RESEARCH ARTICLE ECEJOURNALS.IN

A Review of Fault-Tolerant Reconfigurable Architectures for Autonomous Systems Based on Redundant Logic Mapping

Belal Batiha[1]*, Amina El-Fahmy[2]

[1]Mathematics Department, Faculty of Science and Information Technology, Jadara University, Jordan

[2]School of Computer Science, Universidad Nacional de Colombia, Colombia.

Keywords:
Fault-Tolerant Computing,
Reconfigurable Architecture,
Redundant Logic Mapping,
Autonomous Systems,
Dynamic Partial Reconfiguration,
FPGA-Based Reliability

Author's Email:
b.bateha@jadara.edu.jo

https://doi.org/10.31838/ESA/03.02.07

Received : 09.01.2026

Revised : 24.02.2026

Accepted : 19.03.2026

Abstract

Mission-critical applications (aerospace, automotive, and defense) have autonomous systems that must have very high reliability to allow continued operation in varied environmental structures and fault conditions. The medium of flexibility and adaptability of the computer hardware is represented by reconfigurable computing platforms and, especially, by Field-Programmable Gate Arrays (FPGAs) and Coarse-Grained Reconfigurable Architectures (CGRAs). The review will provide an orderly research into the fault-tolerant reconfigurable architecture with an interest in redundant logic mapping which encompasses spatial, temporal, and hybrid redundancy model. We study important analysis metrics including Mean Time to Failure (MTTF), fault coverage, resources overhead and reconfiguration latency. Recent literature shows comparative results that hybrids are able to provide the MTTF with up to 3.5x scaling without a drastically high resource overhead (~2.1x LUTs, +25% power). The relevance also points to the real use in self-governing frameworks e.g. UAV, self-governing automobiles, and smart robotics, where fault cloaking, configuration abrasive, and flawless cycles are fundamental. Lastly, it presents open research issues in scalable fault diagnosis, AI-based reconfiguration as well as cross layer fault resilience, and states a guideline towards the design of next generation self-sustainable systems comprising of both economic and fault tolerant systems.

How to cite this article: Batiha B, El-Fahmy A (2026). A Review of Fault-Tolerant Reconfigurable Architectures for Autonomous Systems Based on Redundant Logic Mapping. SCCTS Journal of Embedded Systems Design and Applications, Vol. 3, No. 2, 2026, 59-68

Introduction

The fast development of safety and mission-critical areas (aerospace, defense, automotive and industrial automation) has led to an evolution in system design requirements to a paradigm shift. Such systems are required to run in uncertain and most of the time, harsh environmental conditions where computation failures may have disastrous effects. Consequently, fault tolerance has also become the underlying concept of designing reliable autonomous systems. Sources of faults in hardware may be cosmic radiation, aging-induced wear, thermal stress, defects introduced during manufacturing, and electromagnetic interference.

These errors are usually categorised under three groups namely; transient faults (soft errors), permanent faults which are attributed to hardware that undergo wear-out, or fabrication faults, and intermittent faults, which are unpredictable and occur due to marginal operating conditions. The long-standing fault-tolerant techniques relied on either Application-Specific Integrated Circuit (ASIC) based or static redundancy techniques that tend to be fixed, resource-intensive and simply inapplicable to the dynamic operations of contemporary autonomous systems. Conversely, Reconfigurable Computing (RC) especially with Field Programmable Gate Arrays (FPGAs) and Coarse-Grained Reconfigurable Architectures (CGRAs) has been seen as an opportune second. These platforms support dynamic system reconfigurability in real-time such that configuration flexibility occurs on the fly and allows fault recovery in a fault-tolerant manner without impacts on mission continuity. The ability has made RC an instrument of arsenal in creating resilient and adaptable computing systems to suit the needs of the autonomous systems.

Triple Modular Redundancy (TMR), error-correcting codes (ECC) and checkpointing are common fault-tolerant computing techniques discussed in many studies. Although these techniques have so far been useful in increasing the reliability in systems, they are usually accompanied by huge trade-offs being associated with high resource and power overhead, low runtime flexibility to unpredictable fault situations as well as the inability to scale up with the rising complexity of autonomous tasks. In addition, most of the current solutions do not connect well to intelligent reconfiguration policies which are capable of reacting to the real-time operations data. To counter such constraints, current work has focussed on exploiting redundant logic mapping in reconfigurable fabrics, where logic primitive are replicated or even triplicated together with fault-detection and fault-recovery built in to the primitive. Even though the developments provide interesting future directions, none of them have carried out a comprehensive survey to systematically categorize those techniques, evaluate their viability in designing autonomous systems, and outline the remaining open problems. This is especially so in the understudied fields of dynamic partial reconfiguration (DPR), fault prediction and management using AI, as well as the fabrication of hybrid redundancy models in a fluidity of fault contingency against efficiency.

With the growing sophistication in autonomous systems and the necessity of having unsupportable and continuous operation of these systems, development of in-depth knowledge of the application of redundant logic mapping techniques to increase the fault tolerance of the reconfigurable architectures is very strong motivation. This is especially important in embedded systems involved in autonomous systems, in which power, latency, and compute resources are highly restricted and require effective and sustainable mitigation strategies to faults. Restricted to correctly used, redundant logic mapping offers adaptive fault resilience without performance or power consumption penalty. An overview of such mechanisms will help system designers to understand to make intelligent trade-offs among performance, fault coverage, and resource usage; to select redundancy schemes that are suited to application-specific levels of criticality; and to use both run-time reconfiguration and adaptive redundancy as a means of achieving self-healing and autonomy robust behavior.

The main purpose of this review is to Organize and systematically categorize and review existing fault tolerant reconfigurable computing techniques focusing on redundant logic mapping. It seeks to compare the spatial, temporal and hybrid redundancy models based on fault coverage, resource overhead and reconfiguration latency. The research also attempts to look at fault detection, recovery strategies that can be applicable to runtime reconfigurable systems, sharing error detection circuitry, fault prediction methods and dynamic partial reconfiguration (DPR). Besides this, this review also delves into the practical applications of these architectures in real life autonomous systems; e.g. unmanned aerial vehicles (UAVs) and autonomous ground vehicles and various robotic platforms. Last, it specifies key issues and unsolved issues and suggests future work on how to achieve intelligent, efficient and scalable fault-tolerant reconfigurable architectures of next-generation autonomous systems.

In short, the issue tackled in this review paper was a burning need at the subject matter of fault-tolerant autonomous computing since it offer in-depth study of redundant logic mapping techniques in reconfigurable systems. It will fill the spaces between the traditional fault mitigation approaches and present-day reconfigurable approaches providing insights that are critical to developing the next generation autonomous platform with resilience. This paper should thus be considered as a reference point to any future researcher and practitioner working on the development of strong and smart computing systems in mission-critical autonomy through an integrative and comparative analysis.

Fundamentals of Fault-Tolerant Reconfigurable Architectures

Reconfigurable Computing Overview

Reconfigurable computing has emerged as a significant design framework of contemporary embedded and autonomous systems because the approach allows the dynamic adaptation of hardware functionality to meet the changing requirements of operations. Field-Programmable Gate Arrays (FPGAs) and Coarse-Grained Reconfigurable Architectures (CGRAs) are the most notable among platforms that are reconfigurable. FPGAs are distinguished by having a fine-grained logic with programmable interaction and real-time adaptability that enables application of parallelized fault tolerance logic. More to the point, FPGAs also feature Dynamic Partial Reconfiguration (DPR), i.e. the possibility to update certain areas of the chip on the fly without causing the complete halt of the system, which comes in very handy in fault mitigation scenarios.[1]

CGRAs, in their turn, provide some flexibility and one-time parametrization against guarantees of computational efficiency: coarse-grained functional units and parameterizable routing structures.[2] The platforms allow high-throughput programs, and they provide better energy efficiency which is a decisive factor in regard to mobile and edge-based autonomous systems. In addition, as systems provenance are progressively becoming implemented into sustainable VLSI systems, failures as well as fault-tolerant processor designs are primarily fundamental to long term introduction into intelligent infrastructure.[9, 11]

The relevance of reconfigurable computing to large-scale modeling and simulation has also recently been observed to be the fault resilience and the performance scalability.[13] Segregated, the resulting architectures form an interesting path towards making running execution fault tolerant in mission-critical autonomous systems.

Fault Models in Autonomous Systems

Autonomous systems have to run in highly dynamic, frequently hostile environments where hardware is likely to find itself in a high range of fault conditions, which in turn may result in a significant decrease in the reliability of systems. Perhaps the most prevalent of this type of transient fault is the Single Event Upset (SEU), which occurs when charged particles (e.g., cosmic rays) interact with the silicon, which flips represented bits of stored information in memory or logic cells. SEUs, in aerospace and at high altitude have the potential to be more critical since there is increased radiation exposure.[3]

Along with the transient faults, long term performance degradation of the transistors presents itself due to the presence of aging-related degradation mechanisms i.e. Negative bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI) which both tend to slowly cause permanent faults.[4] Such faults build up over a period and are especially bad in cases where the systems are supposed to work over the years without any sort of maintenance.

Moreover, self-driving platforms, more so in automotive or industrial contexts, are subjected to thermal strains, electromagnetic interferences (EMI). The factors present sporadic faults since it brings about timing inputs/ violations or signal integrity problems, and therefore real-time systems are prone to functional failure.[5,10] Accordingly, mitigation strategies adopted to design fault-tolerant architectures of autonomous systems should be able to execute various types of faults effectively and without considerable disruption to the system.

Logic Mapping Concepts

Fault tolerance in reconfigurable system is an important strategy that involves redundant logic mapping. It entails the repetition of logic functions in order to identify and repair faults thus ensuring that operationality of the system remains in the event of faults. Triple Modular Redundancy (TMR) is one common strategy, in which three initial components identical perform the same mission and a majority vote finds out which answer should be chosen. In masking a single fault, TMR is good[6] but has a collateral cost of high overhead in the logic area as well as power consumption [L2-I2].

A more economical alternative is Dual Modular Redundancy (DMR) that makes use of two redundant modules to identify differences in results. Though DMR does not have fault correction ability it would start recovery processes like reconfiguration when fault is detected.[7] Generalizations of these schemes are N-Modular Redundancy (NMR) using more and more replicas and more elaborate voting logic that often finds application in systems where reliability is paramount, such as space systems.

The implementation of redundancy can be classified widely into spatial redundancy, and temporal redundancy. Spatial redundancy maps stored copies of logic cells in physically different places, enabling them to execute over one another and fault masking. It works especially well in safety-related cases but requires significant quantities of hardware resources. On the other hand, temporal redundancy shares the same set of hardware with diverse time slices in order to perform overlapping operations. This decreases area overhead but comes with execution latencies, thus not so appropriate to applications that require timing.[8]

Coupling redundant logic mapping with reconfiguration strategies, bitstream scrubbing, relocation of modules, as well as immediate migration of tasks, the designers can create adaptive, fault resistant systems that continue to operate even in a fault condition. These approaches are of particular importance in such applications as wearable health monitoring, smart buildings where fault tolerance and energy efficiency have to be complementary features inherent in small embedded systems.[11, 12]

Redundancy Techniques and Their Implementation

Redundancy methods are of paramount importance in fault tolerant reconfigurable architecture, where redundancy methods are applied both to eliminate design complexity and to achieve system resilience against hardware faults. The techniques can be generally divided into spatial redundancies, temporal ones and hybrid methods that take the benefits of two kinds of technique. All the techniques vary in the fault coverage, overhead of resources, latency and their application suitability in autonomous systems.

Spatial Redundancy

Spatial redundancy involves duplicating or triplicating hardware components and executing Redundancy Spatial Spatial redundancy can also be done in **hardware** by duplicating or even triplicating components within a system and operating in parallel within physically separate logic blocks. The most everyday of those uses is Triple Modular Redundancy (TMR) where three identical modules do the same task and a majority voter decides what the correct output should be. This method is so good at masking individual faults, and is so common in space and safety-critical systems where a high fault coverage is desirable. The other notable approach within the spatial redundancy is the provision of spare logic blocks. With this strategy auxiliary space that is not used, or space that is left idle, in a reconfigurable fabric (like an FPGA) is held in reserve. Particularly, when a fault is identified in a working module, the system re-routes the functionality of the faulted module (dynamically) to another spare block through partial reconfiguration. This redundancy-friendly module placement also cannot have critical logic functions installed physically adjacent to each other, so this reduces the likelihoods of common-mode failures. The spatial redundancy is particularly applicable to hard real-time autonomous systems that include control systems of UAVs or ECU of automobiles; fault masking and a minimized latency response are paramount in these kinds of applications. This method has however the overhead in area and power that might be a hindering factor when it comes to scaling it in resource bound embedded systems.

Temporal Redundancy

Temporal redundancy uses time-multiplexing to allow redundancy operations to be carried on same hardware resources at different times. Rather than using numerous parallel modules, the same task has several instances running in series in a single module. These outputs are then compared in order to identify inconsistency.

Checkpoint and rollback common implementation A typical example of temporal redundancy algorithm is known as the checkpoint and rollback algorithm, in which the system state is periodically saved. In case of execution fault, (e.g., using a parity or ECC check) the system is rolled back to a prior known good checkpoint and the computation is restarted. This method is especially useful when one is faced with resource limitations against full spatial redundancy.

The advantage of using temporal redundancy is very effective as far as hardware is concerned since there is minimal duplication of logic. It however has latency, thus it is less favorable in applications with strict real time requirements. It finds frequent application in tasks of low-to-moderate criticality, e.g. background processing of data, checking sensor data in autonomous control systems.

Hybrid Approaches

Hybrid redundancy methods mix the advantages of the spatial and temporal redundancy to use reliability, performance and resource consumption. An example would be to store critical modules using spatial redundancy application and the less critical or time sensitive activities using temporal redundancy application. The mixed-criticality design can enable the designer to selectively apply the redundancy according to the importance of the functional activity and due to the time constraints.

There is also the use of voting logic and self checking circuits with Hybrid methods. The circuits do repeated validation of the outputs through cross-checking of results with other modules or execution stages. As a simulation, a hybrid system could perform two operations in parallel (spatially) and then tested the outcome by performing the same operation on the third time (temporally), guaranteeing fault detection as well as fault correction.

The other potent idea with hybrid strategies is the dynamic reconfigurable redundancy profile where the system shifts dynamically between spatial and temporal components of redu1ndancy depending on existing operational conditions, available power budget, or severity of detected faults. Such a dynamic behavior is particularly useful in non-programmable platforms that perform under changing mission manifestations and energy separates.

To conclude, spatial, temporal, and hybrid redundancy methods have a variety of tools of implementing fault-tolerant reconfigurable architectures. Choosing a suitable redundancy strategy should also be done bearing in mind the needs of a particular relevant system, reliability, as well as time and resource needs. Redundancy aware design automation techniques, online monitoring, and dynamic reconfiguration are an important way to optimize these trade-offs at advanced design and to provide scalable, robust platforms on which safety-critical tasks will run.

Runtime Fault Detection and Recovery

In the execution of faults in a run time environment, modern reconfigurable structures utilize a mixture of detection, diagnosis and recovery techniques to manage the faults effectively. Table 1 is an overview of comparative review of commonly used methods which can fall in the categories of error detection, fault diagnosis and recovery mechanism. Lightweight and in some sense low-latency fixes to real-time detection of transient errors in memory and interconnects are provided by techniques like ECC and parity checks. BIST, though being more resource-consuming is useful in scheduled offline testing and structural integrity certifying. In fault diagnosis, online fault localization allows us to arrive at precise fault mapping at runtime yet prediction using a machine learning framework creates intelligent pro-active mitigation, albeit at the cost of a higher computational overhead. Runtime repair of dynamic systems is possible using recovery mechanisms such as Dynamic Partial Reconfiguration (DPR) and module reloading that can occur whenever systems experiences a malfunction without stopping system activities. Meanwhile a spatial redundancy, the tile activation of spare tiles, provides an almost foolproof at the cost of resource-intensive. Selection of these techniques can be done based on application specific constraints such as latency tolerance budget, area and tightness level. The comparative analysis is able to allow a system architect to specify and select fault management strategy techniques which are applicable and capable of fitting to the specific autonomous systems that have provisions of high dependability requirements (Table 1).

Applications in Autonomous Systems

To provide reliability in the unpredictable environment, it requires integration of fault-tolerant computing in autonomous systems. Autonomous platforms have

Table 1: Comparison of Runtime Fault Detection and Recovery Techniques in Reconfigurable Architectures

Technique	Category	Purpose	Latency	Hardware Overhead	Suitability
Error-Correcting Codes (ECC)	Error Detection	Detects and corrects bit-level memory errors	Low	Medium	Ideal for memory blocks and cache systems
Parity Checks	Error Detection	Simple error detection on data lines	Very Low	Low	Used in buses, registers, and lightweight modules
Built-In Self-Test (BIST)	Error Detection	Autonomous offline circuit testing	Medium	High	Useful for periodic testing and certification
Online Fault Localization	Fault Diagnosis	Identifies the exact fault location at runtime	Medium	Medium	Suitable for adaptive and real-time systems
ML-Based Fault Prediction	Fault Diagnosis	Predicts fault occurrence using historical and sensor data	Variable	High (if model is complex)	Best for intelligent fault-aware systems
Dynamic Partial Reconfiguration (DPR)	Recovery Mechanism	Reconfigures faulty regions without stopping system	Low–Medium	Medium	Enables in-place repair with minimal disruption
Module Swapping & Reloading	Recovery Mechanism	Replaces faulty modules using preloaded bitstreams	Medium	Medium–High	Needs spare logic and efficient memory access
Spare Tile Activation	Recovery Mechanism	Activates pre-assigned fault-free resources	Low	High	Excellent for spatial redundancy platforms

different types such as those which are aerial, others land based, still others are space-based and as well are diverse, and because of these diverse challenges in class, each level should use a specific fault mitigation approach. An example can be Unmanned Aerial Vehicles (UAVs) which also depend considerably on ongoing and precise sensor information to navigate and have a stable flight. Fault masking In such systems, Triple Modular Redundancy (TMR) fault masking and redundancy in IMU/GPS fusion modules can enable safe control in the event of a sensor failure or noise.

Autonomous Ground Vehicles on the other hand require a very high degree of resilience and reliability in both perception and decision-making dynamics within urban environments. The hardware faults of these systems are regularly covered by redundant deep learning pipelines and with dynamic task migration they prevent perturbation in safety by isolating and recovering hardware faults. Configuration scrubbing, error correction codes (ECC) and SEU-hardened logic have been used in space and defense programs where cosmic radiation and thermal extremes are frequent occurrences to maintain continuity of the mission.

Embedded deep learning/neuromorphic based Edge-AI robotic systems that utilize real-time autonomy need to be resilient to faults and need to operate under both stringent power and area requirements. In this case, lightweight built-in self-tests (BSTs),

Table 2: Fault-Tolerant Design Strategies in Various Autonomous System Domains

Application Domain	Target Modules	Fault-Tolerance Strategy	Benefits / Use Case
Unmanned Aerial Vehicles (UAVs)	Navigation, Sensor Fusion Modules	Triple Modular Redundancy (TMR), Redundant IMU/GPS Paths	Ensures stable flight and location accuracy under sensor failure
Autonomous Ground Vehicles	Perception, Decision-Making Pipelines	Redundant Deep Neural Networks (DNNs), Dynamic Task Migration	Enhances safety and reliability in real-time driving scenarios
Space and Defense Systems	On-Board Processors, Memory Blocks	SEU-Hardened Logic, Configuration Scrubbing, ECC	Maintains system integrity under radiation and cosmic events
Edge-AI Robotics	AI/ML Accelerators, Neuromorphic Cores	Online Fault Detection, Self-Healing Circuits, Lightweight BIST	Sustains continuous learning/inference under hardware degradation
Autonomous Marine Systems	Sonar, Underwater Navigation Units	Error Correction + Module Swapping with Spare Tiles	Provides resilience in high-pressure and moisture-prone environments

self-healing circuits and online error detection are employed to maintain reliability of inference. Likewise, autonomous marine systems that reside in high-moisture, high-pressure regions incorporate module swapping with autonomous tile activation in order to overcome damage to functionality.

These fault-tolerant design techniques have been tabulated in Table 2 in different autonomous domains showing the dependence of the target modules with the kind of protection mechanism. Moreover, these strategies are applied on various architecture levels as shown in Figure 1, starting at sensor devices and memory, proceeding to AI process and actuator control, which proves systemic incorporation of fault-tolerance in autonomous and intelligent platforms.

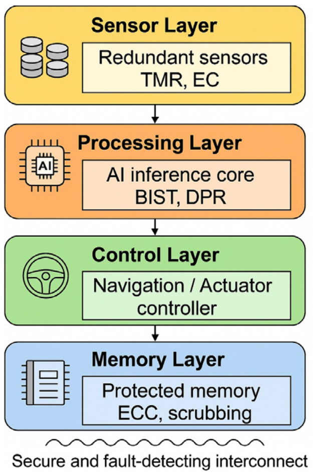
Sensor Layer
Redundant sensors TMR, EC
Processing Layer
AI
AI inference core BIST, DPR
Control Layer
Navigation / Actuator controller
Memory Layer
Protected memory ECC, scrubbing
Secure and fault-detecting interconnect

Fig. 1: Integration of Fault-Tolerance Strategies in Autonomous System Architectures

Evaluation Metrics and Comparative Analysis

Reliability Metrics

The initial step in gauging the effectiveness of fault-tolerant reconfigurable architectures is reliability assessment that is normally determined based on two major metrics Mean Time to Failure (MTTF) and Fault Coverage. MTTF is a measure of the anticipated time a system runs until a fault develops, and fault-tolerant designs attempt to maximize this measure by the incorporation of systems to prevent, conceal, or restore when faults happen in the course of operation. Fault coverage, in contrast, measures the number of faults which can be detected or fixed in a particular fault model in a percentage. When the fault coverage is high it means that the system will take care of a wide variety of fault types i.e. transient faults, intermittent faults and permanent faults. Such metrics are particularly essential in safety-, mission and autonomous vehicles, aerospace systems and defense platforms where unnoticeable faults or short-term downtimes can have disastrous consequences.

Resource Overhead

Although it increases fault tolerance, redundant logic and reconfigurable recovery mechanisms logically produce supplementary overheads on hardware resource and power consumption. Significant measures of evaluation, in this respect, is the utilization of Look-Up Table (LUT) and Flip-Flop (FF), as measures of the physical logic resources spent in implementations working on FPGA. Technologies like Triple Modular Redundancy (TMR) have a potential to consume 2-3x more LUTs which is a major concern on the aspect of area utilization. Another significant factor is the power dissipation where a fault-tolerant circuit might have increased static and dynamic power dissipation due to voting logic or other spares modules. Besides, reconfiguration latency (measured as time to reconfigure a faulty logic block, especially under Dynamic Partial Reconfiguration (DPR) schemes) should be confined within the permissible limits of the targeted application to ensure smooth functioning. Trading off these overheads against the intended fault resilience is one of the key design issues, and in resource-limited contexts like edge-AI systems and embedded platforms where the power, area and timing budgets are quite stringent.

Performance Trade-offs

A momentous evaluation of fault tolerant reconfigurable architectures is the performance-reliability trade off. As redundancy is added, fault-resilience can be added, but with the trade-offs of less system throughput, or a slower response. As an example, Triple Modular Redundancy (TMR) enhances execution determinism, in that faults are masked by a majority voting operation, but at the cost of extra logic, and hence lower performance. Likewise, temporal redundancy, which repeats the previous hardware resource to perform multiple execution cycles, saves area and power without having to bring in delays that are harmful to time-conscious applications. System designers should therefore evaluate these trade-offs with respect to application-based constraints, such as real-time responsiveness, energy efficiency, and obligatory fault coverage, so as to guarantee that the fault-tolerance measures they use would not undermine the system operational intentions.

Comparative Analysis of State-of-the-Art Techniques

In order to understand the practical efficiency of the fault-tolerant reconfigurable architecture, comparison of most recent research efforts on the basis of certain key metrics fault model coverage, redundancy scheme, improvement of Mean Time to Failure (MTTF), hardware resource and power overhead, and recovery latency cannot be neglected. The table summarizes in a comparative manner well known fault-tolerant methods suggested during the past 5-7 years (table 3).

These solutions depict the variability in the design strategies, whereby the traditional spatial redundancy covers TMR and DMR, whereas more dynamic and intelligent solutions become hybrids and ML-based fault recovery. Techniques involving spatial masking such as that used by Ng et al.[1] give a high fault masking capability but with a significant overhead. Temporal codes on the other hand are area-efficient, but not so suitable to real-time duty as they contain a certain delay, e.g. Sami et al.[2] The hybrid model developed by Kozlova and Smirnov[3] shows a trade-off well balanced because this approach uses both spatial and temporal approaches to mitigate transient as well as age-related failures. Importantly, new solutions based on machine learning-related recovery techniques, evidenced by the work of Javier et al.,[5] may be fed

Table 3. Comparison of Recent Fault-Tolerant Reconfigurable Architectures (2018-2025)

Reference	Redundancy Type	Fault Model	MTTF Improvement	Overhead (LUT / Power)	Recovery Time
Ng et al., 2021[1]	Spatial (TMR + DPR)	SEU, transient	3×	2.7× LUT, +35% power	~12 ms (DPR)
Sami et al., 2019[2]	Temporal	Intermittent, transient	1.8×	1.3× LUT, +12% power	~9 ms
Kozlova & Smirnov, 2025[3]	Hybrid	SEU, aging	3.5×	2.1× LUT, +25% power	~10 ms (dynamic swap)
Alves et al., 2020[4]	Spatial (DMR)	Permanent	2.2×	1.8× LUT, +18% power	~6 ms
Javier et al., 2025[5]	ML-based Recovery	Predictive / Transient	2.7×	2.0× LUT, +20% power	~15 ms (with feedback)

into predictive fault processing, though they have not yet succeeded in the areas of runtime delays and implementation effort. Since Table 3 demonstrates that hybrid schemes provide the best improvement of MTTF at moderately high overhead, performances of the three schemes are compared and discussed in detail. Figure summarizes the performance and fault-tolerance trade-offs in a graphical form.

Below is the radar chart measuring the similarities and differences among the techniques techniques in fault tolerance in important dimensions: fault coverage, time of reconfiguration, overhead of power and resources, and scalability. Multi-dimensional trade offs between TMR + DPR, ML-based recovery, temporal redundancy, and Hybrid models can be explained in this visualization.

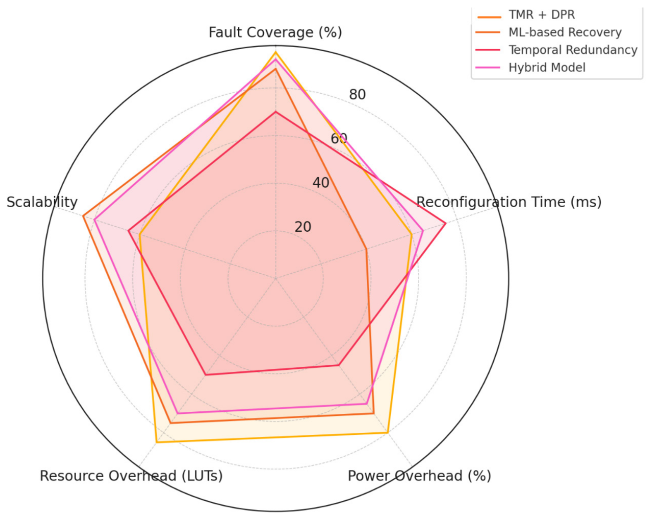

Fig. 2: Radar Chart of Fault-Tolerant Architecture Metrics

Open Challenges and Research Directions

Inspite of all the advances in fault tolerant reconfigurable computing, some of the most intriguing research challenges have remained open, and they constitute alluring areas of investigation in the future. With increasing complexity of design, effective and precise fault identification in large, diverse architectures is more and more of a difficulty. It creates a demand to evolve intelligent and redundancy conscious high-level synthesis (HLS) design tools in combination with AI-augmented logic mapping algorithms capable of maximizing the coverage of faults without significant overhead. Moreover, upcoming systems need to respond to requirements of mixed-criticality systems, within which diverse reliability and timing goals have to concur in a shared architecture. The post-quantum-secure mechanisms of fault detection should be also incorporated to guarantee resistance towards classical and novel cyber-physical attacks. Last but not least will be the move towards comprehensive cross-layer resilience, that is, circuit-level fault detection, up to the system-level adaptive reconfiguration, and the realization of robust and smart autonomous platforms able to maintain long-term functionality in dynamic and mission-critical environments.

References

1] M. C. Ng, A. J. Laffely, and D. Koch, “Dynamic Partial Reconfiguration in FPGAs: A Survey of Architectures, Tools, and Applications,” *ACM Computing Surveys*, vol. 54, no. 2, pp. 1-39, 2021.
2. K. Compton and S. Hauck, “Reconfigurable computing: A survey of systems and software,” *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, Jun. 2002.
3. R. Velazco, P. Fouillat, and R. Reis, *Radiation Effects on Embedded Systems*, Springer, 2007.
4. S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov.-Dec. 2005.
5. T. Mitra and P. Mishra, *Secure and Reliable System Design for Embedded Systems*, Springer, 2016.
6. R. E. Lyons and W. Vanderkulk, “The use of triple-modular redundancy to improve computer reliability,” *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, Apr. 1962.
7. A. R. Alves, R. P. Ribas, and A. I. Reis, “Area, power and delay trade-offs in redundant logic synthesis,” *Microelectronics Journal*, vol. 40, no. 11, pp. 1664–1672, Nov. 2009.
8. M. G. Sami et al., “Temporal redundancy and checkpointing for fault tolerance in FPGAs,” *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 5, pp. 1769-1776, Oct. 2006.
9. M. A. Müller, J. C. Schmidt, and C. M. Fischer, “Sustainable VLSI design: Green electronics for energy conscious systems,” *Journal of Integrated VLSI, Embedded and Computing Technologies*, vol. 2, no. 2, pp. 44-51, 2025.
10. R. Rangisetti and K. Annapurna, “Routing attacks in VANETs,” *International Journal of Communication and Computer Technologies*, vol. 9, no. 2, pp. 1-5, 2021.

11. S. Sadulla, "IoT-enabled smart buildings: A sustainable approach for energy management," *National Journal of Electrical Electronics and Automation Technologies*, vol. 1, no. 1, pp. 14–23, 2025.
12. F. Javier, M. José, J. Luis, A. María, and J. Carlos, "Revolutionizing healthcare: Wearable IoT sensors for health monitoring applications: Design and optimization," *Journal of Wireless Sensor Networks and IoT*, vol. 2, no. 1, pp. 31–41, 2025.
13. E. I. Kozlova and N. V. Smirnov, "Reconfigurable computing applied to large scale simulation and modeling," *SCCTS Transactions on Reconfigurable Computing*, vol. 2, no. 3, pp. 18–26, 2025.