DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING KANCHEEPURAM
CHENNAI - 600127

Synopsis Of

Fault-Tolerant Adder and Multiplier Designs for Mission Critical Systems

A Thesis

To be submitted by

SAKALI RAGHAVENDRA KUMAR

For the award of the degree

Of

DOCTOR OF PHILOSOPHY

1 Abstract

Adders and multipliers serve as critical components in the harsh environmental conditions of space and mission-critical systems, where radiation and high temperatures pose significant challenges. This research addresses the susceptibility of these circuits to Single-Event Upsets (SEUs) induced by high-energy particles such as gamma rays and alpha rays. While existing solutions incorporate redundant-based fault-tolerant techniques, concerns persist regarding hardware overhead and system performance degradation. In response, our work introduces a novel preferential-based approach to fault tolerance, strategically reducing hardware overhead by 40% to 60% based on the input size of adder and multiplier circuits. Despite the efficacy of redundant fault-tolerant techniques, their lack of self-adaptability represents a notable limitation. To overcome this, a bio-inspired hardware approach is pursued, leveraging Evolvable Hardware (EHW). Previous EHW implementations encountered challenges related to scalability and error recovery time.

A substantial contribution of this research is the development of self-healing designs for adders and multipliers, implemented on a single Field-Programmable Gate Array (FPGA) with a noteworthy 50% to 60% reduction in error recovery rates. An integral aspect of this work is the proposal of an Amended Virtual Reconfiguration Circuit (AVRC) for adders and multipliers, streamlining the bitstream and effectively addressing scalability and hardware complexity challenges. This comprehensive approach includes error detection through a reference unit and recovery via a bitstream generation unit. A comparative analysis demonstrates a significant reduction in error rates when compared to existing methodologies. The proposed self-healing designs mark a substantial advancement in fault-tolerant digital circuits, particularly in the demanding contexts of space and mission-critical systems. These advancements not only enhance the reliability of adders and multipliers but also pave the way for more resilient digital circuits in the stringent environments of space and mission-critical systems.

2 Objectives

- Examine the operational characteristics and design approaches of adder and multiplier circuits within the context of digital design used in mission-critical systems.

- Conduct extensive fault injection simulations in adder and multiplier circuits, mimicking the behaviour of SEUs to analyze their impact on circuits.

- Conduct a detailed study on fault-tolerant techniques used to mitigate SEUs in adder and multiplier circuits, including an analysis of their challenges.

- Design unique fault-mitigation approaches for adder and multiplier circuits to address the challenges identified in existing works.

- Evaluate the proposed solutions considering metrics such as resource utilization, delay, and fault recovery time.

- Compare and contrast the efficiency of the proposed solutions to state-of-the-art fault mitigation approaches targeted for adder and multiplier circuits

3 Existing Gaps Which Were Bridged

- **Conventional Fault-Tolerant Technique**
  - TMR and self-checking units are reliable and feasible as fault-tolerant techniques, TMR requires more than $2\times$ to $3\times$ times the original hardware, and similarly, self-checking units require extra MUX and XOR gates, as well as the DMR technique, which scales up the hardware cost by more than approximately $2.5\times$ times compared to the original hardware. This major challenge leads to a degradation in system performance due to increased delay costs.

- **Evolvable Hardware**
  - **Configuration Bitstream Length:** Many existing works have designed adder and multiplier circuits with long bitstreams. This increases the search space and prolongs the recovery rate of faults in the circuit, consequently extending the evolution time of the circuit.
  - **Reconfiguration Technique:** The evolution of configuration bits is not feasible in non-commercial FPGAs designed for military and security applications. These FPGAs lack the DPR (Dynamic Partial Reconfiguration) tool and port for configuring bits, and bitstream access is restricted by encryption. Additionally, non-commercial FPGAs do not provide tools for accessing the bitstream, while in commercial FPGAs, such tools are vendor-specific.
  - **Complex VRC Architecture:** The VRC-based adder and multiplier design require a significant amount of resources, posing issues of insufficient resources in FPGAs. Due to this limitation, many existing works have opted for 2-bit and 3-bit size adders and multipliers.
  - **Fault Recovery Time:** The recovery rate is a critical parameter for fault correction in the circuit. There might be a chance of circuit failure if a fault is not recovered within the required time, especially with genetic algorithms. Longer configuration bitstream lengths, particularly in the case of combinational circuits, can exacerbate this situation.
  - **Extrinsic/Hybrid Implementation:** Many existing works have implemented EHW using an extrinsic/hybrid approach. This may increase the circuit delay and degrade system performance.

4 Most Important Contributions

- **Contribution 1: Redundant-Based Fault-Tolerant Technique for Adder** The main objective of the proposed work is to mitigate single-event upsets (SEU) in both integer and floating-point adders by reducing hardware utilization. We address challenges in existing architectures through detailed analysis to improve system performance.
- **Contribution 2: Redundant-Based Fault-Tolerant Technique for Multiplier** The primary goal of this contribution is to mitigate SEU in both integer and floating-point multipliers by reducing hardware utilization. We address challenges in existing architectures through detailed analysis to enhance system performance.

- **Contribution 3: Self-Healing Adder Design with an Intrinsic Approach** The proposed work aims to design a self-healing adder by addressing challenges in evolvable hardware. The design incorporates an amended virtual reconfigurable circuit and novel ideas to improve system performance. Also highlighted the importance of the configuration bitstream regeneration unit.
- **Contribution 4: Self-Healing Multiplier Design with an Intrinsic Approach** This contribution focuses on designing a self-healing multiplier by addressing challenges in evolvable hardware. In this work, we reduced the configuration bitstream of the AVRC adder to reduce time in the case of configuration bitstream generation. Illustrated the comparison of the hybrid and intrinsic implementation of the proposed work and shown the fault-recovery rate improved performance.

4.1 Redundant Fault-Tolerant Techniques for Adder

Adder is the primary component in datapath elements for processing the applications of mission-critical systems. The mission-critical systems are operated in the radiation environment. Digital components are sensitive to the radiation environment. High energy particles induce into the digital system and flip the data bit. This is known as a Single Event Upset (SEU). The SEU will alter the result of the adder and cause the system failure. Most of the existing works were integrated with either TMR (Triple Modular Redundancy) or Self-Repair Adder (SRA) with adder circuits for mitigating the SEUs. These techniques were reliable and generated the correct result. However hardware overhead is the major challenge in these techniques. Because of this issue, delay will arise and the system performance will be degraded, To address this challenge, we proposed an approach with error analysis results.

In the proposed work, at the initial stage, the behaviour of SEU in adder was studied. The fault-injection simulator was designed to inject the single-bit fault in an adder circuit by mimicking the behaviour of the SEU. The faults were injected randomly into the circuit with respect to the resultant bit position. Later the generated error result was calculated by using the original result of the adder. In this experimental work, we injected the fault from the 0^{th}-bit position to the N-1^{th} position hardware element in different adder sizes such as 8-bit, 16-bit, 32-bit, 64-bit and 128-bit.

4.1.1 Preferential Fault-Tolerant Adder

Based on the outcomes of the SEU analysis, we propose an innovative fault-tolerant approach designed to reduce hardware utilization and enhance system performance, particularly when contrasted with traditional fault-tolerant techniques. Traditional fault-tolerant methods typically demand more hardware elements, which, in turn, elevate the probability of failure. We aim to scale down the hardware overhead associated with a fault-tolerant adder. In this approach, our focus centers on elements positioned in the Most Significant Bit (MSB). In a binary format, MSBs bear greater significance in weight. Consequently, errors occurring in the MSB-positioned elements result in a notably higher error percentage in the adder's output. This approach strategically addresses the pivotal role of MSBs in influencing the accuracy of the adder's results while concurrently aiming to reduce the hardware complexities inherent in conventional fault-tolerant strategies.

In an adder operation, the input with an N-bit size will generate an output with an N+1-bit size. After observing the SEU analysis for different adder lengths, it was identi-

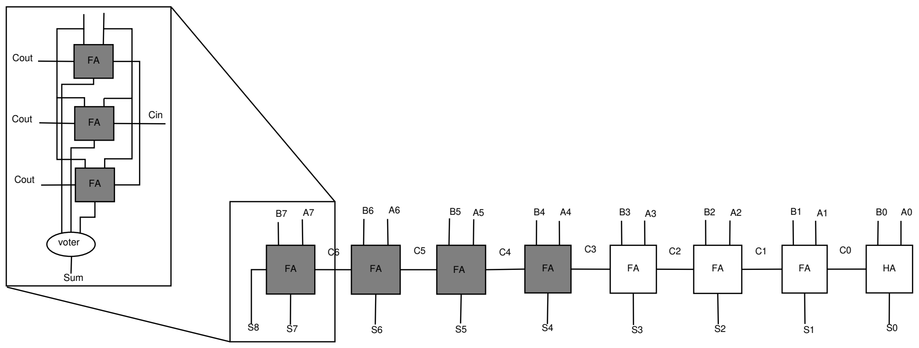

Figure 1: 8-bit fault-tolerant adder with preferential TMR approach

fied that the bit positions that hold the error percentage equal to or greater than 6 are the most critical. This criticality was identified with an error distance when compared with the original, as shown in Table 1. So, the last four MSB positions are considered critical bit positions. Apart from these four MSB positions, fault occurrences in any bit position don't have much impact on the result. So, we designed a fault-tolerant adder by preferring fault-tolerant techniques at the MSB-positioned hardware, as shown in Figure 1. This proposed approach can be used for any kind of adder circuit with fault-tolerant techniques such as redundant technology. This technique can be used for applications such as robotics and rovers, multimedia applications, geospatial satellite applications, and digital signal processing applications where fault occurrences in LSB are acceptable.

Table 1: Error percentage of 8-bit Adder

Carry bit position	Input 1 (Binary value)	Input 2 (Binary value)	Input 1 (Decimal Value)	Input 2 (Decimal Value)	Original Carry	Original sum	Original sum (decimal value) (x1)	Carry bit flip Position	Error Carry	Error sum	Error Sum (decimal value) (x2)	Error distance b/w x1 & x2	Error Percentage
0	11111111	11111111	255	255	11111111	111111110	510	0	11111110	111111100	508	2	0.39
1	11111111	11111111	255	255	11111111	111111110	510	1	11111101	111111010	506	4	0.78
2	11111111	11111111	255	255	11111111	111111110	510	2	11111011	111110110	502	8	1.56
3	11111111	11111111	255	255	11111111	111111110	510	3	11110111	111101110	494	16	3.13
4	11111111	11111111	255	255	11111111	111111110	510	4	11101111	111011110	478	32	6.27
5	11111111	11111111	255	255	11111111	111111110	510	5	11011111	110111110	446	64	12.54
6	11111111	11111111	255	255	11111111	111111110	510	6	10111111	101111110	382	128	25.09
7	11111111	11111111	255	255	11111111	111111110	510	7	01111111	011111110	254	256	50.19

4.1.2 Implementation and Results

Proposed works such as PrTMR adder and PrSRA adder were designed on the Libero SOC 11.8 IDE tool by targeting the A3PE3000 FPGA to analyse LUT utilisation and delay. So, we implemented existing work, and proposed work for different input sizes, such as 8-bit, 16-bit, and 32-bit, using HDL. The implementation results provide details in the form of LUTs and delays. This practical result provides proof that the proposed work is a better solution for implementing fault-tolerant circuits as shown in Table 2 and Table 3.

4.2 Redundant Fault-Tolerant Techniques for Multiplier

Multiplication is an essential operation to perform various numerical calculations. The process of multiplier operations can be complicated and sometimes leads to delays based on the input size. There is a chance of SEUs in combinational circuits due to

Table 2: LUTs Utilization

Adder Input Size	Adder without Fault-Tolerant	Existing Work		Proposed Work	
		TMR based Adder Zhao *et al.* (2018), Li *et al.* (2020)	SRA based Adder SRA based Adder Kumar and Sharma (2016), Palsodkar *et al.* (2018)	PrTMR based Adder	PrSRA based Adder
8-bit	16	57	72	40	48
16-bit	32	113	164	56	79
32-bit	64	226	289	88	113

Table 3: Delay(ns)

Adder Input Size	Adder without Fault-Tolerant	Existing Work		Proposed Work	
		TMR based Adder Zhao *et al.* (2018), Li *et al.* (2020)	SRA based Adder Kumar and Sharma (2016), Palsodkar *et al.* (2018)	PrTMR based Adder	PrSRA based Adder
8-bit	1.829	3.462	4.234	2.131	2.543
16-bit	2.155	3.574	4.412	2.157	2.671
32-bit	2.899	3.693	4.524	2.279	2.823

radiation effects or external disturbance sources. In existing works, multiplier designs were integrated with TMR and SRA to design fault-tolerant multipliers. These conventional fault-tolerant techniques provide reliable results in the radiation environment by mitigating SEUs. The standalone multiplier circuit is complex apart from this fault-tolerant technique enhancement will scale up the hardware and increase the delay. To overcome these challenges, we proposed a novel approach using error analysis. This is similar to the adder.

4.2.1 Preferential Fault-Tolerant Multiplier

In a multiplier, the generation of a $2 \times N$-bit output is driven by two N-bit inputs. While Single Event Upsets (SEUs) may occur at any point within the multiplier during its operation, it is recognized that the partial products play a critical role. Errors at this stage can lead to potential error propagation, particularly given the significance of partial products in ensuring the accuracy of the result. To investigate the impact of errors, we strategically selected the partial products for fault injection, considering the result bit positions ranging from 0-bit to $2 \times N - 1$ bit randomly. Experimental trials were conducted with varying bit lengths, including 4, 8, 16, 32, and 64, using two N-bit inputs. The outcomes of these experiments were meticulously recorded for each bit length of the multiplier. Subsequently, the error analysis results for each multiplier size architecture were utilized to discern critical and non-critical error bit positions within the resultant multiplier.

4.2.2 Results and Implementation

PrTMR based multiplier and PrSRA based multiplier, along with existing work, were implemented on the Xilinx Vivado by targeting the Zync-7000 FPGA. 8×8 (Error

Table 4: 8-Bit Error Percentage Results

Error injection in PP w.r.t Result Bit Position	X (Decimal)	X (Binary)	Y (Decimal)	Y (Binary)	Result (Decimal) X1	Result (Binary)	Result after error injection (Decimal) X2	Result after error injection (Binary)	Error Distance b/w X1 & X2	Error %
0	255	11111111	255	11111111	65025	1111111000000001	65024	1111111000000000	1	0.00153
1	255	11111111	255	11111111	65025	1111111000000001	65023	1111110111111111	2	0.00307
2	255	11111111	255	11111111	65025	1111111000000001	65021	1111110111111101	4	0.00615
3	255	11111111	255	11111111	65025	1111111000000001	65017	1111110111111001	8	0.01230
4	255	11111111	255	11111111	65025	1111111000000001	65009	1111110111110001	16	0.02460
5	255	11111111	255	11111111	65025	1111111000000001	64993	1111110111100001	32	0.04921
6	255	11111111	255	11111111	65025	1111111000000001	64961	1111110111000001	64	0.09842
7	255	11111111	255	11111111	65025	1111111000000001	64897	1111110110000001	128	0.19684
8	255	11111111	255	11111111	65025	1111111000000001	64769	1111110100000001	256	0.39369
9	255	11111111	255	11111111	65025	1111111000000001	64513	1111110000000001	512	0.78738
10	255	11111111	255	11111111	65025	1111111000000001	64001	1111101000000001	1024	1.5747
11	255	11111111	255	11111111	65025	1111111000000001	62977	1111011000000001	2048	3.1495
12	255	11111111	255	11111111	65025	1111111000000001	60929	1110111000000001	4096	6.29911
13	255	11111111	255	11111111	65025	1111111000000001	56833	1101111000000001	8192	12.5982
14	255	11111111	255	11111111	65025	1111111000000001	48641	1011111000000001	16384	25.1964
15	255	11111111	255	11111111	65025	1111111000000001	32257	0111111000000001	32768	50.3929

Analysis shown in Table 4) and 16×16 size multipliers are considered for the implementation of the proposed work. The efficiency of the proposed work was analysed based on the utilisation of LUTs and delay. Table 5 represents the LUTs and delay of existing and proposed work.

Table 5: Utilization of Resource and Delay

Works	**Types**	**Input Size**	**LUTs**	**Delay (ns)**
Existing Work	TMR based Multiplier Zhao *et al.* (2018)	8x8	1256	67.124
		16x16	2600	126.245
	SRA based Multiplier Kumar and Sharma (2016)	8x8	1229	55.21
		16x16	2558	116.32
Proposed Work	PrTMR based Multiplier	8x8	669	47.869
		16x16	1358	89.562
	PrSRA based Multiplier	8x8	629	36.52
		16x16	1298	72.26

4.3 Self-Healing Adder Design with an Intrinsic Approach

Evolvable Hardware (EHW) is a bio-inspired technique used in designing fault-tolerant circuits. The EHW-based fault-tolerant technique overcomes the major limitation like self-healing which is not in conventional fault-tolerant techniques. The other limitation as the size of the circuit increases, the delay may also increase. EHW has a nature of self-healing. Most of the existing works used extrinsic/ hybrid implementation. There are some challenges present in the existing that are listed in Section 3. Those challenges are achieved by our contributions as follows.

- The adder circuit was implemented by amending the structure of the VRC architecture. This approach scales down the hardware complexity and configuration bitstream length.

- Proposed a novel configuration bitstream reconstruction algorithm by replacing the evolutionary algorithm to reduce the fault-recovery rate.
- The intrinsic-based implementation was used to improve the system's performance.

The proposed self-healing adder consists of four major modules such as VRC adder, Reference Output Generator (ROG) unit, error detection unit and configuration bitstream reconstruction unit.

4.3.1 Proposed Amended VRC (AVRC) Adder

The VRC is a kind of reconfiguration mechanism used in evolvable hardware. VRC is designed with MUX elements and this acts as a second fabric layer on FPGA. We have modified the structure of VRC-based specific functionality and reduced the maximum hardware compared to existing works and named as Amended VRC (AVRC). The AVRC adder works based on user configuration bitstream. In this mechanism, the circuit control will be in the hands of the designer. AVRC adder consists group of PEs (Programmable Elements). In the proposed work each PE consists of two 5:1 MUX for selecting inputs and one 4:1 mux for adder functionality. The 4-bit AVRC adder requires a total of 17 PEs. Each PE requires 8 bits and a total of 132 bits are required to operate the AVRC adder.

4.3.2 ROG and Error Detection Unit

The ROG unit is designed using a full-adder LUT, this unit is operated by passing inputs of the original unit. The full-adder truth table information is stored in flash memory. The sum and carry bits are accessed using inputs. Both adders were operated simultaneously. After this point, both results were compared using the XOR operator. If the resultant of XOR has 1's bits, then it enables the signal for configuration bitstream reconstruction.

4.3.3 Configuration Bitstream Reconstruction Unit

The Configuration Bitstream Reconstruction (CBR) unit has been designed to generate the bitstream based on the structure of the AVRC adder. This unit is implemented using integer sequence generators. The sequence generator generates the required bits based on the size of MUX and the size of the adder. The generated bits are concatenated and arranged using the structure of the AVRC adder. This is used instead of the evolutionary algorithm and it doesn't require any search space. The fault-recovery will be improved using proposed CBR unit.

4.3.4 Implementation and Results

The experimental work was conducted with 4-bit and 8-bit adders to assess the efficiency of the proposed approach. The fault-tolerant self-healing adder prototype and the algorithm hardware testing were hosted on the Proasic3e 3000 FPGA. All four modules of proposed were designed in HDL using the Libero SOC design suite 11.8. The proposed work for 4-bit and 8-bit adders takes around 12.32 ns and 26.25 ns to generate fault-free results, respectively. In Jian et al.'s work Jian and Mengfei (2018), the 2-bit adder took approximately 17 milliseconds of evolution time using a hybrid approach. We analyzed Jian's adder with a 4-bit input, and it took around 38 milliseconds to evolve a circuit. In Cancare et al.'s work Cancare *et al.* (2012), the evolution time for a 4-bit adder took approximately 34 minutes to achieve a fault-free result. However, it was implemented using the DPR mechanism with an extrinsic approach. The comparison of the evolution time of existing works and the proposed work is shown in Table 6. In contrast to previous work, the proposed work evolves the circuit more efficiently during fault occurrence.

Table 6: Comparing the evolution time of a proposed work with existing works

Works	Adder size	Reconfiguration mechanism	Evolvable approach	Error recovery time
Existing works	4-bit Cancare *et al.* (2012)	DPR	Extrinsic	2040 s
	4-bit Jian and Mengfei (2018)	VRC	Extrinsic	0.038 s
Proposed work	4-bit	VRC	Intrinsic	12.32 ns
	16-bit	VRC	Intrinsic	26.25 ns

The area occupancy of the proposed work is accounted for in terms of IO Ports, LUTs, and flashROM utilization. Consequently, the resource usage and recovery time of the proposed work are compared with similar works discussed in the related literature. The proposed work exhibits LUT utilization of 53.64%, which is lower than existing works for both sizes of adders. Additionally, IO Ports and FlashROM were utilized 42.77% and 54.9% less, respectively, than in previous works. Hence, the resource utilization of the proposed work is 50% lower than that of existing works as shown in Table 7

Table 7: Hardware utilization of Proposed Work

Adder size	Resources	Available	Proposed Self-Healing Adder with AVRC		Existing EHW Adder with standard VRC	
			Utilization	Utilization%	Utilization	Utilization%
4-bit	LUTs	75264	15	0.0199	32	0.0425
	IO Ports	620	55	8.870	95	15.32
	FlashROM	1024	200	19.53	430	41.99
8-bit	LUTs	75264	33	0.0438	72	0.0956
	IO Ports	620	112	18.06	198	31.94
	FlashROM	1024	400	39.06	840	82.03

4.4 Self-Healing Multiplier Design with an Intrinsic Approach

We introduce a fault-tolerant multiplier designed to counteract single-event upsets. This multiplier architecture draws parallels with evolvable hardware paradigms. Consisting of three integral modules, namely, an amended VRC-based multiplier, an error diagnosis unit, and a configuration bitstream generator, the proposed self-healing multiplier boasts both adaptability and functionality akin to evolvable hardware, particularly in fault recovery. The sole distinction lies in the configuration bitstream generation process. Our innovation centers on a novel Configuration Bitstream Generator (CBG) aimed at improving the error recovery speed. The design encompasses a memory-based multiplier complemented by a comparator dedicated to error detection. In this unit, the outcome of the VRC multiplier is cross-referenced with that of the memory-based multiplier.

Most of the existing works have implemented a 2-bit or 3-bit multiplier. In this work, we present the design of an Amended VRC-based multiplier. To address scalability concerns and streamline memory complexity, we divide the multiplier into three segments. The first segment details the selection of input bits (multiplicand and multiplier) using 4:1 multiplexers. The second segment covers partial product generation via 2:1 multiplexers. The third segment focuses on partial product reduction using VRC adder modules. This modular approach aims to enhance both the multiplier's performance and memory efficiency.

4.4.1 Implementation Results

The performance and effectiveness of the self-healing multiplier were assessed through simulations conducted on an A3PE3000 FPGA employing a 4×4, and 8×8 multipliers and results are shown in Table 8 and Table 9. The proposed approach utilized intrinsic-based methods and operated on a single FPGA. An HDL-based fault injection simulator was developed specifically for targeting internal elements in the circuit. These artificially induced faults were utilized to assess the system's ability to detect and recover from errors, showcasing its reliability and effectiveness even under fault conditions.

Table 8: Resource Utilization of Existing and Proposed Work

Work	**Fault-Tolerant Multiplier**	**Mechanism**	**Approach**	**Multiplier Size**	**#LUTs (Available-75264)**		**IO Ports (Available-620)**	
					Utilized	**Utilization %**	**Utilized**	**Utilization %**
Existing	EHW Multiplier	Standard VRC	Hybrid (on M1A3PE3000)	4×4	225	0.298%	308	49.6%
				8×8	548	0.728%	614	99%
Proposed	Self-Healing Multiplier	Amended VRC		4×4	75	0.099%	134	21%
				8×8	173	0.229%	286	46%
			Intrinsic (on A3PE3000)	4×4	86	0.114%	142	22.9%
				8×8	194	0.257%	302	48.7%

Table 9: Comparison of Proposed Work Resources Reduction w.r.t Existing Work

Proposed Work	**Approach**	**Multiplier Size**	**LUT Reduction %**	**IO Port Reduction %**
Self-Healing Multiplier	Intrinsic	4×4	61.78%	53.89%
		8×8	64.59%	56.49%
	Hybrid	4×4	66.67%	50.81%
		8×8	68.43%	53.42%

The proposed work has been realized through two distinct approaches: embedded microprocessor-based FPGAs and standalone FPGAs. In the context of the proposed work that employs a genetic algorithm, it was implemented using a hybrid approach on the M1A3PE3000 platform. In this particular implementation, a genetic algorithm was deployed on the ARM processor integrated within the FPGA to generate the bitstream. Additionally, an AVRC multiplier and an error detection unit were instantiated on the FPGA to scrutinize hardware resource utilization. However, this approach received limited appreciation due to its longer error recovery time, as shown in Table 10.

Table 10: Comparison of Error Recovery Time in Existing and Proposed Work

Works	**Mechanism**	**Approach**	**Algorithm**	**Multiplier Size**	**Time**
Existing work	VRC	Extrinsic	Genetic Algorithm Wang and Lee (2014)	3×3	<5 min
			EA-HRS Algorithm Huang *et al.* (2015)	4×4	1350.721 s
		Intrinsic	Genetic Algorithm Suhas *et al.* (2021)	2×2	500 ns
			HS Clone Algorithm Suhas *et al.* (2021)		180 ns
		Hybrid	Genetic Algorithm Shang *et al.* (2020)	4×4	3.2 s
				8×8	5.4 s
	DPR		Genetic Algorithm Cancare *et al.* (2012)	6×6	2.6 hrs
Proposed Work	VRC		Genetic Algorithm	4×4	2.6 s
				8×8	4.8 s
		Intrinsic	CBG Algorithm	4×4	15.25 ns
				8×8	29.25 ns

5 Conclusions

In the proposed work, we initially focused on redundant-based fault-tolerant techniques. Challenges were identified during the implementation of redundant-based techniques with adders and multipliers. In the case of TMR and self-check units, hardware costs increased, leading to delays. However, our preferential approach resulted in decreased hardware costs for TMR-based adders and multipliers, as well as SRA-based adders and multipliers. The PrTMR adder demonstrated a reduction in hardware costs ranging from 29% to 61%, depending on the adder size, compared to the TMR adder. Similarly, the PrSRA-based adder showed a hardware cost reduction ranging from 33% to 60% compared to the TMR adder. For multipliers, PrTMR and PrSRA both contributed to a hardware reduction of 46% to 47%, depending on the multiplier size, compared to TMR. Despite these advancements, limitations such as self-healing in redundant-based techniques persisted. To address this, we proposed a self-healing adder and multiplier by considering evolvable hardware as a foundational study. The self-healing adder reduced hardware costs by 50% compared to existing EHW adders. Similarly, the self-healing multiplier with an intrinsic approach exhibited a hardware reduction of 61% to 64% compared to EHW. Importantly, the self-healing multiplier not only improved fault recovery rates but also enhanced overall performance. These findings collectively highlight the significant advancements achieved in fault-tolerant techniques, with notable improvements in hardware efficiency and fault recovery rates.

6 Organization of the Thesis

(a) Chapter 1: Introduction

(b) Chapter 2: Related Works

(c) Chapter 3: Redundant-Based Fault-Tolerant Technique for Adder

(d) Chapter 4: Redundant-Based Fault-Tolerant Technique for Multiplier

(e) Chapter 5: Self-Healing Adder Design with an Intrinsic Approach

(f) Chapter 6: Self-Healing Multiplier Design with an Intrinsic Approach

(g) Chapter 7: Conclusion and Future Scope

7 List of Publications

I REFEREED JOURNALS BASED ON THE THESIS

i. Sakali Raghavendra Kumar, P Balasubramanian, Ramesh Reddy, Sreehari Veeramachaneni, and Noor Mahammad Sk, *"Optimized Fault-Tolerant Adder Design using Error Analysis"* was published in the Journal of Circuits, Systems and Computers - World Scientific, Volume No. 32, Issue No. 06, Article No. 2350091, Year 2023.

ii. Raghavendra Kumar Sakali, Noor Mahammad Shak, *"Intrinsic based Self-Healing Adder Design Using Chromosome Reconstruction Algorithm"* was published in the Journal of Electronic Testing - Springer, Volume No. 39, Year 2023

iii. Raghavendra Kumar Sakali, Sreehari Veeramachaneni, Noor Mahammad SK. *"Preferential Fault-Tolerance Multiplier Design to Mitigate Soft Errors in FPGAs"* was published in the Journal of VLSI Integration - Elsevier, Volume No. 93, Article No. 102068, Year 2023.

iv. Raghavendra Kumar Sakali, Noor Mahammad Sk. *"Intrinsic-based Self-Healing Multiplier with Configuration Bitstream Generator"* was communicated to IEEE Transactions on Circuits and Systems I: Regular Papers).

II PUBLICATIONS IN CONFERENCES BASED ON THE THESIS

i. S Raghavendra Kumar, Noor Mahammad Sk, S Balasubramanian, and G Venkat Reddy, *"Novel Embryonics Adder Architecture with Unicellular Self-Check Unit"*, in 2nd International Conference on Emerging Trends in Engineering (ICETE 2023)

ii. Raghavendra Kumar Sakali, Sreehari Veeramachaneni, Noor Mahammad SK, *"Fault-Tolerant Floating-Point Multiplier Design for Mission Critical Systems"*, proceedings of the 37^{th} International Conference on VLSI Design and 23^{rd} International Conference on Embedded Systems, Jan 2024.

iii. Raghavendra Kumar Sakali, Sreehari Veeramachaneni, Noor Mahammad SK, *"Preferential Fault-Tolerant Based Bfloat16 Multiplier"*, communicated to IEEE International Symposium on Circuits & Systems 2024.

References

1. **Cancare, F.**, **D. B. Bartolini**, **M. Carminati**, **D. Sciuto**, and **M. D. Santambrogio** (2012). On the evolution of hardware circuits via reconfigurable architectures. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, **5**(4), 1–22.

2. **Huang, X.**, **N. Wu**, **X. Zhang**, and **Y. Liu** (2015). An evolutionary algorithm based on novel hybrid repair strategy for combinational logic circuits. *IEICE Electronics Express*, 12–20150765.

3. **Jian, G.** and **Y. Mengfei** (2018). Evolutionary fault tolerance method based on virtual reconfigurable circuit with neural network architecture. *IEEE Transactions on Evolutionary Computation*, **22**(6), 949–960, `doi:10.1109/TEVC.2017.2779874`.

4. **Kumar, P.** and **R. K. Sharma** (2016). Real-time fault tolerant full adder design for critical applications. *Engineering science and technology, an international journal*, **19**(3), 1465–1472.

5. **Li, Y.**, **A. Breitenreiter**, **M. Andjelkovic**, **J. Chen**, **M. Babic**, and **M. Krstic** (2020). Double cell upsets mitigation through triple modular redundancy. *Microelectronics Journal*, **96**, 104683.

6. **Palsodkar, P.**, **P. Palsodkar**, and **R. Giri** (2018). Multiple error self checking-repairing fault tolerant adder-multiplier. *In 2018 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. IEEE.

7. **Shang, Q.**, **L. Chen**, **J. Cui**, and **Y. Lu** (2020). Hardware evolution based on improved simulated annealing algorithm in cyclone v fpsocs. *IEEE Access*, **8**, 64770–64782.

8. **Suhas, S.**, **G. Malhotra**, and **V. Rajini** (2021). Hsclone genetic algorithm implementation on a combinational circuit. *IETE Journal of Research*, 1–9.

9. **Wang, J.** and **C.-H. Lee** (2014). Virtual reconfigurable architecture for evolving combinational logic circuits. *Journal of Central South University*, **21**(5), 1862–1870.

10. **Zhao, Z.**, **N. T. Nguyen**, **D. Agiakatsikas**, **G. Lee**, **E. Cetin**, and **O. Diessel** (2018). Fine-grained module-based error recovery in fpga-based tmr systems. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, **11**(1), 1–23.