Towards Reliable and Secure RISC-V Systems: Survey of Testability and Security Mechanisms

Mahreen Khan, Maria Mushtaq, Ludovic Apvrille

Towards Reliable and Secure RISC-V Systems: Survey of Testability and Security Mechanisms

Mahreen Khan
Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
mahreen.khan@telecom-paris.fr

Maria Mushtaq
Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
maria.mushtaq@telecom-paris.fr

Ludovic Apvrille
Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
ludovic.apvrille@telecom-paris.fr

***Abstract*—RISC-V has emerged as a versatile open-source instruction set architecture, enabling extensible microarchitectures, custom accelerators, and domain-specific processors. Its openness facilitates innovation in testability, safety, and security for safety-critical and security-sensitive applications. This survey provides a comprehensive review of recent research in RISC-V verification and protection mechanisms. We analyze AI-assisted test generation, statistical fault injection frameworks, system-level testing, design-for-test architectures, and hardware-software co-verification methods. In the safety domain, we discuss temporal isolation, performance monitoring, debug support, and fault containment strategies. Security mechanisms, including trusted execution environments, memory protection, cryptographic ISA extensions, post-quantum acceleration, and secure debug practices, are evaluated. Open challenges in scalable test coverage, AI-enabled certification, side-channel resilience, and lifecycle management are highlighted. Finally, we outline future research directions that leverage RISC-V's modularity and openness to enable trustworthy computing systems in automotive, aerospace, telecommunications, and edge computing domains.**

***Index Terms*—RISC-V, testability, functional safety, security, trusted execution environment, post-quantum cryptography, hardware-software co-verification, temporal isolation, design-for-test, cryptographic ISA extensions**

I. Introduction

The RISC-V instruction set architecture (ISA) has gained wide adoption as an open and modular platform for processor design. Its extensible ISA allows researchers and industry practitioners to implement custom accelerators, vector units, and domain-specific cores while maintaining compatibility with a shared software ecosystem. The base RISC-V ISA and its standard extensions define a stable foundation upon which implementations can vary, which creates a challenge for rigorous verification and validation across different configurations and custom extensions [1]. RISC-V also specifies privileged features such as Physical Memory Protection (PMP) and a standardized debug interface to support isolation, memory access control, and system inspection [2] [3].

Surveys on verification methods highlight simulation, formal verification, and automated test planning as parts of comprehensive verification methodologies [1]. Work on secure enclaves and trusted execution environments surveys categorical approaches to isolation and confidentiality in RISC-V systems [4]. However, these existing surveys typically focus on either verification in general or a specific domain such as trusted execution, and do not provide an integrated perspective that spans test generation, safety mechanisms, and security enhancements in RISC-V designs.

This survey fills these gaps by reviewing state-of-the-art methods across test generation, safety architecture, and security mechanisms within the RISC-V ecosystem.

A. Contributions

The main contributions of this survey are:

- A review of advanced RISC-V test generation methods, including AI-assisted pattern generation, statistical fault injection frameworks, system-level testing, design-for-test architectures, and hardware-software co-verification approaches.
- An analysis of safety-oriented architectural features, including privilege-level fault handling, performance monitoring, debug modules, and mechanisms for temporal isolation in multicore systems.
- A detailed examination of security mechanisms, such as trusted execution environments (TEEs), memory protection, cryptographic ISA extensions, post-quantum cryptography acceleration, and secure debug and anti-tampering techniques.
- Identification of open challenges in RISC-V deployment, including scalable test coverage, certification of learning-enabled components, side-channel mitigation, and post-quantum transition.
- Discussion of future research directions that leverage the openness and modularity of RISC-V to enable trustworthy, safety-critical, and security-compliant computing systems.

B. Paper Organization

The remainder of the paper is structured as follows. Section II surveys advances in testability and verification methodologies for RISC-V processors and SoCs. Section III discusses architectural support for temporal isolation and safety features. Section IV covers cryptographic acceleration and secure debug mechanisms. Section V identifies open challenges and future research directions. Section VI concludes the survey with a summary and outlook for RISC-V in critical applications.

II. Advances in RISC-V Testability

This section reviews recent advances in RISC-V testability, organized into five key areas: AI-driven test generation, statistical fault injection for security assessment, system-level test innovations, design-for-test (DfT) architectures, and hardware-software co-verification. These developments collectively enhance the ability to verify, validate, and test RISC-V processors across the design lifecycle, from pre-silicon verification to post-silicon debug and manufacturing test.

A. AI-Driven and Fuzzing-based Test Generation

Building on earlier constrained-random algorithm approaches [5], [6], machine learning techniques have revolutionized test pattern generation for RISC-V processors. Verification of RISC-V processors has also developed on coverage-driven instruction generation techniques [7]. These methods systematically explore the instruction space to maximize functional coverage and uncover corner-case behaviors.

Chen et al. [8] introduced a deep reinforcement learning framework that generates instruction sequences maximizing toggle coverage while minimizing test time. Their approach achieved 95.4% average coverage across various benchmarks. Differential fuzzing approaches, such as DifuzzRTL [9], have also been applied to RISC-V RTL designs, allowing the comparison of multiple implementations under identical instruction sequences to identify inconsistencies.

B. Statistical Fault Injection Evolution for RISC-V Security

The methodology for security assessment of the RISC-V Instruction Set Architecture (ISA) has evolved from ad-hoc testing to sophisticated, simulation-based statistical fault injection (SFI). This evolution directly addresses the need for scalable, reproducible vulnerability analysis in an open ecosystem of diverse hardware implementations [10]. Early security evaluations of RISC-V were limited to physical hardware testing. While providing real-world results, this approach lacked the fine-grained control and observability necessary for deep, root-cause analysis of microarchitectural vulnerabilities and the development of effective countermeasures.

A pivotal shift occurred with the development of specialized simulation frameworks. The gem5-MARVEL framework [10] enables microarchitecture-level resilience analysis of heterogeneous SoC architectures, including fault injection capabilities for vulnerability assessment within a full-system simulator. Its subsequent extension to model multi-core RISC-V systems with shared cache hierarchies was critical for accurate, system-level vulnerability assessment, as it allowed researchers to track fault propagation across cores and coherent memory. This paradigm enables the statistical injection of transient, intermittent, and permanent fault models under highly configurable and repeatable conditions.

The most recent evolution seeks to bridge the gap between simulation flexibility and hardware timing accuracy through hardware-accelerated platforms. The FPGA-based Chiffre platform [11] exemplifies this trend, enabling real-time fault injection into RISC-V cores synthesized on adaptable hardware. This methodological progression, from physical testing to simulated SFI, and now to accelerated hybrid platforms, provides the RISC-V community with a powerful, scalable toolkit for statistically rigorous security evaluation.

C. System-Level Test Innovations

System-level testing for RISC-V processors and SoCs has advanced significantly to address the challenges of functional verification, compliance, and integration validation across simulation, emulation, and silicon environments. Randomized instruction sequence generation and constrained-random stimulus tools, such as Synopsys STING, produce test programs that exercise privilege levels, memory protection, interrupt handling, and other system behaviors in a portable, self-checking manner across simulation, FPGA prototypes, and silicon platforms [12]. These generated tests have been effective at uncovering corner cases such as cache coherence conflicts and fence instruction mishandling that directed tests often miss.

The RISC-V community also standardizes randomized instruction testing frameworks like TestRIG, which uses RVFI-DII interfaces to drive random instruction streams against both reference models and implementations under test, comparing execution traces to detect divergences early in development [13]. TestRIG's approach supports rapid iteration and debugging, and has been applied to validate ISA extensions such as capability architectures.

Comprehensive system-level verification suites are provided by commercial and open-source efforts, such as Breker's RISC-V SoCReady SystemVIP test suite, which synthesizes coverage-driven tests for complex operations, including multicore coherency, atomic instructions, paging, interrupts, system memory protection, and performance profiling. SystemVIP's portability across simulation, emulation, prototype, and post-silicon phases enables unified validation across the design lifecycle [14].

Open-source directed test frameworks like RiESCUE further facilitate customizable test generation and compliance testing with support for multiple RISC-V extensions, self-checking test harnesses, and flexible constraints [15]. Together, these innovations improve functional coverage, reduce manual effort, and help ensure robust RISC-V SoC validation from pre-silicon verification through product release.

D. Design-for-Test Architectures

Design-for-Test (DfT) support is critical for ensuring manufacturability and correctness of complex RISC-V SoCs. Traditional scan-chain techniques and built-in self-test (BIST) remain foundational, but recent research has adapted and evaluated these methods in the context of highly configurable RISC-V designs. Scan-based DfT remains relevant for RISC-V cores, enabling sequential element controllability and observability during test shifts, which supports automatic test pattern generation (ATPG) for structural faults such as stuck-at and transition faults [16].

Functional and compliance verification frameworks for RISC-V benefit from constrained random and reference-model-driven testing, which augment classical DfT by exercising ISA-defined behaviors in simulation and emulation environments. Qiu and Liu propose an integrated UVM-TLM co-simulation framework that co-verifies functional correctness alongside performance models for superscalar RISC-V cores, effectively bridging between high-level verification and structural test validations [17]. Likewise, Galimberti et al. describe functional ISS-driven verification that interfaces a RISC-V instruction set simulator with RTL testbenches to achieve high coverage across pipeline and control flows [18].

For RISC-V vector and ISA extension units, instruction coverage and workload-driven tests have been introduced to supplement structural DfT. Lai et al. develop an instruction coverage analysis methodology based on TSVC (Test Set Vector Coverage) to systematically quantify RVV (RISC-V Vector Extension) coverage, forming the basis for improved test suite design that captures complex vector behaviors [19]. These approaches, while not purely DfT in the classical hardware sense, contribute to comprehensive test architectures that combine structural techniques with functional, compliance, and coverage-driven methods tailored for RISC-V's extensible ISA.

Overall, DfT for RISC-V SoCs is evolving toward hybrid verification strategies that integrate structural scan/BIST with advanced simulation-based frameworks and ISA-aware coverage analysis, enabling more thorough validation of base cores and custom extension blocks across pre-silicon and prototype phases.

E. Hardware-Software Co-Verification

Hardware-software co-verification plays a central role in ensuring that RISC-V implementations correctly realize the instruction set architecture and interact safely with system software. Formal verification approaches have been applied to RISC-V cores to prove ISA compliance and functional correctness at varying abstraction levels. Polynomial Formal Verification (PFV) techniques, for example, use Binary Decision Diagrams (BDDs) and related structures to bound verification complexity and have been demonstrated on multi-cycle RISC-V cores covering both combinational and sequential logic [20]. These methods provide mathematical guarantees of correctness, complementing traditional dynamic verification.

Symbolic execution and co-simulation combine strengths from software and hardware verification domains. In the work by Bruns et al., symbolic execution engines such as KLEE are integrated with co-simulation frameworks that couple an instruction set simulator (ISS) with the RTL of a RISC-V processor under test. This allows symbolic exploration of both hardware and software paths and has been effective in revealing subtle mismatches between simulation and RTL behavior [21]. Such symbolic co-verification expands coverage beyond concrete stimuli, enabling systematic exploration of corner cases inherent to HW-SW interactions.

Hybrid co-verification frameworks also incorporate virtual prototypes and accelerated platforms to balance fidelity and performance. FPGA-assisted emulation frameworks like FERIVer accelerate RTL verification by cross-validating hardware behavior against software models running on the same platform, achieving orders-of-magnitude speedups while preserving functional correctness checks [22]. Complementary to these, unified co-simulation infrastructures using UVM (Universal Verification Methodology) and SystemC/TLM provide scalable environments where software workloads drive hardware models, enabling early detection of integration and performance issues across design iterations [17].

The RISC-V formal verification ecosystem also includes open frameworks such as the riscv-formal project. This framework is widely used to express properties and formally check RTL models against reference ISA specifications using the RISC-V Formal Interface (RVFI), often as part of hybrid verification flows that mix formal, simulation, and co-simulation components. These layered methodologies improve coverage, reduce manual effort, and support rigorous correctness assurance from pre-silicon to prototype stages [23].

Table I summarizes the key RISC-V testability techniques, categorized by approach and representative tools.

III. Security Architecture and Isolation

This section focuses on RISC-V mechanisms that provide isolation, determinism, and architectural foundations for secu-

TABLE I
SUMMARY OF ADVANCES IN RISC-V TESTABILITY

Category	Paper / Tool	Testing Technique
AI-Driven and Fuzzing-Based Test Generation	Chen et al. [8]	Deep reinforcement learning for instruction sequence generation
	DifuzzRTL [9]	Differential fuzzing across RTL implementations
	Lu et al. [7]	Coverage-driven instr. generation
Statistical Fault Injection for Security	gem5-MARVEL [10]	Simulation-based statistical fault injection in full-system
	Chiffre [11]	FPGA-based fault injection
System-Level Test Innovations	Synopsys STING [12]	Constrained-random system-level stimulus generation
	TestRIG [13]	Random instruction testing with RVFI-DII trace comparison
	SystemVIP [14]	Coverage-driven SoC-level verification suite
	RiESCUE [15]	Constrained test generation
Design-for-Test Architectures	Scan/BIST [16]	Scan-chain DfT with ATPG support
	UVM-TLM [17]	Integrated functional and performance co-simulation
	ISS-driven verification [18]	ISS-RTL interfacing for functional coverage
	TSVC for RVV [19]	Instruction coverage quantification for vector extension
Hardware-Software Co-Verification	PFV [20]	Polynomial formal verification using BDD-based techniques
	Symbolic co-simulation [21]	Symbolic execution integrated with ISS-RTL co-simulation
	FERIVer [22]	FPGA-assisted cross-validation of RTL and software models
	riscv-formal [23]	Formal property checking via RVFI interface

rity. It covers temporal isolation techniques, safety-oriented features built into the privileged architecture, trusted execution environments, and memory protection. These hardware–software co-designed approaches exploit the openness of RISC-V to enable analyzable isolation.

Attacker Model: The mechanisms discussed in this section assume a powerful software adversary operating within the system but outside the trusted computing base. The attacker may execute arbitrary code at user or supervisor privilege levels and may control the operating system or other co-resident software components. In multicore environments, the adversary may run concurrently on separate cores and attempt to interfere with victim execution through shared microarchitectural resources such as caches, memory hierarchies, and interconnects. The attacker aims to violate spatial or temporal isolation guarantees by triggering unauthorized memory accesses, inducing privilege escalation, or exploiting timing-based interference channels. The hardware implementation, including machine-mode firmware or security monitors where applicable, is assumed to be trusted and correctly enforcing architectural rules. Physical attacks, invasive fault injection, and malicious hardware modifications are outside the scope of this model unless explicitly stated.

A. Temporal Isolation and Determinism

Temporal isolation requires bounding interference originating from speculation, shared caches, and memory subsystems. In the RISC-V ecosystem, predictability is often achieved by constraining or eliminating microarchitectural features that introduce nondeterminism.

Vicuna introduces a timing-predictable RISC-V vector coprocessor designed explicitly for real-time systems [24]. The architecture removes dynamic scheduling and speculative execution within the vector unit, enabling static timing analysis and tight worst-case execution time (WCET) bounds. Experimental evaluation demonstrates analyzable execution latency across vector workloads without reliance on speculative mechanisms.

The Rocket and BOOM RISC-V cores provide openly documented microarchitectural implementations that have enabled systematic study of speculation and timing behavior [25]. While BOOM is an out-of-order speculative processor, its open RTL description allows researchers to selectively disable speculation, configure cache hierarchies, and evaluate timing interference under controlled conditions. This transparency facilitates experimental assessment of temporal isolation mechanisms.

Open-source RISC-V platforms such as PULP (Parallel Ultra-Low Power) provide lightweight in-order cores (e.g., RI5CY) with simplified pipelines and predictable execution characteristics [26]. These cores are frequently used in safety-oriented embedded contexts due to their reduced microarchitectural complexity and analyzable timing behavior.

Recent analyses of speculative execution in RISC-V cores have shown that transient execution behavior can affect timing predictability [27]. Although speculative attacks were first demonstrated on x86 and ARM, subsequent studies confirm that similar transient behaviors can manifest in speculative RISC-V microarchitectures when branch prediction and out-of-order execution are enabled. This reinforces the importance of carefully configuring RISC-V cores when temporal isolation is required.

Importantly, RISC-V does not standardize microarchitectural-level temporal isolation or Quality-of-Service mechanisms at the ISA level. Instead, determinism is achieved through microarchitectural configuration choices, simplified pipelines, and analyzable coprocessor designs within the open RISC-V hardware ecosystem.

B. Safety-Oriented Architectural Features

Safety-oriented RISC-V systems rely on architectural transparency, standardized debug support, and hardware monitoring capabilities rather than proprietary safety extensions. The open specification enables integration of safety mechanisms consistent with functional safety standards such as ISO 26262.

The RISC-V privileged architecture defines precise trap handling, interrupt delegation, and control and status registers (CSRs) that enable structured fault detection and recovery [2]. Machine-mode exception handling and vectored interrupts provide deterministic fault containment paths, which are essential in safety-critical deployments.

Performance monitoring counters (PMCs), standardized through the RISC-V "Zihpm" extension, provide architectural support for counting hardware events such as cycles, instructions retired, cache events, and branch behavior [28]. These counters can be used for runtime monitoring, anomaly detection, and deadline supervision in safety-aware systems. Because PMCs are architecturally defined, they provide portable observability across compliant RISC-V cores.

The RISC-V debug specification defines a standardized external debug architecture supporting hardware breakpoints, triggers, single-step execution, and abstract command interfaces [3]. The trigger module allows programmable match conditions on instruction addresses, data accesses, and exceptions, which can be used for safety validation and fault analysis. The debug module operates through a dedicated debug transport interface, enabling non-intrusive system inspection during validation and certification processes.

Open-source RISC-V cores such as Rocket and BOOM provide fully documented RTL implementations of interrupt controllers, CSR subsystems, and debug modules [25], [29]. This openness facilitates formal verification and fault-injection campaigns to evaluate diagnostic coverage and failure modes, which are central requirements in safety-certified designs.

In practice, watchdog timers, lockstep execution, and redundancy mechanisms are implemented at the platform or SoC level rather than through ISA-defined extensions. The modular structure of RISC-V allows integration of these safety mechanisms while preserving ISA compliance.

C. Trusted Execution Environments

Trusted Execution Environments (TEEs) on RISC-V platforms build upon the architecture's open privileged specification and Physical Memory Protection (PMP) mechanism to provide isolated execution environments. RISC-V enables customizable hardware–software co-design for enclave isolation.

Sanctum introduced a secure processor architecture that enforces strong isolation between software components without relying on trusted system software [30]. Sanctum eliminates shared microarchitectural state between enclaves and untrusted software, thereby mitigating cache-based side-channel leakage. Sanctum's architectural principles informed later enclave designs in open ISAs, including RISC-V.

Keystone is the first open-source TEE framework specifically designed for RISC-V [31]. It leverages RISC-V's PMP to isolate enclave memory regions and introduces a minimal security monitor operating in machine mode. Keystone supports customizable enclave runtimes and remote attestation while maintaining a trusted computing base (TCB). The framework demonstrates that RISC-V's modular privilege architecture enables flexible and formally analyzable enclave designs.

To address speculative execution attacks within enclave environments, Mi6 proposes a secure enclave architecture that partitions microarchitectural resources such as caches and branch predictors to eliminate cross-domain leakage [32]. Implemented on the open-source RiscyOO out-of-order RISC-V processor, Mi6 demonstrates that strong isolation requires both architectural and microarchitectural enforcement mechanisms. The design highlights the importance of integrating speculative execution controls into enclave-capable RISC-V cores.

Collectively, these works establish that RISC-V TEEs rely on PMP-based memory isolation, minimal privileged security monitors, and microarchitectural partitioning to provide confidentiality and integrity guarantees. The openness of the RISC-V ecosystem enables rigorous evaluation of enclave security properties and facilitates reproducible hardware-level experimentation.

D. Memory Protection and Isolation

Memory protection in RISC-V systems is primarily enforced through the Physical Memory Protection (PMP) mechanism defined in the privileged architecture specification [2]. PMP enables machine-mode software to define region-based access control policies for lower privilege levels, allowing isolation between operating systems, hypervisors, and secure monitors. Regions can be configured with read, write, and execute permissions, and entries may be locked to prevent modification after initialization. This hardware-supported region enforcement forms the foundation for enclave frameworks and secure monitor designs on RISC-V platforms.

Beyond region-based isolation, capability-based protection has been explored through CHERI-RISC-V [33]. CHERI extends the RISC-V ISA with hardware capabilities that encode bounds, permissions, and provenance within pointers. By enforcing spatial memory safety and fine-grained compartmentalization in hardware, CHERI-RISC-V demonstrates how architectural extensions can mitigate memory safety violations at their source. The integration preserves compatibility with the RISC-V software ecosystem while introducing tagged memory and capability-aware load/store instructions.

Together, PMP-based region isolation and capability-based memory protection illustrate two complementary approaches within the RISC-V ecosystem: coarse-grained privilege-enforced isolation and fine-grained pointer-level protection. The openness of the RISC-V specification enables both mechanisms to be formally specified, implemented, and experimentally evaluated across research and industrial platforms. Table II provides an overview of RISC-V security mechanisms.

TABLE II
SUMMARY OF SECURITY AND ISOLATION MECHANISMS IN RISC-V

Category	Paper	Mechanism / Technique
Temporal Isolation	Vicuna [24]	Predictable vector coprocessor
	Rocket/ BOOM [25], [29]	Configurable cores; selective speculation control
	PULP [26]	In-order cores with simple pipeline
Safety Features	RISC-V Priv. Spec [2]	Trap handling, CSRs, deterministic fault containment
	RISC-V PMCs [28]	Performance counters for runtime monitoring
	RISC-V Debug Spec [3]	Debug modules and triggers for fault analysis
	Rocket / BOOM [25], [29]	Open RTL enabling verification and fault injection
Trusted Exe. Environments	Sanctum [30]	Strong enclave isolation; no shared microarchitectural state
	Keystone [31]	PMP-based memory isolation with minimal monitor
	Mi6 [32]	Partitioning of caches and predictors to prevent leakage
Memory Protection	RISC-V PMP [2]	Region-based read/write/execute access control
	CHERI-RISC-V [33]	Capability-based fine-grained memory protection

IV. CRYPTOGRAPHIC AND HARDWARE SECURITY

This section addresses cryptographic acceleration and physical security features in RISC-V systems. It reviews post-quantum cryptography implementations on RISC-V cores, standardized cryptographic ISA extensions, and secure debug/anti-tampering mechanisms. Collectively, these capabilities demonstrate how the RISC-V ecosystem enables both high-performance cryptography and robust platform security.

A. Post-Quantum Cryptography Acceleration

The standardization of post-quantum cryptography (PQC) by NIST has motivated research on efficient implementations of lattice-based and isogeny-based algorithms on RISC-V platforms. Both software-optimized and hardware-accelerated approaches have been explored for embedded and low-power cores.

Banerjee et al. [34] presented the first full hardware implementation of the DTLS 1.3 protocol for IoT devices, integrating a reconfigurable prime-field elliptic curve cryptography (ECC) accelerator with a low-power RISC-V processor.

Fritzmann et al. [35] proposed RISQ-V, a hardware/software co-design that tightly integrates dedicated lattice-based cryptography accelerators into a RISC-V core and extends the ISA with 29 custom instructions. Their ASIC and FPGA implementations achieve significant speedups and energy reductions for schemes such as NewHope, Kyber, and Saber compared to pure software on baseline RISC-V. Wang et al. [36] presented an optimized hardware–software co-design for Kyber and Dilithium on a RISC-V SoC FPGA platform. They integrate customized polynomial accelerators, optimized hashing via RISC-V assembly, and a multi-core acceleration scheme, achieving 3–5× overall speedup across security levels while using less than 5% of FPGA resources.

Isogeny-based cryptography, such as SIKE, has been evaluated on embedded RISC-V cores [37]. While highly computational, these implementations demonstrate feasibility on low-power RISC-V microcontrollers and highlight trade-offs between execution time, memory footprint, and security. Overall, RISC-V's open ISA and flexible microarchitectures allow both software-level optimizations and integration of specialized accelerators, making it a suitable platform for exploring PQC in embedded, IoT, and safety-critical environments.

B. Cryptographic ISA Extensions

RISC-V standardizes cryptographic acceleration through architecturally defined instruction set extensions rather than vendor-specific coprocessors. The RISC-V Scalar Cryptography Extensions (Zk*) provide instruction-level support for symmetric cryptography, hashing, and finite-field arithmetic [38]. These extensions include AES round instructions (Zkne, Zknd), SHA-2 and SHA-3 acceleration (Zknh), and carryless multiplication (Zbc) for Galois/Counter Mode (GCM). By integrating cryptographic primitives directly into the ISA, implementations can achieve improved performance while preserving software portability across compliant cores.

Bit manipulation extensions (Zb*) further support cryptographic workloads [39]. Instructions for bit rotations, generalized bit permutations, and carryless multiplication are essential for efficient implementations of stream ciphers, authenticated encryption schemes, and hashing functions. The standardization of these primitives enables constant-time software implementations without reliance on proprietary instructions.

Entropy generation is defined through the RISC-V Entropy Source Extension (Zkr), which specifies a standard architectural interface for hardware random number generators [38]. The extension provides mechanisms for accessing conditioned entropy and mandates compliance requirements aligned with established randomness standards. By defining entropy access at the ISA level, RISC-V enables portable and auditable

implementations of cryptographically secure random number generation.

Together, the cryptography, bit manipulation, and entropy extensions illustrate the RISC-V approach to security: architecturally specified, openly documented, and implementation-agnostic acceleration mechanisms that preserve interoperability across the ecosystem.

C. Secure Debug and Anti-Tampering

Debug interfaces constitute a critical security boundary in processor-based systems. The RISC-V External Debug Support specification defines a standardized debug module, debug transport module (DTM), and abstract command interface for controlling harts and accessing system state [3]. While the base specification enables powerful inspection and control capabilities, secure deployment requires restricting debug access through hardware lifecycle management and authentication mechanisms implemented at the platform level.

Secure boot and anti-rollback protection are typically implemented through a hardware root of trust anchored in immutable ROM and cryptographic signature verification. RISC-V-based secure platforms such as OpenTitan demonstrate how a minimal mask-ROM boot stage verifies subsequent firmware images using public-key cryptography before transferring control [40]. Version control and anti-rollback protections are enforced by storing monotonic counters or version metadata in non-volatile storage to prevent execution of outdated firmware images.

Device-unique key derivation and entropy generation are commonly integrated alongside secure boot mechanisms. RISC-V systems leverage hardware entropy sources defined in the entropy extension specification to seed cryptographic primitives and derive device-bound secrets [41]. Although physical unclonable functions (PUFs) are implementation-specific rather than ISA-mandated features, they are frequently integrated in RISC-V-based SoCs to support secure provisioning and attestation workflows.

Overall, secure debug and anti-tampering protections in RISC-V systems are achieved through standardized debug interfaces combined with platform-level authentication, lifecycle control, and hardware root-of-trust mechanisms rather than through dedicated ISA-level anti-tamper extensions. Table III summarizes key RISC-V cryptographic acceleration and hardware security mechanisms.

TABLE III
SUMMARY OF CRYPTOGRAPHIC MECHANISMS IN RISC-V

Category	Paper	Mechanism / Technique
Post-Quantum Crypto.	Banerjee et al. [34]	ECC accelerator with DTLS 1.3 on low-power RISC-V
	Fritzmann et al. [35]	Lattice-based accelerator with custom ISA instructions
	Wang et al. [36]	Multi-core FPGA for Kyber/ Dilithium; polynomial accelerators
	Liu et al. [37]	Embedded isogeny-based cryptography on low-power RISC-V
Cryptographic ISA Extensions	Zk*/ Zb*/ Zkr [38], [39]	AES, SHA-2/3, carryless multiply, bit manipulation, entropy source
Secure Debug /Anti-Tampering	RISC-V Debug Spec [3]	Standardized debug modules with controlled hart access
	OpenTitan [40]	Secure boot, anti-rollback, and hardware root-of-trust
	RISC-V Entropy [41]	Device-unique keys and secure entropy for provisioning and attestation

V. OPEN CHALLENGES AND FUTURE DIRECTIONS

Despite substantial progress in RISC-V adoption, several challenges remain in applying the architecture to critical and safety-sensitive domains.

A. Testability Challenges

Complex microarchitectures, including multi-level cache hierarchies, vector extensions, and custom accelerators, increase the test space exponentially. Achieving high functional coverage across cores, peripherals, and software stacks remains a significant challenge. Existing AI-driven and formal verification approaches provide partial solutions, but complete coverage guarantees for large-scale RISC-V SoCs have not yet been established.

System-level validation of RISC-V-based systems remains resource-intensive. The integration of heterogeneous cores, peripheral IP, and operating system stacks requires coordinated simulation, emulation, and silicon-based testing. Standardized methodologies for hybrid verification that can scale to industrial RISC-V designs are still under development.

Analog and mixed-signal testing is increasingly relevant as RISC-V cores are deployed alongside PLLs, ADCs, and power-management blocks. Effective test strategies must bridge conventional digital verification with functional validation of analog components, including their interactions with the processor and memory subsystems.

B. Safety Challenges

Ensuring temporal composability in multicore RISC-V systems remains challenging. Guaranteeing timing isolation across partitions sharing accelerators or memory controllers requires further study. Current approaches rely on static partitioning or conservative WCET analysis, limiting system flexibility.

Certification of AI accelerators integrated with RISC-V cores is an open problem, as safety standards like ISO

26262 lack guidance for learning-enabled components. Supply chain security also concerns hardware trojans and counterfeit IP, requiring integrated verification and provenance tracking throughout the RISC-V development lifecycle.

Deploying RISC-V across domains requires adapting safety mechanisms to meet specific standards. For automotive ISO 26262, ASIL D compliance demands diagnostic coverage for latent faults through lockstep execution or self-test mechanisms, and quantitative fault metrics demonstrated through fault injection campaigns like gem5-MARVEL [10].

Aerospace systems governed by DO-178C/DO-254 require DAL, a verification with structural coverage metrics, including modified condition/decision coverage. RISC-V's open RTL enables this analysis unavailable in proprietary cores. Deterministic timing favors in-order cores such as PULP RISCY [26] over speculative designs unless temporal isolation is rigorously proven.

C. Security Challenges

Transitioning legacy cryptography to post-quantum algorithms within RISC-V systems poses architectural and performance challenges. Maintaining compatibility with existing protocols while implementing lattice-based, hash-based, or code-based schemes requires careful hardware-software co-design. Side-channel resistance continues to demand attention. Attacks leveraging power analysis, electromagnetic emissions, or speculative execution require RISC-V cores to adopt both software and hardware mitigations, including constant-time execution and partitioned microarchitectural resources. Integrating emerging technologies such as quantum key distribution with RISC-V-based systems is largely unexplored. Realizing secure communication channels in hardware and software will necessitate novel co-design approaches that leverage the flexibility of RISC-V.

D. Future Research Directions

Research opportunities include extending RISC-V architectures to support safety-critical anomaly detection through neural-inspired or neuromorphic accelerators. Exploring modular verification frameworks and AI-assisted test generation could reduce the time and cost of certification for safety and security compliance.

Investigating quantum-resistant hardware primitives alongside algorithmic post-quantum cryptography may provide long-term security assurances. Secure provisioning, attestation, and lifecycle management mechanisms should be integrated into RISC-V platforms to support anti-rollback, key derivation, and device authentication.

Bio-inspired security strategies, such as adaptive and self-healing mechanisms, offer potential for resilient RISC-V architectures capable of responding to new attack patterns in deployed systems [42].

VI. Conclusion

The RISC-V ecosystem has evolved significantly in recent years, achieving industrial adoption for safety-critical and security-sensitive applications. This survey reviewed key contributions across testability, safety, and security, highlighting developments in multi-core verification, hardware-enforced isolation, post-quantum cryptography acceleration, and secure boot mechanisms. Emerging trends indicate a convergence of safety and security design, integration of AI for verification and runtime monitoring, and modular system design for scalable certification. Persistent challenges include complex test generation, temporal composability in multicore and heterogeneous systems, post-quantum migration, and lifecycle management for secure deployment. The openness, modularity, and extensibility of RISC-V provide a unique platform to address these challenges. Continued collaboration among academia, industry, and standards organizations will be critical to advance verification methodologies, co-designed safety-security solutions, and quantum-resistant architectures. RISC-V has the potential to become a foundational platform for trustworthy computing systems across automotive, aerospace, telecommunications, and edge computing domains, enabling rigorous safety and security guarantees over the next decade.

References

[1] C. Tain, S. Patil, and H. Al-Asaad, "Survey of verification of risc-v processors," *Journal of Electronic Testing*, vol. 41, no. 2, pp. 111–138, 2025. [Online]. Available: https://doi.org/10.1007/s10836-025-06169-3

[2] RISC-V Privileged Architecture Task Group, "The risc-v instruction set manual, volume ii: Privileged architecture," RISC-V International Specification, 2021, ratified specification, Version 20211203. [Online]. Available: https://docs.riscv.org/reference/isa/priv/priv-index.html

[3] RISC-V Debug Task Group, "Risc-v external debug support version 0.13.2," *RISC-V International Specification*, 2019. [Online]. Available: https://github.com/riscv/riscv-debug-spec/releases

[4] M. Boubakri and B. Zouari, "A survey of risc-v secure enclaves and trusted execution environments," *Electronics*, vol. 14, no. 21, p. 4171, 2025. [Online]. Available: https://doi.org/10.3390/electronics14214171

[5] F. Corno, G. Cumani, M. S. Reorda, and G. Squillero, "Automatic test program generation for pipelined processors," in *Proceedings of the 2003 ACM symposium on Applied computing*, 2003, pp. 736–740.

[6] T. Faller, N. I. Deligiannis, M. Schwörer, M. S. Reorda, and B. Becker, "Constraint-based automatic sbst generation for risc-v processor families," in *2023 IEEE European Test Symposium (ETS)*. IEEE, 2023.

[7] Y. Lu, C. Bai, Y. Zhao, Z. Zheng, Y. Lyu, M. Liu, and B. Yu, "Learning to update test sequences for coverage-guided verification," *ACM Transactions on Design Automation of Electronic Systems*, 2025.

[8] J. Chen, L. Yan, S. Wang, and W. Zheng, "Deep reinforcement learning-based automatic test case generation for hardware verification," *Journal of Artificial Intelligence General science (JAIGS)*, 2024.

[9] J. Hur, S. Song, D. Kwon, E. Baek, J. Kim, and B. Lee, "Difuzzrtl: Differential fuzz testing to find cpu bugs," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1286–1303.

[10] O. Chatzopoulos, G. Papadimitriou, V. Karakostas, and D. Gizopoulos, "Gem5-marvel: Microarchitecture-level resilience analysis of heterogeneous soc architectures," in *IEEE HPCA*, 2024.

[11] S. Eldridge, A. Buyuktosunoglu, and P. Bose, "Chiffre: A configurable hardware fault injection framework for risc-v systems." CARRV, 2018.

[12] Synopsys, "Sting: Risc-v verification tool for portable, constrained random and directed test generation," 2025. [Online]. Available: https://www.synopsys.com/verification/sting.html

[13] A. Joannou, P. Rugg, J. Woodruff, F. A. Fuchs, M. van der Maas, M. Naylor, M. Roe, Watson, and others., "Randomized testing of risc-v cpus using direct instruction injection," 2024.

[14] Breker, "Risc-v socready systemvip: Verification suite for risc-v socs," 2025, https://brekersystems.com/products/riscv-socready/.

[15] Tenstorrent, "Riescue: Risc-v directed test framework and compliance suite," GitHub repository, 2025. [Online]. Available: https://github.com/tenstorrent/riescue

[16] J. Abella *et al.*, "Security, reliability and test aspects of the risc-v ecosystem," *Technical report / RISC-V verification survey*, 2021. [Online]. Available: https://upcommons.upc.edu/bitstreams/f1a4cff8-bde2-4487-aad4-bfd26bdd9419/download

[17] R. Qiu and Y. Liu, "An integrated uvm-tlm co-simulation framework for risc-v functional verification and performance evaluation," *arXiv preprint*, 2025, functional co-simulation framework combining UVM and TLM for verification. [Online]. Available: https://arxiv.org/abs/2505.10145

[18] A. Galimberti, M. Vitali, S. Vittoria, and D. Zoni, "Functional iss-driven verification of superscalar risc-v processors," *arXiv preprint arXiv:2407.21192*, 2024, methodology interfacing RTL with an instruction set simulator for verification. [Online]. Available: https://arxiv.org/abs/2407.21192

[19] H. M. Lai *et al.*, "Risc-v vectorization coverage for hpc: A tsvc-based instruction coverage analysis," in *Proceedings of the ACM Conference on Computing Frontiers*, 2025.

[20] L. Weingarten, K. Datta, and R. Drechsler, "Polynomial formal verification of a risc-v processor using bdd-based methods," *IEEE Transactions on Nanotechnology*, vol. 24, pp. 140–151, 2025.

[21] N. Bruns, V. Herdt, and R. Drechsler, "Processor verification using symbolic execution: A risc-v case-study," in *2023 Design, Automation & Test in Europe Conference (DATE)*, 2023, pp. 1–6.

[22] K. Qin, X. Guo, M. Schulz, and C. Trinitis, "Feriver: An fpga-assisted emulated framework for rtl verification of risc-v processors," *arXiv preprint*, 2025, fPGA-assisted verification combining hardware emulation and software reference models. [Online]. Available: https://arxiv.org/abs/2504.05284

[23] S. E. . riscv-formal community, "riscv-formal: Risc-v formal verification framework," https://github.com/SymbioticEDA/riscv-formal, 2025, formal verification interface (RVFI) used for property checking against RTL.

[24] M. Platzer and P. Puschner, "Vicuna: A timing-predictable risc-v vector coprocessor for scalable parallel computation," *Proceedings of the 33rd Euromicro Conference on Real-Time Systems (ECRTS)*, 2021.

[25] C. Celio *et al.*, "The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor," 2015.

[26] PULP Platform, "Ri5cy user manual," 2019. [Online]. Available: https://pulp-platform.org/docs/ri5cy_user_manual.pdf

[27] L. Gerlach, D. Weber, R. Zhang, and M. Schwarz, "A security risc: microarchitectural attacks on hardware risc-v cpus," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2321–2338.

[28] A. Waterman and K. Asanović, "The risc-v instruction set manual, volume i: Unprivileged isa," *RISC-V International*, 2019.

[29] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio *et al.*, "The rocket chip generator," *IEEE Micro*, vol. 36, no. 2, pp. 6–17, 2016.

[30] V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," *USENIX Security Symposium*, pp. 857–874, 2016. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[31] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys)*, pp. 1–16, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387532

[32] T. Bourgeat, I. Lebedev, A. Wright, S. Zhang, Arvind, and S. Devadas, "Mi6: Secure enclaves in a speculative out-of-order processor," *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 42–56, 2019. [Online]. Available: https://doi.org/10.1145/3352460.3358310

[33] P. Rugg, J. Woodruff, A. Joannou, and S. W. Moore, "A suite of processors to explore cheri-risc-v microarchitecture," *Cambridge Repository Technical Report*, 2023. [Online]. Available: https://doi.org/10.17863/CAM.110446

[34] U. Banerjee, A. Pathak, and A. Chandrakasan, "An Energy-Efficient Reconfigurable DTLS Cryptographic Engine for Securing Internet-of-Things Applications," in *2019 IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 42–44.

[35] T. Fritzmann, G. Sigl, and J. Sepúlveda, "Risq-v: Tightly coupled risc-v accelerators for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 239–280, 2020. [Online]. Available: https://doi.org/10.13154/tches.v2020.i4.239-280

[36] T. Wang, C. Zhang, X. Zhang, D. Gu, and P. Cao, "Optimized hardware–software co-design for kyber and dilithium on risc-v soc fpga," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, pp. 99–135, 2024. [Online]. Available: https://doi.org/10.46586/tches.v2024.i3.99-135

[37] R. Elkhatib, B. Koziel, R. Azarderakhsh, and M. M. Kermani, "Accelerated risc-v for post-quantum sike," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2490–2501, 2022.

[38] RISC-V Cryptography Extensions Task Group, "Risc-v cryptography extensions, volume i: Scalar & entropy source instructions," *RISC-V International Specification*, 2021. [Online]. Available: https://docs.riscv.org/reference/isa/unpriv/scalar-crypto.html

[39] RISC-V Bit Manipulation Task Group, "Risc-v bit manipulation isa extensions (zba, zbb, zbc, zbs, etc.)," *RISC-V International Specification*, 2021. [Online]. Available: https://riscv.org/technical/specifications/

[40] lowRISC CIC and OpenTitan Contributors, "Opentitan: Open source silicon root of trust," Online open source project, 2019–2026, available at https://opentitan.org/ (accessed 2026). [Online]. Available: https://opentitan.org/

[41] RISC-V Cryptography Extensions Task Group, "The risc-v cryptography extensions, volume i: Scalar & entropy source instructions," *RISC-V International Specification*, 2023. [Online]. Available: https://docs.riscv.org/reference/isa/unpriv/scalar-crypto.html

[42] A. Bala, A. Bahnasse, B. El Bhiri, M. Zegrari, and P.-M. Tardif, "Immunity-inspired approaches to cybersecurity: A review," *Procedia Comput. Sci.*, p. 274–281, Jan. 2025. [Online]. Available: https://doi.org/10.1016/j.procs.2025.03.037