(Online) ISSN 2712-0562
Sustainable Engineering and Innovation *Original Research*
Vol. 7, No. 1, 2025, pp.117-126
https://doi.org/10.37868/sei.v7i1.id384

A programmable multi-bit fault injection for embedded system

Rahul Shandilya[1*], R. K. Sharma[2]
[1] School of VLSI Design and Embedded Systems, NIT Kurukshetra-136119, Haryana, India
[2] Dept. of Electronics and Communication Engineering, NIT Kurukshetra-136119, Haryana, India

*Corresponding author E-mail: rss.nitk@gmail.com

Received Oct. 7, 2024
Revised Feb. 14, 2025
Accepted Feb. 25, 2025
Online Mar. 12, 2025

Abstract

Fault injection techniques are commonly used to intentionally introduce attack on embedded systems, specifically advanced Field Programmable Gate Arrays (FPGAs) and microcontrollers. The FPGA-based embedded system uses SRAM for storage of configuration data. Due to technology scaling and growing complexity in FPGA bit files, multiple-bit upset is a primary threat to FPGAs. These devices are also vulnerable to radiation threats in space environments. Currently existing methods have the limitation of low rate of fault injection and the inability to fault inject at the transistor level. To address these issues, this paper proposes burst error modeling and a Fault Injection Self-test Hardware (FISH). FPGA is utilized in the proposed fault injection architecture to efficiently inject Multiple-Bit Upset (MBUs) onto the design's interconnect without altering the value of flip-flops associated with the design path. There is no need to reload the same flops and memory with correct values since their values are unchanged. The AMD Virtex 7 XCV2000T FPGA has been used to evaluate the proposed FISH architecture. Results show that FISH is 2x faster than existing techniques, and it uses only ~4 % CLB overhead for target FPGA. The FPGA resource utilization overhead is also less as compared to other exiting designs, but it depends on the number of fault injection points used. Future research has the potential to examine the use of the True Random Number Generator (TRNG), which operates on a physical phenomenon that is unpredictable.

Keywords: Fault Injection, FPGA, Single Event Upset, Multiple-Bit Upset, LFSR

1. Introduction

Injection of faults is a typical method for assessing a system's ability to handle physical faults [1]. Due to its low power consumption and flexible programming, the Field Programmable Gate Array (FPGA) is commonly used in practical field applications. The FPGA has reconfigurable logic, I/O, and connecting blocks unlike Von Neumann-type devices such as microcontrollers and DSP processors [2]. The term 'hostile environment' is commonly used when referring to environments that could be a hindrance to the reliable operation of field-programmable gate arrays (FPGAs). Testing system resilience is frequently done when systems are implemented in hostile environments where errors are likely to occur [3]. Considering the potential uses of

radiation-tolerant circuits, such as space missions, satellites, and high-energy physics experiments, there has been an interest in investigating fault-tolerant approaches to keep integrated circuits (ICs) working in hostile environments [4].

FPGAs can be used to simulate defects in electronic systems, which is known as an FPGA-based fault injection system technique. A wide range of digital logic operations can be carried out through FPGAs, integrated circuits that can be programmed after manufacturing [5]. To examine the dependability and efficiency of digital systems, FPGA-based fault injectors are used. Safety systems, such as medical devices, aircraft, and automobiles, require this application. By intentionally flipping several bits in the data memory or configuration of the FPGA, FPGA-based fault injectors can be used to conduct experiments simulating the impact of MBUs. By introducing faults based on MBU models, researchers can evaluate how the FPGA behaves in practical fault scenarios and validate MBU mitigation strategies.

The modeling of multiple-bit upset (MBU) aims to study and simulate scenarios where multiple bits in a memory unit get corrupted simultaneously. Data corruption can result from flipping multiple bits in a memory cell, which happens due to high-energy particles, radiation, or other environmental factors [6]. Error detection and correction mechanisms are developed through MBU modeling to reduce the impact of such faults. High-energy physics and aerospace applications for FPGAs have become more popular in the past decade. FPGAs are popular for these applications because of their many advantages, including greater adaptability, cheap cost, and fast turnaround time. This is particularly true when compared to more expensive, specialized alternatives, such as integrated circuits that are designed specifically for specific applications [7].

The utilization of commercial SRAM-based FPGAs in radiation settings is now common because they perform better and are cheaper than radiation-hardened FPGA systems. The cost of commercial SRAM-based FPGAs is undoubtedly lower than radiation-hardened FPGA solutions, but they will either perform better or worse depending on the application's specific needs and limitations and the radiation environment [8]. Commercial FPGAs based on SRAM have higher production numbers and wider market availability, making them more economical than radiation-hardened FPGAs [9]. The use of advanced techniques in silicon manufacturing leads to elevated frequencies and superior performance. Furthermore, FPGAs are machines that are capable of programming. During development, their behavior can be altered to meet different mission objectives [10]. There is a gap in understanding between hardware and software-based fault injection vulnerability detection. Fault injection vulnerabilities can be detected using both hardware and software. An EMP generator is used to achieve hardware-based detection [11].

Considering the significance of reconfiguration tasks in FPGA applications, it is crucial to fully examine the effects of SETs during configuration memory re-writing. An approach has been proposed to evaluate the impact of SET pulses on reconfiguring configuration memory in SRAM-based FPGAs [12]. Additionally, SRAM-based FPGAs have a greater number of memory elements than their ASIC counterparts, which makes them more prone to Single Event Upset (SEU). The higher operating voltages of early SRAMs made them more resistant to soft errors. On the flip side, the node capacitance and operating voltage decrease with every new generation of SRAM [13, 14]. To overcome these challenges, a new architecture for fault injection (FI) has been proposed by utilizing burst error modeling and Fault Injection Self-test Hardware (FISH).

- The built-in Instrumented FPGA-based FI allows for the effective injection of MBUs into the configuration memory of FPGAs.
- Soft error estimation accuracy is ensured by using an adaptive rate for FI.

To evaluate the speedup of the proposed technique, it is necessary in evaluation to test FISH on AMD Virtex 7 XCV2000T target FPGA against different fault injection techniques on the OR1200 processor design which is used as workloads. This research work is organized as follows the following sections. In Section 2, we examined previous research on fault injections and detection technique. In Section 3, there is a detailed explanation of the

proposed approach. In Section 4, the outcome and discussion are discussed. Section 5 provides a summary as well as directions for future research.

2. Literature survey

This section discusses various fault injection method for FPGA-based embedded systems. Velayaudhan et. al. proposed a method BUFIT [15] for injection faults through Built-in circuit inside FPGA for MBUs in configuration memory with adaptive rate up to 53.4 faults/sec. The limitation of this method is the low rate of fault injection. There is a lot of scope to increase the fault injection rate. Lanzieri L. et al. [16] suggested that embedded systems could detect and monitor age. Embedded devices that play crucial roles in reliability or safety-critical applications are experiencing an increasing problem with hardware aging. This work's primary objective is to make future research efforts easier by coordinating all major approaches. Metawie H. et al. [17] proposed a method for introducing faults through the Quick EMUlator (QEMU). The fault model for memory coupling problems can be extended by simulating faults in the control and execution channels of an ARM processor. Their evaluation of a memory exam demonstrates the usefulness of the approach. The limitation of this method is fault injection in simulation environment only and fault injection at application layer, not at transistor level. Richter-Brockmann, J. et al. [18] proposed revisiting hardware faults in adversary models. Moreover, the use of customized models makes it more difficult to compare different designs and evaluate results. Additionally, it demonstrates that the recommended adversary model can be applied to VerFI, a state-of-the-art fault-proof tool.

Claudepierre, L., and his co-authors came up with a TRAITOR platform [19] that is inexpensive and able to generate precise bursts of faults with the help of clock glitches. The errors are caused by the injection of clock glitches, which have high repeatability and reliability. This platform is inexpensive, simple to use, and capable of injecting many spurts of faults. Future development will extract an exact fault model for TRAITOR using the STM32F100RB board. Furthermore, the investigation of software or hardware counter measures is being explored. This method cannot be used for gate level fault injection because it is based on clock glitch and limited to flip-flops level. Liao H. et al. [20] reported that electromagnetic fault injection (EMFI) techniques have a major impact on the security of embedded devices. The idea behind this paper is to develop a new EMFI backside technique that utilizes overclocking and an expanded fault model that incorporates the concept of critical charge. This research plays a major role in the security and fault injection resistance of embedded processors and their instruction set designs. The study's funding is partially supported by contributions from XtremeEDA and NSERC. Breier J. et al. [21] developed a new way to shield implementations from SIFA by using error-correcting codes. They developed an electronic logic analysis instrument that examines the output for errors, recursively runs through all possible inputs, and injects a stuck-at-fault at each gate in the circuit.

Cerveira F. et al. [22] proposed the analysis of exploratory data obtained from fault injection campaigns. The essay utilizes exploratory (big) data analysis techniques, tools, and approaches to organize and execute information extraction in a contemporary perspective on these problems. This has led to the discovery of a previously undiscovered possibility for accelerating the FI process. Several methods were employed to introduce errors in both single-bit and multiple-bit scenarios. The challenges it encounters include the immediate practical need to mitigate hardware aging in current systems and more complex fault models beyond clock glitches. To address these issues, a new FISH has been proposed in this work.

3. Proposed fault injection model

This section describes a simulation framework for MBU injection and its associated design methodology. To reduce the accumulation of MBUs, early estimation of FPGA's sensitivity to run-time MBUs is crucial. This allows for exploring MBU modeling and anticipating its design before implementing an efficient MBU injector. Modeling an event-driven simulator and including functional models for the FPGA is the basis of the MBU injection framework shown in Figure 1. The roadblock rate, MBU injection rate, frame address, and fault list

are all able to be redefined by the designer. Real-time fault injection experiments in dynamic radiation environments can be virtualized using this scalable, configurable, and versatile framework. The FPGA configuration memory contents can be artificially changed through in-built FI to emulate radiation-induced MBUs. To determine how a configuration memory upset will affect the originally implemented design behavior, the FPGA's output is monitored.

To develop an effective FI technique, it is important to have knowledge of different fault models; this knowledge will be different for different FPGA resources. Failed routing resources in FPGAs can be induced by using the Stuck-at-1 or Stuck-at-0 models, and the bit flip fault model can cause faults in FPGAs' memory resources. The radiation experiment conducted recently has revealed that over 48% of the faults are caused by MBUs [23]. The memory unit can experience a maximum of 8-bit upsets per word, and 2-bit, 3-bit, and 4-bit upsets play an important role.

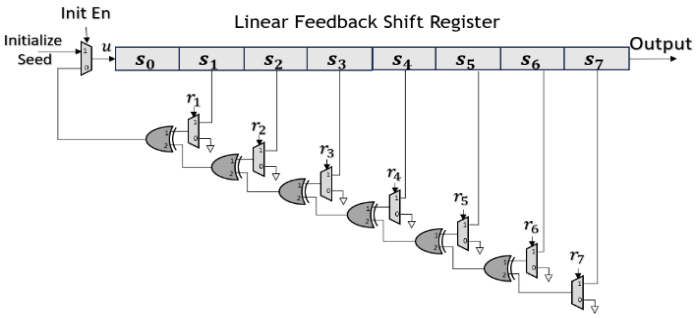

Figure 1. Programmable LFSR circuit

Linear feedback shift register (LFSR) is a shift register that feeds the input with a function of its previous states. The initial value of the LFSR is called the seed. The usage of LFSRs involves generating whitening sequences, pseudo-noise sequences, pseudo-random numbers etc. The feedback function determines a new bit depending on the state of certain taps (selected bits) in the shift register. The choice of taps is based on the LFSR's characteristic polynomial. In general, the feedback function is an exclusive OR of the values of the tapped bits in the shift register. This produces shift register output and it becomes the new input to shift register, and the process continues until the desired number of bits is shifted. In our design, LFSR has programmable taps so that it can operate different shift functions right.

The polynomial equation for 8-bit LFSR is given below which is used to mimic run-time radiation environment in the experiments. Finite field arithmetic can be used to express the configuration of taps for feedback in an LFSR as a polynomial mod 2. The coefficients of the polynomial are required to be 1s or 0s. This is referred to as the reciprocal characteristic polynomial. or feedback polynomial. In the presence of taps at the 7th and 6th bits, the feedback polynomial will be:

$$x^7 + x^6 + 1 \tag{1}$$

In Equation 1, 'one' is a reference to the input of the first bit, i.e. x^0 which is equivalent to 1. The tapped bits are represented by the powers of the terms, counted from the left. Examples of feedback polynomials for shift registers with lengths up to 8 are shown in Table 1.

Table 1. Feedback polynomial functions for programmable taps

Programable Taps $(r_{7\to1})$	**Feedback polynomial**
1100000	$x^7 + x^6 + 1$
110000	$x^6 + x^5 + 1$
10100	$x^5 + x^3 + 1$
1100	$x^4 + x^3 + 1$
110	$x^3 + x^2 + 1$
11	$x^2 + x^1 + 1$

In following LFSR architecture, the input u is set to some the state bits XOR-ed together. its state equation is:

$$u[k] = \bigoplus_{j=0}^{N-1} b_j S_j[k] \quad (2)$$

In Equation 2, coefficients b_j that are either 0 or 1, and symbol $\oplus$ means to XOR all of its inputs together.

Table 2. Percentage of occurrence of fault value generated by proposed LFSR-based MBU model

Fault Type		Occurrence %
Random value generated without fault		~ 23 %
Random Value generated with fault	7th bit position	~13%
	6th bit position	~7%
	5th bit position	~8%
	4th bit position	~9%
	3rd bit position	~12%
	2nd bit position	~11%
	1st bit position	~8%
	0th bit position	~9%

The seed of LFSRs can be either constant or varied. Table 2 shows the modeling of MBUs. The four 8-bit LFSRs are connected in parallel to generate 32-bit random data. The proposed design can be used to upset 1-bit, 2-bit, 3-bit and 4-bit based on the configuration set by the user. Table 2 shows the fault generation rate for the proposed fault injection model, which was recorded for 1M execution times when a single bit flip in a byte was detected.

4. Proposed fault injection self-test hardware

FISH is capable of to create faults in any place inside the circuit with the help of FI element in the design (Figure 3). It can perform single bit and multi bit upset sequence using LFSR. Proposed fault models can be derived from LFSR's seed and feedback polynomial vectors. 32-bit fault injector circuit in Figure 2 is composed of four 8-bit fault injector circuits that are connected to inject a 32-bit fault at a time.

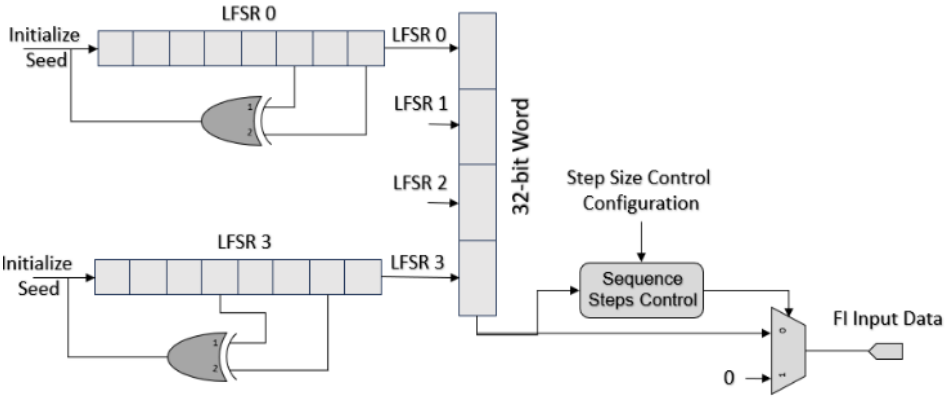

Figure 2. Fault injection sequence generation

It is possible for the fault injector model to inject single, double, triple, or four-bit upsets. The percentage of occurrence of fault value generated by fault injector model is given in Table 2. There is a masking logic to control the steps count to generate the configurable count of bit upset. By selecting error data for FISH and error-free data for normal operation, the configuration signal named as 'FI Enable' is used. 'FI input' can be fed in the fault injection chain sequence at the positive edge of clock. When all the fault input value is reached at desired location of flops then 'FI enable' signal is set to generate the fault at the interconnect where fault injection element placed.

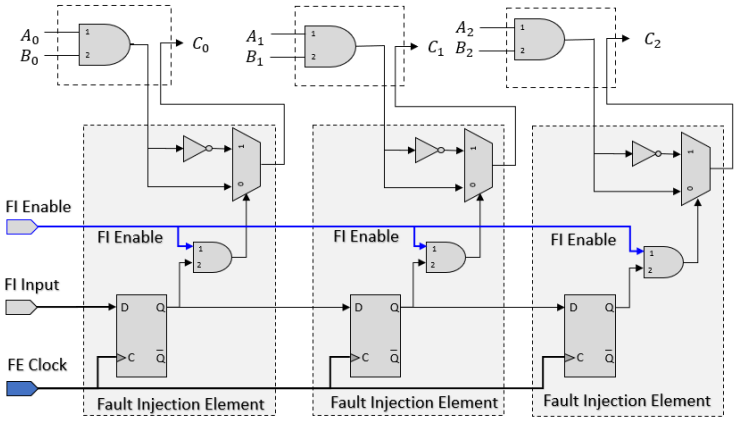

Figure 3. Fault injection chain build using fault injection elements at output of AND logic gates inside TMR logic

The Fault Injection input sequence that feeds in the design can be read-back from Serial in Parallel Out (SIPO) as shown in Figure 4. This captured data is evaluated on the basis of numbers of one and used for classification of fault that generated during the whole process.

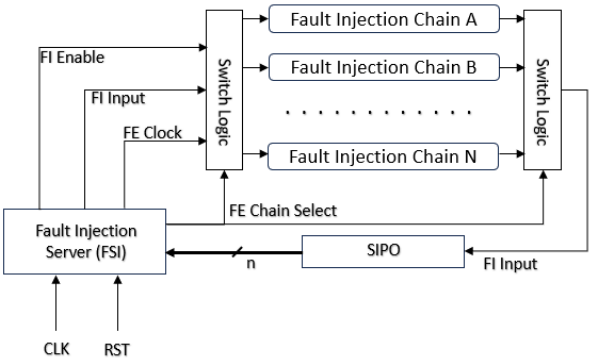

Figure 4. FISH block diagram with SIPO

4. Results and discussion

The proposed FISH is designed to generate faults using an FPGA and is suited for embedded systems. Different radiation environments and memory faults are simulated in the FISH injection framework by FPGA emulation. Fault injection logic has been added to the interconnect based on the fault list identified in the design to simulate faults in different memory locations.

Table 3. Fault injection rate

Injection method	Initialization time (ms)	Write sequence time (ms)	Injection Frequency	
			Time (ms)	Rate (Fault/Sec)
BUFIT	0.7	18	18.7	53.4
SCFIT	18	18	36	27
DPR	-	54	54	18.5
Proposed FISH	5	4	9	111.1

Table 3 compares various fault injection methods in terms of their performance measured at the same operating frequency. The performance of the proposed FISH method compared to BUFIT, DPR [24] and SCFIT [25]. FISH demonstrates a significantly lower total injection time and a higher injection rate, suggesting that it can inject faults more rapidly and efficiently. FISH has the shortest injection time compared to DPR, SCFIT and BUFIT. As shown in Table 3.

Table 4: Speed improvement comparison of fault injection techniques

Workload	DPR (ms)	SCFIT (ms)	BUFIT (ms)	Proposed FISH (ms)	Speed improvement in comparison to :		
					DPR	**SCFIT**	**BUFIT**
Counter	1944	1296	673	303	~6X	~4X	~2X
Bubble sort	7779	5184	2693	1220	~6X	~4X	~2X
4-bit adder	1466	983	512	245	~6X	~4X	~2X
4-bit multiplier	3122	2042	1064	496	~6X	~4X	~2X

The comparison of fault injection time and speed improvement for different workloads is shown in Table 4. Logic for Counter, bubble sort circuit, 4-bit adder and 4-bit multiplier circuits are also implemented with the proposed FISH architecture and compared with existing techniques. The speed of the proposed technique has been enhanced by 2 times, 6 times, and 4 times compared to the existing BUFIT, DPR, and SCFIT, respectively. The initialization delay of 10 clocks is the shortest offered by FISH. The design has eight FI elements, causing a write delay of 8 clocks, leading to a total injection time of 18 clocks. This method achieves the highest injection frequency at 100Mhz in the FPGA, making it the most efficient in injecting faults quickly and frequently. The Table shows that the proposed FISH method has higher fault injection rate and lower injection time compared to BUFIT, DPR and SCFIT, suggesting that FISH is the better option for applications that require quick and frequent fault injections.

Table 5. FPGA resources overhead due to build-in Fault Injection Server

Resource	**OR1200 without FISH**	**OR1200 with FISH**	**% Overhead**		
			OR1200 + FISH	**OR1200 + BUFIT**	**OR1200 + SCFIT**
CLB	3826	3991	4.30%	0.40%	4.80%
FF	2319	2414	4.10%	~0%	5.80%

Different workloads are compared in Table 4 by comparing fault injection times using three techniques: DPR, SCFIT, and the proposed FISH. Table 5 presents the analysis of FPGA resource overhead that is caused by the addition of an FI instrument to the target FPGA. The SCFIT technique requires an extra 6% of CLBs (Configurable Logic Blocks) and 5.8% of FFs (Flip Flops) to be added in addition to the FPGA target. The maximum available FFs in the AMD Virtex 7 XCV2000T FPGA are not enough to meet the required FFs. This outcome demonstrates the practical limitations of utilizing the SCFIT method. The proposed FISH does not have any limitations as compared to existing methods and requires a smaller number of CLBs and FFs resources inside FPGA.

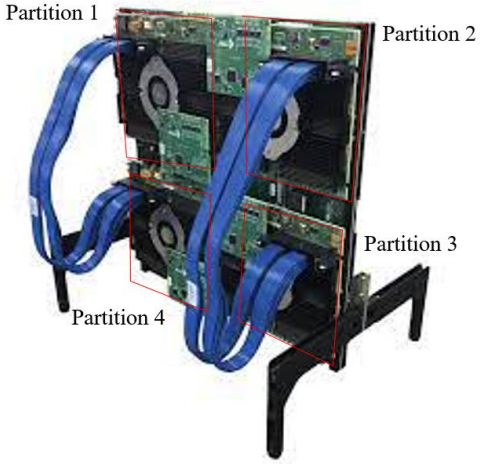

Figure 5. FPGA Environment Setup

A proFPGA quad motherboard was used for prototyping that can accommodate four FPGA tiles with AMD Virtex 7 XCV2000T FPGA devices, shown in Figure 5, which offer scalable high-performance and Multi-FPGA Solutions, which can be easily adapted and expanded by further extensions cards equipped with interconnections, interfaces or memories. It supports various high-speed communication standards, like PCI, Ethernet and USB2.0. With its simple script-based system configuration and launch technique, it allows quick and reproducible test runs.

5. Conclusions

This paper introduces a new FPGA-based technique called FISH, which is based on AMD Virtex 7 XCV2000T. It is capable of producing SEU and MBU. The real-time radiation environments can be emulated by modifying the sizes of these MBUs. MBUs are injected into the memory elements of an FPGA by means of the proposed FI element. Results in the OR1200 processor integration show that FISH is two orders of magnitude faster than existing techniques, and it uses only ~4 % CLB overhead for target FPGA. The aim of this work is to enhance fault injection performance on FPGA by implementing additional features with the proposed FISH. In the future scope of this architecture, LFSRs' deterministic nature and finite cycle length should be solved by utilizing the integration of True Random Number Generator (TRNG) that function is based on an unpredictable physical phenomenon.

Abbreviations

- FPGA - Field Programmable Gate Array
- SRAM - Static Random-Access Memory
- MBU - Multiple-Bit Upset
- FISH - Fault Injection Self-test Hardware
- AMD - Advanced Micro Devices
- CLB - Configurable Logic Block
- TRNG - True Random Number Generator
- SEU - Single Event Upset
- LFSR - Linear Feedback Shift Register
- BUFIT - Built-in Fault Injection Technique
- QEMU - Quick Emulator
- EMP - Electromagnetic Pulse
- SET - Single Event Transient
- TRAITOR - A platform used for generating precise bursts of faults via clock glitches
- SIFA - Software-implemented Fault Attack
- EMFI - Electromagnetic Fault Injection
- NSERC - Natural Sciences and Engineering Research Council of Canada
- ICs - Integrated Circuits
- ASIC - Application-Specific Integrated Circuit

Declaration of competing interest

The authors declare that they have no known financial or non-financial competing interests in any material discussed in this paper.

Funding information

The authors declare that they have received no funding from any financial organization to conduct this research.

Author contribution

The contribution to the paper is as follows: Rahul Shandilya: Study conception and design, literature review, analysis and interpretation of results, writing and editing of the manuscript.; R.K. Sharma: Drafted research outline, project plan, methodology, and provided critical feedback on manuscript content while supervising project execution. All authors approved the final version of the manuscript.

References

[1] A. Gangolli, Q. H. Mahmoud, and A. Azim, "A Systematic Review of Fault Injection Attacks on IoT Systems," *Electronics*, vol. 11, no. 13, p. 2023, Jun. 2022. doi: https://doi.org/10.3390/electronics11132023.

[2] Z. Gao and X. Liu, "An Overview on Fault Diagnosis, Prognosis and Resilient Control for Wind Turbine Systems," *Processes*, vol. 9, no. 2, p. 300, Feb. 2021. doi: https://doi.org/10.3390/pr9020300.

[3] M. Carminati and G. Scandurra, "Impact and trends in embedding field programmable gate arrays and microcontrollers in scientific instrumentation," *Review of Scientific Instruments*, vol. 92, no. 9, p. 091501, Sep. 2021. doi: https://doi.org/10.1063/5.0050999.

[4] A. Appathurai and P. Deepa, "Design for reliablity: A novel counter matrix code for FPGA based quality applications," *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, pp. 56–61, Aug. 2015. doi:10.1109/acqed.2015.7274007.

[5] A. Appathurai and P. Deepa, "Design for reliablity: A novel counter matrix code for FPGA based quality applications," *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, pp. 56–61, Aug. 2015. doi:10.1109/acqed.2015.7274007 .

[6] S. Medjmadj, D. Diallo, and A. Arias, "Mechanical sensor fault-tolerant controller in PMSM drive: experimental evaluation of observers and signal injection for position estimation," *Revue Roumaine Des Sciences Techniques—Série Électrotechnique Et Énergétique*, vol. 66, no. 2, pp.77-83, 2021.

[7] Ahilan A. and Deepa P., "Modified decimal matrix codes in FPGA configuration memory for multiple bit upsets," *2015 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, Jan. 2015. doi:10.1109/iccci.2015.7218146.

[8] T. Given-Wilson, N. Jafri, and A. Legay, "Combined software and hardware fault injection vulnerability detection," *Innovations in Systems and Software Engineering*, vol. 16, no. 2, pp. 101–120, Jun. 2020. doi:10.1007/s11334-020-00364-5.

[9] S. Mandal, S. Sarkar, W. M. Ming, A. Chattopadhyay, and A. Chakrabarti, "Criticality aware soft error mitigation in the configuration memory of SRAM based FPGA," *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, pp. 257–262, Jan. 2019. doi:10.1109/vlsid.2019.00063.

[10] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets Vulnerability Analysis of modern microprocessors," *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 119–130, Nov. 2019. doi:10.1109/iiswc47752.2019.9042036 .

[11] R. V. Wicaksana Putra, M. A. Hanif, and M. Shafique, "Respawn: Energy-efficient fault-tolerance for spiking neural networks considering unreliable memories," *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, Nov. 2021. doi:10.1109/iccad51958.2021.9643524 .

[12] I. Bentchikou, K. Halbaoui, F. Boudjema, D. Boukhetala, and T. Abdelhalim, "Alternative hybrid control of switched systems, an application to machine dc fed by multicellular converter," *Revue Roumaine Des Sciences Techniques—Série Électrotechnique Et Énergétique*, vol. *67, no.* 3, pp.247-252, 2022.

[13] B.M. Kumar, J. Ragaventhiran and V. Neela, "Hybrid optimization integrated intrusion detection system in WSN using ELMAN network," International Journal of Data Science and Artificial Intelligence, vol. 02, no. 02, pp. 55–62 , 2024.

[14] M. Anisha and V.A. Beenu, "Double secure cloud medical data using Euclidean distance-based Okamoto Uchiyama homomorphic encryption," *International Journal of System Design and Computing*, vol. *2, no.* 1, pp.1-7, 2024.

[15] S. T. Velayaudhan and K. Devi, "BUFIT: Fine-grained dynamic burst fault injection tool for embedded field programmable gate array testing," *Revue Roumaine Des Sciences Techniques — Série Électrotechnique Et Énergétique*, vol. 69, no. 3, pp. 303–308, Sep. 2024. doi:10.59277/rrst-ee.2024.69.3.8

[16] L. Lanzieri et al., "A review of techniques for ageing detection and monitoring on embedded systems," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–34, Oct. 2024. doi:10.1145/3695247.

[17] H. Metawie, M. Safar, and M. Watheq El-Kharashi, "An evaluation method for embedded software dependability using QEMU-based fault injection framework," *2022 6th International Conference on System Reliability and Safety (ICSRS)*, pp. 548–555, Nov. 2022. doi:10.1109/icsrs56243.2022.10067534.

[18] J. Richter-Brockmann, P. Sasdrich, and T. Guneysu, "Revisiting fault adversary models – hardware faults in theory and Practice," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 572–585, Feb. 2023. doi:10.1109/tc.2022.3164259 .

[19] Ludovic Claudepierre, Pierre-Yves Péneau, D. Hardy, and Erven Rohou, "TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection," *HAL (Le Centre pour la Communication Scientifique Directe)*, pp. 51–56, May 2021. doi: https://doi.org/10.1145/3457340.3458303.

[20] H. Liao and C. Gebotys, "Methodology for EM Fault Injection: Charge-based Fault Model," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, pp. 256–259, Mar. 2019. doi: https://doi.org/10.23919/date.2019.8715150.

[21] J. Breier, M. Khairallah, X. Hou, and Y. Liu, "A Countermeasure Against Statistical Ineffective Fault Analysis," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3322–3326, Dec. 2020. doi: https://doi.org/10.1109/tcsii.2020.2989184.

[22] F. Cerveira, I. Kocsis, R. Barbosa, H. Madeira, and A. Pataricza, "Exploratory Data Analysis of Fault Injection Campaigns," *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Jul. 2018. doi: https://doi.org/10.1109/qrs.2018.00033.

[23] A. Ramos, R. G. Toral, P. Reviriego, and J. A. Maestro, "An ALU Protection Methodology for Soft Processors on SRAM-Based FPGAs," *IEEE Transactions on Computers*, vol. 68, no. 9, pp. 1404–1410, Mar. 2019. doi: https://doi.org/10.1109/tc.2019.2907238.

[24] Reza Omidi Gosheblagh and K. Mohammadi, "Three-Level Management Algorithm to Increase the SEU Emulation Rate in DPR Based Emulators," *Journal of Electronic Testing*, vol. 30, no. 6, pp. 739–749, Oct. 2014. doi: https://doi.org/10.1007/s10836-014-5489-x.

[25] A. Mohammadi, M. Ebrahimi, Alireza Ejlali, and Seyed Ghassem Miremadi, "SCFIT: a FPGA-based fault injection technique for SEU fault model," *Design, Automation, and Test in Europe*, pp. 586–589, Mar. 2012. doi: https://doi.org/10.5555/2492708.2492854.