SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 2, No. 3, 2025 (pp. 27-36)
RESEARCH ARTICLE ECEJOURNALS.IN

Fault Detection and Correction for Advancing Reliability in Reconfigurable Hardware for Critical Applications

Jaanus Aare Tamm[1], Evelin Kärt Laanemets[2], Andres Peeter Siim[3]*

[1-3]Department of Computer Systems, Tallinn University of Technology, 12618 Tallinn, Estonia

Keywords:
Critical Applications;
Fault Detection;
Fault Tolerance;
Reconfigurable Hardware;
Reliability Enhancement;
System Integrity

Corresponding Author Email:
andpe.siim@taltech.ee

DOI: 10.31838/RCC/02.03.04

Received : 17.07.25
Revised : 26.09.25
Accepted : 31.10.25

Abstract

For critical applications, where system failures can result in severe consequences, the reliability of hardware components is of primal importance. Field Programmable Gate Arrays (FPGAs) have become common reconfigurable hardware in these high stakes environments due to their flexibility and performance advantages. Specifically in a radiation rich environment, the susceptibility of these devices to faults necessitates robust fault detection and correction techniques. In this article, we explore some of the most fundamental approaches used to improve reliability of reconfigurable hardware in high reliability application. While learning how to mitigate faults in FPGAs, we will follow errors from the different sources, how different device architectures respond different ways to them, and programs that help counteract the possibility of this failure. On the single event upset (SEU) to the more complex error patterns, we will explore how engineers are driving the limits of reliability in harsh operating conditions. The discussion of user inserted and embedded techniques for mitigation will explore both but focus on presenting a comprehensive current state of the art on fault tolerant reconfigurable systems. We explore the details of error detection and correction, and how these technologies are helping to build reliable computing for aerospace, defense, and other critical sectors where failure means loss of life.

How to cite this article: Tamm JA, Laanemets EK, Siim AP (2025). Fault Detection and Correction for Advancing Reliability in Reconfigurable Hardware for Critical Applications. SCCTS Transactions on Reconfigurable Computing, Vol. 2, No. 3, 2025, 27-36

A Definitive Guide to Single Event Upsets: Understanding in Digital Devices

Single Event Upsets (SEUs) are a serious threat to operational integrity of digital systems deployed in space or other areas of radiation intense environments. High energy particles strike sensitive regions of semiconductor devices such as the memory element or logic circuit and change their state to these transient errors. SEUs are temporary disturbances of the device that can result in data corruption or unexpected system behavior and do not represent actual damage to the device. This SEU mechanism is based on the fact the charged particle will leave an ionization trail through the semiconductor material. This ionization can create a charge that, if large enough, may turn a memory cell on or off or create a transient pulse in combinational logic. When memory element such as flip flop or SRAM cell is used as an SEU can directly alters stored data. A Single Event Transient (SET) may be generated for combinational logic, which, if captured by sequential logic, may cause an erroneous state change.[1-4]

However, it is important that SEUs be decoupled from the effect of Total Ionizing Dose (TID), another radiation induced effect. TID is a cumulative damage from long term exposure to ionizing radiation whereas SEUs are instantaneous events from a single particle strike. This distinction is critical because mitigation strategies to address these two forms of radiation effects must be very different. The effects of SEUs on digital systems can range from very limited to potentially catastrophic, and are somewhat dependent on the component and the architecture of the system.

An SEU may cause a minor glitch, which either is not noticed or is corrected automatically by existing error handling mechanisms. But in critical applications even a single bit flip can cause system malfunction, data loss, or in worst case scenario catastrophic failure. The goal is to either not create SEUs or detect and mitigate them when they do happen very quickly. In this topic we will showcase these strategies in detail investigating how they operate on particular classes of reconfigurable hardware and how effective they are in ensuring system reliability under radiation induced errors.[5-8]

FPGA Configuration Vulnerabilities and Their Mitigation Strategies

As they can be reprogrammed, FPGAs have become indispensable in many critical applications. This brings both good and bad to the FPGAs: their programmable nature has unique attacking surface, especially in the configuration memory used by it to determine what the FPGA can do. It is important to understand these vulnerabilities and apply appropriate mitigation strategies to providing the reliability of an FPGA based system in harsh environments. An FPGA's configuration memory consists essentially of a large array of bits that specify which pins will route to which pins, which logic will execute, and which elements in the logic will be combined. These configuration bits are particularly vulnerable to SEUs in any FPGA based on SRAM technology. A single bit flip in the configuration memory can have incredible impact on the architecture after much discussion with the Lattice employee. The severity of a configuration upset is depended on the FPGA technology being used. For example, antifuse based FPGAs have a configuration that is physically programmed, which resists radiation induced changes. One hand, SRAM based FPGAs have more flexibility, but are much less stable from the upset environment. FPGAs based on flash are between the extremes, with radiation tolerance slightly better than SRAM, but not as good as antifuse technology (Table 1).[9-12]

Configuration Scrubbing: This technique reads back the configuration memory regularly and checks that the backed off copy and the actual value are the same, and corrects any that are different. In some situations, scrubbing can be operated continuously, and in others it can be operated at set intervals, based on the system and upset rate requirements. Triple Modular Redundancy (TMR): TMR duplicates critical portions of the design by triplicating the logic, and uses majority voting to mask any error in a specific copy of the redundant logic. The principle of this idea can be applied at various levels (from the individual flipflop level to the functional block level). Error Detection and Correction (EDAC) Codes: Automatic detection and correction of single bit errors, with detection of multi bit errors, are possible if EDAC codes are implemented in the configuration memory. Partial Reconfiguration: Partially reconfigurable devices are supported by some advanced FPGAs which allow some sections of the device to be reprogrammed without bringing the rest of the device into operation. In this case, it can be used to more tightly and efficiently resolve configuration errors. Radiation-Hardened FPGAs: Special-purpose radiation hardened FPGAs are available for the most critical applications. Both these devices employ various hardening components at silicon level to minimize the possibility of SEUs in configuration and user memory (Figure 1).[13-15]

Table 1: Strategies for Enhancing System Reliability

Strategy	Rationale
Redundant Component Design	Redundant components help to ensure that if one component fails, another can take over, minimizing the impact of faults on system performance.
Continuous Monitoring	Continuous monitoring allows real-time detection of system malfunctions, enabling early fault detection and immediate corrective actions.
Signature-based Detection	Signature-based detection uses predefined signatures or patterns to identify abnormal behavior, aiding in the detection of specific faults.
Time-Triggered Monitoring	Time-triggered monitoring schedules regular checks based on time intervals, ensuring that faults are detected even if they occur intermittently.
Built-In Self-Testing	Built-In Self-Testing periodically checks hardware integrity, ensuring that faults are detected automatically during normal operation without external tests.
Health Monitoring Frameworks	Health monitoring frameworks track the overall condition of hardware components, allowing proactive maintenance and reducing the chances of failure.

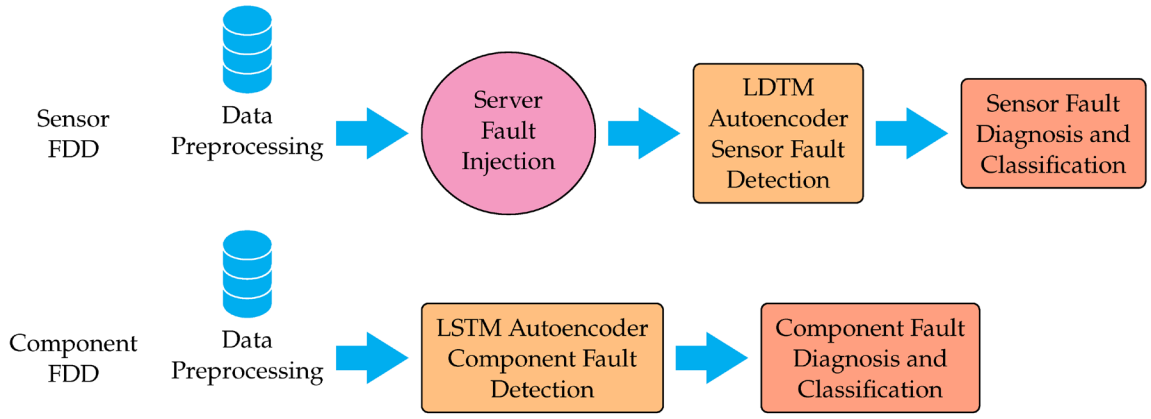

Fig. 1: FPGA Configuration Vulnerabilities and Their Mitigation Strategies

Keep in mind that while these techniques can considerably increase the reliability of FPGA based systems, they typically entail a tradeoff in performance, power consumption and resource utilization. However, when implementing mitigation strategies, designers must bear in mind the particular requirements of their application and the environment in which the application is supposed to operate. In addition, the effectiveness of these techniques can depend on the implementation of the design on the FPGA architecture and in particular, its general characteristics. For example, some routing configurations may be more vulnerably to SEUs, and some mitigation techniques may work for different types of circuit. For this reason, a complete mitigation strategy requires a complete understanding of the FPGA architecture and of the designed structure. Challenges of configuration protection will grow in the future as FPGA technology ever more develops more complex architectures, smaller feature sizes. Yet recent work in this area shows much promise, with new and more efficient mitigation techniques coming into play to meet the growing requirements of critical applications in difficult environments.[16-19]

FPGAs Data Path Vulnerabilities

In FPGA-based systems, SEUs and SETs on the configuration memory itself are a major concern, but the user defined logic and memory elements of the data path are also prone to these upsets. These vulnerabilities can introduce errors in computation, data corruption, or cause an unexpected state transition in sequential logic. Data path vulnerabilities need to be understood and mitigated to guarantee overall reliability of FPGA designs for use in critical applications. Most of the data path in an FPGA is composed of combinational logic implemented in a set of Look Up Tables (LUTs), followed by sequential elements (e.g. flip flops) and a set of interconnect resources that route signals between these elements. Each of these elements can be affected by radiation-induced events in different ways.[20]-24]

Combinational Logic: Combinational logic SETs may produce transient pulses that propagate and capture through a sequential element where it creates erroneous data. SET arising from the logical masking (possibly blocked by subsequent logic gates), electrical masking (attenuation due to electrical properties of the circuit), or temporal masking (missed coincident clock edge) determine the probability of an error caused by the SET. In addition, it must be emphasized that, just like any other technique, the effectiveness of these techniques might depend in particular on a particular FPGA architecture and the nature of the implemented design. As an example, the routing and layout of a design can make it more or less susceptible to SETs and more or less amenable to mitigation techniques. The future of data path protection with FPGA technology due to smaller feature sizes and more complex structuring is likely to become more of a challenge as it continues to advance. While there is still a lot of research to come in this field, new and more efficient techniques to mitigate them are also in the works, and seem likely to keep up with the rising asking of critical applications in hostile environments.[25-29]

Critical Applications Fail-Safe Strategies

In such critical applications with severe systemic failure consequences, robust fail safe strategies need

to be implemented. The strategy developed in these considers that even under the circumstance of a fault the system should still operate safely or will move to a known safe state. But when used on reconfigurable hardware such as FPGAs, fail safe strategies must both account for the unique vulnerabilities of these devices as well as be specific to the application at hand. Consequently, fail safe strategies for FPGAs in critical applications tend to include error detection, error correction, and system level recovery mechanisms. A resilient system against faults of various types is sought, including SEUs, SETs, and more complex modes. An extra cost in terms of bits and TTL logic for encoding, decoding, and so forth.• An FPGA monitoring circuit is used to trigger a reset or failover, if normal operation terminates. components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks.[30-33]

Table 2: Techniques for Reconfigurable Hardware Systems

Technique	Solution
Fault Isolation and Recovery	Fault isolation and recovery isolate the faulty component, rerouting tasks to healthy modules or systems to maintain continuous operations.
Dynamic Reconfiguration	Dynamic reconfiguration allows the hardware to change its configuration during runtime to avoid faults by adapting to new conditions.
Error Masking	Error masking involves hiding errors from the system, allowing operations to continue normally even when some faults are detected.
Reparative Repair Mechanisms	Reparative repair mechanisms automatically fix detected errors by reconfiguring hardware logic or reloading faulty components with updated configurations.
Redundant Computation	Redundant computation executes the same task in multiple components simultaneously, ensuring that results are still obtained if one unit fails.
Resilient Data Flow	Resilient data flow ensures that data continues to move through the system without loss or corruption, even in the presence of faults.

Avoidance of false triggers and recovery was necessary with careful design. In this approach two identical cores run the same operations in parallel and compare their outputs on discrepancies. Components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks. With modern high performance designs synchronization between cores can become very complex. With more states and transitions, implementing state machines handling unexpected cases. Marker state encoding schemes which can detect illegal state transitions.is technique involves triplicating critical components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks. Regular periodic scrubbing of nonvolatile memory looking for and correcting SEUs. Multiple copies of configuration bitstreams and ability to rapidly reload initial configuration. Components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks. In terms of memory overhead and (brief) interruptions during reconfiguration. Break down the system into critical and non critical sections and provide more robust protection to critical areas. Avoid faults from an area in the machine to propagate to other areas. Ing critical components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks. Critical sections (and therefore fault propagation paths) needed to be identified by careful system analysis. A system needs to be designed so that the system continues operation with reduced functionality in the case of partial failures.• We implement prioritization schemes to maintain the most critical function. Involves triplicating critical components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks (Figure 2).

It needs to be carefully planned and has some additional logic needed to manage degraded modes. Applicable minimum requirements from an application, such as acceptable downtime, error rates and recovery times. Where the target environment is expected to be, and it's characteristics such as expected radiation levels or other sources of interference. FPGA platform chosen, capabilities and limitations including available resources and built-in reliability features. The external components or any redundant systems in the overall system architecture. Replicating critical components and using majority voting to mask errors. TMR can be applied at various levels, from individual

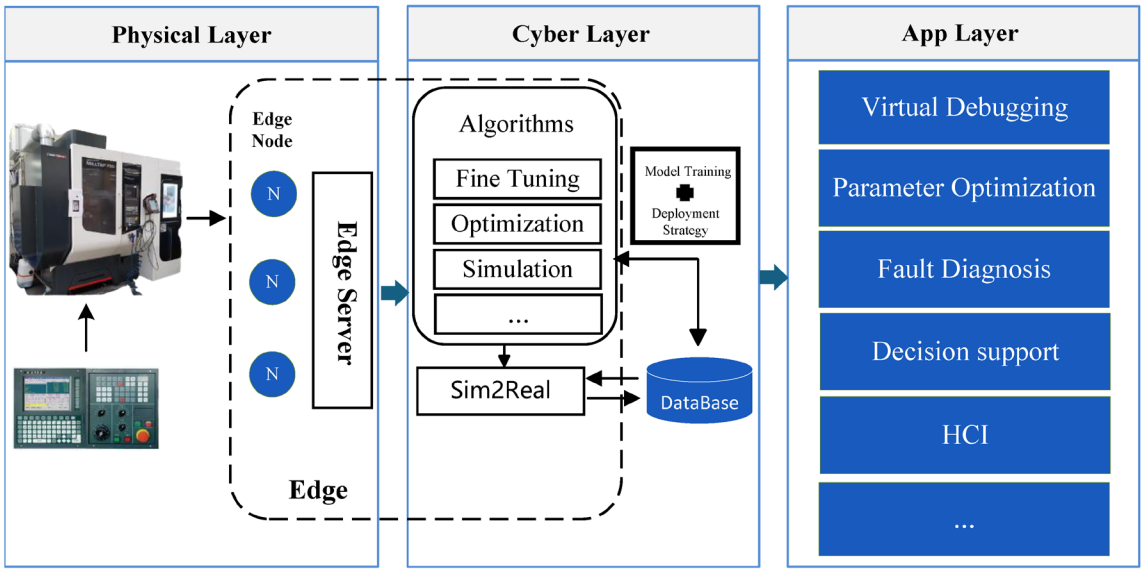

Fig. 2: Critical Applications Fail-Safe Strategies

flip-flops to entire functional blocks. With the increasing complexity of FPGA design and ever more challenging environments in which these devices are placed in service, there continues to be an active area of research and innovation in developing effective fail safe strategies. Finally, promising techniques for improving the reliability of FPGA-based systems in critical applications include emerging techniques that operate on error correction (adaptive error correction) and machine learning based fault prediction.[34-36]

Dual Redundancy and Cold Sparing Techniques are used

Dual redundancy and cold sparing are two conceptually different means of improving system's reliability in the context of fault tolerant design for critical applications. Although the implementations of these techniques differ and their capabilities are different, both are intended to provide some degree of protection from failures that may jeopardize system integrity or performance. Dual Redundancy: Dual redundancy refers to the use of two (or more) identical systems, or components, operating in parallel. The idea is to detect errors by comparing the output of one system to the other. Two of these identical processing units execute the same instructions in parallel. But outputs are compared cycle by cycle, looking for differences. Error detection with high confidence thanks to cycle accurate comparison. Complex synchronization of units, particularly for high performance systems. Careful design is required in handling of asynchronous events (e.g. Interrupts). Doesn't intrinsically correct for errors. Like strict lockstep, but loosely timed two unit. Instead of at each cycle, comparison is done at certain checkpoints. It is more flexible than strict lockstep.

It is easier to implement in complex systems. Can live with a little performance variation between units. Error detection may not occur until the subsequent checkpoint. Non trivial thing is to define appropriate checkpoints. Two independent modules perform the same function and their outputs are compared. The internal operation can be different as long as the results at the end are same. Internal operations can differ as long as results from each round are the same. It's more flexible: it can use different hardware or even different algorithms. It can detect a wide range of error including those caused by hardware fault or software bugging critical components and using majority voting to mask errors. TMR can be applied at various levels, from individual flip-flops to entire functional blocks. While dual redundancy and cold sparing can significantly enhance system reliability, they are often used in conjunction with other techniques such as error correction codes, watchdog timers, and system-level health monitoring. The choice between these approaches, or the decision to use them in combination, depends on factors such as the specific reliability requirements, power constraints, available resources, and the nature of the expected failure modes in the target application.

As FPGA technologies continue to evolve, offering larger devices with more advanced power

management and partial reconfiguration capabilities, the implementation of these redundancy techniques is likely to become more sophisticated and efficient. This evolution will enable the development of even more reliable systems for critical applications in challenging environments. Triple Modular Redundancy (TMR) is a powerful fault-tolerance technique widely used in critical systems to mitigate the effects of Single Event Upsets (SEUs) and other transient faults. In the context of FPGA designs, TMR involves triplicating logic and using majority voting to mask errors. While the basic concept of TMR is straightforward, its implementation in FPGAs can take various forms, each with its own strengths and trade-offs. Best suited for: Systems with the highest reliability requirements on the markets of mission critical systems. A TMR system relies on voters; and voters are by themselves reliable. Techniques of hardened voter designs or voter triplication maybe used. Very importantly, they require optimal voter placement to balance protection, resource usage, and timing performance. TMR errors accumulate simply but does not correct them. This accumulation may be prevented by implementing the feedback from voters to correct upset flip-flops. Before creating single points of failure in feedback paths, care must be taken. The DTMR and GTMR issues are managing multiple clock domains. Care has to be taken with techniques like clock domain crossing (CDC) synchronization. The current designs include using a single clock source and triplicated clock trees for synchronization simplification. Since typically TMR increases resource usage by more than 200% because of logic triplication and additional voting logic, TMR circuits and architectures are required to minimize resource usage, while increasing fault coverage for turbines. They can become difficult to deal with if you need to fit into available FPGA devices. Partial TMR strategies are used to protect only the most critical parts of a design.[37]

While dual redundancy and cold sparing can significantly enhance system reliability, they are often used in conjunction with other techniques such as error correction codes, watchdog timers, and system-level health monitoring. The choice between these approaches, or the decision to use them in combination, depends on factors such as the specific reliability requirements, power constraints, available resources, and the nature of the expected failure modes in the target application. As FPGA technologies continue to evolve, offering larger devices with more advanced power management and partial reconfiguration capabilities, the implementation of these redundancy techniques is likely to become more sophisticated and efficient. This evolution will enable the development of even more reliable systems for critical applications in challenging environments. Triple Modular Redundancy (TMR) is a powerful fault-tolerance technique widely used in critical systems to mitigate the effects of Single Event Upsets (SEUs) and other transient faults. In the context of FPGA designs, TMR involves triplicating logic and using majority voting to mask errors. While the basic concept of TMR is straightforward, its implementation in FPGAs can take various forms, each with its own strengths and trade-offs (Figure 3).

However, as FPGA technology advances, with larger devices and increasingly elaborate features, such as built in error correction, or hardened TMR cells, implementation of TMR is most likely to become increasingly efficient and effective. And, fundamental tradeoffs between reliability, resource usage, and performance will continue to influence the field, driving designers to develop increasingly complex TMR strategies ranging from the most basic to those optimized to the most critical application [38].

State Machine Design For Fail Safe Systems

Many digital systems based on state machines, e.g., controllers and coordinators, control critical and often crucial operations, and the state machines involved need to be robust. This is particularly important in the case of FPGAs used in high reliability applications, where state machines must behave predictably and safely in the presence of SEUs and other faults. Fail safe state machine design is a set of techniques of detecting errors, preventing an illegal state transition and ensuring system integrity even in the case of unexpected events.

Checkpoint and Rollback: Save state periodically and roll back to the last known good state on error detection. All inputs to the state machine clock are synchronized to prevent metastability issues. Designs multi stage synchronizers for inputs coming from different clock domains. Minimize errors caused by changes in value using Gray code for multi bit inputs. The completeness and correctness of the state machine design is verified using formal methods. Confirm that all legal states and transitions exist and is no illegal transitions possible. A scan chain implementation is

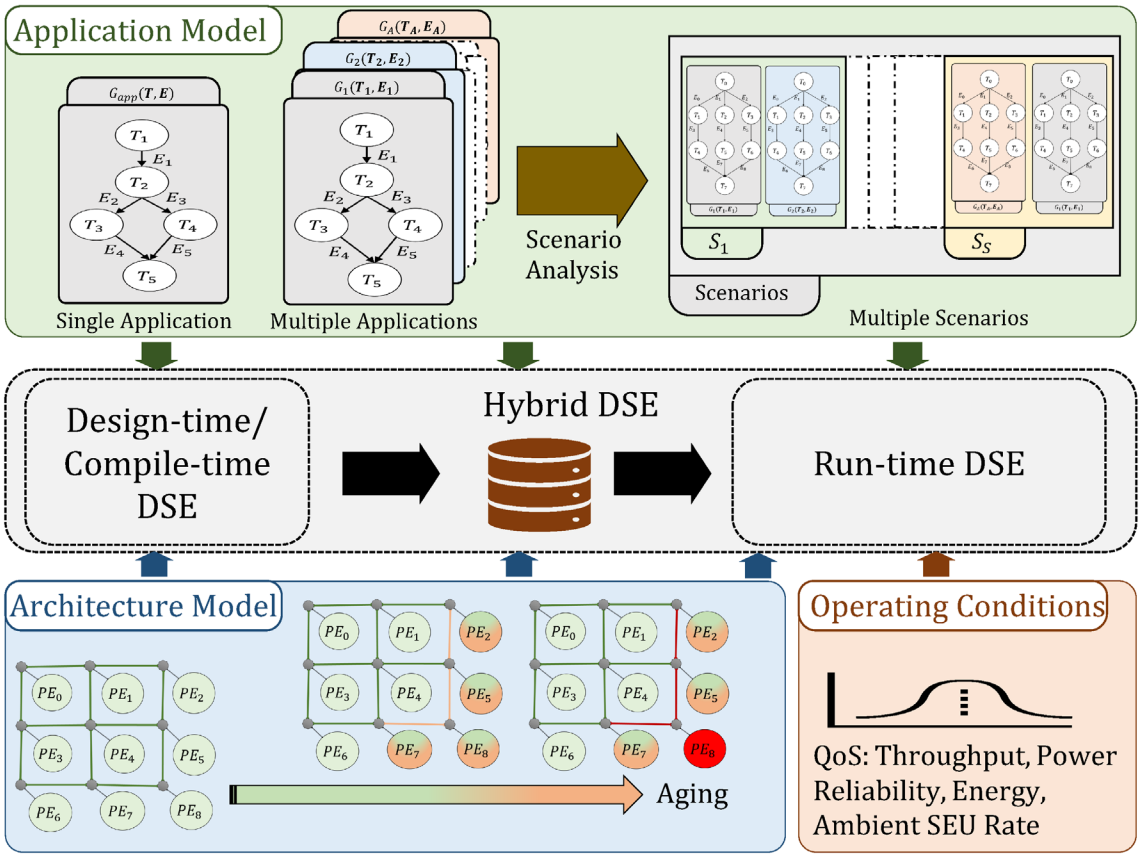

Fig. 3: Dual Redundancy and Cold Sparing Techniques are used

also achieved to improve testability and diagnostics. A way to add observability points to monitor internal state machine behavior. Fault injection design for error detection and recovery mechanism validation. Upper, more complex encodings and error detect logic increases LUT and FF usage. There must be balance between reliability and resource efficiency. Faster maximum clock frequency is susceptible to additional logic for error detection and handling. Timing analysis may be careful enough to need pipelining, or perhaps even pipelining is necessary. The configuration bits defining the state machine logic themselves are SEU susceptible in SRAM based FPGAs. Can require additional protection of the form of scrubbing configuration or by use of hardened logic elements.

For designs using partial reconfiguration, be sure that state machine protective mechanisms are friendly to reconfigurable regions. Power consumption is generally increased over basic state machine implementations. Selective application of protection techniques can be required for power sensitive applications. If possible, using any built in error detection or correction features within the FPGA architecture. In event of having available hardened memory elements or specialized state machine encoding support consider. Among the FPGA design tools, some provide automatic state machine protection features. They also cover the strengths and limitations of these tools, and how they fit in alongside custom protection strategies. totality of the application. As FPGA technologies continue to evolve, offering larger devices with more advanced features such as hardened TMR cells or built-in error correction, the implementation of TMR is likely to become more efficient and effective. However, the fundamental trade-offs between reliability, resource usage, and performance will continue to drive innovation in this field, pushing designers to develop ever more sophisticated TMR strategies tailored to the unique requirements of each critical application.

While most FPGA technologies will continue to evolve, we can expect that newer FPGA technologies will have more built in support of fail-safe state machine design, with hardened voting logic, better error detection circuits and more sophisticated configuration protection mechanisms. Yet, the

principles of redundancy, error detection, and safe recovery will most likely continue to form the core of designs of robust state machines for critical applications. Challenge: Behavior verification across several periods including the cumulative effect of multiple faults. Increased fault injection rates, i.e., accelerated life testing. Statistical modeling and analysis, and Verification of fault-tolerant FPGA designs presents unique challenges that go beyond traditional functional verification. The goal is not only to ensure that the design operates correctly under normal conditions but also to verify that it behaves as expected in the presence of faults. This requires a comprehensive approach that combines various verification techniques and tools.

Conclusion

State machines are fundamental components in many digital systems, often controlling critical operations and decision-making processes. In the context of FPGAs used in high-reliability applications, ensuring that state machines behave predictably and safely in the presence of SEUs and other faults is crucial. Fail-safe state machine design encompasses a range of techniques aimed at detecting errors, preventing illegal state transitions, and maintaining system integrity even in the face of unexpected events. Implement a global reset that brings the state machine to a known, safe initial state. Define a sequence of states that the machine transitions through to recover from errors. Periodically save the state and roll back to the last known good state upon error detection. Reliability, performance and resource utilization trade within an implementation of fail safe state machines. The particular techniques employed must be chosen for application and target FPGA characteristics. In most critical applications, a combination of techniques is used for defense in depth against several failure modes.

References:

1. Gangadhar, C., Moutteyan, M., Vallabhuni, R. R., Vijayan, V. P., Sharma, N., & Theivadas, R. (2023). Analysis of optimization algorithms for stability and convergence for natural language processing using deep learning algorithms. *Measurement: Sensors*, *27*, 100784.
2. Babu, D. V., Basha, S. A., Kavitha, D., Nisha, A. S. A., Vallabhuni, R. R., & Radha, N. (2023). Digital code modulation-based MIMO system for underwater localization and navigation using MAP algorithm. *Soft Computing*, 1-9.
3. Selvam, L., Garg, S., Prasad, R. M., Qamar, S., Lakshmi, K. M., & Ratna, V. R. (2023). Collaborative autonomous system based wireless security in signal processing using deep learning techniques. *Optik*, *272*, 170313.
4. Jiang, S., Ma, Z., Zeng, X., Xu, C., Zhang, M., Zhang, C., & Liu, Y. (2020, July). Scylla: Qoe-aware continuous mobile vision with fpga-based dynamic deep neural network reconfiguration. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications* (pp. 1369-1378). IEEE.
5. Çambay, V. Y., Uçar, A., & Arserim, M. A. (2019, September). Object detection on FPGAs and GPUs by using accelerated deep learning. In *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)* (pp. 1-5). IEEE.
6. Nakahara, H., & Sasao, T. (2018, May). A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector. In *2018 IEEE international symposium on circuits and systems (ISCAS)* (pp. 1-5). IEEE.
7. Marcel, S., & Rodriguez, Y. (2010, October). Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia* (pp. 1485-1488).
8. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.
9. Bernardeschi, C., Cassano, L., Do menici, A., & Sterpone, L. (2014). ASSESS: A simulator of soft errors in the configuration memory of SRAM-based FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *33*(9), 1342-1355.
10. Karwatzki, D., von Hofen, M., Baruschka, L., & Mertens, A. (2014, October). Operation of modular multilevel matrix converters with failed branches. In *IECON 2014-40th Annual Conference of the IEEE Industrial Electronics Society* (pp. 1650-1656). IEEE.
11. Fan, B., Wang, K., Zheng, Z., Xu, L., & Li, Y. (2017). Optimized branch current control of modular multilevel matrix converters under branch fault conditions. *IEEE Transactions on Power Electronics*, *33*(6), 4578-4583.
12. Cimpoesu, E. M., Ciubotaru, B. D., & Stefanoiu, D. (2013, May). Fault detection and diagnosis using parameter estimation with recursive least squares. In *2013 19th International Conference on Control Systems and Computer Science* (pp. 18-23). IEEE.
13. Geist, A., Brewer, C., Davis, M., Franconi, N., Heyward, S., Wise, T., ... & Flatley, T. (2019). SpaceCube v3. 0 NASA next-generation high-performance processor for science applications.
14. George, A. D., & Wilson, C. M. (2018). Onboard processing with hybrid and reconfigurable computing on small satellites. *Proceedings of the IEEE*, *106*(3), 458-470.

15. Guo, K., Zeng, S., Yu, J., Wang, Y., & Yang, H. (2019). [DL] A survey of FPGA-based neural network inference accelerators. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *12*(1), 1-26.
16. Tzimpragos, G., Kachris, C., Djordjevic, I. B., Cvijetic, M., Soudris, D., & Tomkos, I. (2014). A survey on FEC codes for 100 G and beyond optical networks. *IEEE Communications Surveys & Tutorials*, *18*(1), 209-221.
17. Arockia Bazil Raj, A., & Padmavathi, S. (2016). Statistical analysis of accurate prediction of local atmospheric optical attenuation with a new model according to weather together with beam wandering compensation system: a season-wise experimental investigation. *Journal of Modern Optics*, *63*(13), 1286-1296.
18. Kopeika, N., & Raj, A. A. B. (2023). Special Issue on "Optical and RF Atmospheric Propagation". *Sensors*, *23*(7), 3644.
19. Han, Q., Fan, M., Niu, L., & Quan, G. (2015, March). Energy minimization for fault tolerant scheduling of periodic fixed-priority applications on multiprocessor platforms. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 830-835). IEEE.
20. Salehi, M., Tavana, M. K., Rehman, S., Shafique, M., Ejlali, A., & Henkel, J. (2016). Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *24*(7), 2426-2437.
21. Zhu, D., & Aydin, H. (2006, November). Energy management for real-time embedded systems with reliability requirements. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design* (pp. 528-534).
22. Li, J., & Wang, Y. (2021, August). Transient fault tolerance on multicore processor in amp mode. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)* (pp. 332-337). IEEE.
23. Fuchs, C. M., Murillo, N. M., Plaat, A., van der Kouwe, E., Harsono, D., & Wang, P. (2018, December). Software-Defined Dependable Computing for Spacecraft. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 231-232). IEEE.
24. Benites, L. A., Benevenuti, F., De Oliveira, Á. B., Kastensmidt, F. L., Added, N., Aguiar, V. A., ... & Guazzelli, M. A. (2019). Reliability calculation with respect to functional failures induced by radiation in TMR arm cortex-M0 softcore embedded into SRAM-based FPGA. *IEEE Transactions on Nuclear Science*, *66*(7), 1433-1440.
25. Li, J., & Wang, Y. (2021, August). Transient fault tolerance on multicore processor in amp mode. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)* (pp. 332-337). IEEE.
26. Fuchs, C. M., Murillo, N. M., Plaat, A., van der Kouwe, E., Harsono, D., & Wang, P. (2018, December). Software-Defined Dependable Computing for Spacecraft. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 231-232). IEEE..
27. Benites, L. A., Benevenuti, F., De Oliveira, Á. B., Kastensmidt, F. L., Added, N., Aguiar, V. A., ... & Guazzelli, M. A. (2019). Reliability calculation with respect to functional failures induced by radiation in TMR arm cortex-M0 softcore embedded into SRAM-based FPGA. *IEEE Transactions on Nuclear Science*, *66*(7), 1433-1440.
28. Kastensmidt, F. L., Sterpone, L., Carro, L., & Reorda, M. S. (2005, March). On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Design, Automation and Test in Europe* (pp. 1290-1295). IEEE.
29. Keller, A. M., & Wirthlin, M. J. (2016). Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs. *IEEE Transactions on Nuclear Science*, *64*(1), 519-528.
30. Sterpone, L. (2008, June). A novel design flow for the performance optimization of fault tolerant circuits on SRAM-based FPGA's. In *2008 NASA/ESA Conference on Adaptive Hardware and Systems* (pp. 157-163). IEEE.
31. Siwakoti, Y. P., & Town, G. E. (2013, June). Design of FPGA-controlled power electronics and drives using MATLAB Simulink. In *2013 IEEE ECCE Asia Downunder* (pp. 571-577). IEEE.
32. Butler, R. W., & Johnson, S. C. (1995). *Techniques for modeling the reliability of fault-tolerant systems with the Markov state-space approach* (No. NAS 1.61: 1348).
33. Kumar, V., Singh, L. K., Singh, P., Singh, K. V., Maurya, A. K., & Tripathi, A. K. (2018). Parameter estimation for quantitative dependability analysis of safety-critical and control systems of NPP. *IEEE Transactions on Nuclear Science*, *65*(5), 1080-1090.
34. Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., & Burger, D. (2011, June). Dark silicon and the end of multicore scaling. In *Proceedings of the 38th annual international symposium on Computer architecture* (pp. 365-376).
35. Salamun, K., Pavić, I., Džapo, H., & Čuljak, I. (2023). Weakly Hard Real-Time Model for Control Systems: A Survey. *Sensors*, *23*(10), 4652.
36. Ranjbar, B., Nguyen, T. D., Ejlali, A., & Kumar, A. (2020). Power-aware runtime scheduler for mixed-criticality systems on multicore platform. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *40*(10), 2009-2023.
37. Lefèvre, S., Vasquez, D., & Laugier, C. (2014). A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, *1*, 1-14.
38. Wegener, M., Herrmann, F., Koch, L., Savelsberg, R., & Andert, J. (2021). Longitudinal vehicle motion prediction in urban settings with traffic light interaction. *IEEE Transactions on Intelligent Vehicles*, *8*(1), 204-215.
39. Muyanja, A., Nabende, P., Okunzi, J., & Kagarura, M. (2025). Metamaterials for revolutionizing modern appli-

cations and metasurfaces. Progress in Electronics and Communication Engineering, 2**(2), 21-30. https://doi.org/10.31838/PECE/02.02.03**

40. Surendar, A. (2024). Internet of medical things (IoMT): Challenges and innovations in embedded system design. *SCCTS Journal of Embedded Systems Design and Applications, 1*(1), 43-48. https://doi.org/10.31838/ESA/01.01.08
41. Kavitha, M. (2024). Environmental monitoring using IoT-based wireless sensor networks: A case study. *Journal of Wireless Sensor Networks and IoT, 1*(1), 50-55. https://doi.org/10.31838/WSNIOT/01.01.08
42. Danh, N. T. (2025). Advanced geotechnical engineering techniques. *Innovative Reviews in Engineering and Science, 2*(1), 22-33. https://doi.org/10.31838/INES/02.01.03
43. Kavitha, M. (2024). Energy-efficient algorithms for machine learning on embedded systems. *Journal of Integrated VLSI, Embedded and Computing Technologies, 1*(1), 16-20. https://doi.org/10.31838/JIVCT/01.01.04
44. Kavitha, M. (2023). Beamforming techniques for optimizing massive MIMO and spatial multiplexing. *National Journal of RF Engineering and Wireless Communication, 1*(1), 30-38. https://doi.org/10.31838/RFMW/01.01.04
45. Goyal, D., Hemrajani, N., & Paliwal, K. (2013). GPH algorithm: Improved CBC improved BIFID cipher symmetric key algorithm. *International Journal of Communication and Computer Technologies, 1*(2), 83-86. https://doi.org/10.31838/IJCCTS/01.02.03
46. Ibrahim, N., Rajalakshmi, N. R., & Hammadeh, K. (2024). A Novel Machine Learning Model for Early Detection of Advanced Persistent Threats Utilizing Semi-Synthetic Network Traffic Data. Journal of VLSI Circuits and Systems, 6(2), 31–39. https://doi.org/10.31838/jvcs/06.02.04
47. Maidanov, K., & Fratlin, H. (2023). Antennas and propagation of waves connecting the world wirelessly. *National Journal of Antennas and Propagation, 5*(1), 1–5.