Characterization of a RISC-V System-on-Chip under Neutron Radiation

Douglas Almeida dos Santos, Lucas Matana Luza, Maria Kastriotou, Carlo Cazzaniga, Cesar A Zeferino, Douglas Rossi Melo, Luigi Dilillo

Title: Characterization of a RISC-V System-on-Chip under Neutron Radiation

Author(s): Douglas A. Santos, Lucas M. Luza, Maria Kastriotou, Carlo Cazzaniga, Cesar A. Zeferino, Douglas R. Melo, and Luigi Dilillo

DOI: 10.1109/DTIS53253.2021.9505054

Published: 09 August 2021

Document version: Post-print version (Final draft)

Please cite the original version:
D. A. Santos *et al*., "Characterization of a RISC-V System-on-Chip under Neutron Radiation," 2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2021, pp. 1-6, doi: 10.1109/DTIS53253.2021.9505054.

Characterization of a RISC-V System-on-Chip under Neutron Radiation

Douglas A. Santos*†, Lucas M. Luza*, Maria Kastriotou ‡, Carlo Cazzaniga ‡,
Cesar A. Zeferino†, Douglas R. Melo†, and Luigi Dilillo*§
* University of Montpellier, LIRMM UMR 5506, Montpellier, France
† University of Vale do Itajaí, Laboratory of Embedded and Distributed Systems (LEDS), Brazil
‡ ISIS Facility, STFC, Rutherford Appleton Laboratory, United Kingdom
§ Centre National de la Recherche Scientifique (CNRS), France
dalmeidados@lirmm.fr, lucas.matana-luza@lirmm.fr, maria.kastriotou@stfc.ac.uk,
carlo.cazzaniga@stfc.ac.uk, zeferino@univali.br, drm@univali.br, dilillo@lirmm.fr

Abstract

Systems for harsh environments often use embedded processors for tasks that require reliability. However, harsh environments cause faulty behavior in electronics, which eventually lead to system failure. Therefore, embedded processors must use techniques to improve their reliability. In this context, this work presents the implementation and characterization of a RISC-V-based system-on-chip. We characterized our implementation by carrying out test campaigns at the ChipIr irradiation facility. This facility provides a beamline for testing electronics against neutrons, mimicking atmospheric-like environments. With this first test campaign, we identified the most critical parts of our system-on-chip and essential tips to improve the test effectiveness. In the second test campaign, we used an improved version of the system setup with higher reliability error observability features. The version embedding all the hardening techniques could correct or mitigate 98.1 % of the detected upsets under irradiation.

Index Terms

RISC-V, System-on-Chip, Fault Tolerance, Neutrons, Radiation

I. Introduction

Embedded systems are often used in harsh environments, in which ionizing particles interact with semiconductors causing faulty behavior. The atmospheric environment, for example, presents a cascade of secondary particles, including neutrons, protons, and electrons, due to the interaction of the atmospheric gases and galactic cosmic rays [1]. The interaction of these particles with electronics can lead to transient, permanent, or intermittent faults [2]. As a result, electronic systems may suffer malfunctions and catastrophic failures.

Among the effects that can affect the behavior of electronic devices, a major role is expressed by Single-Event Effects (SEEs). These effects arise from ionizing particles penetrating sensitive nodes of electronic systems [3]. SEEs may lead to many events, such as Single Event Transient (SET) and Single Event Upset (SEU). SETs are characterized by transient turbulence in the signals, which can lead to incorrect circuit action. SEUs are similar to SETs, but they affect the memory elements with bit-flips [2]. According to [4], SEUs have a very high occurrence probability and must be considered when designing embedded systems.

Several techniques are used to harden systems against SEUs and one of them is the use of redundancy. According to [5], there are three classes of redundancy: spatial (also referred to as ”physical”), temporal, and information. Spatial redundancy comprises the utilization of more than one instance of the same component. TMR (Triple Modular Redundancy) is an example of spatial redundancy, which consists of instantiating a module three times and using a majority voter to select the most frequent output to mitigate errors. Temporal redundancy is the utilization of the same instance more than once. Information redundancy is the addition of redundant information to the data, such as ECCs (Error-Correcting Codes), that can detect and correct errors that may affect system behavior.

Several processors were designed to target harsh environments, applying one or more techniques to harden the processors. The LEON4-FT [6] processor applies ECC in RAM blocks to mitigate SEUs. MIPS Crypto [7] uses TMR and matrix encoding techniques to protect its buffer units. MIPSFT [8] employs Hamming code to protect registers and TMR for providing fault tolerance to the message passing interface.

RISC-V is an emerging processor architecture that was specified by [9]. Its design is similar to MIPS, and it also focuses on a simple design to simplify the implementation. Even though several open-source processors use this

This work was financed by CNPq – the Brazilian National Council for Scientific and Technological Development – Processes 315287/2018-7 and 436982/2018-8, the Region d’Occitanie – Contract 20007368/ALDOCT-000932, and the École Doctorale I2S de l’Université de Montpellier.

architecture, there are not many implementations of this architecture for reliable applications. The authors in [10], [11] adapted existing versions of RISC-V processors using a tool to harden the design at the netlist level. The work of [12] explored the soft-error hardening of the decoding stage of the processor's pipeline. They applied Hsiao ECC in the register file and TMR in the pipeline registers, decoder, and controller.

One way to evaluate the efficiency of the fault tolerance and characterize processors is through neutron irradiation, which causes SEUs [13]. The authors of the works [10], [11] adapted open-source RISC-V SoCs (Systems-on-Chip) and characterized them against neutron radiation. Their hardening process consists of using the BL-TMR tool [14] to triplicate the design at the netlist level. This tool can identify and triplicate critical nodes in the netlist, improving the system's reliability based on user-defined granularity parameters. Their main focus was on evaluating the operation of the processors when running on an SRAM-based FPGA (Field-Programmable Gate-Array) device. The work of [15] evaluated the implementation of a hybrid fault-tolerant LEON3 processor in an SRAM-based FPGA against neutron radiation. This processor employs EDAC (Error Detection and Correction) code and scrubbing in the RAM, ROM, and cache memories. They implemented single-event correction with duplication in the register file. For the FPGA, they applied configuration memory scrubbing, which ensures that the programming bitstream is correct. As a result of the processor assessment against the neutron radiation, they reported a 4.13 times reliability improvement when the hardened implementation is used. It is worth noticing that the authors compared the system with hardening only in the memory versus the system with all the fault-tolerant techniques.

In this context, we developed a low-cost RISC-V core with basic hardening features in a previous work [16]. This core was a single-cycle processor implementation that employed Hamming ECC to protect all internal registers and TMR to protect the Arithmetic and Logic Unit (ALU) and the control unit. We evaluated this simple processor's reliability by running simulations with SEU and SET fault injections. As a result, we obtained an average error propagation rate improvement of 50.4 %, considering both SEUs and SETs and the worst-case scenario.

Here, the proposed work presents the assessment of an improved version of the previous implementation. We improved that processor to provide the means to use it as a microcontroller and evaluated its reliability features through two test campaigns in the ChipIr irradiation facility. Furthermore, for the implementation, we employed a flash-based FPGA innerly resilient to SEUs in the configuration memory [17] in order to focus on the microcontroller reliability features and filter the errors in the FPGA.

The remainder of this article is organized as follows. Section II describes the experimental setup and the irradiation facility. Section III presents the preliminary characterization campaign. Section IV describes the design of the second improved RISC-V SoC implementation. Subsequently, Section V discuss the characterization results. Finally, Section VI gives conclusions and final remarks.

II. Experimental Setup

The goal of this work is to analyze the behavior of the implemented SoC against neutron radiation. For this purpose, we carried out tests at the Rutherford Appleton Laboratories – UK, more specifically at the ChipIr beamline [18]. The ChipIr instrument mimics the atmospheric neutron spectrum, which causes SEE on electronic devices. This instrument irradiates the device under test (DUT) with a flux up to 10^9 times greater than the typical natural radiation environment [19]. This high flux enables accelerated testing of electronic devices. According to [20], the average flux provided by this beamline is 5.6×10^6 n/cm^2/s for energies above 10 MeV.

We used the M2S010 FPGA device from the Smartfusion2 family by Microsemi. This family of devices is designed for reliable applications [21] and is highly immune to SEUs because the FPGA configuration is flash-based, which is resistant to this type of SEE [22]. The chosen FPGA, SoC M2S010-VFG400 FPGA, was embedded in the Trenz SMF2000 board.

While the FPGA configuration memory is flash-based, the Block RAMs (BRAMs) use an SRAM technology, and the Flip-Flops are D-type (DFF). We are using both BRAMS and DFFs in our design, which are susceptible to SEUs, as shown in the device characterization made in [17].

We carried out two test campaigns at ChipIr. The first test campaign [23] consisted of a preliminary characterization of the microcontroller to analyze its critical parts, which is described in Section III. Based on this preliminary characterization, we improved the implementation of hardening techniques in the microcontroller and characterized it in a second test campaign, in which we also substantially improved the test setup, as described in Section IV.

III. Preliminary SoC Characterization

We characterized an SoC based on the core developed in the previous work [16]. The core implements the entire integer instruction-set (RV32I), not including system calls and fence instructions. The main contribution of the previous work was the processor's hardening against SEUs by applying Hamming code and TMR techniques at critical components of the internal microarchitecture. We applied Hamming code with single-error correction at the

register file and Program Counter (PC); and TMR at the ALU and the control. We focused on a small footprint to implement this core. Thus, we implemented a single-cycle processor microarchitecture that lacks interruptions and Control and Status Registers (CSRs).

We extended the implementation of the core by adding a state machine, changing the core into a multi-cycle processor. The state machine enabled the processor to support SRAM memories since a grant signal now controls the load and store instructions' execution. It also enables an external SoC wrapper to multiplex the signal between the data memory and the AMBA AXI4-lite bus.

We implemented an SoC top module with two separate memories, the instruction and data, which the memory programmer initializes. The memory programmer is also responsible for resetting and starting the execution of the core. After the core starts, the memories are no longer accessible by the memory programmer but only by the core. The accesses from the core to write and read to/from the data memory may have two targets: the data memory or the AXI4-lite bus. A multiplexer is responsible for selecting the access target based on the provided address. During this process, the core is stopped and waits for the grant signal.

We executed the preliminary test in the ChipIr facility using two M2S010 Smartfusion2 boards simultaneously. Both boards were exposed to the same radiation fluence and side-by-side. Fluence is the total number of particles per unit area irradiated in the board, defined as neutrons per square centimeter.

For the processor's execution, we followed some steps in a computer to test the hardened and non-hardened implementations of the microcontroller. The steps consist in (i) programming the bitstream, which alternates between the implementations; (ii) initialization of instruction and data memories; (iii) start the processor and monitor the UART output; (iv) stops the processor when it identifies a timeout (thirty seconds of inactivity) or a hundred lines with errors.

The benchmark chosen for this preliminary test is composed of both the sum of vectors and Fast Fourier Transform (FFT) algorithms. The sum of vectors algorithm consists of generating and summing two 256-element vectors and storing the result in a third 256-element vector. After that, the result vector is printed to the UART. The FFT algorithm generates an input 256-element vector, computes the Fourier transform, and stores the result into the input vector, which is then printed out.

We evaluated the errors in the UART output by comparing it to a golden execution free of errors. We are considering failure as any different value in the output or a timeout in the execution. Regarding the executions, we evaluated only those with at least one perfect complete execution of both algorithms. This filtering guarantees that the failures are arising from the processor's execution and not the initialization of the memory. After applying this filter, the number of executions was reduced from 284 to 251. We classified the executions according to the failure type. For that, we compared the output to a golden execution (with no radiation) and classified the UART output or outcome, as (i) 1- and 2-*bit upset*, when the output value has the respective hamming distance in comparison with the golden value; (ii) *hang*, which represents the executions that were executing perfectly but did not finish; and (iii) *mismatch*, which represents outputs with more than two bit upsets and errors in the printed string.

Table I presents the failure classification for the executions in each board and implementation variation. We notice that, for most failure types, we do not have clear different results between hardened and non-hardened implementations. Between Boards 0 and 1, the percentage results for hang, 1-bit upset, and mismatch failures were inconsistent. While on Board 0 these failures had fewer occurrences in the non-hardened implementation, Board 2 had fewer occurrences in the hardened implementation.

TABLE I
FAILURE CLASSIFICATION OF THE EXECUTIONS

Failure type	Board 0		Board 1	
	Non-hardened	Hardened	Non-hardened	Hardened
1-bit upset	25.86 %	23.73 %	27.69 %	34.78 %
2-bit upset	8.62 %	1.69 %	7.69 %	2.90 %
Hang	25.86 %	18.64 %	16.92 %	18.84 %
Mismatch	39.66 %	55.94 %	47.70 %	43.48 %

Since the results look inconclusive at first analysis, we used the data from the device characterization made by the FPGA producer in [17] to help their interpretation. For this characterization, several Microsemi devices were tested against neutron radiation, in particular the Smartfusion2 family. Among the reported results, the authors obtained an estimated failure rate for the logical resources of the FPGA. Although they did not use the same device that we used, we expect a similar behavior for our FPGA. Therefore, we have compared our results with the characterization of the M2S050 device.

Based on the average fluence that leads to a failure, we estimated that we had, on average, two upsets in the memory and 0.05 in the internal registers. The entire test campaign had 14 upsets in internal registers and 504 upsets in the memories. Therefore, the sources of failure in our SoC should be mainly located in the instruction and data memories since the fluence of our experiment was not large enough to cause a statistically significant number of upsets in the internal registers.

At the end of the preliminary test campaign, we could also express several conclusions besides the characterization results. The lack of observation techniques implemented in this first architecture restrained a detailed analysis of the system's fault source and propagation. Besides that, considering the type of beam and fluence, the number of DUTs could be increased to achieve better statistics concerning the occurred faults, especially when considering the processor itself. Furthermore, the choice of the benchmark algorithms (sum of vectors and FFT) could also be improved to stimulate more widely and effectively all the system's structure. Finally, and most importantly, we concluded that greater attention must be brought to the system memory since it represents the weakest part of the SoC. In particular, the instruction memory is the most critical one since, besides the sensitivity of the data itself, this part of the memory is not refreshed with new data during operation like data memory, thus enabling error accumulation.

IV. Proposed RISC-V System-on-Chip

For the improved RISC-V SoC implementation, we started with the implementation presented in Section III and took into account all the lessons learned in the preliminary characterization campaign, as detailed above. Since the weakest part of the SoC was identified to be the instruction memory, we placed the instructions in a flash-based memory, which is more resilient to SEUs than the BRAM (SRAM-based). We still used BRAM to implement the data memory, but we hardened it by applying Hamming code with single error correction and double error detection (SECDED). We also learned that we need observation points to analyze the results properly. Hence, we implemented error counters in the core architecture, which can count the number of 1- and 2-bit upsets and are accessible for debugging. In order to improve the test of the processor, we employed the Coremark benchmark, which uses more functionalities of the processor's architecture. Also, we improved the statistics of the results by increasing the number of DUTs to four. Fig. 1 presents the final version of the SoC.

We used the device's eNVM (Embedded Non-Volatile Memory) as our on-chip **flash memory** for the application data. It is accessed through the AMBA bus and the APB3 bridge. We had to add this bridge because the flash memory is not directly accessible to the FPGA but only through the device's Microcontroller Subsystem (MSS). The eNVM stores the entire application code, so at the beginning of the execution, the processor loads all the data sections from the flash into the data memory. In order to ensure that the loading was performed correctly, we also calculate the CRC-32 (Cyclic Redundancy Check-32) and check with the result value.

The implementation of the **data memory SECDED** consists of increasing the data width from 32 to 39 bits, which corresponds to the required ECC data. We implemented a Hamming encoder to compute the ECC and a Hamming decoder for the error-correcting and detecting capabilities.

Regarding the hardening of the core **registers**, we extended the implementation of its Hamming decoders to support the double error detection, besides the single error correction.

From the RISC-V privileged specification **CSRs** [24], we implemented the 64-bit cycle counter, registers for user traps, and machine information. Besides the specification CSRs, we also implemented custom registers to configure the core's hardening capabilities and provide event upsets counting, and the SoC reset cause.

The **hardening configuration CSR** activates flags to the hamming decoders and the TMR voters to indicate if the errors should be corrected. This register improves the test campaign since the FPGA does not need reconfiguration to test different hardening implementations. The hardening configuration works by activating an error correction signal for the Hamming decoders and TMR voters. Therefore, at the processor initialization, the software configures this register according to the desired hardening capabilities.

Regarding **observability**, we implemented CSRs for error counting. Two counters were implemented for each hamming decoder plus one for each TMR voter. In total, we implemented ten error counters: for the Hamming decoders of the PC, data memory, and both read ports of the register file, and for the control and ALU TMR voters. The error counters increment each single- or double-bit flip detected from the hamming decoders and for each error mitigated by the TMR voters.

At the SoC level, we also added a **Watchdog Timer** (WDT). This WDT understands when the processor stops responding and resets the entire SoC. This peripheral is connected to the AMBA bus, and it is enabled by default. If the timer value reaches zero, the watchdog reset triggers for the entire SoC. Since we did not connect the error counters to the reset and implemented the reset cause CSR, every time we identify a watchdog reset, the processor

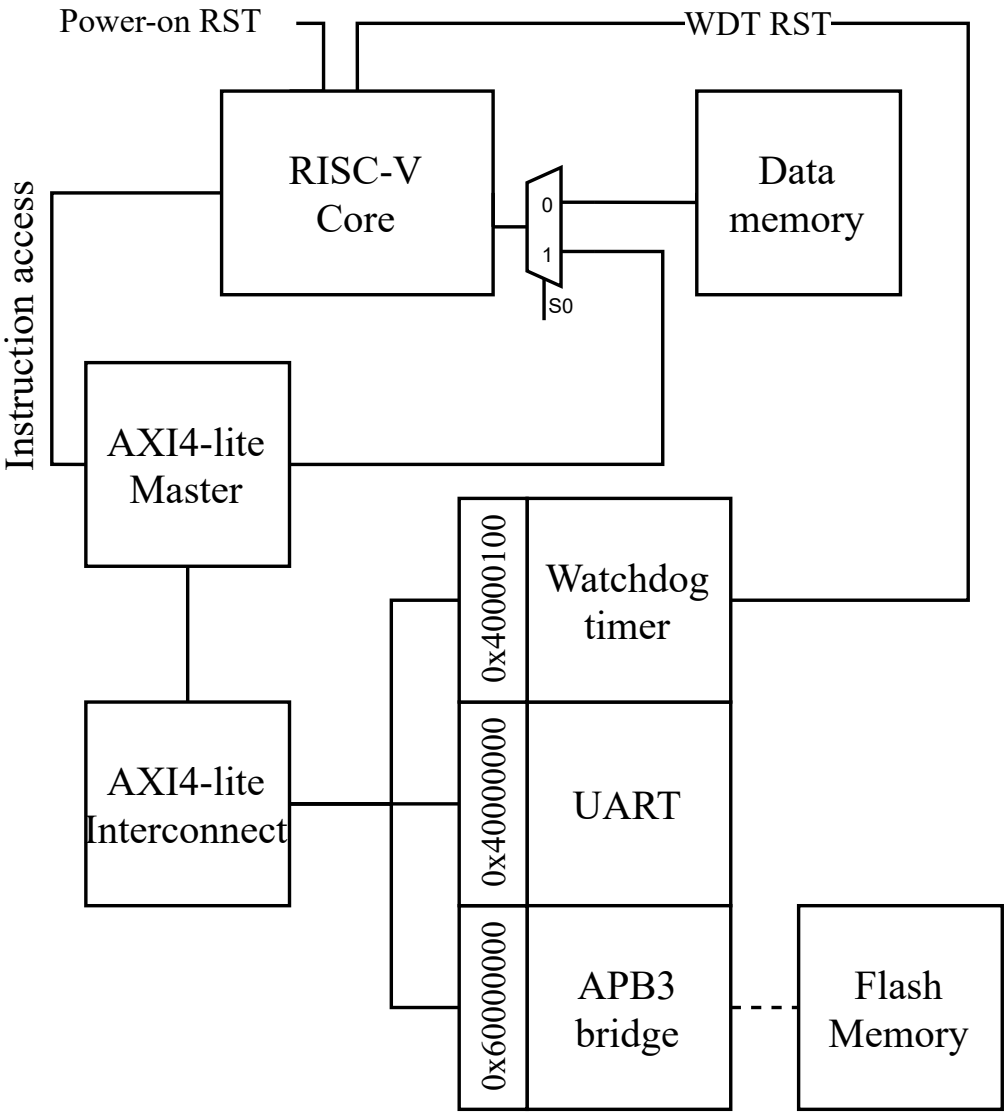

Fig. 1. RISC-V System-on-Chip

prints to the UART output the current values from the error counters. This implementation enables further analysis of the errors.

A. Implementation in the FPGA

The application of fault tolerance at the components increases the number of logical resources required. Table II presents the results obtained from the Libero v2021.1 tool with the SoC running at 50 MHz and targeting the FPGA M2S010-VFG400. We noticed that the hardening application in the processor increased the usage of logic elements by 50.7 % in comparison to the implementation without hardening. This increase is due to the triplicating of the ALU and control modules and the SECDED application at internal registers. The logic elements report combines a 4-input Look-Up Table (LUT-4) and a DFF since each logic element comprises a LUT-4 and a flip-flop. When we apply SECDED in the data memory, we notice an increase in the usage of RAM18K blocks, which is due to the 7-bit increase in the data width. Besides the increase in memory blocks, we noticed an increase of 22.8 % in logic elements when we harden only the memory; this results from the implementation of the hamming encoder and decoder. Implementation Proc+Mem resulted in an increase of 75 % in logic elements usage compared to implementation None.

Besides the logic elements overhead, the SoC suffers a decrease in the maximum operating frequency when hardening techniques are applied. The estimated maximum operating frequency drops from 33.71 MHz to 25.61 MHz.

TABLE II
RESOURCE USAGE IN THE SMARTFUSION2 FPGA

Hardening	LUT-4	DFF	Logic elements	RAM18K	Fmax (MHz)
None	4344	2150	4649	16	33.71
Processor	6686	2385	7005	16	24.55
Memory	5397	2622	5710	20	33.61
Proc+Mem	7824	2852	8140	20	25.61

The reason is mainly that Hamming encoders and decoders are in the system's critical path. However, since the critical path is never fully activated, the SoC can operate at a frequency higher than estimated. Our current SoC is running with a clock of 50 MHz.

B. Characterization experiments

We characterized the RISC-V SoC implementation by running the Coremark benchmark simultaneously in four Smartfusion2 boards; two of the boards are the same used in the preliminary test campaign. All four boards were positioned close to each other and exposed to the same radiation fluence.

The Coremark is a benchmark developed by the Embedded Microprocessor Benchmark Consortium (EMBC), and its goal is to test the performance of microcontrollers. According to the Coremark white paper [25], the benchmark is composed of the operations: (*i*) list processing, which performs reversing, searching, or sorting operations; (*ii*) matrices processing, executing a multiplication with a constant, a vector, or another matrix; and (*iii*) state machine processing, which tests an input string to detect if it is a number. The verification of the correct Coremark operation is done by using CRC-16 on the elements of the processed list.

For our test campaign, we defined the Coremark parameters to improve our analysis. We run the benchmark in a single thread since our algorithm runs as a bare-metal code and does not support context changing. We set the number of Coremark iterations to 200, which takes about 12 minutes to execute. Besides that, we also defined the debug flag of the Coremark, which enables debug printing of the Coremark operations.

At the beginning of each execution during the test campaign, the software sets the hardening CSR according to the previously defined tests. The tests are, in order: (*i*) hardening in the processor and the memory; (*ii*) hardening only in the memory; (*iii*) no hardening; and (*iv*) hardening only in the processor. In each Coremark iteration, we set 10 seconds to the watchdog timer and print the number of errors from the counters. At the end of each execution, a WDT reset is forced by the software, and each execution starts from a reset state.

V. CHARACTERIZATION RESULTS

In the second test campaign, we tested the proposed RISC-V SoC implementation against neutron radiation. In total, the devices were irradiated with a cumulative fluence of 45.9×10^{10} n/cm^2, distributed in the 221 Coremark's executions and all hardening configurations.

In Table III, the benchmark executions are classified according to the UART output. The *match* classification corresponds to the executions that finished perfectly (i.e., all the errors were mitigated or did not affect the execution). The *mismatch* classification comprises executions that printed a different value than expected by the golden run. Moreover, the *timeout* classification corresponds to the executions that did not print lines with errors and suffered a watchdog reset before the execution finished. The columns correspond to the hardened component of the SoC. The columns are: (*i*) None, when no hardening were configured; (*ii*) Processor, in which hardening configured only for the processor; (*iii*) Memory, with hardening configured only for the data memory; and (*iv*) Processor+Memory, when all hardening configurations were enabled.

We noticed that the hardening application in the processor improved the number of *match* executions by 5.31 times compared to the none configuration. This improvement increases to 33.48 times when the hardening in the memory is configured. The improvement obtained from hardening only the memory to hardening both the processor and the memory was small (1.34 %) because most of the errors come from upsets in the data memory. It is worth noting that the only configuration that had *timeout* failures was the one without hardening.

Regardless of the errors in the UART output, the Coremark operation may still run correctly. The errors in the UART could be due to several reasons, such as an error in the transmission, the AMBA bus, and the UART peripheral. Therefore, Table IV presents the classification of the executions based on the Coremark error counter. The *correct* classification means that the Coremark did not detect errors in the execution, while the *error* classification is the

TABLE III
CLASSIFICATION OF THE EXECUTIONS

Outcome	None	Processor	Memory	Processor+Memory
Match	1.92 %	10.20 %	64.29 %	65.15 %
Mismatch	90.38 %	89.80 %	35.71 %	34.85 %
Timeout	7.69 %	0.00 %	0.00 %	0.00 %

contrary. The *timeout* classification is for the executions that failed before reaching the end of the execution. We noticed an improvement of 5.4 % between the none and processor hardening configurations.

On the other hand, when we compare the None configuration to the memory hardening, it improves 34.4 %. The only execution with memory hardening that failed was due to a non-corrected upset in the registers, which affected the processor's control and led to a *timeout* failure. Considering the Coremark result, the configuration processor+memory never failed.

TABLE IV
COREMARK RESULT CLASSIFICATION

Outcome	None	Processor	Memory	Processor + Memory
Correct	73.08 %	77.08 %	98.21 %	100.00 %
Error	1.92 %	2.08 %	0.00 %	0.00 %
Timeout	25.00 %	20.83 %	1.79 %	0.00 %

Table V presents the detected number of upsets during the test campaign for the characterization of the RISC-V SoC and the percentage of them that were corrected. Since most of the upsets are in the memory, the processor-only hardening corrects solely 2.04 % of all the upsets, which correspond to the number of upsets in the processor's internal registers. On the other hand, when we apply hardening in the memory, we increase the percentage of corrected upsets to 95.28 %. When we hardened both the memory and the processor, we noticed an increase of 2.82% in the percentage of corrected errors. It is worth noting that the latest configuration cannot correct all the errors because, in our implementation, the double bit upsets are not correctable.

TABLE V
NUMBER OF UPSETS DURING RISC-V SOC CHARACTERIZATION

Hardening	Upsets	Corrected
None	165	0.00 %
Processor	147	2.04 %
Memory	106	95.28 %
Processor+Memory	105	98.10 %

Regarding the integrity checks, the FPGA bitstream configuration never suffered upsets. Furthermore, the CRC-32 check performed in the application's initialization never detected errors, regardless of the hardening configuration.

VI. CONCLUSION

In this work, we started from the baseline core from previous work and extended it to an SoC to support more complex applications. Subsequently, we performed a preliminary test campaign in order to characterize it in an atmospheric-like neutron environment. From this first test campaign, we observed that the lack of error observation techniques restrained a detailed analysis of the system's fault source and propagation. Besides that, we concluded that the number of DUTs should be increased to have more statistics. Another issue of this preliminary characterization was the lack of a proper benchmark algorithm to test the functionality of the SoC. Finally, the primary source of the errors was identified in the memory.

Afterward, we used these preliminary characterization results to develop an improved version of the SoC. In this second RISC-V SoC, we hardened the memories by placing the instructions in a flash memory, which has high inner resilience to SEUs and applying SECDED in the SRAM memory used for data storage. We also developed observation points by implementing error counters in the core architecture. These error counters were constantly read out to analyze the error propagation within the system. Besides that, we used the Coremark as a benchmark application of the SoC, which better utilizes SoC functionalities.

Finally, we characterized the improved RISC-V SoC implementation in a second test campaign. The results have shown that the SoC could correct most of the upsets caused by neutron radiation. Furthermore, the SoC hardening configuration in both the processor and the memory executed the Coremark benchmark without errors in all the executions.

In future works, we intend to improve the characterization of the SoC by carrying out test campaigns with particles present in space environments, such as heavy ions and protons. The source code of the processor is available at [26].

REFERENCES

[1] P. Cannon, M. Angling, L. Barclay, C. Curry, C. Dyer, R. Edwards, G. Greene, M. Hapgood, R. B. Horne, D. Jackson *et al.*, *Extreme space weather: impacts on engineered systems and infrastructure*. Royal Academy of Engineering, 2013.

[2] M. Yang, G. Hua, Y. Feng, and J. Gong, *Fault-tolerance techniques for spacecraft control computers*, 1st ed. Wiley Publishing, 2017.

[3] E. Petersen, *Single event effects in aerospace*. John Wiley & Sons, 2011.

[4] R. Velazco, R. Ecoffet, and F. Faure, "How to characterize the problem of SEU in processors representative errors observed on flight," in *11th IEEE International On-Line Testing Symposium*, 2005, pp. 303–308.

[5] D. J. Sorin, "Fault tolerant computer architecture," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–104, 2009.

[6] M. Hjorth *et al.*, "GR740: Rad-hard quad-core LEON4FT system-on-chip," in *DASIA 2015-DAta Systems in Aerospace*, vol. 732, 2015.

[7] B. Ustaoglu and B. O. Yalcin, "Fault tolerant register file design for MIPS AES-crypto microprocessor," in *2015 IEEE International Conf. on Electronics, Circuits, and Systems (ICECS)*, Dec 2015, pp. 442–445.

[8] M. Didehban, S. Khosjbakht, H. R. Zarandi, and S. Pourmozaffari, "Reducing of soft error effects on a MIPS-based dual-core processor," in *2010 15th CSI International Symp. on Computer Architecture and Digital Systems*, Sep. 2010, pp. 151–152.

[9] E. A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: User-Level ISA Document Version 20191213*, RISC-V Foundation, December 2019.

[10] A. E. Wilson and M. Wirthlin, "Neutron radiation testing of fault tolerant RISC-V soft processor on Xilinx SRAM-based FPGAs," in *2019 IEEE Space Computing Conf. (SCC)*, July 2019, pp. 25–32.

[11] A. E. Wilson, S. Larsen, C. Wilson, C. Thurlow, and M. Wirthlin, "Neutron radiation testing of a TMR VexRiscv soft processor on SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2021.

[12] N. Marcello, "Design of a fault tolerant instruction decode stage in RISC-V core against soft and hard errors," Ph.D. dissertation, Politecnico di Torino, 2021.

[13] E. Normand, "Correlation of inflight neutron dosimeter and SEU measurements with atmospheric neutron model," *IEEE Transactions on Nuclear Science*, vol. 48, no. 6, pp. 1996–2003, 2001.

[14] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA design robustness with partial TMR," in *2006 IEEE International Reliability Physics Symp. Proc.*, March 2006, pp. 226–232.

[15] A. Lindoso, L. Entrena, M. García-Valderas, and L. Parra, "A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA," *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 374–381, 2017.

[16] D. A. Santos, L. M. Luza, C. A. Zeferino, L. Dilillo, and D. R. Melo, "A low-cost fault-tolerant RISC-V processor for space systems," in *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2020, pp. 1–5.

[17] D. Dsilva, J.-J. Wang, N. Rezzak, and N. Jat, "Neutron SEE testing of the 65nm SmartFusion2 flash-based FPGA," in *2015 IEEE Radiation Effects Data Workshop (REDW)*, 2015, pp. 1–5.

[18] C. Cazzaniga and C. D. Frost, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," *Journal of Physics: Conference Series*, vol. 1021, p. 012037, may 2018. [Online]. Available: https://doi.org/10.1088/1742-6596/1021/1/012037

[19] C. Cazzaniga, M. Bagatin, S. Gerardin, A. Costantino, and C. D. Frost, "First tests of a new facility for device-level, board-level and system-level neutron irradiation of microelectronics," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[20] D. Chiesa, M. Nastasi, C. Cazzaniga, M. Rebai, L. Arcidiacono, E. Previtali, G. Gorini, and C. D. Frost, "Measurement of the neutron flux at spallation sources using multi-foil activation," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 902, pp. 14–24, 2018.

[21] *IGLOO2 and SmartFusion2 Datasheet Version DS0128*, Microsemi, 2018, rev. 12.

[22] *SmartFusion2 SoC FPGA Version PB0115*, Microsemi, 2018, rev. 27.

[23] L. Dilillo *et al.*, "Evaluation of a fault-tolerant RISC-V," *STFC ISIS Neutron and Muon Source*, Nov. 2020.

[24] E. A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20190608-Priv-MSU-Ratified*, RISC-V Foundation, December 2019.

[25] S. Gal-On and M. Levy, *Exploring coremark a benchmark maximizing simplicity and efficacy*, Embedded Microprocessor Benchmark Consortium, 2012.

[26] HARV. HArdened Risc-V. [Online]. Available: http://xarc.org/harv