On the Reliability of Real-time Operating System on Embedded Soft Processor for Space Applications

On the Reliability of Real-time Operating System on Embedded Soft Processor for Space Applications

Andrea Portaluri, Sarah Azimi, Corrado De Sio, Daniele Rizzieri, Luca Sterpone

Politecnico di Torino, Torino, Italy
{andrea.portaluri, sarah.azimi, corrado.desio, daniele.rizzieri, luca.sterpone}@polito.it

Abstract. The interest of the space industry in Real-Time Operating Systems for achieving stringent real-time requirement is drastically increasing. Among the different available hardware architectures, the solution of RTOS implemented on soft processors embedded in programmable devices is one of the most efficient and flexible solution for the mission deployment. However, radiation-induced failures are a severe concern affecting the reliability of electronic systems in space applications. In this paper, we investigate the impact of radiation-induced architectural faults affecting the reliability of application running on a Xilinx Microblaze embedded soft-processor within FreeRTOS Operating System. We developed a fault model through a proton radiation test, while the effects of the faults are evaluated in terms of Mean Time To Failure and Mean Executions To Failure, by a fault injection campaign using detected fault models. Finally, the occurrence and contribution to the error rate of specific MBUs events based on different shapes and sizes are evaluated through dedicated fault injection campaigns.

1 Introduction

The usage of Field Programmable Gate Arrays (FPGA) has been tremendously increasing recently, especially for space applications. With satellite lifetimes increased far beyond 10 years, hardware reconfigurability in flight has become a demanded requirement [1]. Soft-core processors are one of the cores commonly implemented using the programmable logic of the FPGAs [2]. Among the available solutions, Microblaze is an industry leader in FPGA-based soft processing solutions. Due to the flexible architecture and configuration options, it became highly suitable for embedded applications. Moreover, it requires few resources for implementation on programmable hardware. The increasing task complexity required for embedded systems led to the decrease of bare-metal applications and the migration toward the adoption of Real-Time Operating Systems (RTOSs) which provide an efficient solution for meeting stringent real-time requirements [3], especially in safety-critical applications to manage the execution of multiple applications sharing resources.

FreeRTOS is an excellent choice when there are multiple tasks to be executed in an organized and predictable fashion. It is a real-time operating system with deterministic and predictable task scheduling which allows the important task to meet hard deadlines for executions. FreeRTOS allows to schedule tasks by priority and time slicing which

is designed to run on small microprocessors which need to perform multiple tasks deterministically.

Nowadays, due to the transistors scaling and increasing of the number of available resources in a device, radiation-induced failure is becoming an important challenge to overcome [4][5], especially focusing on mission-critical space applications. To show the significant impact of these effects, it is worth to mention the user-observable Mean Time To Failure (MTTF) of an Apple iPhone 3 operating at commercial aircraft altitude can reduce to 1 year [6].

Therefore, when using an operating system running on a soft microprocessor in mission-critical applications, the reliability issues deriving from the exposure of the devices to ionizing radiation, such as Single Event Upsets (SEUs), should be considered [7][8][9][10]. Differently from hardwired microprocessors, the netlist of soft microprocessors such as Microblaze is implemented in the programmable hardware relying on the configuration memory (CRAM) of the FPGA. This memory can be corrupted by SEU [9][11], leading to the hardware micro-architectural faults which can propagate to the application layer and, in the case of the usage of a microprocessor supporting an operating system, it can lead to catastrophic results, especially in mission-critical applications [12].

The main contribution of this work is oriented toward the reliability evaluation of a Real-time Operating System on the Microblaze embedded soft processor for space application. To do so, we performed a detailed evaluation of the impact of radiation-induced architectural faults affecting the application benchmarks running on the FreeRTOS of the Microblaze embedded soft processor. The analysis has been performed through a fault injection campaign while an accurate fault model consisting of different clusters patterns of Multiple Bit Upset (MBU) has been identified through proton radiation performed at Paul Scherrer Institute (PSI) radiation facility. We evaluated the effect of faults during the execution of different software applications on FreeRTOS supported by Microblaze implemented on Zynq-7020 FPGA while we performed a deep investigation on the outcome of the software application, reporting the system reliability in terms of calculated Mean Time To Failure and Mean Executions to Failure as an impact of radiation-induced errors.

Please notice that the developed platform is not targeting the software-level fault injections but targeting the hardware faults and their impact on the execution of the software running in the operating system.

2 Related works

Recently, reliability has become one of the main challenges of embedded systems applied to mission-critical applications. Several works elaborate on the software-level techniques for evaluating the sensitivity to Single Event Upsets (SEUs) of the embedded operating system. Commonly, these approaches are based on modifying the original kernel of the embedded operating system or altering either the memory that the OS uses or the parameters of system calls.

In [15], the vulnerability of FreeRTOS has been evaluated through a software-based fault injection methodology that targets the most relevant variables and data structures of the OS. An automatic method for fault injection into program and data memory is

presented in [16]. The authors of [17] proposed a detailed analysis and hardening architecture based on lockstep synchronization supporting FreeRTOS. The heavy ion irradiation test presented in [18] targets the SRAM and the special purpose registers of an ARM microcontroller to evaluate the impact of the radiation-induced SEU. However, software application-level methods do not take into account the impact of faults occurring at the architectural level of the soft-core processor on the functionality of the operating system.

Other approaches are based on the simulation of HDL description of microprocessors [19]. The advantage of these methods is the feasibility of injecting upsets into any CPU register and structure at any time however, these methods are time expensive.

3 Background on Radiation-induced Effect on Reconfigurable Logic

Modern programmable hardware devices, such as Field Programmable Gate Arrays (FPGAs), are reconfigurable integrated circuits that can be programmed to implement any digital hardware circuit. FPGA consists of configurable logic blocks that can be connected via prefabricated programmable interconnects. The functionality of all the FPGA's programmable blocks and interconnections are controlled using millions of static random access memory (SRAM) cells that are programmed to realize a specific function. The user describes the desired function of the hardware in one of the hardware languages such as VHDL, Verilog, or recently high-level synthesis (HLS). Utilizing the commercial FPGA computer-aided design (CAD) flow, the hardware design is compiled into a bitstream file which is used to program all the FPGA's configuration SRAM cells [20].

Bitstream is a configuration array of binary data. Each bit of the bitstream file is responsible to configure logic blocks such as Look-Up Tables (LUTs), Flip-Flops (FFs), configuration Logic Block (CLB), and interconnections among them.

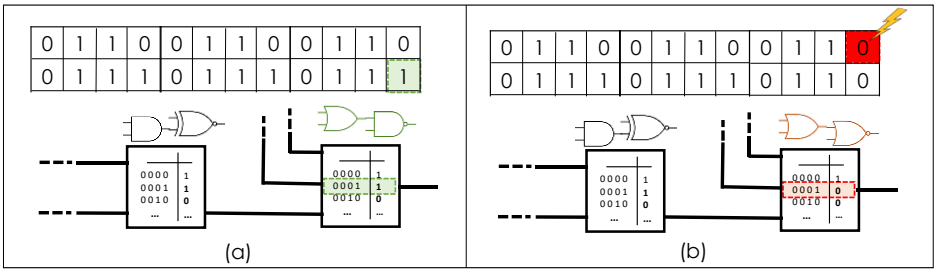

Fig. 1. The Radiation Particle Creating an SEU on the Configuration Memory of FPGA.

The SRAM cell representing the bitstream of the implemented hardware consists of millions of transistors. When a high-energy particle is interacting within the silicon of the device and releases its energy in these transistors, it can lead to the modification of the content of the memory cell (SRAM cell). This phenomenon is known as Single Event Upset (SEU) which corrupts the information stored in the memory cell,

producing a fault that can lead to error and eventual malfunction of the system implemented on the FPGA. For example, as is represented in Figure 1, an SEU in the bitstream can change the configuration of a LUT from NAND to NOR and therefore, change the functionality of the circuit and eventual failure.

In order to consider an accurate fault model affecting the hardware under the study to evaluate the reliability of FreeRTOS on soft-core Microblaze, we have performed a radiation test as the closest scenario to the space application one. The performed test and extracted fault models are described in the following section.

4 Proton Radiation Test-based Reliability Fault Model

In order to achieve accurate reliability analysis, we have performed a proton radiation test at the Paul Scherrer Institute (PSI) Proton Facility in Switzerland. From this experiment, we have identified an accurate fault model for performing fault injection campaigns.

A Zynq-7020 device has been irradiated with a proton beam with energies between 29 and 200 MeV. The value of energies and fluxes used during the radiation test experiment is represented in Table 1. During the experiment, the configuration memory of the device has been continuously monitored through a periodic reading of the content every 5 seconds. The snapshots of the configuration memory content have been then analyzed for detecting the occurrence of Multiple Bit Upsets (MBUs). The flux of the particles has been tuned to keep a few bitflips in configuration memory in each snapshot.

Table 1. Radiation Test Conditions: Energy, Flux and Fluence

Energy [MeV]	Flux [$cm^{-2}s^{-1}$]	Fluence [cm^{-2}]
29.31	$4.124 \cdot 10^7$	$9.173 \cdot 10^{10}$
50.80	$4.024 \cdot 10^7$	$6.064 \cdot 10^{10}$
69.71	$4.110 \cdot 10^7$	$2.124 \cdot 10^{10}$
101.34	$4.319 \cdot 10^7$	$2.415 \cdot 10^{10}$
151.18	$4.094 \cdot 10^7$	$1.226 \cdot 10^{10}$
200	$4.144 \cdot 10^7$	$3.942 \cdot 10^{10}$

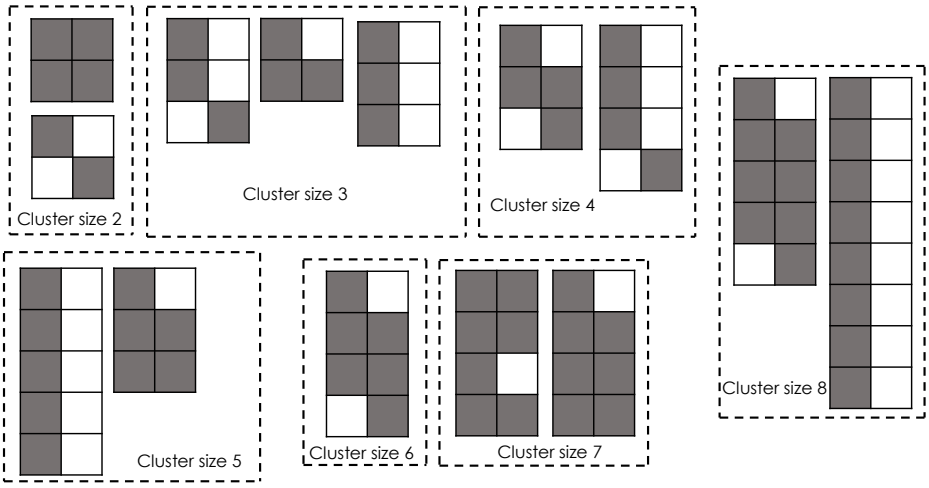

Fig. 2. Detected cluster sizes and shapes during the Zynq-7020 proton test.

Table 2. Cross-section of different cluster sizes

Cluster Size	Cross section per Particle [cm^2]
1	$1.87 \cdot 10^{-8}$
2	$1.11 \cdot 10^{-8}$
3	$1.70 \cdot 10^{-9}$
4	$5.99 \cdot 10^{-10}$
5	$1.73 \cdot 10^{-10}$
6	$1.02 \cdot 10^{-10}$
7	$5.65 \cdot 10^{-11}$
8	$2.64 \cdot 10^{-11}$

The few bitflips and the large size of the configuration memory (more than 100 million bits) allowed us to detect groups of SEUs with a strong correlation both in time and space. Therefore, it has been possible to select a group of bits with a high probability to have occurred as a result of a Single Event Multiple Upsets (SEMUs).

Figure 2 illustrates the observed pattern for MBUs resulting from the radiation test experiment while Table 2 shows the cross-section per particle of each cluster size observed during the whole radiation test. It can be noticed how the contribution of SEMUs is not negligible compared to SEUs. Indeed, more than 40% of the detected events have been SEMUs. The proposed cluster of faults has been used as the fault model in the fault injection campaign. In order to obtain an analysis as closest as possible to reality, the occurrence rate of the different clusters during the fault emulation has been weighted on the cross-section reported in Table 2. Since the high occurrence rate of SEMUs, evaluating only an SEU fault model will result in a loose approximation of the observed events.

5 The Reliability Analysis Workflow

The fault model collected through the proton test has been used to perform an accurate radiation analysis on the impact of radiation-induced faults on the Microblaze embedded soft processor porting FreeRTOS through fault injection campaigns.

5.1 The implemented Hardware/Software Platform

The current section elaborates on the implemented hardware platform, Microblaze soft-core, supporting FreeRTOS and software benchmarks applications.

Hardware Platform The Microblaze embedded soft processor is a Reduced Instruction Set Computer (RISC) optimized for FPGA deployment. It is highly configurable, allowing the selection of a specific set of features required by the design. Therefore, it has been chosen as the platform for supporting FreeRTOS. FreeRTOS, as a deterministic Real-time operating system, allows concurrency among several tasks with different priority levels, supporting a preemption mechanism to switch between task's execution [21].

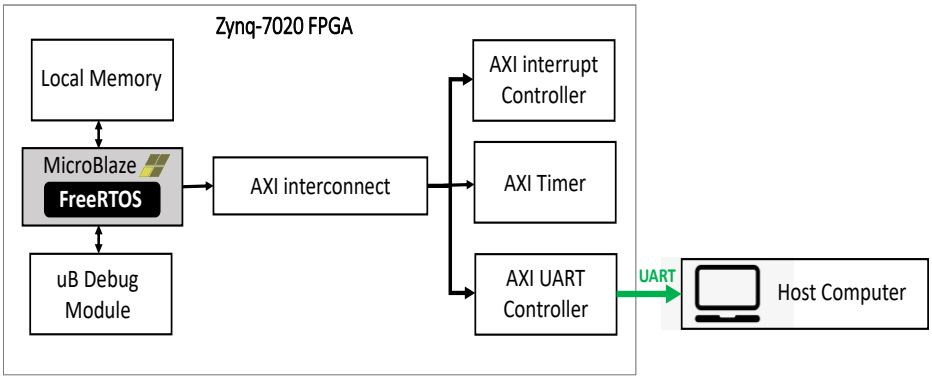

Fig. 3. The implemented hardware platform.

Another important feature that characterizes the Microblaze porting of FreeRTOS is the possibility to instantiate exception handlers to cope with the standard exception conditions defined by Microblaze soft-core [22].

A Xilinx 28 nm CMOS Zynq-7020 FPGA is chosen as a target hardware device.

Figure 3 represents the implemented hardware while Table 3 reports the device utilization when implementing a Microblaze porting FreeRTOS. As it is can be observed from the table, the implemented design used few resources of the FPGA. Therefore, fault injection campaigns are performed selectively to target only a subset of the whole configuration memory of the FPGA where the circuit is implemented.

Table 3. Resource Utilization of the Hardware Platform

Resources	Used [#]	Available [#]	Usage [%]
LUTs	2,596	53,200	4.88
Logic Slices	966	13,300	7.26
Flip-Flops	2,998	106,400	2.51
BRAM	32	140	22.86

Software Platform As a software application suite, a set of software benchmarks has been chosen to run on FreeRTOS while exploiting its main functionalities. To exploit the capability of FreeRTOS to schedule the execution of different tasks, for each software application, three different tasks (software benchmark) with the same priority have been instantiated. Therefore, three tasks running on FreeRTOS have the same instruction code, however, they are operating on different input data while sharing the processor execution time. The three selected software benchmarks are:

- matmul: matrix multiplication between large matrices of integers.
- matconv: matrix convolution between large matrices of integers.
- dijkstra: computation of shortest path between nodes in a large graph using the Dijkstra algorithm.

5.2 Fault Injection Analysis

Reliability analysis of the applications under test against radiation-induced hardware architectural faults in soft microprocessors has been performed by fault injection campaigns. The PyXEL platform has been used as a supporting fault injection framework [23]. PyXEL is an open-source Python-based platform easing the execution of FPGA fault injection campaigns. The platform has been instrumented to inject MBU patterns identified during the proton radiation test in the configuration memory of the FPGA. The device under test is connected to the host computer running the PyXEL experiment manager through a serial connection allowing to run the experiments on the platform and collecting results. A timeout mechanism is used to handle the halt and loop of the processor due to the injected faults.

Two fault injection campaigns have been performed. The former has been carried out based on the distribution of SEUs and MBUs represented in Figure 2 and Table 2. In the latter, each detected cluster has been extensively tested in order to estimate the impact of the different MBU clusters on the application failures. As it has been mentioned before, during the fault injection campaigns, only the part of the whole configuration memory implementing the circuit under test is targeted by the fault injection task in order to reduce the injection space to the resources used by the implemented netlist.

6 Experimental Analysis and Results

Results of the two fault injection campaigns based on fault models collected through a proton radiation test campaign targeting the Zynq-7020 device have been collected, categorized, and discussed.

Error Classification Four categories have been identified. They are defined as follows:

- *Correct*: the FreeRTOS succeeded in executing the application and produced an output that matches the golden one.
- *Silent Data Corruption*: the task execution on FreeRTOS terminates but the produced output data does not match with the golden one.
- *Halt*: the FreeRTOS does not complete the task. It can be due to different causes, such as infinite loops and application timeout.
- *Raising Exceptions:* an exception is generated in the FreeRTOS (i.e., at the software level) as a result of a fault affecting Microblaze architecture (i.e., netlist modification due to configuration memory corruption).

Moreover, we have performed a detailed investigation on the cause of each raised exception and classified them as follows:

- *FSL_EXCEPTION*: data bus error exception.
- *UNALIGNED_ACCESS*: attempt to perform unsupported unaligned access to memory.
- *ILLEGAL_OPCODE*: attempt to execute an illegal opcode.
- *AXI_D_EXCEPTION*: data system bus timeout.

Experimental Results For the first fault injection campaign, we injected and singularly evaluated 10,000 faults, generated considering the cluster distribution represented in Table 2 and Figure 2. As these data show, the occurrence rate of different MBU cluster sizes is not constant, so this first campaign has been done to evaluate the robustness of the different applications in a real-world effects scenario. The error rates observed with the described fault configuration are reported in Figure 4.

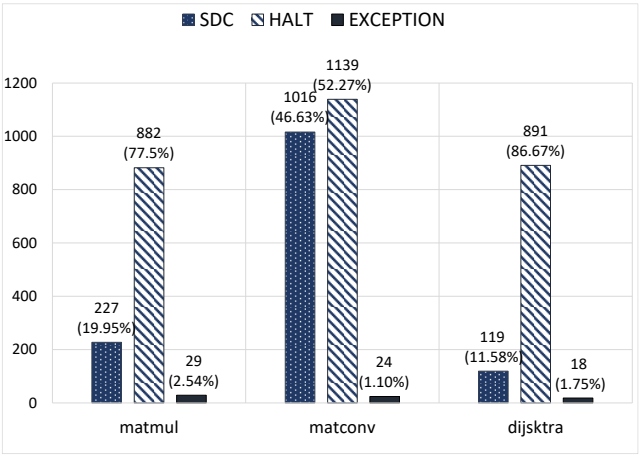

Fig. 4. Error Rates for each application with distributed MBU injection

As it can be observed in Figure 4, most of the errors cause the HALT of the system while errors resulting in SDC and rising EXCEPTION are observed less. This information can be used to identify critical scenarios such as HALT in order to provide sufficient and efficient mitigation techniques.

Moving forward, for the second campaign of fault injection, we focused on performing a deep analysis of the criticality of the MBU occurrences with respect to SEU in terms of corruption in the system. Therefore, the second campaign is performed considering 5,000 fault injections for each cluster size (40,000 fault injections in total) without considering the distribution, aiming to estimate the impact that the different MBU cluster sizes – considered one at a time – have on the three applications. The obtained results are presented in Figure 5.

Results Discussion Data show that the three applications have been impacted differently. In particular, the *matconv* has registered the highest number of total errors (including all the four categories) with 2,179 corruptions over 10,000 injections while *matmul* and *dijkstra* collected 1,028 and 1,026 errors, respectively. It can be noticed that the highest value always belongs to the *Halt* label for all the software applications, most likely due to the corruption of communication modules (e.g. UART controller) within the design.
As can be seen in Figure 5, the cluster size has a marginal effect on the error rate. It may be related to the fact that bits associated with a specific hardware resource are located closely in the configuration memory. If that part of configuration memory is selected as a fault location, the size of the injected cluster will only marginally increase the corruption of the used logic resource.

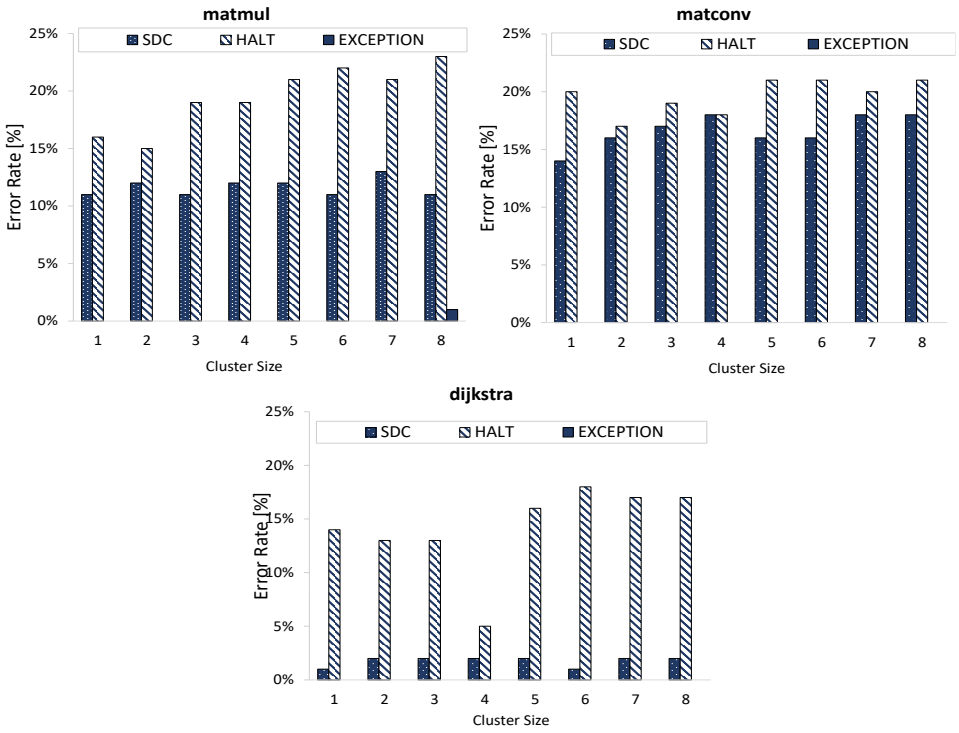

Fig. 5. Error rates for each application with fixed MBU injection

To clarify further the sensitivity of the analyzed software and platform, the mean-time-to-failure (MTTF) and the mean-executions-to-failure (METF) have been evaluated for the various applications, based on the cross-sections of different cluster sizes at different energies, and the error rate of the applications against the specific clusters resulting from the second fault injection campaign. The MTTF is the mean time between two faulty outcomes of the application. The MTTF is reported in Figure 6 based on energies and the fluxes reported in Table 1. The METF has been computed using the average execution time of the software applications. As is expected, for the particles with lower energies, the Mean Execution To Failure as well as Mean Time to Failure is higher which means that fewer errors or failures have been observed. By the particles moving toward the one with higher energies, the failure rate increases which leads to a smaller value for MTTF and METF.

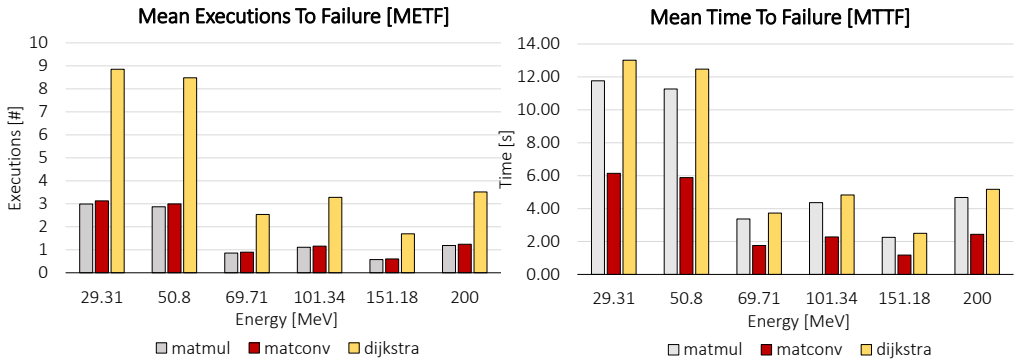

Fig. 6. METf and MTTF for energies ad fluxes reported in Table 1

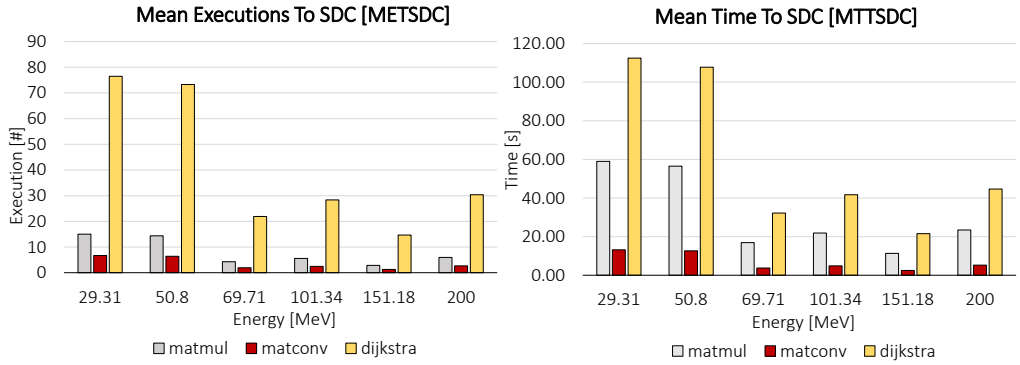

Fig. 7. METSDC and MTTSDC for energies ad fluxes reported in Table 1

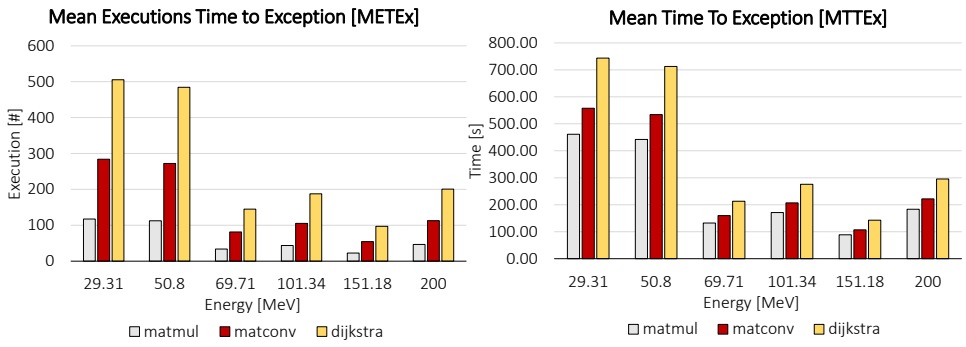

Fig. 8. METEx and MTTEx for energies ad fluxes reported in Table 1

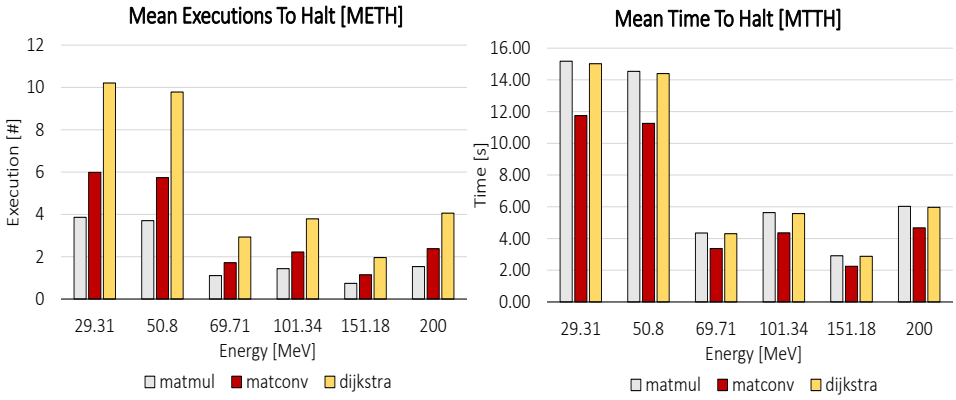

Fig. 9. METH and MTTH for energies ad fluxes reported in Table 1

Moving forward, we performed a deeper analysis to compare the failure causing SDC, Halt, and Exception. As is reported in Figures 7, 8, and 9, for all three applications, Mean Execution/Time to HALT has the lowest value and therefore highest failure rate with respect to SDC and Exception. This leads to the conclusion that the system failure due to Halt is the most critical one which should be taken into account for future mitigation techniques in order to provide a real-time operating system that is tolerant against radiation-induced failure and therefore suitable for space applications.

7 Conclusions and Future works

In this paper, we provided an evaluation of the reliability of the software applications running on Soft Core Microblaze running FreeRTOS. The impact of radiation-induced architectural failures identified through a proton radiation test, affecting different applications has been evaluated. The occurrence and contribution to the error rate of specific MBUs events based on different shapes and sizes have been evaluated in detail. Moreover, we performed a deep reliability analysis in order to classify the radiation-induced errors causing the failure of the running software application in terms of Mean Time/Execution To Failure. In the future, we plan to consider both software and hardware approaches for mitigating radiation-induced errors on applications running within FreeRTOS.

References

[1] Hofmann, et al., "An FPGA based on-board processor platform for space application," NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 17-22, 2012, DOI: 10.1109/AHS.2012.6268653.

[2] S. Azimi, C. De Sio, D. Rizzieri, L. Sterpone, “Analysis of Single Event Effects on Embedded Processor,” MPDI Electronics, vol 10, 2021, DOI: 10.3390/electronics10243160.

[3] "IEEE Standard for a Real-Time Operating System (RTOS) for Small-Scale Embedded Systems," in IEEE Std 2050-2018, pp.1-333, 2018, DOI: 10.1109/IEEESTD.2018.8445674.

[4] L. Sterpone, B. Du, S. Azimi, “Radiation-induced single event transients modeling and testing on nanometric flash-based technologies”, in Microelectronics, vol.55, pp. 2087 – 2091, 2015, DOI: 10.1016/j.microrel.2015.07.035.

[5] S. Azimi, C. De Sio, L. Sterpone, “Analysis of radiation-induced transient errors on 7 nm FinFET technology”, in Microelectronics Reliability, vol. 126, 2021, DOI: 10.1016/j.microrel.2021.114319.

[6] Y. Chen, "Cosmic Ray Effects on Personal Entertainment Applications for Smartphones," IEEE Radiation Effects Data Workshop (REDW), 2013, pp. 1-4, DOI: 10.1109/REDW.2013.6658194.

[7] De Sio, S. Azimi, A. Portaluri, L. Sterpone., "SEU Evaluation of Hardened-by-Replication Software in RISC- V Soft Processor," IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1-6, 2021, DOI: 10.1109/DFT52944.2021.9568342.

[8] Á. B. de Oliveira et al., "Evaluating Soft Core RISC-V Processor in SRAM-Based FPGA Under Radiation Effects," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1503-1510, July 2020, DOI: 10.1109/TNS.2020.2995729.

[9] P. Rech, et al., "Reliability Analysis of Operating Systems for Embedded SoC," 2015 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2015, pp. 1-5, DOI: 10.1109/RADECS.2015.7365659.

[10] S. Azimi, B. Du, L. Sterpone, "On the prediction of radiation-induced SETs in flash-based FPGAs", in Elsevier Microelectronics Reliability, pp. 230-234, 2016, DOI: 10.1016/j.microrel.2016.07.106.

[11] S. Azimi and L. Sterpone, "Digital Design Techniques for Dependable High Performance Computing," 2020 IEEE International Test Conference (ITC), 2020, pp. 1-10, DOI: 10.1109/ITC44778.2020.9325281.

[12] B. Du, S. Azimi, et al., "Ultrahigh Energy Heavy Ion Test Beam on Xilinx Kintex-7 SRAM-Based FPGA," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1813-1819, 2019, DOI: 10.1109/TNS.2019.2915207.

[13] S. Azimi, C. De Sio and L. Sterpone, "A Radiation-Hardened CMOS Full-Adder Based on Layout Selective Transistor Duplication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 8, pp. 1596-1600, Aug. 2021, doi: 10.1109/TVLSI.2021.3086897.

[14] P. V. Nekrasov, et al., "Investigation of Single Event Functional Interrupts in Microcontroller with PIC17 Architecture," 2015 15th European Conference on Radiation and Its Effects on Components and Systems (RADECS), pp. 1-4, 2015, DOI: 10.1109/TNS.2019.2915207.

[15] D. Mamone, et al., "On the Analysis of Real-time Operating System Reliability in Embedded Systems," IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1-6, 2020, DOI: 10.1109/DFT50435.2020.9250861.

[16] I. O. Loskutov et al., "Investigation of Operating System Influence on Single Event Functional Interrupts Using Fault Injection and Hardware Error Detection in ARM Microcontroller," International Siberian Conference on Control and Communications (SIBCON), pp. 1-4, 2021, DOI: 10.1109/SIBCON50419.2021.9438916.

[17] P. M. Aviles, et al., "Radiation Testing of a Multiprocessor Macrosynchronized Lockstep Architecture With FreeRTOS," in IEEE Transactions on Nuclear Science, vol. 69, no. 3, pp. 462-469, March 2022, DOI: 10.1109/TNS.2021.3129164.

[18] I. O. Loskutov, et al., "SEFI cross-section evaluation by fault injection software approach and hardware detection," IEEE 30th International Conference on Microelectronics (MIEL), pp. 251-254, 2017, DOI: 10.1109/MIEL.2017.8190114.

[19] W. Mansour and R. Velazco, "SEU fault-injection in VHDL-based processors: A case study", Journal of Electronic Testing: Theory and Applications (JETTA), vol. 29, no. 1, pp. 87-94, 2013, DOI: 10.1109/LATW.2012.6261258.

[20] A. Boutros and V. Betz, "FPGA Architecture: Principles and Progression," in IEEE Circuits and Systems Magazine, vol. 21, no. 2, pp. 4-29, Secondquarter 2021, DOI: 10.1109/MCAS.2021.3071607.

[21] FreeRTOS, “Xilinx Microblaze Port” informative webpage. [Online]. Available: https://bit.ly/3r5Y3ph. Accessed on: April 06, 2022.

[22] Xilinx, “MicroBlaze Processor Reference Guide”, UG984 v2021.2, Oct 27, 2021, pp. 80-89. [Online].L. Bozzoli, et al., "PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing," International Symposium on Rapid System Prototyping (RSP), pp. 70-75, 2018, DOI: 10.1109/RSP.2018.8632000.

[23] L. Bozzoli, et al., "PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing," International Symposium on Rapid System Prototyping (RSP), pp. 70-75, 2018, DOI: 10.1109/RSP.2018.8632000.