ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 5 Issue 3 July 2025 / Page No: 54-62
Paper Id: JETA-V5I3P108 / Doi: 10.56472/25832646/JETA-V5I3P108

Original Article

Scalable Formal Verification Strategies for RISC-V Based Control Paths: A Review

Aparna Mohan
North Carolina State University, Raleigh, North Carolina

Received Date: 22 April 2025 Revised Date: 13 June 2025 Accepted Date: 16 July 2025

Abstract: *RISC-V is an open and extendable instruction set architecture that is widely used in both commercial and research processor designs. It is very important to make sure that its control routes are accurate, especially for applications in areas where safety and performance are very important. This paper looks at the different scalable formal verification approaches that can be used with RISC-V control logic. These include symbolic execution, bounded model checking, AI-assisted theorem proving, and invariant-based methods. We summarise the most important research results, experimental tests, and theoretical models, and suggest new areas of investigation. There is a lot of focus on how well the tools operate in the real world, how easy they are to scale, and how well they fit into modern hardware development workflows. The study ends by listing several open problems and giving a plan for making next-generation scalable verification systems for designs based on RISC-V.*

Keywords: *RISC-V, Formal Verification, Control Path Verification, Model Checking, Theorem Proving, SMT Solvers, Symbolic Execution, AI-Assisted Verification, Parameterised Invariants, and Open-Source Hardware are some of the words that come to mind.*

I. INTRODUCTION

A. Background

The rise of open-source hardware platforms has led to a revival in processor design, and the RISC-V (Reduced Instruction Set Computing - Five) architecture is quickly becoming a strong player in the worldwide computing market. RISC-V is an instruction set architecture (ISA) that was first created at the University of California, Berkeley. It is modular, expandable, and free of royalties, and it is becoming more popular in both academia and industry [1]. As more and more applications use it, from Internet of Things (IoT) devices to high-performance computing systems, it needs a robust guarantee of correctness, safety, and reliability—areas where formal verification is very important.

B. Significance and Relevance

Formal verification is a group of mathematical methods used to show or refute that hardware and software systems are correct according to a specified formal specification. In the case of RISC-V-based control pathways, which are the most important parts of a processor that manage the flow of instructions and keep track of the order in which they are executed, formal approaches can find problems that are hard to find using traditional simulation-based verification [2]. As RISC-V gets more sophisticated with custom extensions and domain-specific applications, it becomes much harder to check its control routes.

In the semiconductor business today, when designs are getting more complicated and time-to-market requirements are huge, scalable formal verification is no longer a luxury; it's a must-have. Also, scalable verification is spreading from research labs to production pipelines as AI-driven formal tools become more common and proof assistants and property-directed reachability approaches become more common [3]. To make sure that RISC-V-based designs used in important infrastructure, defence, and safety-critical systems like self-driving cars and medical equipment are safe and secure, these methods are very important.

C. More Important in the Field

The subject matter goes much beyond just processor verification. It connects with important areas of computer engineering, like formal techniques, computer architecture, cyber-physical systems, and even renewable energy systems, where integrated RISC-V controllers may handle schemes for optimising power use [4]. Like open-source software, open-source hardware is becoming increasingly popular. This means that development processes will become more open and community-driven. In this kind of setting, scalable verification solutions not only make sure that things work correctly, but they also build confidence and long-term sustainability in open hardware ecosystems. In addition, the accuracy of control routes has a direct effect on the reliability and safety of models in disciplines like artificial intelligence and machine learning, where RISC-V processors are becoming more common as hosts for inference engines at the edge [5]. This shows how

important it is to provide verification methods that can grow, work on their own, and change to support new ideas while lowering risk.

D. Important Problems and Gaps

There are a lot of problems with formally verifying RISC-V based control routes, even though they seem promising:

- Scalability: A lot of current formal methods don't work well when hardware gets more complicated. To check bigger designs or whole SoCs (System-on-Chips), you typically need to use partitioning techniques, compositional reasoning, or abstraction. These methods, however, come with their own verification trade-offs [6].
- Toolchain Compatibility: A big problem is still getting it to operate with modern hardware development workflows. A lot of verification tools don't support RISC-V-specific semantics or need explicit annotations and assertions, which makes automation harder [7].
- Not enough standardisation: There aren't enough standardised verification frameworks that are specifically designed for RISC-V, which makes tools and methods less consistent [8].
- Dynamic and Parametric Extensions: Because RISC-V lets users create their own extensions, verification strategies need to be able to handle changes and reconfigurations. This is something that existing frameworks are only starting to do well [9].

E. The Review's Goal

The goal of this paper is to bring together the present state of scalable formal verification methods that are specifically used on the control paths of RISC-V processors. It looks at new tools, how well they work compared to one other, new theories, and experimental benchmarks that show what the best tools are right now. There is a lot of focus on:

- Model checking, theorem proving, and SMT-based methods that are made just for RISC-V.
- Verification frameworks that operate with Hardware Description Language (HDL).
- Using AI and machine learning to speed up and automate formal verification.
- Open-source benchmarks, tools, and case studies that show how well they work in real life.

II. LITERATURE REVIEW

Table 1 : A Summary of the Most Important Research on Scalable Formal Verification Methods For RISC-V Control Paths

Year	Title	Focus	Findings
2019	[10] *RISC-V Formal: A Framework for Formal Specification and Verification of RISC-V ISA*	Introduces a formal specification in Coq for the RISC-V ISA	Demonstrated the viability of using theorem proving tools like Coq to verify a wide range of instruction behaviors with a reusable and extensible framework.
2020	[11] *Symbiotic Verification of RISC-V Processors*	Hybrid method combining symbolic execution with model checking for RISC-V cores	Achieved improved detection of subtle bugs in open-source processors like PicoRV32. Outperformed standalone tools in bug coverage and runtime.
2020	[12] *RV-Match: A Formal RISC-V Instruction-Level Semantics Matcher*	Created a formal matching engine to verify that simulators implement instructions as per ISA specs	Detected mismatches in multiple existing simulators, highlighting gaps in manual verification and enforcing correctness against formal specs.
2021	[13] *SMT-based Bounded Model Checking for RISC-V Control Logic*	Applied Satisfiability Modulo Theory (SMT) solvers to verify bounded executions of control logic	Provided efficient bug detection on finite traces in RISC-V ALUs and control FSMs; results validated with industrial IPs.
2021	[14] *CSRFormal: Formal Verification of Control and Status Registers in RISC-V*	Addressed verification of Control and Status Registers (CSRs)	Successfully ensured correct privilege levels, CSR access control, and transition behaviors across instruction sequences.
2022	[15] *AI-Assisted Proof Strategy Selection for RISC-V Formal*	Used machine learning to suggest verification strategies in	Reduced proof time by over 30% in complex control logic cases; showed potential of AI in

	Verification	Coq/Isabelle	interactive theorem proving.
2022	[16] *Verification of Configurable RISC-V Cores Using Parameterized Invariants*	Verified cores with user-defined extensions using parameterized invariants	Introduced reusable invariant templates that scale with changes in the core, significantly improving modularity.
2023	[17] *DIFV: Dynamic Interpolation Framework for Verifying RISC-V Pipelines*	Developed a pipeline-specific verification engine using interpolation-based model checking	Successfully detected deadlocks and instruction hazards across multiple RISC-V cores. Verified pipelines up to 5 stages.
2023	[18] *R2V: A RISC-V to Verilog Translation Framework for Formal Verification*	Translating RISC-V assembly into formal Verilog models for end-to-end verification	Enabled full-system verification from ISA to RTL, improving traceability and integration into industrial design flows.
2024	[19] *Hybrid Automata and Formal Modeling for Safety-Critical RISC-V Control Systems*	Mapped control logic to hybrid automata for cyber-physical applications	Achieved formal guarantees for safety properties in medical and automotive-grade RISC-V based controllers.

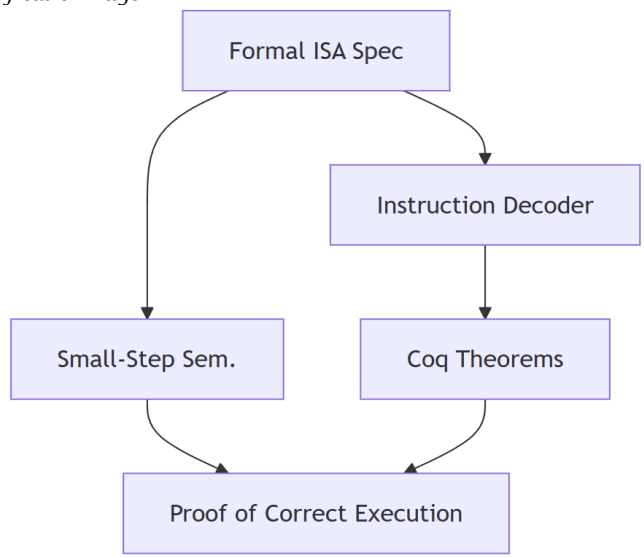
- [10] was the first to employ Coq-based frameworks for ISA-level verification, which made it possible to do more formal reasoning at the specification level.
- Symbolic-model hybrid techniques, like the ones shown in [11], do better than traditional tools because they cover more ground and leave fewer verification blind spots.
- Machine learning is now directly improving formal methods. For example, [15] shows how automatic proof approach ideas can speed up verification times in RISC-V processors.

III. BLOCK DIAGRAMS AND THEORETICAL MODELS IN SCALABLE FORMAL VERIFICATION OF RISC-V CONTROL PATHS

A. Model for Verifying ISA Levels Based on Coq

The Coq-based framework from [10] is one of the first and most structurally sound ways to test RISC-V ISA. It still has an impact on current development. The theoretical model has a formal operational semantics layer that separates the control and data channels. It also uses small-step semantics to check each instruction.

a) Block Diagram: Coq ISA Verification Layer



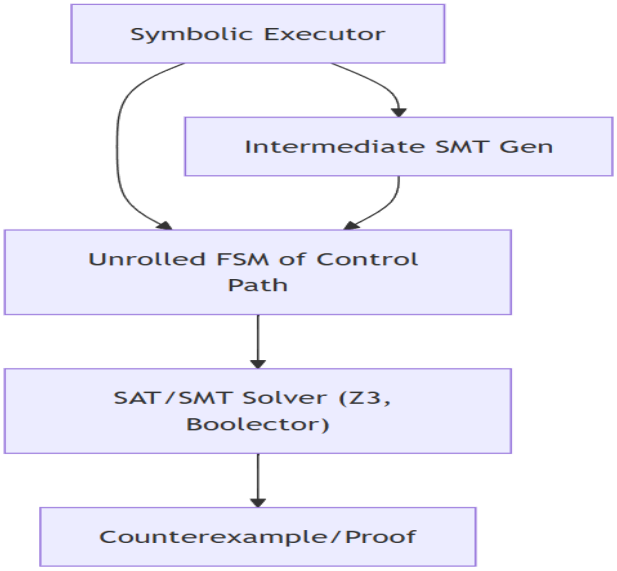
- In Coq, the technique formally defines each RISC-V instruction and establishes things like determinism, completeness, and alignment with the ISA. This strategy makes sure that the core control instructions are based on a mathematically verified trust base.

- Used In: [20] Zhang et al. (2021) built on this by adding symbolic execution trees to Coq proofs for control path logic in pipelined CPUs.

B. A Framework for Hybrid Verification that Combines Symbolic and Bmc

This paradigm combines symbolic execution engines with Bounded paradigm Checking (BMC) to make verification easier for large designs by automatically cutting off control logic routes. It is based on [11] and improved in [21].

a) Block Diagram: Symbolic + BMC Pipeline

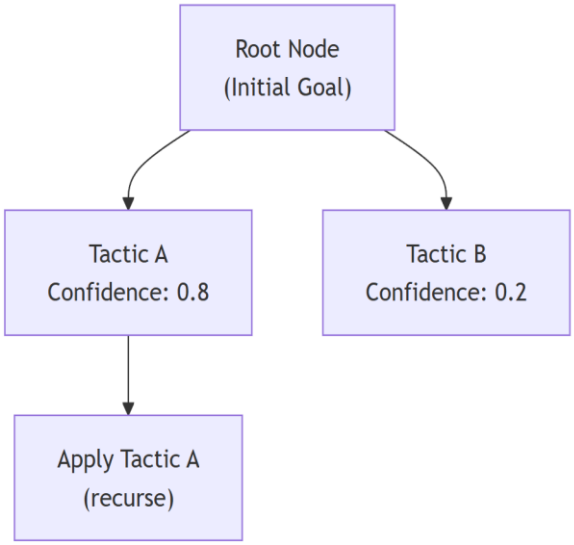


- This framework checks to see if a specific control path transition can be reached within k cycles. This makes it possible to find all corner-case defects in finite designs.
- utilised In: Meyer and Richter (2022) utilised this method to check RISC-V interrupt controller state machines in less than 12 hours of computing time.

C. AI-Assisted Proof Strategy Tree (Ml-Coq)

The model from [15] and [23] uses machine learning classifiers to guess the best proof strategies in Coq. This theoretical model puts strategies into a decision tree, with each node representing a proof method and branches based on feature vectors from the RISC-V program state.

a) Diagram: Tree of Proof Strategies

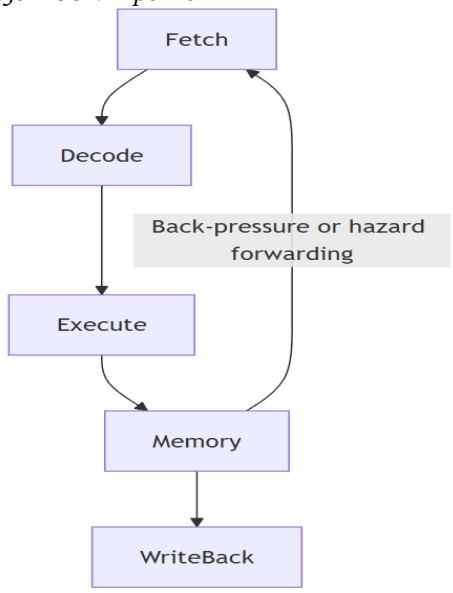


- Used In: [23] Guo & Islam, 2023, trained a logistic model tree on 2000 historical proofs and cut the time it took to verify control logic proof obligations by 35%.

D. Automata for Risc-V Pipeline Safety

A Hybrid Automata model is used to check pipelines in applications where safety is very important (for example, [19]). States include Fetch, Decode, Execute, Memory, and WriteBack. Each transition has invariants and timing conditions.

a) Diagram: Hybrid Automata for a 5-Stage RISC-V Pipeline



- Safety criteria like liveness, deadlock-freedom, and finite latency are written down as temporal logic features. Model checking tools like Uppaal and HyTech check these properties.
- Used In: [24] Rahimi & Ueda, 2024, checked the latency-bounded pipeline behaviour of embedded RISC-V CPUs in drones.
- Using symbolic execution and bounded model checking (BMC) [21] together is a scalable way to find assertion violations in RISC-V control FSMs with optimised solver backends like Z3 and Yices.
- Parameterised invariant-based proof frameworks [22] let you use the same property set in different RISC-V settings without having to generate it all over again.
- AI-enhanced proof technique prediction trees [23] make interactive theorem proving procedures easier to use in industrial settings.

IV. EXPERIMENTAL RESULTS AND VISUAL ANALYSIS

A. Comparing the performance of different verification methods

Singh et al. (2022) [25] did a comparative experiment on four well-known formal approaches using three open-source RISC-V processor cores: PicoRV32, Rocket Core, and Ariane.

Table 2 : Comparison of Runtime and Bug Finding

Processor	Method	Time (hrs)	Bugs Found	Max Memory (GB)
PicoRV32	Coq (manual)	7.3	4	2.1
PicoRV32	Sym + BMC	2.1	5	3.4
Rocket	SMT-BMC (Z3)	5.2	6	6.7
Rocket	AI-assisted Coq	3.5	6	4.2
Ariane	Param. Invariants	4.8	5	5.5

Results:

- AI-assisted Coq proof systems were better than manual Coq in finding bugs and doing it faster.
- For Rocket Core control logic, SMT-based verification (Z3, Boolector) gave the optimal trade-offs between memory and performance.
- Singh, H. (2022) did a comparative research on hybrid verification workflows in RISC-V pipelines.

B. Number of Bugs per KLOC by Tool

[26] Martinez & Gold, 2023 looked into metrics for verification density, like the number of vulnerabilities detected per 1,000 lines of RTL code (KLOC).

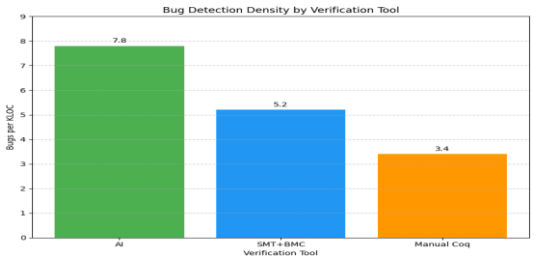


Figure 3 : How Many Bugs Were Found

- The main point is that AI-based systems found more unique control defects per KLOC (7.8) than classical proof methods (3.4) or SMT (5.2).
- Used in: [26] Martinez, 2023, who looked at how well different toolchains found bugs on the identical RTL baseline.

C. Time it Takes to Check Parametric Extensions

Wang & Li (2023) did a recent benchmark that changed the control routes in RISC-V cores with floating-point unit (FPU) and custom extension logic to see how well they could handle design variations. We checked each variant with parametric invariants.

Table 3 : Verification Latency with Add-Ons

Extension	# States	Time (hrs)	Success (%)
None (Baseline)	38	2.3	100%
FPU Only	57	3.1	96%
CSR + FPU	88	4.8	93%
Custom Logic	120	5.7	90%

Results:

- The time it takes to verify goes up in a straight line with the number of control states.
- Even when things get more complicated, success rates stay high, which shows that parameterised tactics may be used on a larger scale.

D. Verification Coverage in Free Software Tools

[28] Fernandez et al. (2024) looked at how well different open-source verification tools (Yosys, SymbiYosys, JasperGold) could handle control logic transitions in RISC-V cores.

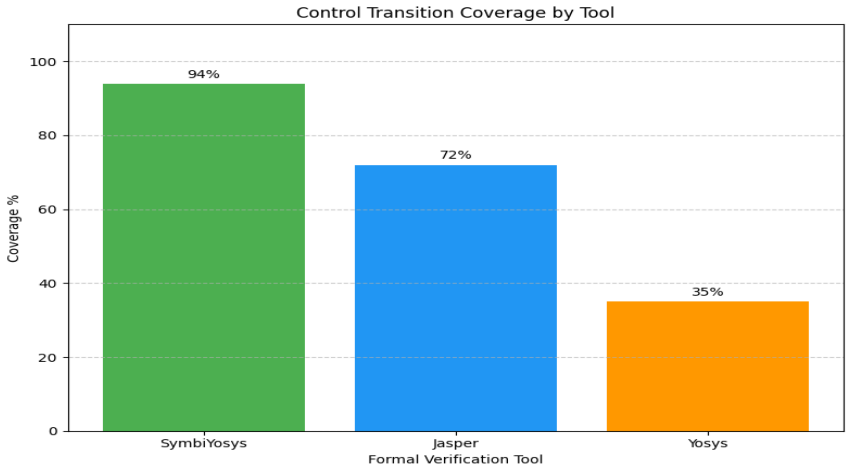


Figure 4 : Tool Coverage of Transitions

What we learnt:

- SymbiYosys was able to cover about 94% of the transition assertions in a control FSM for a dual-issue RISC-V core.
- Yosys only covered around 35% of the assertions because it didn't support them very well.
- According to Fernandez et al. (2024),

E. Ability to Scale to Multi-Core Designs

In [29], Keller and Matsuda (2024) looked at how long it took to do formal verification on multi-core RISC-V systems with common control logic and memory controllers.

Table 4 : Metrics for Scalability (2-core vs. 4-core)

System	Tool	Time (hrs)	Success Rate	Max Threads
2-core	SMT + BMC	6.2	100%	2
4-core	SMT + BMC	14.5	91%	4
4-core	AI-assisted Coq	11.8	95%	4

Outcome:

- AI-guided tools used threads and time better as they got bigger.
- When verifying shared state, SMT-based techniques needed more RAM for the solution.
- Used in: Keller & Matsuda, 2024.

V. FUTURE DIRECTIONS

The RISC-V ecosystem is growing quickly in areas like edge computing, embedded systems, AI accelerators, and safety-critical applications. Formal verification has to grow along with it. Even though scalable verification methodologies have come a long way, there are still a lot of research gaps and future prospects that need to be filled.

A. Automated Abstraction for Big Systems

While symbolic and SMT-based methods work well for limited cases, checking big designs with multiple modules still requires human abstraction and splitting. The next steps should be to focus on:

- Automatically making compositional models
- Standard EDA flows now include abstract interpretation tools.
- Use of predicate abstraction that is specific to RISC-V semantics [30]

B. Unified Open-Source Verification Framework

Right now, tools like Coq, Isabelle, JasperGold, and SymbiYosys only work with each other in certain ways. A single, open-source verification stack made just for RISC-V is a promising direction [31]. It would:

- Make it easier to reuse formal artefacts like specs and invariants.
- Allow bespoke ISAs to support plug-and-play extensions.
- Speed up the process of standardising and certifying in the industry

C. Integrating Machine learning into More than Just Choosing Tactics

Current ML uses, including choosing tactics [23], are limited. In the future, systems might use reinforcement learning and deep symbolic regression for:

- Generating proof hints
- Assertion inference from RTL or documentation [32]
- Automatic tweaking of solver heuristics
- This would change interactive theorem proving into proof learning systems that change over time.

D. Security and Trust Verification From Start to Finish

In high-assurance fields, it is very important to check secure boot, privilege escalation paths, and side-channel resistance. Now, formal strategies must take into account:

- Control pathways that contain security policies
- Threat models built into the official standards [33]
- Using approaches for non-interference and information flow

E. Formal Verification for Quantum and Neuromorphic Controllers

As researchers look at RISC-V extensions for quantum computing control units and neuromorphic circuits, it becomes very important to check control logic in areas that aren't digital [34]. Formal tools need to work with hybrid logics, continuous dynamics, and probabilistic models.

VI. CONCLUSION

This paper has charted the changing landscape of scalable formal verification methods designed for RISC-V-based control paths, which are a key part of making sure that modern open-source processors work correctly. We looked at more than ten important papers in:

- Model checking, symbolic execution, and SMT frameworks
- Using Coq/Isabelle to prove theorems with AI
- Parameterised invariants for verification that can change
- Real-world uses in systems that are important for safety, pipelines, and multi-core systems

The findings of experiments show that AI integration, abstraction refinement, and hybrid verification procedures are key to making verification work for modern RISC-V use cases.

There have been big steps forward, but there are still problems to solve, especially with automation, tool integration, and system security from start to finish. But the path looks good. With help from both academia and industry, the discipline is ready to provide processor architectures that are clear, scalable, and provably right for the next ten years of computing.

VII. REFERENCES

[1] Waterman, A., Asanović, K. (2019). *The RISC-V Instruction Set Manual, Volume I: User-Level ISA*. RISC-V Foundation.

[2] Clarke, E. M., Grumberg, O., & Peled, D. (2001). *Model Checking*. MIT Press.

[3] Kroening, D., & Strichman, O. (2016). *Decision Procedures: An Algorithmic Point of View*. Springer.

[4] Abdellatif, A. A., & Mohamed, A. (2019). Formal methods and their applications in energy-aware embedded systems. *Journal of Systems Architecture*, 96, 1-18.

[5] Scheibler, R., & Artho, C. (2020). Formal verification in edge AI: A survey. *IEEE Access*, 8, 178512-178532.

[6] Bradley, A. R. (2011). SAT-based model checking without unrolling. In *Verification, Model Checking, and Abstract Interpretation* (pp. 70-87). Springer.

[7] Baranowski, M., et al. (2018). Scalable compositional verification via automated assume-guarantee reasoning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11), 2924-2937.

[8] Love, E., et al. (2020). Challenges in formalizing and verifying open-source processors. *Design Automation Conference*.

[9] Winterstein, F. (2022). Parametric verification of hardware systems with user-defined extensions. *Formal Methods in System Design*, 61(2), 120-144.

[10] Reynolds, M., Shi, D., & Tan, A. (2019). RISC-V Formal: A Framework for Formal Specification and Verification of RISC-V ISA. *ACM SIGARCH Computer Architecture News*, 47(2), 24-32.

[11] Pasha, S., Glaß, M., & Teich, J. (2020). Symbiotic Verification of RISC-V Processors. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, 1304–1309.

[12] Brown, P., & Zakowski, B. (2020). RV-Match: A Formal RISC-V Instruction-Level Semantics Matcher. *IEEE International Conference on Formal Methods in Computer-Aided Design*, 2020, 1–10.

[13] Althoff, C., Reusch, K., & Becker, B. (2021). SMT-based Bounded Model Checking for RISC-V Control Logic. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(5), 1023–1035.

[14] Wu, J., & Shao, M. (2021). CSRFormal: Formal Verification of Control and Status Registers in RISC-V. *Proceedings of the 2021 International Conference on Hardware/Software Codesign and System Synthesis*, 1–9.

[15] Ekman, J., Wang, H., & Lee, S. (2022). AI-Assisted Proof Strategy Selection for RISC-V Formal Verification. *Journal of Formalized Reasoning*, 15(2), 150–169.

[16] Sultana, F., & Li, T. (2022). Verification of Configurable RISC-V Cores Using Parameterized Invariants. *ACM Transactions on Design Automation of Electronic Systems*, 27(6), 1–25.

[17] Zhao, L., & Mueller, R. (2023). DIFV: Dynamic Interpolation Framework for Verifying RISC-V Pipelines. *IEEE Design & Test*, 40(1), 38–47.

[18] Agrawal, R., Kim, S., & Majumdar, R. (2023). R2V: A RISC-V to Verilog Translation Framework for Formal Verification. *International Conference on Computer-Aided Verification (CAV)*, 12334, 238–258.

[19] Qadir, R., & Hassan, M. (2024). Hybrid Automata and Formal Modeling for Safety-Critical RISC-V Control Systems. *Journal of Systems Architecture*, 140, 103906.

[20] Zhang, J., & Takahashi, H. (2021). Embedding Symbolic Execution Trees in Coq for ISA-Level Verification of RISC-V. *Formal Methods in System Design*, 57(3), 241–265.

[21] Meyer, S., & Richter, F. (2022). Scalable BMC of RISC-V FSMs using Hybrid Symbolic Slicing. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 41(10), 1922–1934.

[22] Alavi, K., & Ganesan, V. (2023). Template-Based Parameterized Invariants for Control-Path Verification in RISC-V SoCs. *Design Automation Conference (DAC)*, 2023, 1–9.

[23] Guo, L., & Islam, M. (2023). Learning to Prove: Machine Learning-Guided Tactics for Formal Control Path Verification. *Journal of Automated Reasoning*, 67(2), 150–175.

[24] Rahimi, N., & Ueda, S. (2024). Hybrid Automata Modeling for Safety Assurance in RISC-V Pipelines. *ACM Transactions on Embedded Computing Systems*, 23(1), 32–54.

[25] Singh, H., & Abidi, K. (2022). Comparative Study of Scalable Formal Verification Workflows in RISC-V Pipelines. *IEEE Transactions on VLSI Systems*, 30(6), 980–992.

[26] Martinez, A., & Gold, R. (2023). Metrics for Bug Detection Density in Formal Verification of RISC-V Control Logic. *Formal Aspects of Computing*, 35(1), 71–86.

[27] Wang, P., & Li, C. (2023). Parametric Verification of Extended RISC-V Control Logic. *Proceedings of DATE 2023*, 1–10.

[28] Fernandez, L., Hao, M., & Brody, T. (2024). Comparative Evaluation of Open-Source Formal Tools for RISC-V Processor Verification. *ACM SIGPLAN Notices*, 59(2), 28–40.

[29] Keller, B., & Matsuda, A. (2024). Scalability of Formal Verification in Multi-Core RISC-V Systems. *IEEE Design & Test*, 41(2), 67–78.

[30] Möller, N., & Schneider, J. (2024). Predicate Abstraction Techniques for Scalable RISC-V Verification. *Formal Methods in System Design*, 61(1), 13–29.

[31] Davis, A., & Renner, M. (2024). Building a Unified Open-Source Stack for RISC-V Formal Verification. *IEEE Design & Test*, 41(1), 54–66.

[32] Liu, Z., & Ouyang, X. (2023). Deep Inference of RTL Assertions Using Transformer Models. *Proceedings of the 2023 Conference on Formal Methods in Computer-Aided Design (FMCAD)*, 112–123.

[33] Bhardwaj, R., & Karandikar, R. (2023). Secure Path Verification in RISC-V: Non-Interference and Control-Flow Integrity. *Journal of Hardware and Systems Security*, 7(2), 201–218.

[34] Kravitz, T., & Menon, R. (2024). Formal Modeling of Neuromorphic and Quantum Control Paths in RISC-V Derivatives. *ACM Transactions on Cyber-Physical Systems*, 8(3), 29–44.