SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 2, No. 1, 2025 (pp. 24-32)
RESEARCH ARTICLE ECEJOURNALS.IN

Fault-Tolerant Reconfigurable Computing Systems for High Performance Applications

Mohammad Alizadeh[1], Ali Esfahani Pour[2], Hossein Mahmoudian[3]*

[1-3]School of Electrical Engineering Iran University of Science and Technology Tehran, Iran

Keywords:
Fault Tolerance;
High Performance;
Reconfigurable Computing;
Reliability;
System Resilience;
Performance Optimization

Corresponding Author Email:
hosseinmah@elec.iust.ac.ir

DOI: 10.31838/RCC/02.01.04

Received : 27.08.24
Revised : 02.11.24
Accepted : 17.12.24

Abstract

Reconfigurable hardware architectures are transforming the realm of high performance computing. In the face of explosive computational demands (exponentially growing across a wide range of domains ranging from financial modeling to bioinformatics) the need for robust and adaptable computing systems is never higher. In this paper, it will discuss the design of some of the fault tolerant reconfigurable computing systems, the applications, the challenges and the new ideas in the cutting edge solution in high performance computation. Field Programmable Gate Arrays (FPGAs) based reconfigurable computing systems present a unique combination of flexibility and performance that is difficult to equal in traditional computing systems. These systems enable, by allowing hardware configuration to be changed dynamically, to obtain optimization to the task at hand resulting in performance and energy efficiency gains. While these systems are being deployed to mission critical applications, reliability and fault tolerance must be guaranteed. In this article, we conduct a thorough study of the core concepts related to the reconfigurable systems, their fault tolerant designs, innovative architectures developed to overcome the reliability problems, and the real world application where these systems are strongly influencing. This article seeks to explore a detailed perspective encompassing FPGA based fault detection and masking techniques, as well as the implications of the fault tolerant reconfigurable computing for high performance computing landscapes, from current state to future directions.

How to cite this article: Alizadeh M, Pour AE, Mahmoudian H (2025). Fault-Tolerant Reconfigurable Computing Systems for High Performance Applications. SCCTS Transactions on Reconfigurable Computing, Vol. 2, No. 1, 2025, 24-32

Reconfigurable Computing Architectures Understanding

The reconfigurable computing architectures mark an important paradigm shift in how we problem solve computationally. Unlike fixed hardware systems, reconfigurable platforms feature the unique capability to reconfigure its hardware structure depending on the corresponding algorithmic requirements. This flexibility is gained through the use of programmable logic devices, which usually result in Field Programmable Gate Arrays (FPGAs). The concept of hardware plasticity is the heart of reconfigurable computing. A configurable logic block (CLB) is an array of logic elements, such as flip flops and complex functions, circuitry to implement multiplexers, registers, latches, etc., interconnected by programmable routing resources. As an example of these CLBs, they are configurable to produce digital circuits ranging from simple logic gates to complex arithmetic units. Custom data paths are created using programmable interconnects, which result in custom hardware tailored to the needs of a particular application.

Reconfigurable architectures not only provide low cost, but also offer high performance since they can utilize parallelism to achieve high performance. These systems take advantage of the fact that by having multiple processing elements that can

operate concurrently, the overall performance can be substantially better than that of traditional sequential processors on a class of problems. In applications where data parallelism is high, or where custom data flow architectures are required, this parallelism is especially helpful.[1-4]

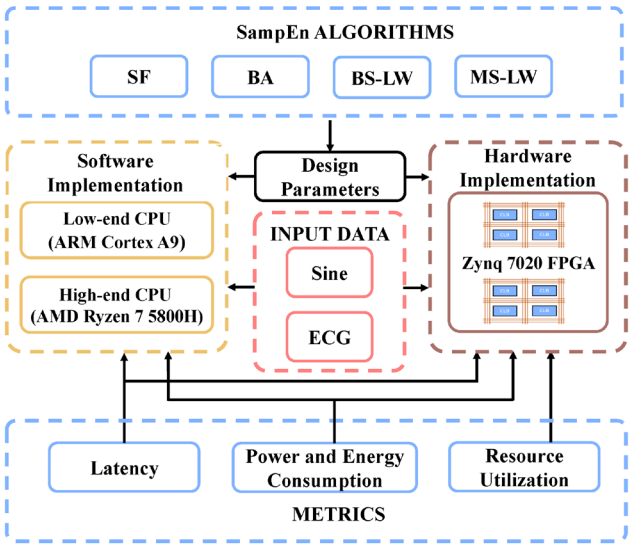

Fig. 1. Reconfigurable Computing Architectures Understanding

Runtime reconfiguration is yet another important aspect of reconfigurable computing. This feature serves the purpose of dynamic modification of the hardware configuration during the program execution, allowing the system to respond to altering its computational requirements. Runtime reconfiguration allows time multiplexing of hardware resources, time multiplexing of adaptive algorithms, and even self modifying circuits. But, of course, this comes at a price: reconfigurable architectures are flexible. FPGAs are programmable, therefore overhead is incurred in the area, power consumption, maximum operating frequency, compared to the Application Specific Integrated Circuits (ASICs). Choosing reconfigurable platforms for their applications, designers must carefully strike the balance between flexibility and performance. Yet, the benefits of reconfigurable computing have driven adoption in a spectrum of high performance applications. Having applications in domains ranging from signal processing and cryptography to bioinformatics and financial modeling, reconfigurable systems show their effectiveness in those which require high performance but also flexibility. With the increasing focus on the design of fault tolerant reconfigurable systems a few fundamental and unique characteristics and challenges for these flexible architectures must be kept in mind. As strong as reconfigurable computing is, these same features introduce new considerations for its reliability and fault tolerance [5]-[9].

Fault Tolerance Needs in High Performance Computing

Reconfigurable hardware is becoming a vital component for high performance computing (HPC) environments, which are facing demand constraints in computational power and energy efficiency. Given that these systems are being scaled up to solve more complex problems, the probability of faults occurring both before or during use also increases. Thus, it requires a sound fault tolerance approach to secure computations' reliability and integrity.

The need for fault tolerance in HPC systems stems from several factors:

Scale and Complexity: Modern HPC systems often consist of thousands of components interconnected. The probability of component failures increases linearly with the system size. Even with high quality components, there is simply too much in play in terms of elements over prolonged operation periods, thus the statistical inevitability of faults is a reality. Long-Running Computations: Among the applications of HPC there are many that involve simulating or analyzing something that can run for days or even weeks. With these long running jobs these single faults can invalidate results or resulted in lots of delay and wasted resources.

Critical Applications: The critical domains of weather forecasting, financial modeling, and scientific research often use HPC systems. Such applications are fault tolerant critical systems, and error in these applications may impact their users very significantly (Table 1).

Energy Considerations: It's important to keep in mind, as HPC systems continue to push performance to the limits, power consumption becomes a significant factor. Many fault tolerant designs can be designed to keep a high efficiency rate without wasting computational effort.

Data Integrity: Data on very large scales, or extremely valuable/dangerous data completes the metadata about such HPC applications. To minimize errors caused by human error and noisy measurements,

Table 1: Fault-Tolerance Mechanisms in Reconfigurable Computing Systems

Mechanism	Fault-Tolerance Strategy
Triple Modular Redundancy (TMR)	Triple Modular Redundancy (TMR) involves replicating critical components three times to ensure that the system can tolerate a single failure without affecting operation.
Dynamic Reconfiguration	Dynamic reconfiguration allows the system to replace faulty components on the fly, improving reliability by adapting hardware resources to changing conditions.
Error Detection and Correction Codes	Error detection and correction codes ensure that data transmitted within the system is verified and corrected, minimizing the risk of system malfunctions due to corrupted data.
Self-Healing Systems	Self-healing systems are designed to detect faults and automatically recover by rerouting tasks or bypassing the failed components to maintain continuous operation.
Redundant Processing Units	Redundant processing units offer backup components that take over tasks when the primary unit fails, ensuring uninterrupted operation of critical applications.
Checkpointing and Rollback	Checkpointing and rollback techniques periodically save system states so that in case of failure, the system can restore its state and continue operation from the last stable point.

fault-tolerant mechanisms are necessary to provide the integrity needed for this data in computing.

The issue of fault tolerance is more colored by the need for it with respect to reconfigurable computing. The programmable nature of FPGAs introduces unique challenges and opportunities for implementing fault-tolerant designs:

- **Configuration Errors:** Even the process of loading a new configuration onto an FPGA can themselves be prone to errors. It is critical to ensure the configuration bitstream integrity and to verify programming correctly.
- **Soft Errors:** Like other semiconductor devices, FPGAs are susceptible to the soft errors due to cosmic rays or other sources of ionizing radiation. Transient faults can corrupt the state of logic elements or memory cells, but the effect can be faulty logic elements or memory cells, or faulty computations.
- **Wear-out Effects:** Wear-out effects of FPGAs tend to generate permanent faults in some parts of the device over time. These long term reliability concerns must be included in fault tolerant designs.

Dynamic Reconfiguration: The capacity to reconfigure FPGAs at runtime provides the opportunity to leverage them as implementing platforms for new adaptive fault tolerance mechanisms. Nevertheless it complicates the process of assuring the reliability of the reconfiguration process itself. The fault tolerance challenges that must be addressed are multi-faceted and necessitate a multi-faceted fault tolerance solution which includes hardware and software techniques. For designers of fault-tolerant reconfigurable systems operating in high performance computing environments, a number of strategies are available from redundancy, and error detection and correction schemes to highly sophisticated runtime monitoring and recovery mechanisms. In the following sections we will investigate how the many aspects of designing fault tolerant reconfigurable systems are being addressed by the number of techniques and architectures which are now being designed to actually tackle the problem.[10-12]

Fundamental Principles of Fault Tolerant Design

Fault tolerant reconfigurable computing systems must be designed to have a deep understanding of the fundamental relations that are essential for reliable system design. The principles described here form the cornerstone of more sophisticated fault tolerance techniques. Let's explore the key concepts that guide the development of robust, fault-tolerant reconfigurable systems:

Redundancy

Perhaps the most important principle in fault tolerant design is redundancy. It involves additional components or subsystems that in the event of a failure will control. In reconfigurable systems, redundancy can be implemented at various levels (Figure 2):

Hardware Redundancy: It consists of duplicating critical components and/or whole functional units. As an example, triple modular redundancy (TMR) employs three identical modules performing a task that have a voter circuit for deciding on the correct output.

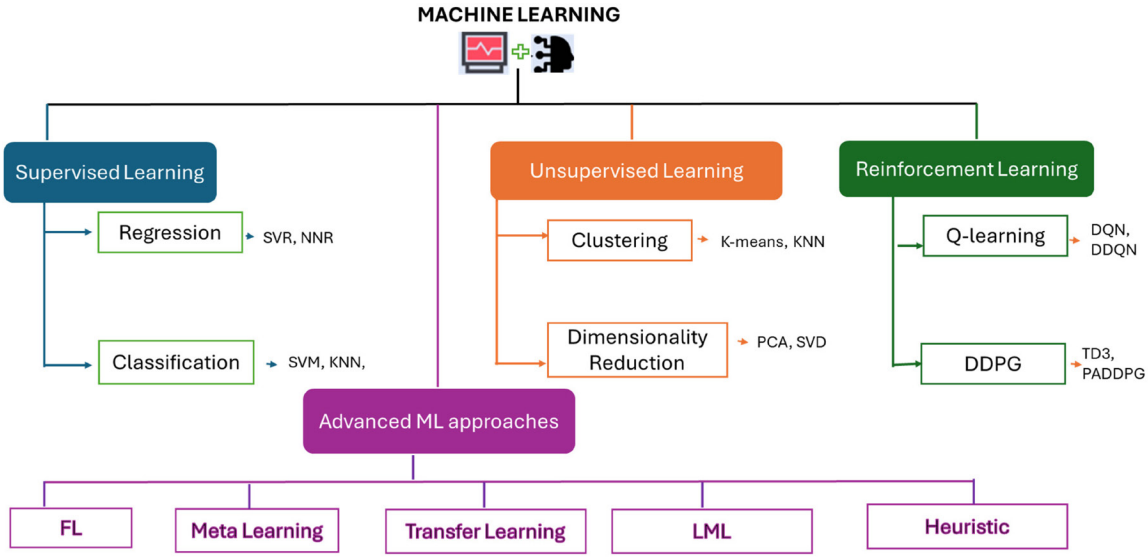

Fig. 2: Fundamental Principles of Fault Tolerant Design

Information Redundancy: It's a technique for adding more bits to data to detect and correct errors. Past examples in the data example are error correcting codes (ECCs) which are a common type of the data, allowing the system to detect and correct some type of bit errors.

Time Redundancy: Transient faults are detected with repetitions of operations multiple times. We find this approach useful for detecting soft errors on FPGAs.

Software Redundancy: Different versions of software algorithms are run to check against and catch errors.

Fault Detection and Isolation

To achieve fault tolerance however, it is first necessary to identify where the faults are and what they are. Fault detection mechanisms in reconfigurable systems often include:

- Built-In Self-Test (BIST): Diagnostic tests and fault detection performed on the FPGA fabric distributed circuitry.
- Concurrent Error Detection: Real time monitoring techniques that identify errors as they occur in system operation.
- Signature Analysis: Methods of compressing circuit outputs into a simple form which can be checked easily for errors.

When a problem is identified it is isolated up to the point it can't spread and affect the remainder of the system. It could mean that faulty components would be disabled or that data flows will redirect around affected areas (Table 2).

Table 2: Performance Benefits of Fault-Tolerant Reconfigurable Computing Systems

Benefit	Value Proposition
Increased Reliability	Increased reliability is achieved by integrating fault-tolerant mechanisms that prevent system failures and ensure continuous operation in critical applications.
Enhanced System Availability	Enhanced system availability ensures that high-performance applications remain operational despite faults, which is vital in mission-critical tasks.
Fault Recovery Speed	Fault recovery speed improves as the system can quickly identify and address issues without significant delays, enhancing performance in real-time environments.
Improved Data Integrity	Improved data integrity ensures that transmitted or processed data remains accurate, reducing the chances of errors that could compromise system output.
Extended System Lifespan	Extended system lifespan is facilitated by fault-tolerant mechanisms that reduce the impact of hardware failures, resulting in fewer repairs and replacements.
Reduced System Downtime	Reduced system downtime is a key benefit of fault-tolerant designs, enabling the system to continue operating even when certain components fail.

Error Correction and Recovery

The system must possess mechanisms of correcting errors and recovery normal operation after detecting a fault. Some common approaches include:

- **Forward Error Correction:** The repetition of information without transmitting it is used to correct errors.
- **Checkpoint and Rollback:** Keeping system state snapshot every once in awhile, and roll back to a known good state if there are errors.
- **Dynamic Reconfiguration:** Reusing reconfigurable chip's capability of reprogramming faulty areas or loading alternative configurations.

Graceful Degradation

In some cases it may not be possible recover from a fault completely. Graceful degradation permits the system to degrade gracefully as it continues operating however much assistance is still available under the reduced or degraded level of performance or functionality. This principle is especially applicable in reconfigurable systems where partial reconfiguration can be employed to offer reduced functionality versions of critical components.[13-14]

Fault Tolerance Metrics

To evaluate the effectiveness of fault-tolerant designs, several metrics are commonly used:

- **Reliability:** Probability that a system will perform its intended function according to the specified performance measures under stated condition.
- **Availability:** The percent of time that a system is in a functional condition.
- **Mean Time Between Failures (MTBF):** The mean time between system failure.
- **Mean Time To Repair (MTTR):** Time taken, on average, to repair a failed system.
- These metrics enable designers to quantify the available fault tolerance in their systems and to make decisions regarding tradeoffs between reliability, performance and cost.

Design for Testability

To keep fault tolerant systems, importance is attached to incorporating features that allow for testing and fault diagnosis. This includes:

- **Scan Chains:** 'Easily tested serial paths through the circuit are provided to test internal states.'
- **Boundary Scan:** Method for establishing and testing interconnects between chips.
- **Debug Interfaces:** Monitoring and control mechanisms of the internal state of the system at run time.

With observation of these fundamental principles, designers can build a strong case for fault tolerant reconfigurable computing systems. Nevertheless, the less generic nature of FPGAs, along with the needs of high performance applications, calls for other special techniques, which we will discuss later on [15]-[18].

Fault Detection and Masking Techniques based on FPGA

The unique opportunities that Field Programmable Gate Arrays (FPGAs) present us for the implementation of sophisticated fault detection and masking techniques are highlighted with the aid of two (2) illustrative examples. The reconfigurable nature of these devices enables adaptive fault tolerance mechanisms to dynamically adapt to changing conditions. Let's explore some of the key FPGA-based techniques for fault detection and masking:

Concurrent Error Detection (CED)

We have already described Concurrent Error Detection as a powerful way to noisily detect faults during normal system operation. In FPGA-based systems, CED can be implemented through various methods:
Duplication with Comparison: It entails making two copies of a circuit, with exact construction, and then comparing the outputs. A fault is indicated by any discrepency. However, this simple approach comes at a high area overhead (Figure 3).

Parity Prediction: For a circuit, we compute parity bits for the expected outputs, and then we check why the actual parity bits are different from ours. Full duplication is more area efficient than this method, except it may miss some types of errors.

Residue Checking: It consists of computing in two formally different number systems---one is the original number system, and the other is a residue number system. If these results differ, there's error. In particular, this method is often good for arithmetic circuits.

Time-Redundant Computation: Transient faults get detected by executing the same computation many

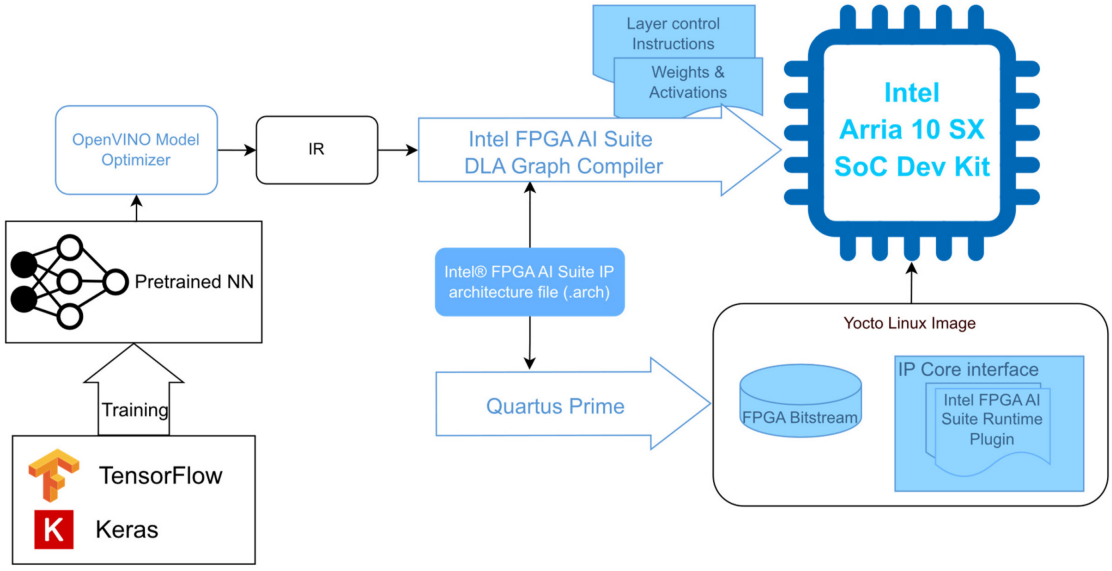

Fig. 3: Concurrent Error Detection (CED)

times, using possibly different encodings of the input. The use of FPGAs' reconfigurability in this context is used to implement time multiplexed redundancy.

Dynamic Reconfiguration: Using runtime reconfiguration as the basis to allow FPGAs to be used as implementing platforms for new adaptive fault tolerance mechanisms opens the door for their further deployment. However it complicates assurance of the reliability of the reconfiguration process itself.

The fault tolerance challenges are multi faceted and call out for a multi facet fault tolerance solution including hardware and software techniques. Redundancy, error detection and correction, and highly sophisticated runtime monitoring and recovery strategies are available to designers of fault tolerant reconfigurable systems operating in high performance computing environments. The following sections examine how various techniques and architectures are being developed to actually face the problem of designing fault tolerant reconfigurable systems.

In order to design fault tolerant reconfigurable computing systems with a deep understanding of the relations that are necessary for reliable system design, a deep understanding of the fundamental relations is needed. The techniques described here are the basis for more sophisticated fault tolerance techniques. Let's explore the key concepts that guide the development of robust, fault-tolerant reconfigurable systems. Redundancy is perhaps the most important principle in fault tolerant design. It consists of additional or subsystem components or subcomponents to control in the event of failure. In reconfigurable systems, redundancy can be implemented at various levels.[19-22]

Fault Detection and Isolation

However, for fault tolerance it is necessary to find and define the faults, first. Fault detection mechanisms in reconfigurable systems often include:

- **Built-In Self-Test (BIST):** Performed diagnostic tests and fault detection on the FPGA fabric distributed circuitry.
- **Concurrent Error Detection:** Techniques for real time monitoring to detect errors as they manifest in system operation.
- **Signature Analysis:** Circuit outputs compressed into a simple form to make them easy to check for errors.

If there are something to fix it isolates till it cannot spread and hit the rest of the system. This could mean disabled parts or that data flows will skip over the problematic areas.

At the same time, the system needs to have the ways of error correction and recovery from fault for normal operation. Some common approaches include:

- **Forward Error Correction:** To correct errors, information is repeated without transmitting it.

- **Checkpoint and Rollback:** Roll back to a known good state if there are errors, but keeping system state snapshot every once in awhile.
- **Dynamic Reconfiguration:** Regeneration of faulty areas in reconfigurable chip or loading of a different configuration.[23-24]

Graceful Degradation

Here it may not be possible to recover completely from a fault. Gracious degradation means that the system degrades gracefully as it continues working, however much of the System's performance or functionality remains available. This principle is also applicable when partial reconfiguration can be used to provide reduced functionality versions of some critical components in reconfigurable systems.

Fault Tolerance Metrics

To evaluate the effectiveness of fault-tolerant designs, several metrics are commonly used:

- **Reliability:** Chance of a system being able to perform its specified function within prescribed requirements, taking into account its status.
- **Availability:** The proportion of a systems time spent in a functional condition.
- **Mean Time Between Failures (MTBF):** Mean time of system failure.
- **Mean Time To Repair (MTTR):** The average time taken to repair a system that failed.

These metrics provide designers with a measure of the available fault tolerance that can be achieved in different kinds of systems and facilitate decisions regarding tradeoffs between reliability, performance and cost.

Design for Testability

So that fault tolerant systems are kept, featues are used to incorporate testing and fault diagnosis. This includes:

- Scan Chains: 'To test internal states, easily tested serial paths through the circuit are provided.'
- **Boundary Scan:** A method for interconnecting chips.
- **Debug Interfaces:** At run time, monitoring and control mechanisms of the internal state of the system.

Designers build the case for fault tolerant reconfigurable computing systems with observation of these basic principles. However, other special techniques are needed, although the less generic nature of FPGAs and the need for high performance applications require other techniques, which we will discuss later on.

Fault Detection and Masking Techniques implemented in FPGA

With the aid of two (2) illustrative examples, the unique opportunities for applying sophisticated fault detection and masking techniques afforded by Field Programmable Gate Arrays (FPGAs) are highlighted. Adaptive fault tolerance mechanisms using these devices are reconfigurable, providing adaptive fault tolerance adaptations to changing environments. Let's explore some of the key FPGA-based techniques for fault detection and masking. Concurrent Error Detection is a strong way to noisily detect faults throughout normal system operation that we have already described above. In FPGA-based systems, CED can be implemented through various methods. The communication infrastructure in a fault-tolerant reconfigurable system plays a critical role in both normal operation and fault recovery. With consideration of these architectural aspects, designers can build reconfigurable computing systems which are both highly performant and resistant to a broad spectrum of fault possibility. The battlefield is to build a flexible, flexible architecture that responds to changes and gracefully handle faults when they do occur. To address the quest for more reliable and fault tolerant FPGA based systems, new architectures beyond conventional design approaches have been developed. These novel architectures exploit the specific advantages that FPGAs provide to realize new fault tolerance methods in hardware. Let's explore some of the cutting-edge FPGA architectures designed with fault tolerance as a primary consideration. Self-healing FPGAs represent a paradigm shift in fault-tolerant design, incorporating mechanisms for autonomous fault detection, isolation, and recovery. Combining the strengths of traditional CPUs with the flexibility of FPGAs can lead to highly robust fault-tolerant systems:

Conclusion

These innovative designs leverage the reconfigurable nature of FPGAs to implement fault tolerance

through time-multiplexing. These FPGA architectures constitute the state-of the art of the fault tolerant design for reconfigurable systems. These architectures promise to deliver unprecedented levels of reliability and resilience for high performance computing applications, while using advanced fault tolerance mechanisms that are incorporated directly into the hardware fabric. With further research in this field, we will also witness an evolution of ever more sophisticated fault tolerant FPGA architectures in the realm of reliable reconfigurable computing. In recent years, reconfigurable computing research has begun to offer fault tolerant computing systems that are being applied to a wide spectrum of industries and domains, where high performance, reliability and adaptability are critical. Let's explore some of the key real-world applications where these systems are making a significant impact:

References:

1. Yin, P., Wang, C., Waris, H., Liu, W., Han, Y., & Lombardi, F. (2020). Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning. *IEEE Transactions on Sustainable Computing*, *6*(4), 612-625.
2. Zacharelos, E., Nunziata, I., Saggese, G., Strollo, A. G., & Napoli, E. (2022). Approximate recursive multipliers using low power building blocks. *IEEE Transactions on Emerging Topics in Computing*, *10*(3), 1315-1330.
3. Waris, H., Wang, C., Xu, C., & Liu, W. (2021). AxRMs: Approximate recursive multipliers using high-performance building blocks. *IEEE Transactions on Emerging Topics in Computing*, *10*(2), 1229-1235.
4. Goldstein, B. F., Srinivasan, S., Das, D., Banerjee, K., Santiago, L., Ferreira, V. C., ... & França, F. M. (2020, February). Reliability evaluation of compressed deep learning models. In *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)* (pp. 1-5). IEEE.
5. Aranda, L. A., Wessman, N. J., Santos, L., Sánchez-Macián, A., Andersson, J., Weigand, R., & Maestro, J. A. (2020). Analysis of the critical bits of a RISC-V processor implemented in an SRAM-based FPGA for space applications. *Electronics*, *9*(1), 175.
6. Wilson, A. E., & Wirthlin, M. (2019, July). Neutron radiation testing of fault tolerant RISC-V soft processor on Xilinx SRAM-based FPGAs. In *2019 IEEE Space Computing Conference (SCC)* (pp. 25-32). IEEE.
7. Ozen, E., & Orailoglu, A. (2019, December). Sanity-check: Boosting the reliability of safety-critical deep neural network applications. In *2019 IEEE 28th Asian Test Symposium (ATS)* (pp. 7-75). IEEE.
8. Vijay, V., Sreevani, M., Rekha, E. M., Moses, K., Pittala, C. S., Shaik, K. S., ... & Vallabhuni, R. R. (2022). A Review On N-Bit Ripple-Carry Adder, Carry-Select Adder And Carry-Skip Adder. *Journal of VLSI circuits and systems*, *4*(01), 27-32.
9. Zhao, K., Di, S., Li, S., Liang, X., Zhai, Y., Chen, J., ... & Chen, Z. (2020). FT-CNN: Algorithm-based fault tolerance for convolutional neural networks. *IEEE Transactions on Parallel and Distributed Systems*, *32*(7), 1677-1689.
10. Zhang, Y., Lin, S., Wang, R., Wang, Y., Wang, Y., Qian, W., & Huang, R. (2020, March). When sorting network meets parallel bitstreams: A fault-tolerant parallel ternary neural network accelerator based on stochastic computing. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1287-1290). IEEE.
11. Gala, N., Menon, A., Bodduna, R., Madhusudan, G. S., & Kamakoti, V. (2016, January). SHAKTI processors: An open-source hardware initiative. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)* (pp. 7-8). IEEE Computer Society.
12. Gupta, S., Gala, N., Madhusudan, G. S., & Kamakoti, V. (2015, November). SHAKTI-F: A fault tolerant microprocessor architecture. In *2015 IEEE 24th Asian Test Symposium (ATS)* (pp. 163-168). IEEE.
13. Ramos, A., Toral, R. G., Reviriego, P., & Maestro, J. A. (2019). An ALU protection methodology for soft processors on SRAM-based FPGAs. *IEEE Transactions on Computers*, *68*(9), 1404-1410.
14. Coelho, C. N., Kuusela, A., Li, S., Zhuang, H., Ngadiuba, J., Aarrestad, T. K., ... & Summers, S. (2021). Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, *3*(8), 675-686.
15. Babu, P. A., Nagaraju, V. S., & Vallabhuni, R. R. (2021, June). Speech emotion recognition system with librosa. In *2021 10th IEEE international conference on communication systems and network technologies (CSNT)* (pp. 421-424). IEEE.
16. Gao, Z., Wei, X., Zhang, H., Li, W., Ge, G., Wang, Y., & Reviriego, P. (2020, October). Reliability evaluation of pruned neural networks against errors on parameters. In *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)* (pp. 1-6). IEEE.
17. Dutta, A., & Touba, N. A. (2007, May). Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code. In *25th IEEE VLSI Test Symposium (VTS'07)* (pp. 349-354). IEEE.
18. Radhakrishnan, S., Nirmalraj, T., Ashwin, S., Elamaran, V., & Karn, R. K. (2018, March). Fault tolerant carry save adders-A NMR configuration approach. In *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)* (pp. 210-215). IEEE.
19. Shaker, M. N., Hussien, A., Alkady, G. I., Amer, H. H., & Adly, I. (2019, June). Mitigating the effect of multiple

event upsets in fpga-based automotive applications. In *2019 8th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1-4). IEEE.

20. Vallabhuni, R. R., Shruthi, P., Kavya, G., & Chandana, S. S. (2020, December). 6Transistor SRAM cell designed using 18nm FinFET technology. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 1584-1589). IEEE.
21. Mansour, W., & Velazco, R. (2013). SEU fault-injection in VHDL-based processors: A case study. *Journal of Electronic Testing*, *29*, 87-94.
22. Rahman, S. M., Ibtisum, S., Bazgir, E., & Barai, T. (2023). The significance of machine learning in clinical disease diagnosis: A review. *arXiv preprint arXiv:2310.16978*.
23. Rahman, S. M., Ibtisum, S., Podder, P., & Hossain, S. M. (2023). Progression and challenges of IoT in healthcare: A short review. *arXiv preprint arXiv:2311.12869*.
24. Ibtisum, S., Bazgir, E., Rahman, S. A., & Hossain, S. S. (2023). A comparative analysis of big data processing paradigms: Mapreduce vs. apache spark. *World Journal of Advanced Research and Reviews*, *20*(1), 1089-1098.
25. Klavin, C. (2024). Analysing antennas with artificial electromagnetic structures for advanced performance in communication system architectures. *National Journal of Antennas and Propagation, 6*(1), 23–30.
26. Anandhi, S., Rajendrakumar, R., Padmapriya, T., Manikanthan, S. V., Jebanazer, J. J., & Rajasekhar, J. (2024). Implementation of VLSI Systems Incorporating Advanced Cryptography Model for FPGA-IoT Application. Journal of VLSI Circuits and Systems, 6(2), 107–114. https://doi.org/10.31838/jvcs/06.02.12
27. Roper, S., & Bar, P. (2024). Secure computing protocols without revealing the inputs to each of the various participants. *International Journal of Communication and Computer Technologies, 12*(2), 31-39. https://doi.org/10.31838/IJCCTS/12.02.04
28. 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
29. 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
30. Ismail, K., & Khalil, N. H. (2025). Strategies and solutions in advanced control system engineering. *Innovative Reviews in Engineering and Science, 2*(2), 25-32. https://doi.org/10.31838/INES/02.02.04
31. Uvarajan, K. P. (2024). Integration of blockchain technology with wireless sensor networks for enhanced IoT security. Journal of Wireless Sensor Networks and IoT, 1(1), 23-30. https://doi.org/10.31838/WSNIOT/01.01.04
32. Kumar, T. M. S. (2024). Security challenges and solutions in RF-based IoT networks: A comprehensive review. *SCCTS Journal of Embedded Systems Design and Applications, 1*(1), 19-24. https://doi.org/10.31838/ESA/01.01.04
33. Sathish Kumar, T. M. (2024). Low-power design techniques for Internet of Things (IoT) devices: Current trends and future directions. *Progress in Electronics and Communication Engineering, 1*(1), 19–25. https://doi.org/10.31838/PECE/01.01.04