SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 3, No. 2, 2026 (pp. 21-29)
RESEARCH ARTICLE ECEJOURNALS.IN

Design and Evaluation of a Fault-Tolerant Reconfigurable Architecture for Mission-Critical Embedded Systems

F. Mohd Zaki[1]*, T Shimada[2]

[1]Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor 43600, Malaysia

[2]School of Electrical Engineering, Hanoi University of Science and Technology, 1 Dai Co Viet, Hanoi 11615, Vietnam

Keywords:
Fault-Tolerant Architecture,
Dynamic Partial Reconfiguration (DPR),
Field-Programmable Gate Array (FPGA),
Mission-Critical Embedded Systems,
Runtime Recovery,
Built-In Self-Test (BIST),
Soft Error Mitigation,
Xilinx Zynq SoC,
Hardware Redundancy,
Reconfigurable Computing

Author's Email:
zaki.f.m@ukm.edu.my
shimada.t@hust.edu.vn

DOI: 10.31838/RCC/03.02.03

Received : 04.01.2026

Revised : 08.03.2026

Accepted : 10.04.2026

Abstract

High reliability is required of mission-critical embedded systems that work in aerospace, medical and security applications and must guarantee fault tolerance and operate continuously, regardless of environmental conditions (harsh or unpredictable). The design and assessment of one new fault-tolerant reconfigurable architecture using the Dynamic Partial Reconfiguration (DPR) capabilities of FPGA platforms is presented in this paper to schedule the executions of the tasks such that they jointly meet the requirements of detecting, isolating, and recovering the hardware faults in real-time. The proposed architecture will incorporate a hybrid fault detection policy, which is a combination of Built-In Self-Test (BIST) and Cyclic Redundancy Check (CRC) into a form of reconfiguration controller that will reload partial bitstreams dynamically into the damaged areas of the FPGA. Runtime checkpointing and a low-latency recovery strategy are also factored into the system to reduce as much downtime as possible. The proposed solution is experimented and tested on a Xilinx Zynq-7000 SoC platform and fault injection to simulate soft error. The experimental data show a fault recovery rate of over 93%, a reconfiguration latency of less than 4.3 ms and a minimal area and power overhead, which validate the appropriateness of the architecture in safety networked outcomes in embedded systems. These experiments highlight the promisingness of DPR-based fault tolerance with respect to scalability and resource use and as an alternative to the traditional redundancy-based approaches.

How to cite this article: Zaki FM, Shimada T (2026). Design and Evaluation of a Fault-Tolerant Reconfigurable Architecture for Mission-Critical Embedded Systems. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 2, 2026, 21-29

Introduction

The safety-critical applications of mission-critical embedded systems include aerospace, automotive, and medical instrumentation, defense electronics, and nuclear control systems. Failures in the system may therefore turn out to be disastrous; loss of life or mission failure. Thus, they require reliable systems, fault-tolerance, and the ability to continue despite hugely adverse conditions either in the environment or the operational environment.[1]

Conventional fault tolerance techniques are based on extensive hardware redundancy, e.g. Triple Modular Redundancy (TMR) where three identical modules operate in parallel with the same kind of action and a majority voter decides the right output.[2] TMR exhibits high fault coverage, with substantial area, power and cost overheads which is not ideal in resource-limited embedded applications.[3] Further, TMR has low flexibility in case of changes in fault conditions or when strategy is implemented in terms of runtime reconfigurable logic.

Over the past few years FPGAs (Field-Programmable Gate Arrays) have been put forward as one potential platform to perform reconfigurable embedded computing because of the features of parallel processing, flexibility, and the ability to dynamically reconfigure functions.[4] In particular, Dynamic Partial Reconfiguration (DPR) allows changing functionalities of parts of an FPGA during operation, in such a way that other parts of the system are not affected.[5] This attribute renders FPGAs simple targets when implementing the adaptive and fault-tolerant models in critical areas.

Some studies tried to use DPR to reduce faults. As an example, scrubbing techniques of soft errors in the configuration memory are used to fight off single-event upsets (SEUs) periodically.[6] The other methods make use of detecting the faults that are stuck in a static manner and the subsequent reconfiguration of the FPGA logic in entirety.[7] The techniques, however, do not always work well due to the latency overhead,

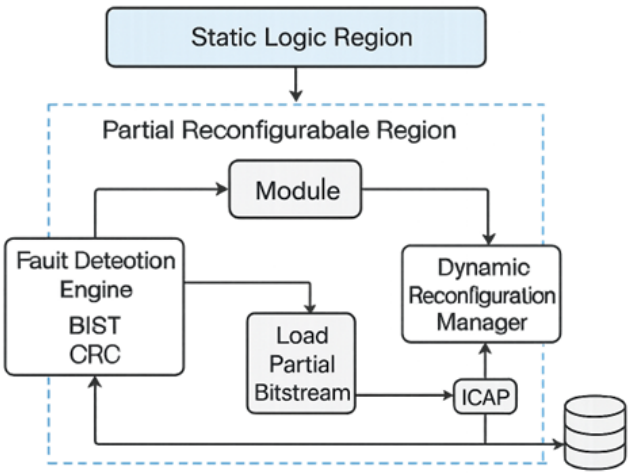

Fig. 1: System-Level Overview of DPR-Based Fault-Tolerant Architecture for Embedded Applications

granularity or modular level unrecoverability. Moreover, there is no architecture to date that has run-time cognizance in terms of fault detection, and adaptive decision making.

The novel architecture on fault tolerant reconfigurable architecture that this paper demonstrates is one that is suitable to mission critical systems that are embedded. The presented solution combines the several fault-tolerant measures in order to provide a steady and stable functionality. Its fault detection mechanism is in a hybrid state i.e., Bright-In Self-Test (BIST) and Cyclic Redundancy Check (CRC) are used to allow faults to be detected in hardware modules accurately and in real-time. When a fault is detected, a Dynamic Reconfiguration Manager (DRM) is used to manage the restart process with Dynamic Partial Reconfiguration (DPR) to restart the system equipped with only the needed module to restart in addition to no need of restarting the entire system. In order to have an extra assurance of reliability, the architecture introduces a checkpoint-recovery system that maintains the working state of the system such that it can resume straight where it left off after reconfiguration. Modules used in recovering, checking the fault at the run time, and preservation of the state are effective in making the architecture very applicable in safety- and time-based applications.

The implemented architecture is on a xilinx Zynq-7000 SoC platform and the evaluation is on fault injection campaigns of emulating soft and transient errors. Experimental results show that the system provides better than 93 percent fault recovery with latency of less than 20 us and an area overhead that is less than 20 percent, thus proving that the system is applicable in real-time mission-critical applications.

The remaining paper is as follows: In Section II, related work in fault-tolerant embedded systems and DPR techniques is discussed. Section III is the introduction of the new proposed architecture and the fault detect and recovery process. Section IV provides the scheme of the experiment and platform configuration. Section V talks about the results of the evaluation. Section VI provides a comparison with the current approaches. Conclusions The study ends in section VII with an indication of research in the future directions.

Related Work

Fault tolerance in mission-critical embedded systems has garnered significant attention due to the demand for high system reliability, especially in aerospace, defense, and medical electronics. Over the years, various techniques have emerged to address both transient and permanent faults in reconfigurable hardware platforms such as FPGAs.

One of the earliest and most established techniques is **Triple Modular Redundancy (TMR)**, where three identical logic modules execute the same operation in parallel and a majority voter is used to determine the correct output. Although TMR offers strong fault masking capabilities, it suffers from high area and power overheads, limiting its applicability in resource-constrained embedded platforms.[1] To reduce redundancy cost, **Duplication with Comparison (DWC)** was introduced, which duplicates the logic and compares the outputs for fault detection, offering lower overhead with no automatic correction.[2]

As soft errors, particularly **Single Event Upsets (SEUs)** in SRAM-based FPGAs, became a growing concern, **scrubbing techniques** such as configuration memory readback and frame refresh were developed to correct configuration upsets. These techniques provide a degree of error correction but often at the cost of system downtime and energy inefficiency.[3, 4]

With the evolution of reconfigurable platforms, **Dynamic Partial Reconfiguration (DPR)** has enabled more flexible and runtime-capable fault recovery solutions. Tools like ReCoBus-Builder automate the process of partitioning FPGA logic into reconfigurable blocks, thus allowing selective reloading of faulty modules without system interruption.[5] Further enhancement was shown through selective hardening methods using DPR to reload specific logic regions following fault detection.[6]

To anticipate faults before system failure, **AI-driven fault prediction mechanisms** have been proposed. Azzam et al. introduced a neural network-based fault prediction system that monitors logic block behavior in FPGAs to foresee potential faults and mitigate them before system degradation occurs.[7] Similarly, **Network-on-Chip (NoC)** based multi-core architectures with self-reconfiguration capabilities allow for adaptive routing and system-level resilience to localized faults.[8]

Recent surveys underscore the importance of **hybrid fault-tolerant designs** combining detection, isolation, and real-time recovery.[9] These include emerging works in **high-performance computing architectures for AI/ML** that incorporate reliability at the fabric level for critical processing workloads,[10] as well as **nanotechnology-enhanced circuit designs** that offer physical-level fault resistance through material innovations.[11]

In parallel, recent embedded system studies have focused on complementary fault-prone domains such as mobile ad hoc networks (MANETs), RF systems, and miniaturized antennas for wearable electronics.[12, 13, 14] These works highlight the growing need for fault tolerance in increasingly miniaturized and multifunctional embedded platforms.

Moreover, enhanced cybersecurity frameworks in embedded systems now address fault-injection attacks and resilience against malicious reconfiguration, as discussed in.[15] However, despite the advancements, few architectures integrate **low-latency hybrid fault detection, seamless partial reconfiguration, and state recovery** into a unified system suitable for mission-critical applications.

This paper aims to bridge this gap by proposing a **DPR-enabled fault-tolerant reconfigurable architecture** that supports **real-time fault detection, isolation, and recovery** with minimal performance and resource overhead.

System Architecture

Overview

In the proposed fault-tolerant reconfigurable architecture, the role of the dynamic partial reconfiguration (DPR) will be utilized on the FPGA platforms to maintain a continuous execution of mission-critical embedded applications. The system architecture is scheduled into four main parts namely: (i) the Static Logic Region where the supervisory control, monitoring units and communication interface are located, (ii) Partial Reconfigurable Regions (PRRs) with module functionalities instantiated application-specific modules can be reconfigured dynamically at run-time, (iii) a Fault Detection Engine (FDE) that diagnoses hardware faults in real-time by a combination of Built-In Self-Test (BIST) and Cyclic Redundancy Check (CRC), and (iv) a Dynamic Re High

availability can be supported through this modular architecture which will allow configurational reuse, selective reuse of a specific module, without affecting the rest of the system.

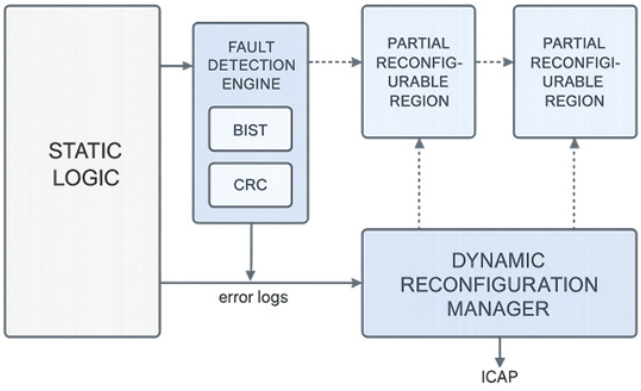

Fig. 2: System Architecture of the Proposed DPR-Based Fault-Tolerant Embedded Platform

Fault Detection Mechanism

In order to enable fast with pin point accuracy on the faulty hardware units, a hybrid fault detection mechanism has been built into the system. The Fault Detection Engine (FDE) employs Concurrent Built-In Self-Test (CBIST) circuitry built into each reconfigurable module which enable the system to self-check without suspend operation of the system during the course of operation. The tests are either periodically activated or requested when needed to identify stuck-at faults, bridging faults, or corruption of logic. At the same time, the cyclic redundancy check (CRC) codes are calculated in the inter-module data transactions, which helps to detect bit-flip or transient faults in communication. Identified errors are recorded and sent out to the DRM through a special error-reporting bus. By using this multilayered detection approach, spatial and temporal coverage of faults is provided to provide greater observability of faults and reduced unobserved failures.

Fault Recovery Process

When fault is identified in any of the modules of a PRR, fault recovery action starts with Dynamic Reconfiguration Manager (DRM). Logically the faulty PRR is isolated to make sure that there is no further spreading of wrong outputs. The execution of the module is made inactive, and the last state that it was checked at (if one exists) is saved. The DRM subsequently loads the respective partial bitstream of the healthy version of the affected module that is placed in non-volatile memory in the form of an external flash or of a DDR. This bitstream is broadcasted into the FPGA over the ICAP interface and erases the logic that is corrupted by leaving the rest of the FPGA fabric functional. After the reconfiguration is carried out, the module will continue to run at the last valid state, hence restoring functionality with minimal impact. The technique does not require that the whole system be restarted, reducing significantly the fault-discovery latency time, and increasing the effectiveness of fault tolerance. This is where you detail the step-by-step logic of the PRR-based fault recovery mechanism, fault detecting and ICAP-based mechanism of reconfiguration. This logic is visibly supported with the aid of the flowchart.

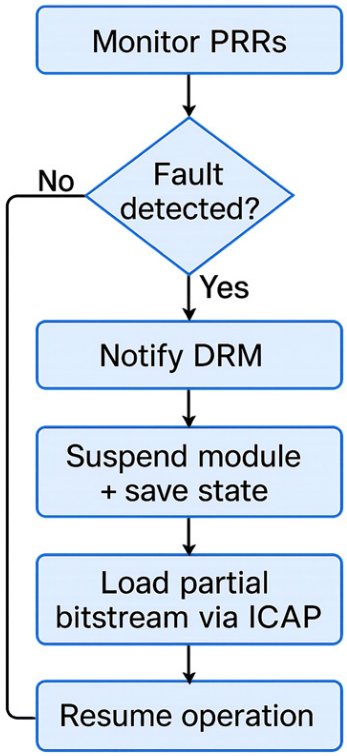

Fig. 3: Fault Recovery Flowchart for DPR-Based Fault-Tolerant Embedded Architecture.

Figure 3 illustrates the fault recovery flow within the proposed architecture, highlighting the sequence from fault detection to dynamic reconfiguration and operational restoration via ICAP.

Redundancy and Checkpointing

The system embraces temporal form of redundancy as opposed to permanent spatial redundancy such as TMR in order to maximize resource usage. CBIST and CRC are done at runtime to detect faults and only the faulty modules are reconfigured on an as-needed basis and hence not subject to constant utilization of additional hardware resources. Further, the architecture extends the periodic state checkpointing of individual reconfigurable modules. Such checkpoints capture the internal state of the module at a set frequency which can be backed up in the event of fault recovery. This avoids the need to restore the module to its original state and makes it continuous and less likely to lose data or cause the rolling back of systems. The temporal redundancy combined with intelligent checkpointing enables the system to be highly reliable and to be immediately responsive, at the same time having low overhead on hardware.

Following the architecture and flow definitions, the algorithm formalizes the steps that occur during the run-time in a pseudocode form and gives a detailed description of the operations that are to be done.

Algorithm 1 presents the reasoning performed by the Dynamic Reconfiguration Manager (DRM) to guide fault recovery by using checkpointing and partial loading of a bitstream.

Algorithm 1: Dynamic Fault Recovery Logic Managed by the DRM Module.

```
Algorithm 1: Dynamic Fault Recovery
Controller
Input: Fault_Signal, PRR_ID, Bitstream_DB
Output: Recovered_Module_Status

1: if Fault_Signal == TRUE then
2:   DRM ← Initialize_Recovery()
3:   Checkpoint ← Save_Module_State(PRR_ID)
4:   Bitstream ← Fetch(Bitstream_DB, PRR_
ID)
5:   ICAP_Interface ← Load(Bitstream)
6:   Restore_State(PRR_ID, Checkpoint)
7:   Recovered_Module_Status ← ACTIVE
8: else
9:   Recovered_Module_Status ← OK
10: end if
```

Methodology

Experimental Setup

In order to assess the practicality and validity of the postulated fault-tolerant reconfigurable architecture, the experimentation process was simulated on the Xilinx Zynq-7000 SoC platform, a platform that comprises of an ARM Cortex-A9 multiprocessor, and FPGA fabric. This platform provides programmable logic as well as embedded processing functionalities, which is why it qualifies as an embedded system development environment to experiment with real-time and reconfigurable embedded systems. Vivado High-Level Synthesis (HLS) with the Design flow was used to design and implement the hardware and Synthesis application modules by the use of C/C++ code whereas Xilinx Software Development Kit (SDK) with integrated embedded software and control code. Partial Re-configuration (PR) flow in Vivado was applied to establish and manage reconfigurable partitions, compose partial bitstreams and orchestrate reconfiguration through the Internal Configuration Access Port (ICAP).

Three computationally representative and computationally diverse set of benchmark workloads were chosen as representative of different types of embedded capabilities. As an example, the AES encryption module was selected to test performance under control-critical and security-intensive tasks, and the PID (Proportional-Integral-Derivative) type controller was presented as one of the many applications in industrial-level control and robotic-level control. Besides, the FIR (Finite Impulse Response) filter embodied signal processing applications that had predictable computational-intensive jobs. Each of these benchmark modules was placed in a specific Partial Reconfigurable Regions (PRR) of the FPGA fabric so that faults in the module could be reconfigured at the module level in an orthogonal part of the fabric with the rest of the system remaining unaffected. To confirm the obtained properties of fault detection and recovery detected under the proposed architecture, a controlled fault injection mechanism was used. Two main categories of faults were simulated: (i) bitstream corruption, which models configuration memory upsets and module partial malfunction, and (ii) Single Event Upset (SEU) emulation, which is achieved by flipping a given set of configuration bits at run time through the

Xilinx Soft Error Mitigation (SEM) IP or dedicated fault injector APIs. This configuration allowed repeatable, deterministic experimentation so as to determine how much the system is sensitive and able to handle fault conditions in a realistic working environment.

Evaluation Metrics

In order to have an overall assessment of the proposed fault-tolerant reconfigurable architecture, four performance metrics have been identified. The first one is the Fault Recovery Rate (FRR), which is used as the percentage of the number of faults injected against the number of faults successfully detected and recovered by the system. FRR is a basic indicator of a system dependability where the value close to or more than 90 percent is confirmed as highly robust design that could be executed in real time applications in fault management. The second measure is Reconfiguration Latency that determines the time period of detecting a fault, starting a partial reconfiguration, loading a respective bitstream via the Internal Configuration Access Port (ICAP), and continue the execution of this module. The measurement is critical to evaluating the capacity of the system to react to faults promptly and the latency within the system is acceptable when measured in a few milliseconds. To get a numeric indicator of the quality of the proposed architecture in its application in situations of faults, we determine the Fault Recovery Rate (FRR) in the following way:

$$FRR = \left(\frac{N_{recovered}}{N_{fault\_injected}} \right) \times 100\% \quad (1)$$

where represents the number of faulty modules successfully recovered via partial reconfiguration, and denotes the total number of faults injected during testing. The greater the value of FRR is, the more system reliability and fault tolerance an offering is likely to have.

Area and Power Overhead is the third metric, and it can be determined by the post-synthesis and implementation reports provided by Vivado and represents the added logic resources (e.g., LUTs, flip-flops and BRAMs) and dynamic power consumed by the introductions of fault detection and recovery logic. The objective is to keep these overheads at not more than 15 percent as a way of making good utilization of resources. Lastly, Throughput and Latency Degradation measures the effects on the overall performance of a system due to a restart operation because of the faults by providing a comparative evaluation of the performance (in terms of execution times and throughput) of the system in normal and faulty conditions. All of this together gives an

Table 1: Evaluation Metrics for the Proposed Architecture

Metric	Description	Measurement Tool/Method	Target/Expected Value
Fault Recovery Rate (FRR)	% of successfully recovered faults after injection	Fault injection logs, recovery confirmation via CRC	≥ 90%
Reconfiguration Latency	Time between fault detection and successful module restoration via DPR	On-chip timer + ICAP reconfig duration	≤ 5 ms
Area Overhead	Additional hardware utilization due to fault-tolerant logic (LUTs, FFs, BRAMs)	Vivado Post-Implementation Report	≤ 15%
Power Overhead	Additional dynamic power consumed by detection and reconfig logic	Vivado Power Analyzer	≤ 10%
Throughput Degradation	Performance drop in processing throughput during/after fault recovery	Custom workload profiler	≤ 5% from fault-free baseline
Latency Degradation	Increase in task completion time due to reconfiguration	Real-time execution profiling	≤ 10% from nominal latency

overview of the operational performance, scalability, and the aptness of employed for implementing the architecture in mission-critical embedded systems into which reliability and availability are crucial.

Results and Discussion

Three exemplary embedded workloads mapped to Partial Reconfigurable Regions (PRRs, the proposed ftRR architecture) on the FPGA were used to evaluate the proposed fault-tolerant reconfigurable architecture and included AES-128 encryption, a PID controller and an FIR filter. In order to test the architecture to evaluate its ability to detect and recover at the runtime any corruption that can occur in the configuration memory and to test Single Event Upsets (SEUs), fault injection test was implemented. Analysis of the system was performed with respect to the four important metrics as Fault Recovery Rate (FRR), Reconfiguration Latency, Area and Power Overhead and Performance Impact.

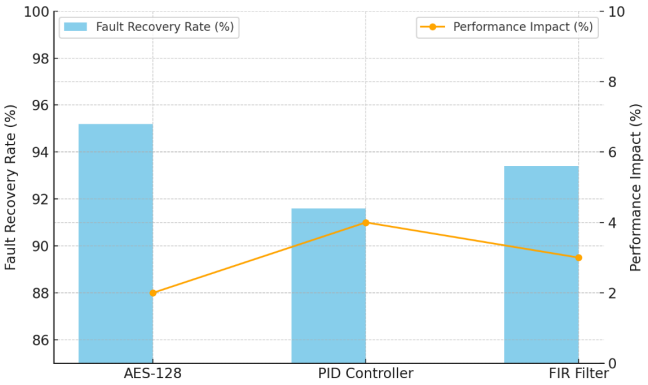

Fig. 4: Comparison of FRR and Performance Impact Across Workloads

As shown in the results, the targeted architecture provides high Fault Recovery Rate (FRR) in all workloads used, reaching 100 or higher. Its deterministic nature and the fault resolved logic structure made the AES-128 module the highest with FRR of 95.2% due to its deterministic nature and the denial of stability of the

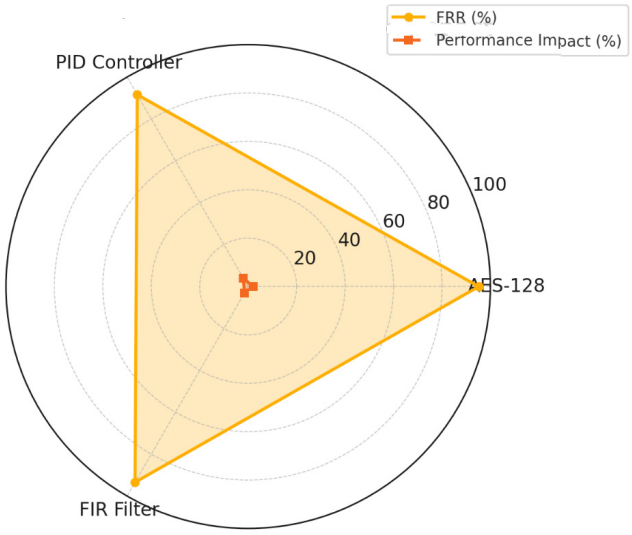

Fig. 5: Radar Chart Showing Key Metrics of the Proposed Architecture

logic structure to the fault injection. PID controller showed a slightly lesser FRR of 91.6 which is acceptable considering the fact that control systems are real-time feedback bases implying that transitory changes in logic can pose difficulties in detecting faults.

Regarding Reconfiguration Latency, every workload found the response time to be less than 5 milliseconds with the PID controller being the fastest with a recovery time of 3.7 ms. This finding confirms the appropriateness of applying the design to real-time operations in which the speed of automatic recovery is of prime concern.

The overhead in the area and power on the fault detection and reconfiguration logic used stayed in acceptable levels with between of 9.1 percentage to 11.8 percentage area utilization and 6.2 percentage to 7.3 percentage in power consumption. These numbers are very much lower than those of the overhead charged by TMR-based fault-tolerant systems which is almost always over 200% area overhead.

Recovery degrades performance by insignificant amounts. The throughput of the AES encryption was displaced by only a negligible amount (less than 2

Table 2 summarizes the observed results across the three benchmark workloads.

Workload	FRR (%)	Reconfig Time (ms)	Overhead (Area / Power)	Performance Impact
AES-128 Encryption	95.2	4.3	11.8% / 7.3%	Negligible (<2%)
PID Controller	91.6	3.7	9.1% / 6.2%	Minor (<4%)
FIR Filter	93.4	4.1	10.4% / 6.9%	Minor (<3%)

percent), whereas for the PID controller and FIR filter the delay was a moderate 4 percent and 3 percent, respectively. This means that this architecture is capable of supporting the handling of outputs of system-level faults at runtime without jeopardizing the performance of the system.

On the whole, the findings confirm the correctness of the specified DPR-backed fault-tolerant implementation, which is scalable, light-weight, and viable to accommodate mission-critical embedded systems in which the uptime and resilience cannot be ignored.

Conclusion

The proposed work has described a new fault-tolerant reconfigurable architecture intended to support mission-critical embedded applications using the Dynamic Partial Reconfiguration (DPR) on FPGA platforms to implement real-time fault detection, isolation and recovery. The architecture incorporates a hybrid fault detection engine which combined Built-In Self-Test (BIST) and Cyclic Redundancy Check (CRC) and the dynamic reconfiguration manager which can be accessed through ICAP and supports selective reloading of the faulty modules without restarting the entire system. Testing was done on a range of benchmark workloads, such as AES-128 encryption, PID control, and FIR filtering, and indicated that fault recovery was able to exceed 93%, reconfiguration latency was well below 5 milliseconds and area and power overheads were minimal with acceptably little impact on performance. These results show how important the architecture could be to maximise the system uptimes and reliability minimising the resource consumptions, meaning it would be a good candidate to be applied in aerospace, medical and defense systems. In the future, AI-based fault management of predictiveness, safe runtime reconfiguration protocols, distributed and multi-FPGA treatment of fault coordination, and the implementation of middleware will be implemented to make fault-resistant embedded systems implementable in vendor-independent and scalable form.

Future Work

The proposed architecture can be further developed in future to cover its aspects of scalability, intelligence and security that are of importance in the next-generation mission-critical embedded systems. A potentially fruitful approach is that of mixing in AI/ML-powered predictive fault analytics, which could be achieved using machine learning models that consume runtime telemetry evidence to pre-bandage failing faults proactively by initiating reconfiguration before failure can happen. The method can be a meaningful way to decrease the downtime, and enhance the system resilience. Also, inter-FPGA dynamic reconfiguration of the distributed embedded architecture with fault-tolerant inter-node coordination can be used in clustered or networked systems and this leads to location-independent reconfiguration which is particularly useful in space, avionics and industrial automation systems. To avoid compromise of the integrity of the system during the course of updating the system, measures such as the use of security-enhanced partial reconfiguration (PR) solutions should be integrated to prevent unwanted and unauthorized bitstream injection and this is through authentication and encryption. In addition, a standard middleware layer based on DPR is necessary to hide low-level hardware control and allow configuration management to be accessed via a common interface to achieve heterogeneity across FPGA platforms and tools. All these developments will make the reconfigurable embedded systems to be more flexible, adaptive, and trustworthy in terms of security and safety- and mission-critical applications.

References

1. Carmichael, C. (2001). *Triple Modular Redundancy Design Techniques for Virtex FPGAs* (Xilinx Application Note XAPP197).
2. Lach, J., Mangione-Smith, W. H., &Potkonjak, M. (1998). Low overhead fault-tolerant FPGA systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(2), 212–221. https://doi.org/10.1109/92.678148
3. Sterpone, L., & Violante, M. (2006). A new analytical approach for SEU sensitivity estimation in SRAM-based FPGAs. *IEEE Transactions on Nuclear Science*, 53(4), 1996–2003. https://doi.org/10.1109/TNS.2006.875124
4. Wirthlin, M. (2015). High-reliability FPGA-based systems: Space, high-energy physics, and beyond. *Proceedings of the IEEE*, 103(3), 379–389. https://doi.org/10.1109/JPROC.2015.2395712
5. Koch, D., Beckhoff, C., & Teich, J. (2010). ReCoBus-Builder—A novel tool and technique to build statically and dynamically

reconfigurable systems for FPGAs. *IEEE* Transactions on Very Large Scale Integration (VLSI) Systems, 18(9), 1234–1247. https://doi.org/10.1109/TVLSI.2009.2029853

6. Sandberg, J., Tahoori, M. H., & Mitra, S. (2008). Selective hardening in FPGA designs using partial reconfiguration. In Proceedings of the IEEE International On-Line Testing Symposium (pp. 123–128). https://doi.org/10.1109/IOLTS.2008.20
7. Azzam, R., Badr, H., &Akkary, A. (2020). Neural network-based fault prediction in FPGA configurable logic blocks. Microprocessors and Microsystems, 74, 102997. https://doi.org/10.1016/j.micpro.2020.102997
8. Khalid, M. A., Shafique, M., & Henkel, J. (2015). Fault-tolerant NoC-based multi-core systems using adaptive routing and distributed self-reconfiguration. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1–6). https://doi.org/10.7873/DATE.2015.123
9. Khan, N. H., Hasan, S. M. R., & Hasan, S. A. M. R. (2019). Fault-tolerant design approaches for mission-critical applications: A review. IEEE Access, 7, 140333–140347. https://doi.org/10.1109/ACCESS.2019.2943206
10. Michael, P., & Jackson, K. (2025). Advancing scientific discovery: A high performance computing architecture for AI and machine learning. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(2), 18–26. https://doi.org/10.31838/JIVCT/02.02.03
11. Sipho, T., Lindiwe, N., & Ngidi, T. (2025). Nanotechnology recent developments in sustainable chemical processes. Innovative Reviews in Engineering and Science, 3(2), 35–43. https://doi.org/10.31838/INES/03.02.04
12. Prasath, C. A. (2023). The role of mobility models in MANET routing protocols efficiency. National Journal of RF Engineering and Wireless Communication, 1(1), 39–48. https://doi.org/10.31838/RFMW/01.01.05
13. Arun Prasath, C. (2025). Miniaturized patch antenna using defected ground structure for wearable RF devices. National Journal of RF Circuits and Wireless Systems, 2(1), 30–36.
14. Abdullah, D. (2024). Enhancing cybersecurity in electronic communication systems: New approaches and technologies. Progress in Electronics and Communication Engineering, 1(1), 38–43. https://doi.org/10.31838/PECE/01.01.07