Nuclear Engineering and Technology 54 (2022) 2031–2036

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

journal homepage: www.elsevier.com/locate/net

Original Article

Fault injection and failure analysis on Xilinx 16 nm FinFET Ultrascale+ MPSoC

Weitao Yang [a, b], Yonghong Li [a, *], Chaohui He [a]

[a] School of Nuclear Science & Technology, Xi'an Jiaotong University, Xi'an, China
[b] Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy

ARTICLE INFO

Article history:
Received 7 May 2021
Received in revised form
16 December 2021
Accepted 17 December 2021
Available online 21 December 2021

Keywords:
Fault injection
Failure analysis
Ultrascale+ MPSoC
Failure modes and effects analysis

ABSTRACT

Energetic particle strikes the device and induces data corruption in the configuration memory (CRAM), causing errors and even malfunctions in a system on chip (SoC). Software-based fault injection is a convenient way to assess device performance. In this paper, dynamic partial reconfiguration (DPR) is adopted to make fault injection on a Xilinx 16 nm FinFET Ultrascale+ MPSoC. And the reconfiguration module implements the Sobel and Gaussian image filtering, respectively. Fault injections are executed on the static and reconfiguration modules' bitstreams, respectively. Another contribution is that the failure modes and effects analysis (FMEA) method is applied to evaluate the system reliability, according to the obtained injection results. This paper proposes a software-based solution to estimate programmable device vulnerability.

1. Introduction

When advanced static random-access memory (SRAM)-based field-programmable gate arrays (FPGAs) are impinged by energetic particles, they suffer from soft errors leading to unexpected outcomes, such as hang or data corruption [1–3]. The particles can be heavy ions, protons, and neutrons [4]. Nowadays, the application of advanced SRAM-based FPGAs extends rapidly, including safety-critical missions, such as nuclear power plants [5]. It urges to assess the reliability of the device thoroughly.

System vulnerability can be judged via fault injection (FI), and two approaches are usually available [6]. One is hardware-based FI, and the other is software-based. The former can come out of accelerator irradiation. However, it suffers from an expensive budget, inadequate hours, or uncontrollable injection locations. In comparison, the latter is achieved through software design kits. Nevertheless, the drawback is also visible since this approach takes a long time.

A significant merit of the SRAM-based FPGAs is that they can be repeatedly programmed fully or partially [7]. This re-programmability provides more options to assess devices' robustness and performance. For example, dynamic partial reconfiguration (DPR) is one unique choice in the SRAM-based FPGAs. And several studies have been reported in SRAM-based FPGA related to DPR. In Ref. [8], a low area requirement, low delay DPR crossbar switch is presented. In Ref. [9], a reduced amount of reconfigurable partitions with better usage of resources from DPR design is discussed. A technique with a library of 11 partial reconfigurable regions and 16 video processing cores operating is designed on the Xilinx PYNQ system in Ref. [10]. In Ref. [11], a DPR system used for time-varying image constraints for video communications applications is proposed. The author introduced a DPR processor system for implementing single-pixel operations in Ref. [12].

DPR can perform fault injection or mitigate soft error in the SRAM-based FPGA from an external or internal port. The port can be a Joint Test Action Group (JTAG), Internal Configuration Access Port (ICAP), or the Processor Configuration Access Port (PCAP). Compared with other ports, PCAP is more flexible in Xilinx Ultrascale+ series devices [13].

Image processing is a widespread application, for instance, in aerospace crafts, automobiles, and drones. It is meaningful to evaluate and reduce risk effectively and then guarantee the applications are more resilient in an advanced system. In this manuscript, two image filters are benchmarked and implemented using DPR. And fault injection based on the DPR is also executed to assess soft errors in a Xilinx 16 nm FinFET Ultrascale+ MPSoC. Regarding the 16 nm Ultrascale+ MPSoC, a few manuscripts have involved the

* Corresponding author.
E-mail address: yonghongli@mail.xjtu.edu.cn (Y. Li).

https://doi.org/10.1016/j.net.2021.12.022

reliability tests in recent years.

The reliability impact of applying binary quantization to convolutional layers of neural networks is discussed in Ref. [14]. In Ref. [15], a 64 MeV mono-energetic proton source is employed to evaluate the soft error mitigation IP capability to detect and correct the single event upset event. In Ref. [16], the single event effect is explored under the irradiations of neutrons, 64 MeV protons, and thermal neutrons. The ultra-high energy beam is applied to examine the device sensitivity in Ref. [17]. In Ref. [18], a novel error rate estimation approach is proposed to explore the MPSoC vulnerability.

Unlike the reported effort, in this work, except for applying DPR to perform the fault injection, we also adopt the failure modes and effects analysis (FMEA) method to discuss system sensitivity further. FMEA is widely applied in various conditions to analyze a complex system's failure and influence [19–22]. We combine the DPR and FMEA to evaluate the reliability of the Ultrascale+ MPSoC for the first time and expect to provide another solution in the MPSoC soft error assessment.

2. MPSoC dynamic partial reconfiguration

The DPR design is implemented on an SRAM-based programmable device, and the reconfiguration modules (RMs) are Sobel and Gaussian image filter processing [23].

2.1. Reconfigurable device

The device under test (DUT) is the Xilinx XCZU3EG-1SFVA625 Zynq® UltraScale+ MPSoC. It mainly comprises a processing system (PS) and programmable logic (PL). Multicores are integrated into the PS, including Cortex®-A53 64-bit quad-core and Cortex-R5 dual-core real-time processor. And the PL is the 16 nm FinFET technology field-programmable gate array (FPGA). Other resources are also available on this device, such as DDR, Dual QSPI Flash, USB, and MIO.

2.2. Schematic of DPR

The DPR covers the static and reconfiguration module designs. It involves block design in Vivdo 2019.2 and program design in Vitis 2019.2. The SD card and DDR, which are out of the chip, are also used to restore bitstreams. Although JTAG and ICAP are also available in the device, JTAG needs a specific interface, cable, and ICAP requires extra resource utilization. PCAP integrated into the MPSoC without additional requirements is an ideal option to read/write bitstream between the PS and PL in the Ultrascale+ MPSoC. Xilinx provides the xil_library functions, which allows accessing bitstream via PCAP directly. Fig .1 draws the schematic of the design.

Fig. 1 briefs the basic connections and communications among the resources. In Fig. 1, the SD card keeps full and partial bitstreams and the BOOT.bin files. PS transfers them to the DDR. Then, PS loads the full and partial bitstreams to the configuration memory of PL by the PCAP interface to achieve functions of the static and reconfiguration modules.

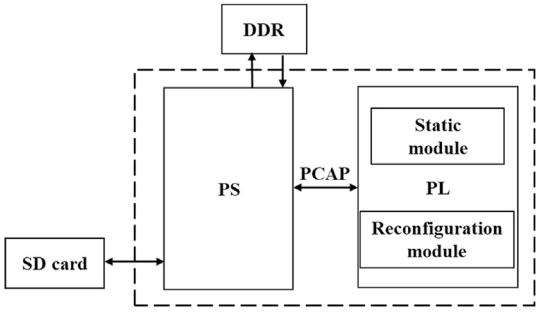

Fig. 1. Schematic of DPR in Ultrascale+ MPSoC.

Two RMs, the Sobel and Gaussian image filter processing, are created in a reconfiguration region [23]. A black box defines the necessary ports of the reconfiguration module. And the static module is locked before loading another reconfiguration module. Table 1 lists the resource utilization of both reconfiguration module designs, respectively. The utilized resources contain configuration logic block (CLB), the look-up table (LUT), global clock buffer (BUFG), digital signal processor (DSP), etc.

2.3. Bitstream in DPR

Bitstreams are generated in the Vivado design. As Table 2 displays, during the DPR, six bitstreams are generated totally. They include three full and partial bitstreams, respectively. The full bitstream corresponds to the static and RM modules (except for the black box). The partial bitstream for RM is just related to the corresponding algorithm. The full bitstream is 5568668 bytes, and the partial one is 476320 bytes.

The Blank full bitstream and Gaussian or Sobel partial bitstream are converted to .bin files. Then, they are stored in the SD card and loaded into CRAM via DDR to reach algorithm functions. Fault can be injected into the Blank full or partial bitstreams.

3. DPR fault injection

Fault injection is realized through modifying bit information in the bitstream. It flips 1-0 or 0-1 to emulate single event upset (SEU) events in the CRAM. This phenomenon frequently emerges while the device is applied in a harsh radiation environment. That can lead to catastrophic consequences sometimes.

Usually, fault injection for the SRAM-based FPGA can be implemented by soft error mitigation (SEM) IP or dynamic reconfiguration. However, SEM IP takes up specific resources. At the same time, there are also some limitations in the SEM IP fault injection. For instance, fault cannot be injected into the SEM IP corresponding bits. These do not appear in the DPR fault injection. In addition, the DPR fault injection can intuitively track the abnormality caused by which module's configuration bit flipping. Thus, the DPR fault injection is adopted in the project.

3.1. Injection procedures

The following steps are necessary for the DPR design. As Fig. 2 displays, it firstly requires transferring bitstreams from the SD card to DDR. Then, the full bitstream is loaded into CRAM from DDR. Thirdly, a reconfiguration module is selected. After that, the corresponding partial bitstream is loaded into CRAM. At last, the RM algorithm is executed. During the DPR, the RM and partial bitstream can be switched repeatedly and flexibly.

Fault injection is performed before the second or fourth step to mimic SEU in the full and partial bitstream, respectively.

Before the second step, for a target bit in the full bitstream, the fault is injected by modifying this bit via XOR operation with 1. With this operation, the '1' or '0' in the bit can be flipped to '0' or '1'. Then, the entire full bitstream containing this injected bit is loaded into CRAM. After executing other procedures, the effect of the fault injection can be investigated.

Once the former three steps are executed, the XOR operation can also be executed before the fourth step for a target bit in the partial bitstream. It can implement fault injection in the partial bitstream.

Table 1
Resource utilization of Sobel and Gaussian design.

Design	CLB(L)	CLB(R)	CARRY8	CLB	LUT(L)	LUT(M)	BFUG	DSP
Sobel	1099 (1.56%)	1241 (0.88%)	14 (0.16%)	256 (2.90%)	1017 (1.44%)	82 (0.28%)	1 (0.51%)	/
Gaussian	1019 (1.44%)	1241 (0.88%)	5 (0.06%)	239 (2.71%)	937 (1.33%)	82 (0.28%)	1 (0.51%)	5 (1.39%)

Table 2
Generated bitstream during the DPR design.

Design	Bitstream	Corresponding module
Sobel	Sobel full bitstream	Static+Sobel
	Sobel partial bitstream	Sobel
Gaussian	Gaussian full bitstream	Static+Gaussian
	Gaussian partial bitstream	Gaussian
Blank	Blank full bitstream	Static+Black box
	Blank partial bitstream	Black box

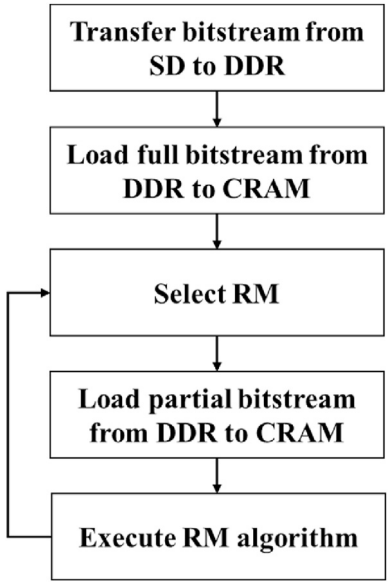

Fig. 2. Flow of achieving reconfiguration module algorithm.

After loading the entire partial bitstream containing the injected bit and executing the algorithm, influence from this fault injection can be obtained. However, it's time-consuming and inefficient if all bits are injected for the full and partial bitstreams.

3.2. Injection bit

The essential bit is the bit that relates to the current design, and SEU emerging in this bit may lead to errors. The essential bits data (EBD) and essential bits configuration (EBC) files generated in the Vivado design are two files for the essential bit. The EBD file describes whether the bit is essential, and '1′ in the file means the corresponding bit is essential. The EBC file presents the specific content information about this essential bit. And the information in EBC is identical to a part of the configuration memory. However, it only produces the essential bit files belonging to the full bitstream without corresponding to the individual partial bitstream.

The EBD file decides the injection bit locations concerning the Blank full bitstream injection (BFBI). And a total of 609658 bits are essential. 100000 bits are randomly extracted from that to generate the full bitstream FI script. The script contains accurate byte offset and bit address information. The length of the partial bitstream is

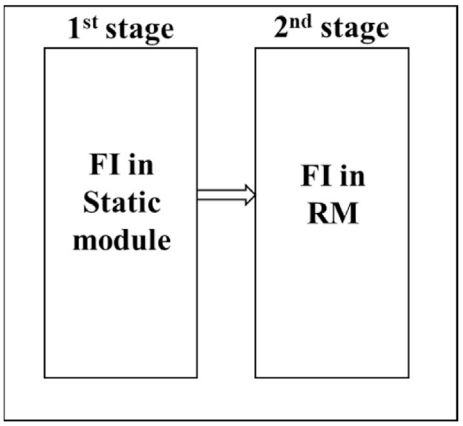

Fig. 3. The overall fault injection strategy.

less than that of the full. As mentioned above, it can only generate the essential bit files for the full bitstream directly in the Vivado design. Thus the fault injection script can't rely on the essential bit files for the partial bitstream. '0′ and '1′ are the contents of the partial bitstreams. It's difficult to determine which '0′ or '1′ is essential. Here, we choose all '1′ as the target bits (it can also choose all '0′). For Sobel and Gaussian partial bitstream injection (SPBI and GPBI), 276089 and 273343 bits are '1′, and they are all injected, respectively.

3.3. Fault injection execution

The entire injection is divided into two stages, as drawn in Fig. 3. First of all, the full bitstream injection is conducted. Two RMs share the Blank full bitstream related to the static module. Therefore, fault injection on the Blank full bitstream executes once only.

For a target bit, injection is executed via writing the data to the offset from the FI script. No matter the full or partial bitstream, the target location from the script is injected one after the other. For one injection, it can or not lead to soft error. If the injected bit induces a soft error, information about the type of error, specific bit, and byte offsets are outputted in the record. At the same time, this injection bit is recovered immediately. During the recovery, if the bit is about BFBI, the original full bitstream is retransferred and reloaded to CRAM again. If it is about the SPBI and GPBI, the corresponding partial bitstream is retransferred and reloaded again.

If no abnormality is detected for the injected bit, this bit is considered uncritical for the design. For the uncritical bit injection, it doesn't execute the recovery operation, and the next recovery operation will clear these uncritical bit injections.

Finally, fault injection for the full bitstream is executed once. For the two partial bitstreams, they are carried out, respectively. During all injections, the real-time messages are refreshed and kept in texts.

4. Results and analysis

The fault injection took about 262 h, and a total of 6822 soft errors were detected during all injections. The errors are loading

bitstream failure (LBF), calculation results error (CRE), and system halt (SH). Table 3 presents the detail about each kind of errors.

4.1. Injection results

Finally, for the BFBI, we detected 87 errors from the 100000 injections, and they are all the system halt. Every SH error is recovered by repower the test board.

For the SPBI and GPBI, 3905 and 2830 soft errors are detected, respectively. Soft errors are similar and illustrated in Table 4. The errors include loading bitstream failure, calculation results error, and system halt. More importantly, the occurrence locations of the detected errors for LBF and SH are identical. Take the system halt error as an example, and Table 5 displays the offset and bit information of the SH error in SPBI and GPBI. It can be seen that the offset and bit in both injections are indeed duplicated.

The loading bitstream failure and calculation results error are processed by retransferring and reloading the partial bitstream to CRAM. At the same time, it requires to repower the test board to solve the system halt error.

4.2. Result analysis

Soft error sensitivity (SES) is the ratio of detected errors in all injected ones and can be calculated in (1) [24]. The BFBI is the same for both partial bitstream injections, and the SES of BFBI is 8.70×10^{-4}.

$$SES = \frac{Ne}{N_i} \tag{1}$$

SES: soft error sensitivity, N_e: number of the detected error, and N_i: the number of injected errors.

Concerning the CRE, the SES values of SPBI and GPBI are 1.37×10^{-2} and 9.92×10^{-3}, respectively. The partial bitstream length of Sobel and Gaussian is 476320 bytes and 3810560 bits. Although the injected fault numbers of SPBI and GPBI are different, as mentioned above, the LBF and SH errors are duplicated, including the offset and bit address. It is reasonable. Apart from the different algorithms, other parts are identical for the two RMs.

Comparing the partial bitstreams of Sobel and Gaussian, 19263 bytes are different in all 476320 bytes. Among the 19263 bytes, '1′ in the Sobel partial bitstream is 34000 bits, while 31254 bits in the Gaussian partial bitstream. The discrepancy is 2746 bits, equal to the disparity between '1′ in Sobel and Gaussian partial bitstreams (276089 and 273343 bits, respectively). The number of the remaining '1′ for both RMs is 242089. Therefore, the duplicated errors in both injections can be considered from the 242089 bits' injections. Further, LBF and SH's SES values for SPBI and GPBI are 4.67×10^{-4} and 1.65×10^{-5}, respectively.

5. FMEA from DPR fault injection

The fault injection emulates SEU in the CRAM of the target device, and the obtained errors are the effect of different kinds of system failure. The DPR design is composed of the static and reconfiguration modules, and it should follow the specific operation sequences for the two modules' execution. Hence, the FMEA method is more suitable to analyze the results further in this project. The FMEA module is applied to assess the observed errors from the DPR injection to analyze the DPR design's sensitivity systematically. That's the significant difference of current work with other reported DPR-based fault injection efforts [25–27].

Table 3
Detected error types during the injection.

Detected error	Detail of the error
LBF	Bitstream can't be loaded to CRAM from DDR
CRE	At least one is different from the expected results
SH	The program stops running, and no fresh messages

Table 4
Detected errors in the partial bitstream injection.

Injection	LBF	CRE	SH
SPBI	113	3788	4
GPBI	113	2713	4

Table 5
System halt error detail in partial bitstream injection.

Injection	Offset	Bit
SPBI	367032	4
	367032	5
	367034	5
	367035	0
GPBI	367032	4
	367032	5
	367034	5
	367035	0

5.1. FMEA construction

To construct the FMEA, the following information: top event, module, failure mode, failure rate, severity, and risk priority number (RPN), should be determined. For the Ultrascale+ MPSoC, the top event is the system malfunction, and the module includes the static and reconfiguration ones. According to the fault injection, the module, failure mode, failure rate, and processing methods in the FMEA are presented in Table 6. In this table, the failure rate is mainly affected by the number of detected errors [1–33]. The uncertainty of the failure rate can be obtained by $n^{1/2}/N$ (n: corresponding detected error number, N: corresponding injected bit number) [28].

In this study, the module covers the Static, the Sobel, and Gaussian RMs. The Static module's failure mode is SH with 8.70×10^{-4}, and this failure is processed by repowering the test board. The failure modes and processing methods for Sobel and Gaussian RM are the same, except for the CRE failure rates are different.

5.2. System risk assessment

A failure mode's RPN characterizes its impact on the system risk consequences for the system-level risk assessment. The larger the RPN value for a failure mode, the greater the impact on the system vulnerability. Like the RPN of failure mode, the larger the RPN value for a module or component, the greater the influence on the system

Table 6
Parameters in the FMEA.

Module	Failure mode	Failure rate	Processing methods
Static Module	SH	$(8.70 \pm 0.93) \times 10^{-4}$	Repowering
Sobel RM	LBF	$(4.67 \pm 0.44) \times 10^{-4}$	Reloading bitstream
	CRE	$(1.37 \pm 0.02) \times 10^{-2}$	Reloading bitstream
	SH	$(1.65 \pm 0.83) \times 10^{-5}$	Repowering
Gaussian RM	LBF	$(4.67 \pm 0.44) \times 10^{-4}$	Reloading bitstream
	CRE	$(9.92 \pm 0.19) \times 10^{-3}$	Reloading bitstream
	SH	$(1.65 \pm 0.83) \times 10^{-5}$	Repowering

reliability. The larger RPN also indicates the priority of taking measures to this failure mode or component [29,30]. As Xilinx uses Ultra-Low Alpha (ULA) material, the average alpha particles flux from package impurities is estimated to be 0.001 cm^{-2} h^{-1} [31]. And the 16 nm FinFET CRAM upset rate from package alpha particle impurities is about 0.1 FIT/Mb [16]. The static and reconfiguration module's total bits are 44549344 and 3810560 bits. Therefore, for the DPR design, the SEU failure rates of the static and reconfiguration modules are 4.45 and 0.38 FIT, respectively. There are two components and three kinds of failure modes for the DPR design, represented as follows. {*C(1), C(2)*} = {*Static module, Module*} and {*FM(1), FM(2), FM(3)*} = {*SH, LBF, CRE*}. The *i*th component's RPN (*RPN_C(i)*) and the *k*th failure mode's RPN (*RPN_FM(k)*) can be calculated by (2) and (3), respectively.

$$RPN\_C(i) = FR\_C(i) \times \sum_{k=1}^{z} P(i, FM(k)) \times S\_FM(k),\ 1 \prec k \prec z \qquad (2)$$

$$RPN\_FM(k) = S\_FM(k) \times \sum_{i=1}^{n} FR\_C(i) \times P(i, FM(k)),\ 1 \prec i \prec n \qquad (3)$$

RPN_C(i): RPN of the component, FR_C(i): SEU rate of the component, P (i, FM(K)): the probability of FM(K) if a failure occurs in the component, S_FM(k): severity rate of the failure mode.

For the DPR design, {*FR_C(1), FR_C(2)*} is {*4.45 FIT, 0.38 FIT*} and taking into the influence and processing methods of the failure modes, {*S_FM(1), S_FM(2), S_FM(3)*} is {*10, 6, 4*}. Hence, the values of *RPN_C(i)* and *RPN_FM(k)* are follows.

$$RPN\_C(1) = 4.45 \times (8.70 \times 10^{-4} \times 10 + 0 \times 6 + 0 \times 4) = (3.87 \pm 0.41) \times 10^{-2}\ FIT$$

$$RPN\_C(2) = 0.38 \times (1.65 \times 10^{-5} \times 10 + 4.67 \times 10^{-4} \times 6 + 1.18 \times 10^{-2} \times 4) = (1.91 \pm 0.04) \times 10^{-2}\ FIT$$

(1.18×10^{-2} is the average CRE of SBPI and GBPI, other failure rates are also averages)

$$RPN\_FM(1) = 10 \times (8.70 \times 10^{-4} \times 4.45 + 1.65 \times 10^{-5} \times 0.38) = (3.88 \pm 0.42) \times 10^{-2}\ FIT$$

$$RPN\_FM(2) = 6 \times (0 \times 4.45 + 4.67 \times 10^{-4} \times 0.38) = (1.06 \pm 0.10) \times 10^{-3}\ FIT$$

$$RPN\_FM(3) = 4 \times (0 \times 4.45 + 1.18 \times 10^{-2} \times 0.38) = (1.79 \pm 0.10) \times 10^{-2}\ FIT$$

It can be seen that the *RPN_C(1)> RPN_C(2)* and *RPN_ FM(1) >RPN_ FM(3)> RPN_ FM(2).* That demonstrates the static module and system halt error should be paid more attention to SEE hardening. In Ref. [32], Xilinx describes an example to improve the reliability of the DPR design on Ultrascale+ MPSoC recently. It combines the isolation design flow and dynamic partial reconfiguration to reduce soft errors. In the future, we will try to implant this solution into our design to examine its performance.

In [33], aiming at the Xilinx 16 nm FinFET Ultrascale+ MPSoC, fault injection based on soft error mitigation (SEM) and fault tree analysis have been conducted. The SEM IP depends on the specific block and takes up resources on the FPGA. If the fault is injected into the SEM IP corresponding bits, the operation will fail. It is different from Ref. [33]. DPR-based injection operates without using other FPGA resources. Moreover, the FMEA points out different bitstreams and failure modes severity sequences.

6. Conclusion

Fault injection on the Xilinx 16 nm FinFET Ultrascale+ MPSoC has been conducted depending on dynamic partial reconfiguration. And the reconfiguration module implements the Sobel and Gaussian image filter algorithms. Three kinds of errors, including calculation results error, loading bitstream failure, and system halt, are investigated. The failure modes and effects analysis method is applied further based on the obtained results. The fault injection and analysis illustrate that the static module and system halt need to take priority hardening measures. This work provides a solution, depending on DPR and FMEA, to assess the reliability of the SRAM-based Ultrascale+ MPSoC.

Declaration of competing 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.

Acknowledgment

Project supported by National Natural Science Foundation of China (Grant Nos. 11575138, 11835006, 11690040, and 11690043), National Key Laboratory of Materials Behavior and Evaluation Technology in Space Environment, Harbin Institute of Technology (Grant No.6142910190304), and China Scholarships Council (Grant No. 201906280343).

References

[1] H. Asadi, M.B. Tahoori, B. Mullins, D. Kaeli, Kevin Granlund, Soft error susceptibility analysis of SRAM-based FPGAs in high-performance information systems, IEEE Trans. Nucl. Sci. 54 (6) (2007) 2714–2726.
[2] T.S. Nidhin, A. Bhattacharyya, R.P. Behera, T. Jayanthi, K. Velusamy, Understanding radiation effects in SRAM-based field programmable gate arrays for implementing instrumentation and control systems of nuclear power plants, Nucl. Eng. Technol. 49 (2017) 1589–1599.
[3] F. Siegle, T. Vladimirova, J. Ilstad, O. Emam, Mitigation of radiation effects in SRAM-based FPGAs for Space applications, ACM Comput. Surv. 47 (2) (2015) 37.
[4] W.T. Yang, Y.H. Li, Y.X. Guo, H.Y. Zhao, Y. Yang Li, P. Pei Li, C.H. He, G. Guo, J. Jie Liu, S.S. Yang, H. An, Investigation of single event effect in 28-nm system-on-chip with multi patterns, Chin. Phys. B 29 (10) (2020) 108504.
[5] J. Kim, E. Kim, J. Yoo, Y. Lee, G. Cho, An integrated software testing framework for FPGABased controllers in nuclear power plants, Nucl. Eng. Technol. 48 (2016) 470–481.
[6] M. Kooli, G.D. Natale, A survey on simulation-based fault injection tools for complex systems, in: 2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Santorini, Greece, 2014.
[7] C. Bernardeschi, L. Cassano, A. Domenici, SRAM-based FPGA systems for safety-critical applications: a survey on design standards and proposed methodologies, J. Comput. Sci. Technol. 30 (2) (2015) 373–390.
[8] C.H. Hoo, A. Kumar, An area-efficient partially reconfigurable crossbar switch with low reconfiguration delay, in: 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, 2012.
[9] V.M.G. Martins, P.R.C. Villa, R. Travessini, M.D. Berejuck, E.A. Bezerr, A dynamic partial reconfiguration design flow for permanent faults mitigation in FPGAs, Microelectron. Reliab. 83 (2018) 50–63.
[10] A.E. Wilson, M. Wirthlin, Reconfigurable real-time video pipelines on SRAM-based FPGAs, in: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 2019.
[11] Y.B. Jiang, M.S. Pattichis, A dynamically reconfigurable architecture system for time-varying image constraints (drastic) for motion jpeg, J. Real-Time Image. Pr 14 (2) (2018) 395–411.
[12] D. Llamocca, M. Pattichis, A dynamically reconfigurable pixel processor system based on power/energy-performance-accuracy optimization, IEEE Trans. Circ. Syst. Video Technol. 23 (3) (2013) 488–502.
[13] C. Kohn, XAPP1159 Partial Reconfiguration of a Hardware Accelerator on Zynq-7000 All Programmable SoC Devices, 2013.
[14] F. Libano, B. Wilson, M. Wirthlin, P. Rech, J. Brunhaver, Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs, IEEE Trans. Nucl. Sci. 67 (7) (2020) 1478–1484.

[15] P. Maillard, M.J. Hart, P. Chang, Y.P. Chen, M. Welter, R. Le, R. Ismail, J. Barton, E. Crabill, Single-event evaluation of Xilinx 16nm UltraScale+TM single event mitigation IP, in: 2018 IEEE Radiation Effects Data Workshop (REDW), Waikoloa Village, HI, USA, 2018.
[16] P. Maillard, M.J. Hart, J. Barton, et al., Neutron, 64 MeV proton & alpha single-event characterization of Xilinx 16nm FinFET Zynq® UltraScale+™ MPSoC, in: 2017 IEEE Radiation Effects Data Workshop (REDW), New Orleans, LA, USA, 2017.
[17] M. Glorieux, A. Evans, T. Lange, et al., Single-event characterization of Xilinx UltraScale+r MPSOC under standard and ultra-high energy heavy-ion irradiation, in: 2018 IEEE Radiation Effects Data Workshop (REDW), Waikoloa Village, HI, USA, 2018.
[18] L. Sterpone, S. Azimi, L. Bozzoli, et al., A novel error rate estimation approach for UltraScale+ SRAM-based FPGAs, in: 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Edinburgh, 2018.
[19] P. McNelles, L.X. Lu, Field programmable gate array reliability Analysis using the dynamic flowgraph methodology, Nucl. Eng. Technol. 48 (2016) 1192–1205.
[20] P. Conmy, I. Bate, Component-based safety analysis of FPGAs, IEEE Trans. Ind. Inf. 6 (2) (2010) 195–205.
[21] M.J. Peng, H. Wang, S.S. Chen, G.L. Xia, Y.K. Liu, X. Yang, A. Ayodejia, An intelligent hybrid methodology of on-line system-level fault diagnosis for nuclear power plant, Nucl. Eng. Technol. 50 (3) (2018) 396–410.
[22] R. Mariani, G. Boschi, A systematic approach for failure modes and effects analysis of system-on-chips, in: 13th IEEE International On-Line Testing Symposium (IOLTS 2007), Crete, Greece, 2007.
[23] http://ivpcl.unm.edu/ivpclpages/Research/drastic/PRWebPage/PR_Sub.php.
[24] S. Kim, A.K. Somani, Soft error sensitivity characterization for microprocessor dependability enhancement strategy, in: IEEE International Conference on Dependable Systems and Networks, Washington, D. C, 2002.
[25] G. Fakhreddine, S. Fouad, E. Mohamed, G. Bertrand, Fast SRAM-FPGA fault injection platform based on dynamic partial reconfiguration, in: 2014 26th International Conference on Microelectronics (ICM), Doha, 2014.
[26] T.M. Tom, J. Plusquelic, D. Brian, Information leakage analysis using accelerated fault injection emulation of a RISC-V microprocessor, SAND2020-0503C, https://www.osti.gov/servlets/purl/1761288.
[27] P. Arturo, R. Alfonso, O. Andres, G.A. David, J. Álvaro, A.V. Miguel, D.L.T. Eduardo, Run-time reconfigurable MPSoC-based on-board processor for vision-based Space navigation, IEEE Access 8 (2020) 59891–59905.
[28] W.T. Yang, X.C. Du, J.L. Guo, J.Z. Wei, G.H. Du, C.H. He, W.J. Liu, S.S. Shen, C.L. Huang, Y.H. Li, Y.Y. Fan, Preliminary single event effect distribution investigation on 28 nm SoC using heavy ion microbeam, Nucl. Instrum. Methods Phys. Res. B 450 (2019) 323–326.
[29] Y.Y. Chen, Y.C. Wang, J. Peng, SoC-level fault injection methodology in SystemC design Platform, in: Asia Simulation Conference-International Conference on System Simulation and Scientific Computing, Beijing, China, 2008.
[30] Y.Y. Chen, C.H. Hsu, K.L. Leu, SoC-level risk assessment using FMEA approach in system design with SystemC, in: IEEE International Symposium on Industrial Embedded Systems, Lausanne, Switzerland, 2009.
[31] Xilinx Inc., UG116 Device Reliability Report, 2015 v10.2.1.
[32] Xilinx Inc., XAPP1361, Isolation Design Flow + Dynamic Function eXchange Example, 2021. V1.0.
[33] W.T. Yang, B.Y. Du, C.H. He, S. Luca, Reliability assessment on 16 nm ultrascale+ MPSoC using fault injection and fault tree analysis, Microelectron. Reliab. 120 (2021) 114122.