The American Journal of Applied Sciences
ISSN 2689-0992 | Open Access

TYPE Original Research
PAGE NO. 17-22

A Comprehensive Study on Fault-Tolerant RISC-V Architectures and Safety-Aware Cyber-Physical Systems for Reliable Autonomous and Space Computing

OPEN ACCESS

SUBMITTED 21 August 2024
ACCEPTED 28 August 2024
PUBLISHED 22 September 2024
VOLUME Vol.06 Issue 09 2024

Dr. Adrian M. Keller
Department of Computer Engineering Helios Institute of Technology, Zurich, Switzerland

Abstract: This article synthesizes contemporary theory and practice in fault-tolerant microarchitectures with a focus on RISC-V processor cores, dual-core lockstep designs, software-implemented redundancy, and latency-aware provisioning in prediction-serving pipelines. Drawing exclusively on the provided literature, it constructs an integrative perspective that links radiation- and transient-fault mitigation strategies from aerospace and satellite applications to safety-critical domains such as automotive zonal controllers and edge prediction services. The structured abstract presents: (a) background emphasizing the confluence of reliability, performance, and cost pressures; (b) methods describing comparative, design-oriented, and analytical approaches distilled from the references; (c) key thematic results synthesizing hardware-and-software co-design patterns, statistical injection frameworks, thread protection techniques, and system-level provisioning trade-offs; and (d) conclusions outlining research directions, practical recommendations, and limitations. This contribution does not present new empirical measurements but offers a deep theoretical elaboration that clarifies design choices, exposes nuanced trade-offs, and proposes a unified framework for future experimental validation and standardization.

Keywords: Fault tolerance, RISC-V, lockstep architectures, radiation effects, safety co-design, prediction-serving pipelines

Introduction

The contemporary landscape of embedded and distributed computing is characterized by an accelerating demand for computational capability at the edge, strong constraints on cost and power, and

pervasive expectations of reliability in hostile environments. These pressures converge across domains as diverse as spaceborne systems—where ionizing radiation and single-event upsets (SEUs) present existential threats to computation—and automotive zonal controllers—where safety-critical operation demands predictable, certified behavior under fault conditions (Fayyaz & Vladimirova, 2014; Abdul Karim, 2023). At the same time, novel service models such as prediction-serving pipelines deliver low-latency, high-throughput inference close to users, and they rely on careful provisioning and scaling to balance performance objectives against resource usage (Crankshaw et al., 2020). These research vectors share a common problem statement: how to design architectures and systems that robustly tolerate faults—transient, permanent, and systemic—without prohibitive increases in cost, energy, or latency.

Extant literature provides a multiplicity of partial solutions. Hardware-level redundancy and lockstep execution approaches promise deterministic error masking and straightforward certification paths when implemented on diverse platforms (Abdul Karim, 2023; Blasi et al., 2019). RISC-V, an open ISA, has become a fertile ground for fault-tolerant microcontroller and processor research because it enables architectural experimentation with minimal licensing friction (Li et al., 2022; Santos et al., 2020). Complementary to hardware redundancy, software-implemented approaches offer flexibility, lower upfront cost, and faster iteration cycles; these approaches include checkpoint/recovery, thread-level monitoring, and semantic recovery schemes (Goloubeva et al., 2006; Gomez-Cornejo et al., 2013). The safety and security co-design movement emphasizes that achieving dependable systems requires semantics-aware architectural patterns and tooling that bring safety constraints into the architectural design process (Dantas & Nigam, 2023).

However, several gaps persist. First, the literature often treats radiation-hardened designs for space and low-latency edge services as separate domains, yielding fragmented design heuristics that are not readily reconcilable for multi-domain platforms (Ginosar, 2012; Foudeh et al., 2021). Second, statistical methods for fault-injection and confidence estimation—necessary for predicting field reliability without exhaustive testing—remain underutilized or inconsistent across studies, complicating cross-comparison and engineering decision-making (Leveugle et al., 2009). Third, trade-offs between latency-aware provisioning in prediction-serving systems and reliability provisioning in fault-tolerant hardware are rarely integrated, even though many modern applications (e.g., autonomous drones performing power-line inspections) combine both real-time inference and radiation or environmental exposure (Foudeh et al., 2021; Crankshaw et al., 2020). This article synthesizes the provided references to close these gaps by constructing an integrated, theory-rich framework for fault-tolerant architectural design that explicitly accounts for domain-specific constraints, statistical validation, and operational provisioning.

Methodology

The methodological approach taken in this synthesis is analytical and comparative rather than experimental. It consists of three complementary activities: (1) extraction and classification of design primitives from the reference set, (2) construction of an integrative design space that maps primitives to system-level goals, and (3) theoretical elaboration of trade-offs, supported by citations to the primary sources.

Extraction and classification. Each reference was examined for architectural mechanisms, validation techniques, and system-level practices relevant to fault tolerance. The mechanisms were categorized into hardware redundancy (e.g., lockstep cores, dual-core lockstep, thread-controlled watchdogs), software redundancy (e.g., software-implemented hardware fault tolerance, checkpointing, thread-level redundancy), validation methodologies (e.g., statistical fault injection, neutron radiation testing), and system provisioning strategies (e.g., latency-aware scaling for prediction-serving). The classification emphasizes cross-cutting properties: determinism, detection latency, recovery latency, resource overhead (area, power, and cost), implementation complexity, and suitability for certification.

Integrative design space construction. Building upon the classification, design primitives were placed within a conceptual design space spanned by axes representing (A) fault model severity (transient vs. permanent), (B) temporal constraints (soft real-time vs. hard real-time), (C) resource constraints (area/power/cost sensitivity), and (D) verification maturity (from simulation to radiation testing). The mapping associates each primitive with regions of the design space where it offers favorable trade-offs. For instance, full lockstep hardware maps to high-severity fault models and hard real-time constraints but to high resource overhead; software-implemented redundancy maps to lower-cost regions where recovery latency can be tolerated (Blasi et al., 2019; Goloubeva et al., 2006).

Theoretical elaboration and synthesis. For each region and primitive, the article develops detailed arguments

about mechanism operation, interactions with adjacent primitives, cumulative overhead, and validation expectations. Theoretical discussion draws on statistical injection methods for quantifying fault coverage and confidence intervals (Leveugle et al., 2009), empirical radiation testing to anchor expectations under extreme conditions (Wilson et al., 2019), and modern provisioning approaches for low-latency pipelines (Crankshaw et al., 2020). Where literature makes explicit empirical claims (e.g., neutron testing results), those claims are restated and integrated; where literature proposes architectures (e.g., Duckcore, RISC-V fault-tolerant microcontrollers), the paper extrapolates design consequences and possible combinations.

Limitations of methodology. Because this work synthesizes rather than empirically measures, its contributions are conceptual and prescriptive. The emphasis on theoretical elaboration is intentional to produce a rigorous roadmap that experimentalists and system designers can use to instantiate and test architectures in domain-specific contexts (Gomaa et al., 2003; Santos et al., 2020).

Results

This section distills the principal outcomes of the analytical process: a taxonomy of resilient primitives, an articulated mapping between domain demands and recommended patterns, and a set of theoretical performance-reliability trade-offs that inform practical design choices.

Taxonomy of resilient primitives. The literature suggests a rich palette of primitives:

Full lockstep hardware redundancy: Two or more cores execute the same instruction stream synchronously, with built-in comparators to detect divergence. This approach offers deterministic failure detection and simple fail-stop semantics that assist certification processes (Abdul Karim, 2023; Blasi et al., 2019).

Partial/thread-level hardware protection with watchdogs: This primitive combines hardware thread isolation and watchdog timers to detect stuck threads or livelock conditions, allowing selective protection that reduces area/power overhead compared with full-core duplication (Blasi et al., 2019).

Software-implemented hardware fault tolerance (SIHFT): Techniques delivered in software to emulate hardware redundancy through instruction-level duplication, N-version programming, or checkpoint/restart. SIHFT shifts cost from silicon to code, permitting deployment on commodity devices while accepting variable detection and recovery latencies (Goloubeva et al., 2006).

Radiation-tested soft cores and hardened microcontrollers: Empirical approaches emphasize that soft cores implemented on SRAM-based FPGAs require radiation testing (e.g., neutron testing) to understand field failure rates; results guide architectural hardening or operational mitigation (Wilson et al., 2019; Santos et al., 2020).

Statistical fault injection: A validation primitive to estimate fault coverage and confidence using probabilistic sampling of fault modes; it provides a structured way to reason about expected failure rates without exhaustive real-world testing (Leveugle et al., 2009)

System-level latency-aware provisioning: For services such as online prediction-serving, dynamic provisioning minimizes latency while considering resource budgets; it is orthogonal but complementary to resilience measures, and it's essential when inference is part of a safety-critical pipeline (Crankshaw et al., 2020).

Mapping primitives to domains. By mapping the taxonomy to domain-specific demands, the analysis produces the following recommendations:

Space systems and satellites: Favor hardware-level redundancy (e.g., lockstep or dual-core lockstep) combined with radiation-tested components. The severe fault model (ionizing radiation, single-event latchups) and certification needs argue for deterministic, low-detection-latency mechanisms; software techniques can be layered for additional coverage but are insufficient as sole mitigation (Fayyaz & Vladimirova, 2014; Wilson et al., 2019; Ginosar, 2012).

Automotive zonal controllers and safety-critical vehicles: Dual-core lockstep and hardware thread protection are compelling because automotive systems require predictable behavior and need to meet safety standards; however, cost constraints favor partial protection where acceptable and SIHFT in adjacently non-critical domains (Abdul Karim, 2023; Blasi et al., 2019).

Unmanned aerial vehicles and edge inference platforms: These combine environmental exposure with strict latency constraints for perception and control. A hybrid approach is recommended: hardware redundancy for critical control loops, software-implemented validation for inference tasks, and latency-aware provisioning for prediction-serving pipelines to guarantee the required response times (Foudeh et al., 2021; Crankshaw et al., 2020).

Quantitative reasoning and validation practices. The synthesis emphasizes that confidence in resilience claims depends on rigorous validation. Statistical injection methods provide a path to estimate fault rates and characterize error distributions; these methods complement radiation testing and operational field data (Leveugle et al., 2009; Wilson et al., 2019). For instance, statistical fault injection can identify weak coverage regimes that radiation testing should prioritize, thereby creating an efficient validation schedule that maximizes information per test dollar.

Interaction effects and systemic risk. A salient result is the recognition that resilience mechanisms interact nonlinearly. For example, redundant cores that share the same supply domain can experience correlated failures under a single-event transient that affects power rails, reducing the assumed independence advantage of duplication. Similarly, software redundancy that relies on the same compiler and runtime (e.g., GCC toolchains) can suffer correlated faults due to shared compiler bugs or runtime library vulnerabilities (Hagen, 2006). Thus, achieving true diversity—across hardware, software, and operational modes—is crucial for high-assurance systems (Gomaa et al., 2003; Goloubeva et al., 2006).

Design overheads and amortized cost. While hardware redundancy can be labeled "expensive" in area and power, the analysis shows that amortized cost against mission-critical failure consequences often favors hardware-level investment in domains with high failure impact (e.g., space systems, medical devices). Conversely, for consumer-edge devices or soft real-time prediction-serving services, software-based mitigations and intelligent provisioning often yield a better cost-benefit ratio (Santos et al., 2020; Crankshaw et al., 2020).

Discussion

This section interprets the results in depth, explores counter-arguments, identifies limitations, and sketches future research directions that emerge from synthesizing the literature.

Interpreting the synthesis: converging evidence for hybrid strategies. A dominant theme in the references is that no single mitigation strategy universally dominates; instead, hybrid solutions that combine hardware redundancy, software monitoring, and system-level provisioning produce robust, cost-aware outcomes (Blasi et al., 2019; Goloubeva et al., 2006; Crankshaw et al., 2020). The theoretical foundations of hybridization rest on diversity and layered detection: hardware offers low-latency detection and simple semantics for safety certification, while software adds semantic checks, graceful degradation, and field-upgradable logic. System-level provisioning, particularly for latency-sensitive inference pipelines, must be cognizant of redundancy-induced resource needs to avoid undermining latency guarantees.

Trade-offs: latency vs. reliability vs. cost. An inescapable trade-off triangle governs design decisions. Full hardware redundancy moves a design toward high reliability and deterministic behavior at the cost of latency-insensitive resource consumption (e.g., additional cores may increase power draw but do not necessarily increase critical path latency). Software redundancy can be more lightweight in hardware cost but often increases detection and recovery latency, which may be unacceptable for hard real-time control loops (Gomaa et al., 2003; Blasi et al., 2019). System-level provisioning can mitigate some latency costs by allocating more compute capacity dynamically, but this inflates operating costs and complicates scheduling in constrained environments (Crankshaw et al., 2020).

Counter-arguments and nuanced positions. Some proponents of software-implemented techniques argue that advances in compiler-assisted duplication and lightweight transactional memory make software approaches sufficiently robust even for aggressive environments (Goloubeva et al., 2006). The counterpoint—grounded in empirical radiation testing (Wilson et al., 2019)—is that certain classes of faults (e.g., single-event functional interrupts, configuration memory upsets in SRAM-based FPGAs) can undermine software-layer assumptions and require lower-level mitigation. Therefore, software approaches are necessary but not sufficient in several high-risk domains.

Statistical validation: strengths and pitfalls. Statistical fault injection provides a scalable method for estimating failure probabilities, but its effectiveness depends critically on the fidelity of the injection model and the representativeness of the sample space (Leveugle et al., 2009). Poorly parameterized injections can produce optimistic assurances; conversely, overly conservative models may force unnecessary design complexity. Integrating statistical injection with targeted physical testing (e.g., neutron radiation tests for FPGA soft cores) yields balanced and economically defensible validation strategies (Wilson et al., 2019). The literature suggests using statistical injection early in the design cycle to guide focus and physical testing later to validate the most critical failure modes (Leveugle et al., 2009).

Certification and standards implications. The adoption of RISC-V in fault-tolerant research underscores the

need for ecosystem-level standards that reconcile openness with safety assurance (Li et al., 2022; Blasi et al., 2019). Automotive standards require traceability and deterministic behavior, which favor hardware redundancy and watchdog-controlled isolation; space systems mandate radiation hardening and rigorous testing (Abdul Karim, 2023; Fayyaz & Vladimirova, 2014). A key implication is that design choices should be sensitive to certification paths: adopting software-only mitigations may complicate certification and liability assignments in regulated industries.

Operational considerations: deployment, maintenance, and field updates. One advantage of software-implemented approaches is field upgradability, enabling iterative hardening and bug fixes post-deployment; hardware redundancy lacks this flexibility. Consequently, a pragmatic approach is to combine hardened hardware for the most critical functions with flexible software for peripheral tasks and for deploying patches as new vulnerabilities or failure modes appear (Santos et al., 2020; Goloubeva et al., 2006). This pattern is particularly relevant for long-lived space and industrial assets where post-launch updates are either impossible or extremely costly.

Limitations of the synthesis. While the integrative framework draws robustly from the provided literature, its limitations include: (1) absence of new experimental validation within this article, (2) potential biases inherited from the cited works' experimental methodologies, and (3) the reliance on the set of references supplied, which—while diverse—does not exhaust the full global literature on the topic. These limitations underscore the need for empirical follow-ups, ideally cross-domain testbeds that evaluate hybrid strategies under representative mission profiles (Gomez-Cornejo et al., 2013; Santos et al., 2020).

Future research directions. Several promising avenues arise:

Cross-domain testbeds that combine radiation testing, automotive environmental stressors, and real-time inference workloads to evaluate hybrid strategies holistically (Wilson et al., 2019; Foudeh et al., 2021).

Improved statistical fault-injection frameworks that integrate correlated-failure models to capture supply-domain and software-toolchain correlations (Leveugle et al., 2009; Hagen, 2006).

Co-design methodologies that formalize safety-security pattern libraries and semantically rich architectural templates to accelerate certified deployments (Dantas & Nigam, 2023)

Research into economical diversity: methods to achieve uncorrelated redundancy without duplicating expensive silicon, such as mixed-vendor soft cores, compiler-level diversity, and heterogenous isolation schemes (Gomaa et al., 2003; Goloubeva et al., 2006).

Conclusion

This synthesis has articulated a unified perspective on fault-tolerant architecture design by systematically drawing on the provided literature. The central thesis is that resilient systems are best realized through hybrid approaches that combine hardware redundancy for deterministic, low-latency detection and fail-stop semantics; software-implemented techniques for flexibility, field updateability, and semantic checks; statistical injection and targeted physical testing for validation; and system-level provisioning to ensure latency and resource constraints are met in operational settings. Domain-specific constraints—whether radiation exposure in space, safety certification in automotive domains, or stringent latency in edge inference—guide the weighting of these components within solutions.

Key practical takeaways for designers include: (1) prioritize hardware redundancy for mission-critical control functions where certification and determinism are essential (Abdul Karim, 2023; Blasi et al., 2019); (2) use software techniques to augment and extend detection capabilities while enabling maintainability (Goloubeva et al., 2006); (3) employ statistical injection early and physical testing for the most impactful failure modes (Leveugle et al., 2009; Wilson et al., 2019); and (4) integrate latency-aware provisioning when prediction-serving operations are involved to avoid inadvertent performance reliability trade-offs (Crankshaw et al., 2020).

Ultimately, the pathway to robust, cost-effective resilient architectures is neither purely hardware nor purely software—it is a carefully engineered confluence of both, validated by rigorous statistical and physical testing regimes, and tailored to the demands of each application domain. The conceptual framework and detailed trade-off elaborations presented here aim to guide future empirical work and standardization efforts that will bring these hybrid strategies into production systems with confidence.

References

1. Crankshaw, D., Sela, G. E., Mo, X., Zumar, C., Stoica, I., Gonzalez, J., & Tumanov, A. (2020, October). InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In Proceedings of the

11th ACM Symposium on Cloud Computing (pp. 477-491).
2. Dantas, Y. G., & Nigam, V. (2023). Automating safety and security co-design through semantically rich architecture patterns. ACM Transactions on Cyber-Physical Systems, 7(1), 1-28.
3. Foudeh, H. A., Luk, P. C. K., & Whidborne, J. F. (2021). An advanced unmanned aerial vehicle (UAV) approach via learning-based control for overhead power line monitoring: A comprehensive review. IEEE Access, 9, 130410-130433.
4. Li, J., et al. (2022). Duckcore: A fault-tolerant processor core architecture based on the RISC-V ISA. Electronics, 11(1).
5. Blasi, L., et al. (2019). A RISC-V fault-tolerant microcontroller core architecture based on a hardware thread full/partial protection and a thread-controlled watchdog timer. In APPLEPIES, 2019, pp. 505–511.
6. Wilson, A. E., et al. (2019). Neutron radiation testing of fault tolerant RISC-V soft processor on Xilinx SRAM-based FPGAs. In IEEE SCC, 2019, pp. 25–32.
7. Santos, D. A., et al. (2020). A low-cost fault-tolerant RISC-V processor for space systems. In DTIS, 2020, pp. 1–5.
8. Leveugle, R., et al. (2009). Statistical fault injection: Quantified error and confidence. In IEEE/ACM DATE, 2009, pp. 502–506.
9. Fayyaz, M., & Vladimirova, T. (2014). Fault-tolerant distributed approach to satellite on-board computer design. In 2014 IEEE Aerospace Conference. ISSN 1095-323X.
10. Ginosar, R. (2012). Survey of processors for space. In DASIA, 2012, pp. 1–5.
11. Goloubeva, O., et al. (2006). Software-implemented hardware fault tolerance. Springer Science & Business Media, 2006.
12. Abdul Salam Abdul Karim. (2023). Fault-Tolerant Dual-Core Lockstep Architecture for Automotive Zonal Controllers Using NXP S32G Processors. International Journal of Intelligent Systems and Applications in Engineering, 11(11s), 877–885. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/7749
13. Gomaa, M. A., et al. (2003). Transient-fault recovery for chip multiprocessors. IEEE Micro, 23(6), 76–83.
14. Gomez-Cornejo, J., et al. (2013). Fast context reloading lockstep approach for SEUs mitigation in a FPGA soft core processor. In IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, pp. 2261–2266.
15. Hagen, W. von. (2006). The Definitive Guide to GCC. 2nd ed. Apress.