International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 40-46
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

FIS: A Programmable Multi-Bit Fault Injection Server for Embedded System

Rahul SHANDILYA[1]*, R.K. SHARMA[2]

[1]School of VLSI Design and Embedded Systems, NIT Kurukshetra-136119, Haryana, India
***Corresponding Author Email:** rss.nitk@gmail.com - **ORCID:** 0009-0006-3073-092X

[2]Dept. of Electronics and Communication Engineering, NIT Kurukshetra-136119, Haryana, India
Email: mail2drks@gmail.com - **ORCID:** 0009-0009-2430-669X

Article Info:

DOI: 10.22399/ijcesen.674
Received : 21 October 2024
Accepted : 05 December 2024

Keywords :

Fault Injection,
FPGA,
Single Event Upset,
Multiple-Bit Upset,
LFSR.

Abstract:

The Fault injection technique is commonly used to intentionally introducing attack on embedded systems, specifically advanced 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. To address these issues, this paper proposes burst error modeling and a Fault Injection Server (FIS). 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 Xilinx Zynq-7000 FPGA has been used to evaluate the proposed FIS architecture, and It is able to perform two times faster than existing techniques. The FPGA resource utilization overhead also less as compared to other exiting design but it depends on number of fault injection points used.

1. Introduction

Injection of faults is a typical method for assessing a system's ability to handle physical faults [1-5]. 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 [6-10]. 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 [11]. 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 [12-23].

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 [17]. 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 [12]. 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 [24].

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 [15]. Commercial FPGAs based on SRAM have higher production numbers and wider market availability, making them more economical than radiation-hardened FPGAs. [18].

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. [20]. 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 [13].

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. [6]. 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 [2,3]. To overcome these challenges, a new architecture for fault injection (FI) has been proposed by utilizing burst error modeling and fault injection server (FIS).

• 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 FIS on Zynq-7000 target FPGA against different fault injection technique on the OR1200 processor design which is used as workloads.

This research work is organized as the following sections. In Section 2, we examine previous research on fault injection 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. Material and Methods

2.1 Related Works

This section discusses various fault injection method for FPGA-based embedded systems.

In 2024, Velayaudhan et. al. [4] proposed a method for injection faults through Built-in circuit inside FPGA for MBUs in configuration memory with adaptive rate up to 53.4 faults/sec. There is a lot of scope to increase the fault injection rate.

In 2023, Lanzieri L. et al. [1] 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.

In 2022, Metawie H. et al. [8] 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.

In 2022, Richter-Brockmann, J. et al. [7] 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.

In 2021, Claudepierre, L., and his coauthors came up with a TRAITOR platform 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.

Liao H. et al. [21] reported in 2019 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.

In 2020, Breier J. et al. [16] 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 in 2018. 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 FIS has been proposed in this work.

2.2 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 [19]. 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.

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 right functions.

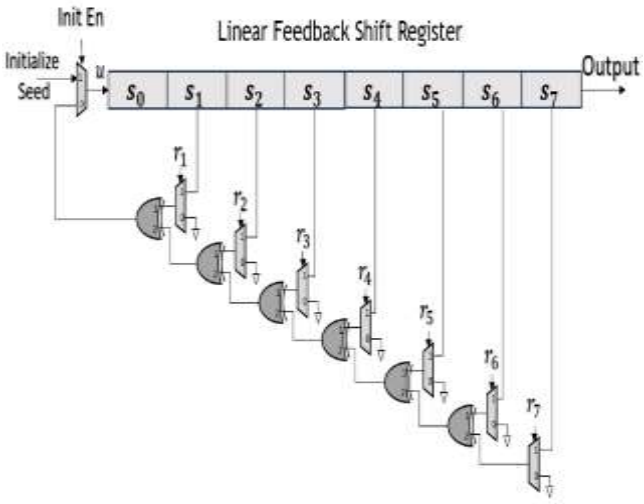

Figure 1. *Programmable LFSR circuit*

The polynomial equation for 8-bit LFSR is given below which 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:

$$x^7 + x^6 + 1 \quad (1)$$

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 the 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:

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

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	7^{th} bit position	~13%
	6^{th} bit position	~7%
	5^{th} bit position	~8%
	4^{th} bit position	~9%
	3^{rd} bit position	~12%
	2^{nd} bit position	~11%
	1^{st} bit position	~8%
	0^{th} bit position	~9%

The seed of LFSRs can be either constant and 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.

2.3 Proposed Fault Injection Server

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 together to inject a 32-bit fault at a time. FIS is capable of to create fault in any place inside the circuit with the help of FI element in the design, (figure 3).

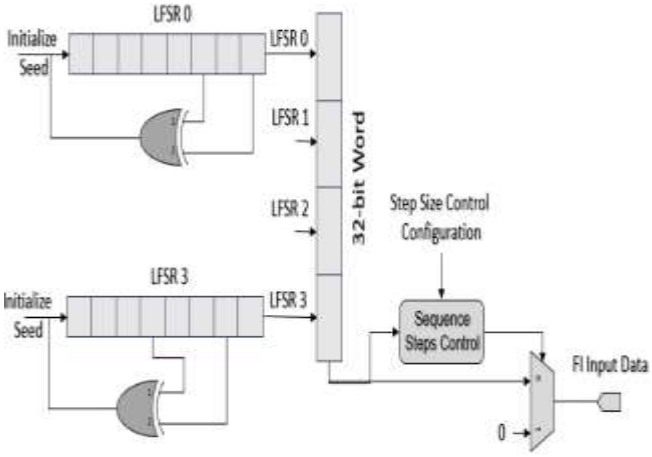

Figure 2. *Fault injection sequence generation*

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 FIS and error-free data for normal operation, the configuration signal named as 'FI Enable' is used. 'FI input' can be feed 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. The Fault Injection input sequence that feed in the design can be read-back from data register 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.

3. Results and Discussions

The proposed FIS is designed to generate faults using an FPGA and is suited for embedded systems.

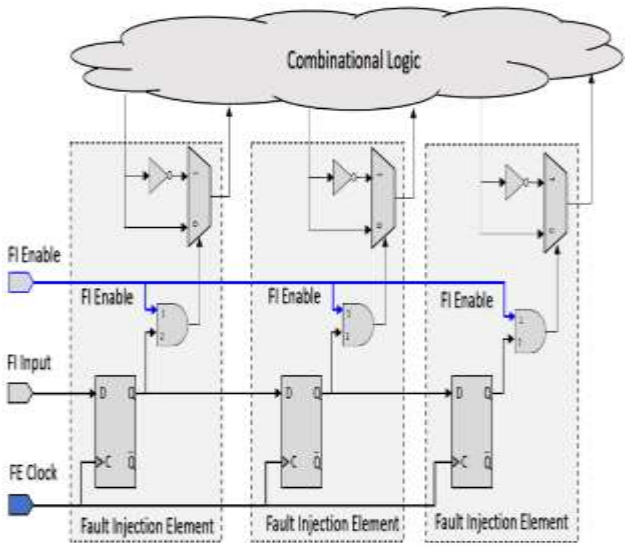

Figure 3. *Fault injection chain build using fault injection elements*

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 FIS	5	4	9	111.1

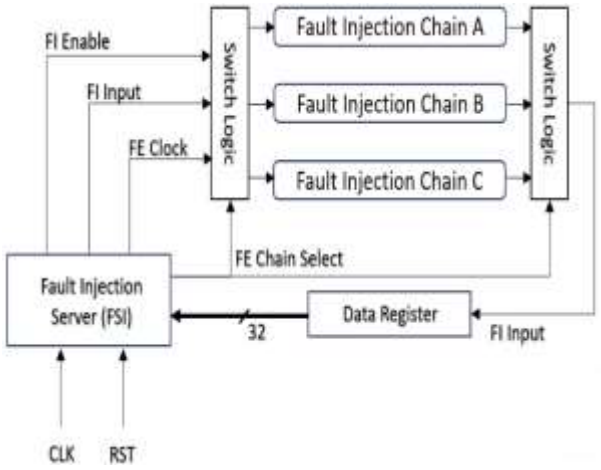

Figure 4. *Fault injection server block diagram*

Different radiation environments and memory faults are simulated in the FIS 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 compares various fault injection methods in terms of their performance measured at the same operating frequency. The performance of the proposed FIS method compared to BUFIT, DPR and SCFIT. FIS demonstrates a significantly lower total injection time and a higher injection rate, suggesting that it can inject faults more rapidly and efficiently. FIS has the shortest injection time compared to DPR, SCFIT and BUFIT. As shown in Table 4. The comparison of fault injection time and speed improvement for different workloads is shown in Table 5. Logic for Counter, bubble sort circuit, 4-bit adder and 4-bit multiplier circuits are also implemented with the proposed FIS 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 FIS. 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 FIS method has higher fault injection rate and lower injection time compared to BUFIT, DPR and SCFIT, Suggesting that FIS is the better option for applications that require quick and frequent fault injections. Different workloads are compared in Table 4 by comparing fault injection times using three techniques: DPR, SCFIT, and the proposed FIS. 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

Table 4. *Speed improvement comparison of fault injection techniques*

Workload	DPR (ms)	SCFIT (ms)	BUFIT (ms)	Proposed FIS (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

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

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

addition to the FPGA target. The maximum available FFs in the Xilinx Zynq-7000 FPGA are not enough to meet the required FFs. This outcome demonstrates the practical limitations of utilizing the SCFIT method. The proposed FIS does not have any limitations as compared to existing methods and requires less number of CLBs and FFs resources inside FPGA.

4. Conclusions

This paper introduces a new FPGA-based technique called FIS, which is based on Xilinx Zynq-7000 FPGA. 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 FIS 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 FIS. In addition, LFSRs' deterministic nature and finite cycle length should be solved by exploring the integration of TRNGs, which may be more appropriate in the future.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] L. Lanzieri et al. (2024). A review of techniques for ageing detection and monitoring on embedded systems. *ACM Computing Surveys*. 57(1): 1–34. DOI:10.1145/3695247.

[2] B.M. Kumar, J. Ragaventhiran, V. Neela. (2024). Hybrid optimization integrated intrusion detection system in WSN using ELMAN network. *International Journal of Data Science and Artificial Intelligence*. 02(02): 55–62.

[3] Anisha, M., and V. Adlin Beenu. (2024). Double secure cloud medical data using Euclidean distance-based Okamoto Uchiyama homomorphic encryption. *International Journal of System Design and Computing*. 2(01): 1-7.

[4] Velayaudhan, Sindhu Thazhathethil, and Kalpana Devi. (2024). BUFIT: Fine-Grained Dynamic Burst Fault Injection Tool For Embedded Field Programmable Gate Array Testing. *Revue Roumaine Des Sciences Techniques—Série Electrotechnique Et Energétique*. 69(3): 303-308. DOI:10.59277/RRST-EE.2024.69.3.8.

[5] Gangolli, Aakash, Qusay H. Mahmoud, and Akramul Azim. (2022). A systematic review of fault injection attacks on IOT systems. *Electronics*. 11(13): 2023. DOI:10.3390/electronics11132023

[6] Bentchikou, Ibrahim, et al. (2022). Alternative Hybrid Control of Switched Systems. An Application To Machine DC Fed By Multicellular Converter. *Revue Roumaine Des Sciences Techniques—Série Électrotechnique Et Énergétique*. 67(3): 247-252.

[7] Richter-Brockmann, Jan, Pascal Sasdrich, and Tim Güneysu. (2022). Revisiting fault adversary models–hardware faults in theory and practice. *IEEE Transactions on Computers*. 72(2): 572-585. DOI: 10.1109/TC.2022.3164259.

[8] Metawie, Haytham, Mona Safar, and M. Watheq El-Kharashi. (2022). An Evaluation Method for Embedded Software Dependability Using QEMU-Based Fault Injection Framework. *6th International Conference on System Reliability and Safety (ICSRS), IEEE*. DOI:10.1109/ICSRS56243.2022.10067534.

[9] Yang, Weitao, Yonghong Li, and Chaohui He. (2022). Fault injection and failure analysis on Xilinx 16 nm FinFET Ultrascale+ MPSoC. *Nuclear Engineering and Technology*. 54(6): 2031-2036. DOI:10.1016/j.net.2021.12.022.

[10] Gao, Zhiwei, and Xiaoxu Liu. (2021). An overview on fault diagnosis, prognosis and resilient control for wind turbine systems. *Processes*. 9(2): 300. DOI:10.3390/pr9020300.

[11] Carminati, M., and G. Scandurra. (2021). Impact and trends in embedding field programmable gate arrays and microcontrollers in scientific instrumentation. *Review of Scientific Instruments*. 92(9): 091501. DOI:10.1063/5.0050999.

[12] Medjmadj, Slimane, Demba Diallo, and Antoni Arias. (2021). Mechanical sensor fault-tolerant controller in PMSM drive: experimental evaluation of observers and signal injection for position estimation. *Revue Roumaine Des Sciences Techniques—Série Electrotechnique Et Energétique*. 66(2): 77-83.

[13] Putra, Rachmad Vidya Wicaksana, Muhammad Abdullah Hanif, and Muhammad Shafique. (2021). Respawn: Energy-efficient fault-tolerance for spiking neural networks considering unreliable memories. *IEEE/ACM International Conference On Computer*

Aided Design (ICCAD), IEEE. DOI: 10.1109/ICCAD51958.2021.9643524.

[14] Claudepierre, Ludovic, et al. (2021). TRAITOR: a low-cost evaluation platform for multifault injection. Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems. DOI:10.1145/3457340.3458303.

[15] Given-Wilson, Thomas, Nisrine Jafri, and Axel Legay. (2020). Combined software and hardware fault injection vulnerability detection. *Innovations in Systems and Software Engineering*. 16(2): 101-120. DOI:10.1007/s11334-020-00364-5.

[16] Breier, Jakub, et al. (2020). A countermeasure against statistical ineffective fault analysis. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 67(12): 3322-3326. DOI:10.1109/TCSII.2020.2989184.

[17] Aranda, Luis Alberto, Alfonso Sánchez-Macián, and Juan Antonio Maestro. (2019). ACME: A tool to improve configuration memory fault injection in SRAM-based FPGAs. *IEEE Access*. 7: 128153-128161. DOI:10.1109/ACCESS.2019.2939858.

[18] Mandal, Swagata, et al. (2019). Criticality aware soft error mitigation in the configuration memory of SRAM based FPGA. *32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), IEEE*. DOI:10.1109/VLSID.2019.00063.

[19] Ramos, Alexis, et al. (2019). An ALU protection methodology for soft processors on SRAM-based FPGAs. *IEEE Transactions on Computers*. 68(9): 1404-1410. DOI:10.1109/TC.2019.2907238.

[20] Chatzidimitriou, Athanasios, et al. (2019). Multi-bit upsets vulnerability analysis of modern microprocessors. *IEEE International Symposium on Workload Characterization (IISWC), IEEE*. DOI:10.1109/IISWC47752.2019.9042036.

[21] Liao, Haohao, and Catherine Gebotys. (2019). Methodology for em fault injection: Charge-based fault model. *Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE*. DOI:10.23919/DATE.2019.8715150.

[22] Cerveira, Frederico, et al. (2018). Exploratory data analysis of fault injection campaigns. *IEEE International Conference on Software Quality, Reliability and Security (QRS), IEEE*. DOI:10.1109/QRS.2018.00033.

[23] Appathurai, Ahilan, and P. Deepa. (2015). Design for reliablity: A novel counter matrix code for FPGA based quality applications. *6th Asia Symposium on Quality Electronic Design (ASQED). IEEE*. DOI:10.1109/ACQED.2015.7274007.

[24] Ahilan, A., and P. Deepa. (2015). Modified Decimal Matrix Codes in FPGA configuration memory for multiple bit upsets. *International Conference on Computer Communication and Informatics (ICCCI). IEEE*. DOI:10.1109/ICCCI.2015.7218146.