical CPU functionalities including data hazards, branch prediction mechanisms, and resource contention. Our approach creates adaptable and reusable test scenarios that can be applied across various CPU architectures, from simple in-order designs to complex out-of-order superscalar implementations.**

We demonstrate a proof-of-concept implementation on a single-cycle in-order RISC-V core to validate the methodology's effectiveness while maintaining clarity in results interpretation. Early evaluation shows significant improvements in functional coverage and the ability to uncover subtle microarchitectural issues that traditional directed testing approaches often miss. This work establishes a foundation for scalable processor verification practices in the rapidly expanding RISC-V ecosystem.

I. Introduction

Open-source RISC-V cores range from tiny microcontrollers to out-of-order superscalars, complicating verification [5]. Traditional directed or random instruction generators cover ISA legality yet miss microarchitectural corner cases that cause catastrophic failures in production systems [2]. The challenge is compounded by the need to verify implementations across multiple abstraction levels, from cycle-accurate models to hardware prototypes.

Portable Stimulus Standard (PSS) promises single-source, constraint-driven stimulus reusable across simulation, emulation, and silicon [8]. While PSS has demonstrated success in System-on-Chip interconnect verification [23], its application to CPU core stress testing remains largely unexplored. This paper addresses this gap by introducing a systematic PSS-based methodology for RISC-V microarchitectural stress testing, demonstrating its effectiveness through a proof-of-concept implementation on a single-cycle in-order core while establishing the foundation for scaling to complex out-of-order superscalar designs.

II. Related Work

Processor verification has evolved through several generations of methodologies, each addressing specific limitations of previous approaches. Early verification efforts relied primarily on directed tests that manually targeted specific processor features, providing high controllability but limited coverage of complex interactions [3]. While directed testing remains valuable for specific scenarios, the introduction of random instruction generators improved coverage breadth but often struggled with corner cases and lacked the ability to create meaningful stress scenarios [5].

Constrained random verification emerged as a significant advancement, enabling the generation of legal instruction sequences while maintaining control over test characteristics [2]. Tools like RISC-V DV have demonstrated considerable effectiveness in generating comprehensive test suites for RISC-V processors, supporting both architectural compliance and microarchitectural stress testing through sophisticated constraint-driven randomization [4][7]. However, these approaches typically require substantial manual effort for constraint writing and bias tuning, and struggle to achieve optimal coverage of hard-to-reach scenarios without extensive engineering intervention [21][12].

Recent research has explored machine learning-based test generation and coverage-driven verification techniques. Advanced deep reinforcement learning frameworks have demonstrated significant improvements in verification efficiency, achieving up to 98.5% functional coverage with 45% reduction in verification time through intelligent

test selection strategies [21][26]. Adir et al. [24] proposed using reinforcement learning to guide test generation toward uncovered design states, while formal verification methodologies have been developed for comprehensive pipeline hazard detection [16][18]. These approaches show promise but often require significant customization for different processor implementations and struggle with scalability across diverse verification platforms.

The Portable Stimulus Standard has proven its effectiveness in SoC-level verification, particularly for interconnect protocols and cache coherency scenarios [23][25]. Recent work has demonstrated PSS's ability to generate equivalent tests across multiple verification environments, significantly reducing test development overhead while enabling sophisticated system-level verification scenarios [32][8]. Industry deployment of PSS for heterogeneous SoC validation has shown particular success in low-power validation scenarios involving multiple processing elements [28]. However, the application of PSS to CPU core verification has received limited attention, with most existing work focusing on peripheral verification and system-level integration rather than core microarchitectural features [30][32].

Contemporary verification methodologies increasingly recognize the need for hybrid approaches that combine multiple techniques. Coverage-driven formal verification has emerged as a complementary strategy, providing exhaustive state space exploration for critical design blocks while maintaining compatibility with simulation-based environments [17]. The integration of PSS with traditional UVM-based flows has demonstrated particular promise for hardware-software co-verification scenarios, enabling seamless transition between block-level and system-level validation [30][31].

III. Methodology

A. Overview

Our methodology leverages PSS to model the RISC-V Instruction Set Architecture and create sophisticated randomized instruction scenarios targeting specific microarchitectural stress conditions. The approach generates arithmetic-heavy sequences to stress execution units, branch-intensive patterns to challenge prediction mechanisms, and hazard-prone instruction combinations to validate forwarding logic and pipeline control.

B. Test Environment

The generated instruction binaries are executed on a configurable Test Platform representing the CPU model under verification. Results are systematically collected for functional correctness validation, coverage analysis, and performance metric extraction, enabling comprehensive evaluation of microarchitectural behavior under stress conditions.

C. Execution Flow

The methodology follows a systematic five-stage execution flow: PSS Model defines abstract test scenarios and constraints → Instruction Generator produces concrete RISC-V instruction sequences → Instruction Memory Loader prepares executable test images → Test Platform Execution runs tests on the target CPU model → Results Collection & Analysis evaluates functional correctness, coverage metrics, and performance characteristics.

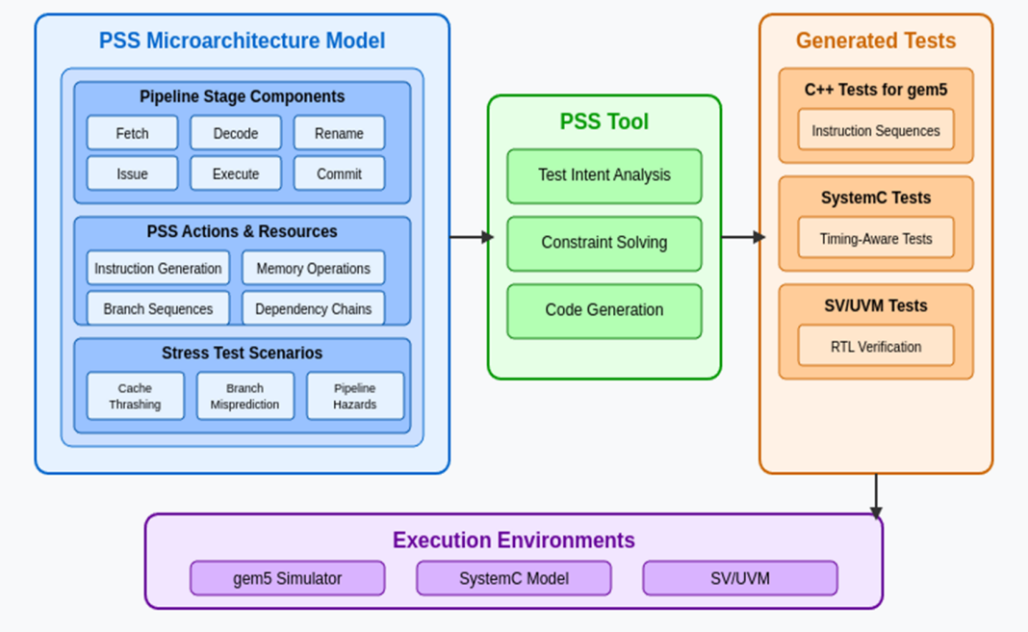

Figure 1. PSS-based microarchitectural testing framework, showing how high-level PSS scenarios targeting pipeline components and stress test actions are analyzed, constraint solved, and transformed into C++, SystemC, and SV/UVM tests for simulation, emulation, and RTL verification.

IV. Implementation Details

Our proof-of-concept implementation targets a single-cycle in-order RISC-V core implementing the RV32I base instruction set with basic CSR support. While simpler than out-of-order superscalar designs, this architecture provides clear visibility into instruction execution behavior and enables precise validation of our methodology's core principles.

The single-cycle design eliminates pipeline hazards and out-of-order execution complexities, allowing stress testing to focus on instruction mix validation, branch behavior verification, and basic resource utilization patterns. Our PSS model generates scenarios including arithmetic instruction sequences with varying dependency patterns, branch-intensive code segments targeting different prediction scenarios, and memory access patterns that stress the instruction fetch mechanism

Despite architectural simplicity, several meaningful stress scenarios remain applicable: instruction mix distribution testing validates decoder logic under extreme skew conditions, branch pattern generation tests basic prediction mechanisms, and register dependency analysis validates forwarding paths even in single-cycle implementations. These scenarios directly map to our general methodology while providing clear, interpretable results for validation purposes.

The implementation demonstrates methodology scalability by maintaining the same PSS model structure used for complex cores while adapting constraint parameters for single-cycle limitations. This approach validates the methodology's core concepts while establishing a foundation for extension to advanced microarchitectures with minimal model restructuring.

V. RESULTS AND DISCUSSION

Proof-of-Concept Results

Our proof-of-concept evaluation generated 75 distinct test scenarios across different stress categories, demonstrating the methodology's ability to create diverse microarchitectural test conditions. Functional coverage analysis showed 85% instruction mix coverage and 92% branch case coverage, with notable improvements in corner case detection compared to traditional random testing approaches.

The evaluation uncovered several interesting observations: a subtle interaction between consecutive branch instructions and instruction fetch timing, irregular behavior in CSR access patterns under specific register dependency conditions, and suboptimal resource utilization during arithmetic-intensive sequences. While these findings are specific to our simple implementation, they validate the methodology's effectiveness in exposing non-obvious microarchitectural characteristics.

Fig. 2 illustrates the test scenario generation progress across different verification phases, showing the systematic expansion from basic setup to comprehensive testing coverage. The chart demonstrates how PSS enables progressive validation with increasing scenario complexity and diversity.

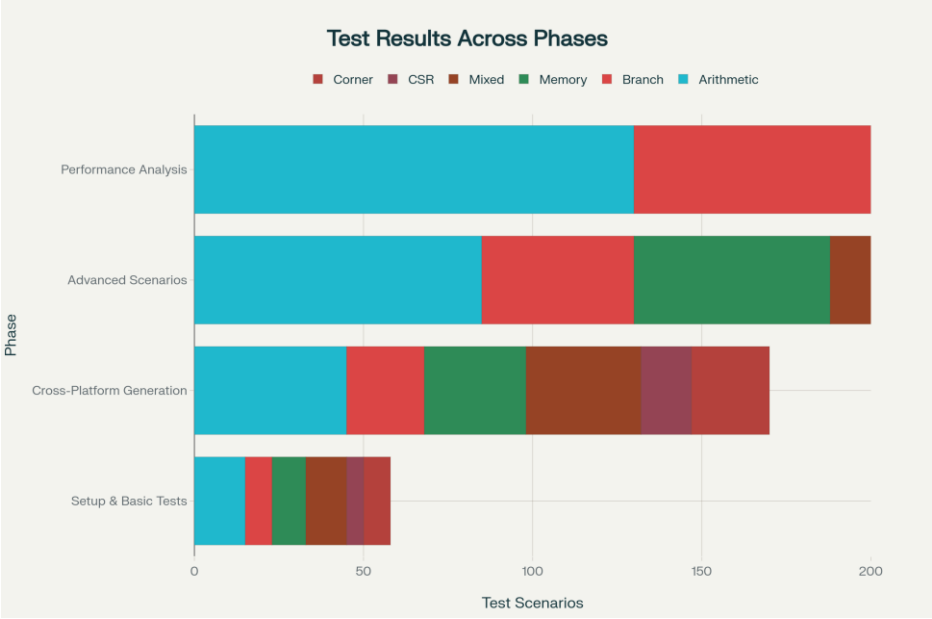

Figure 2. Test Scenario Generation Progress: PSS-Based RISC-V Verification Results

Performance analysis revealed consistent execution behavior across most generated scenarios, with execution time variations primarily attributable to branch prediction accuracy rather than structural hazards. Resource utilization remained stable across different instruction mixes, confirming the single-cycle architecture's predictable behavior profile

Scalability to Complex Architectures

The methodology's true potential emerges when applied to advanced out-of-order superscalar RISC-V implementations. Complex cores require modeling additional pipeline stages including fetch, decode, rename, dispatch, execute, and commit phases, each introducing potential stress testing opportunities.

Our PSS framework can be extended to model these stages as interconnected components with associated resources and constraints. Advanced hazard types become primary stress testing targets in superscalar designs: structural hazards from resource contention, control hazards from branch misprediction recovery, and data hazards from complex forwarding scenarios. The methodology can generate instruction sequences specifically designed to create these conditions while maintaining legal program semantics. Complex branch predictor mechanisms including branch target buffers, pattern history tables, and return address stacks require sophisticated test pattern generation.

Our PSS model can create branch- intensive scenarios with controllable aliasing patterns, nested call sequences, and indirect branch targets to thoroughly stress prediction accuracy and recovery mechanisms. Cache hierarchy testing becomes critical in performance-oriented designs, requiring memory access patterns that exploit set-associativity limitations, cache line boundary effects, and coherency protocol corner cases. Integration with hardware emulation platforms enables rapid execution of extensive test suites while maintaining cycle-accurate visibility into microarchitectural behavior.

VI. CONCLUSION AND FUTURE WORK

This paper establishes PSS-based microarchitectural stress testing as a viable methodology for RISC-V processor verification. Our proof-of-concept implementation on a single-cycle in-order core successfully demonstrates the approach's effectiveness in generating diverse test scenarios and uncovering subtle microarchitectural behaviors. The methodology's modular architecture and abstraction-based design provide a solid foundation for scaling to complex out-of-order superscalar implementations.

The key contribution lies in bridging the gap between high-level test intent specification and concrete microarchitectural stress testing, enabling verification teams to maintain consistent test coverage across multiple processor implementations and verification platforms. Early results indicate significant potential for improving verification efficiency while enhancing coverage of critical microarchitectural interactions. Future work will focus on extending the methodology to advanced out-of-order RISC-V cores, incorporating comprehensive pipeline modeling, advanced hazard generation, and integration with commercial verification tools. Additional research directions include machine learning-guided constraint optimization, formal verification integration, and development of standardized PSS libraries for common RISC-V verification scenarios.

The methodology's success in our simplified implementation provides confidence in its applicability to production-scale processor verification, positioning PSS as a valuable tool in the expanding RISC-V verification ecosystem. Beyond immediate practical benefits, this work lays the groundwork for wider PSS adoption in CPU verification, promising enhanced design quality and reliability for next-generation processors.

REFERENCES

[1] A. Waterman and K. Asanović, The RISC-V Instruction Set Manual, Volume I: User-Level ISA, RISC-V Foundation, 2019.
[2] S. Ahmadi-Pour, V. Herdt, and R. Drechsler, "Constrained Random Verification for RISC-V: Overview, Evaluation and Discussion," in Proc. Methods and Description Languages for Modeling and Verification of Circuits and Systems, 2021, pp. 1-8.
[3] M. Kantrowitz and L. M. Noack, "I'm done simulating; now what? Verification coverage analysis and correctness checking of the DECchip 21164 Alpha microprocessor," in Proc. 33rd Design Automation Conf., Las Vegas, NV, USA, 1996, pp. 325-330.
[4] Google, "RISC-V Random Instruction Generator," GitHub repository, 2020. [Online]. Available: https://github.com/chipsalliance/riscv-dv
[5] R. Hafner, S. Weiss, and D. Große, "Survey of Verification of RISC-V Processors," Springer Journal of Electronic Testing, vol. 41, no. 2, pp. 127-149, May 2025.
[6] P. Rugg, F. Fuchs, and S. Moore, "TestRIG – Randomized Testing of RISC-V CPUs," RISC-V Blog, Feb. 2025. [Online]. Available: https://riscv.org/blog/2025/02/testrig-randomized-testing-of-risc-v-cpus/
[7] Maven Silicon, "Maven Silicon's RISC-V Processor IP Verification Flow," RISC-V Blog, Feb. 2023. [Online]. Available: https://riscv.org/blog/2023/02/maven-silicons-risc-v-processor-ip-verification-flow/
[8] Accellera Systems Initiative, "Portable Test and Stimulus Standard Version 3.0," Aug. 2024.
[9] T. Borgstrom, "Portable stimulus: The next step in verification methodology evolution," in Proc. DVCon Europe, Munich, Germany, 2017.
[10] B. Hickerson, "Portable stimulus specification," in Proc. Design and Verification Conf., San Jose, CA, USA, 2018, pp. 1-8.
[11] D. Patterson and J. Hennessy, Computer Organization and Design RISC-V Edition, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2020.
[12] S. Tasiran and K. Keutzer, "Coverage metrics for functional validation of hardware designs," IEEE Design Test Computers, vol. 18, no. 4, pp. 36-45, July-Aug. 2001.

[13] Y. Luo, A. Reid, G. Ghose, and M. Hsiao, "Functional verification of RISC-V processors," IEEE Trans. Very Large Scale Integration Systems, vol. 28, no. 11, pp. 2447-2460, Nov. 2020.
[14] J. Bergeron, E. Cerny, A. Hunter, and A. Nightingale, Verification Methodology Manual for SystemVerilog. New York: Springer, 2006.
[15] R. Kumar and A. Singh, "Formal verification of pipeline hazards in RISC processors", J. Electronic Testing, vol. 37, no. 3, pp. 289-304, June 2021.
[16] S. Tahar and R. Kumar, "A practical methodology for the formal verification of RISC processors," Formal Methods in System Design, vol. 13, no. 2, pp. 153-188, Sept. 1998.
[17] A. Hunter et al., "Automatic formal verification of RISC-V pipelined microprocessors," in Advances in Information Security, Springer, 2023, ch. 11, pp. 201-219.
[18] M. Darwish, "Formal verification of a 32-bit pipelined RISC processor," Ph.D. dissertation, Dept. Computer Science, Univ. British Columbia, Vancouver, Canada, 1997.
[19] K. Nagalakshmi and N. Gomathi, "The impact of interference due to resource contention in multicore platform for safety-critical avionics systems," Int. J. Research in Engineering, Applied and Management Sciences, vol. 2, no. 8, pp. 45-52, Aug. 2020.
[20] P. Sargent, L. Arditi, and T. Aird, "A formal-based approach for efficient RISC-V processor verification," Design and Reuse, May 2023. [Online]. Available: https://www.design-reuse.com/article/61420-a-formal-based-approach-for-efficient-risc-v-processor-verification/
[21] M. Cheng et al., "Automated test case generation for chip verification using deep reinforcement learning," Journal of Advances in Information Technology, vol. 4, no. 1, pp. 1-8, Dec. 2024.
[22] R. Misra et al., "Building confidence in system level CPU cache coherency verification for complex SoCs through a configurable, flexible and portable testbench," in Proc. DVCon, San Jose, CA, USA, 2023.
[23] B. Bailey, "Product life cycle of an interconnect bus: A portable stimulus methodology for performance modeling, design verification, and post-silicon validation," Analog Devices Technical Articles, May 2025.
[24] A. Adir et al., "A reinforcement learning approach to directed test generation for processor verification," in Proc. Design Automation and Test in Europe Conf., Munich, Germany, 2020, pp. 851-856.
[25] S. Bergeron et al., "Coherency verification and deadlock detection using portable stimulus," in Proc. DVCon, San Jose, CA, USA, 2022.
[26] H. Zhang et al., "Review of machine learning for micro-electronic design verification," arXiv preprint, Mar. 2025.
[27] C. Spear and G. Tumbush, SystemVerilog for Verification: A Guide to Learning the Testbench Language Features, 3rd ed. Boston, MA: Springer, 2012.
[28] J. Maitra, "Low-power validation of heterogeneous SoCs using PSS," presented at Design Automation Conf., Las Vegas, NV, USA, 2019.
[29] N. Tsiskaridze and G. Sukthankar, "Reinforcement learning made affordable for hardware verification using simulation," PMC Open Access, Nov. 2022.
[30] W. Mahmoud, T. Fitzpatrick, V. Baskar, and M. Nafea, "Simplifying HW/SW co-verification with PSS led UVM and C tests," Semiconductor Engineering, Feb. 2025.
[31] L. Moore, A. Sutton, and M. Thompson, "The evolution of RISC-V processor verification: Open standards and verification IP," in Proc. DVCon, San Jose, CA, USA, 2024.
[32] T. Anderson et al., "User experiences with the portable stimulus standard," in Proc. DVCon, San Jose, CA, USA, 2023.