Hybrid Hardening Approach for a Fault-Tolerant RISC-V System-on-Chip

Douglas A Santos, Pablo M Aviles, André M P Mattos, Mario García-Valderas, Luis Entrena, Almudena Lindoso, Luigi Dilillo

Title: Hybrid Hardening Approach for a Fault-Tolerant RISC-V System-on-Chip

Author(s): Douglas A. Santos, Pablo M. Aviles, André M. P. Mattos, Mario García-Valderas, Luis Entrena, Almudena Lindoso, and Luigi Dilillo.

DOI: https://doi.org/10.1109/TNS.2024.3406021

Document version: Pre-print version (Submitted draft)

Please cite the original version:
D. A. Santos et al., "Hybrid Hardening Approach for a Fault-Tolerant RISC-V System-on-Chip," in IEEE Transactions on Nuclear Science, doi: 10.1109/TNS.2024.3406021.

Hybrid Hardening Approach for a Fault-Tolerant RISC-V System-on-Chip

Douglas A. Santos, Pablo M. Aviles, André M. P. Mattos, Mario García-Valderas, Luis Entrena, Almudena Lindoso, and Luigi Dilillo,

Abstract

The New Space era has driven a wide array of applications in novel space missions with an increasing demand for processors with high computational capabilities while simultaneously maintaining low power consumption, flexibility, and reliability. Soft-core processors based on Commercial Off-The-Shelf (COTS) technologies have emerged as a viable alternative that can meet these requirements but need to be hardened to operate in harsh environments. In this work, we propose and apply fault tolerance strategies based on software recovery using checkpoint and rollback operations to extend the capabilities of a hardened soft-core RISC-V-based System-on-Chip. The proposed strategy relies on the fault awareness provided by the SoC hardening to trigger the software recovery without the addition of dedicated structures or processor cores. Notably, we investigate the effectiveness of this multilayer hardening strategy, which combines software recoverability and hardware redundancy. For that, a neutron irradiation campaign was performed. The results show the effectiveness of the proposed approach, achieving 45.09% of effective software recovery operations with low overhead for performance and resource utilization.

Index Terms

Fault Tolerance, Reliability, Neutron, Soft Error, Radiation, Checkpoint, Rollback, RISC-V

I. Introduction

In recent years, an increasing interest in space exploration and commercial applications related to the space sector has been observed, not only by governments and agencies but also by private companies, which became known as the New Space era. SpaceX [1] and Blue Origin [2] are two private companies in the space sector that have achieved significant milestones. SpaceX became the first private company to transport humans to the International Space Station (ISS) in 2020. Other notable examples include the deployment of global high-speed satellite internet constellations [3], Earth observation services, and Martian exploration missions [4]. In this context, processors are being employed to carry out increasingly complex tasks that demand high computational capabilities [5]–[7], all while striving to maintain low costs, power consumption, and reliability. Despite the multitude of hard-core processors available in the market, soft processors are becoming popular for applications that require high performance and low power consumption. For instance, they are more customizable, and can be tailored to predefined specifications and modified to accommodate design alterations during the development process (designers can change the number of cores, the size of the cache, and other features to meet the specific needs of the application). However, these features are not sufficient for soft-core processors to be used in the space sector. An important element that can lead to malfunction in space electronic systems is radiation. Thus, safety concerns are imposed when harsh environments are targeted. Processors used in radiation environments must be designed to withstand or mitigate their effects.

RISC-V [8] soft-core processors have recently surfaced as viable options for compute-intensive tasks, thanks to their performance-enhancing architecture, open-source design, scalability, flexibility, and simplified instruction sets. RISC-V soft-core processors are expected to be used in a variety of future applications, including space applications [9]. As an example, the Trisat-R nanosatellite [10], launched in July 2022, is equipped with a SkyLabs NANOhpm On-Board Computer (OBC) powered by a RISC-V NOEL-V processor [11].

Several RISC-V soft-core implementations can be found in the literature [12], [13], even some of which have been developed or evaluated for their use in harsh environments [14], [15]. However, despite a growing ecosystem based on the architecture, not many readily available and validated fault-tolerant RISC-V processors exist. For this reason, in previous works [16], [17], we introduced and characterized a fault-tolerant System-on-Chip (SoC), HARV-SoC. The SoC is compliant with the RISC-V architecture, combining physical and information redundancy to achieve a balanced solution between reliability and hardening overheads. It has a multi-cycle processor core that supports the RISC-V integer instruction set (RV32I), Control and Status

This work was supported in part by the European Union's Horizon 2020 research and innovation program under grant agreement no. 101008126, by the Region d'Occitanie and the École Doctorale I2S from the University of Montpellier (contract no. 20007368/ALDOCT-000932), and by project PID2022-138696OB-C21 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe".

The authors would like to acknowledge the support from the irradiation facility personnel, notably Carlo Cazzaniga and Maria Kastriotou. Also, we state that the three first co-authors had an equivalent contribution to this work, in which the order merely defines their affinity with the work subject.

D. A. Santos, A. M. P. Mattos, and L. Dilillo are with IES, University of Montpellier, CNRS, Montpellier, France. Contact: {douglas.santos, andre.martins-pio-de-mattos, luigi.dilillo}@umontpellier.fr.

P. M. Aviles, M. García-Valderas, L. Entrena, A. Lindoso are with the Electronic Technology Department, University Carlos III of Madrid, Leganes-Madrid, Spain. Contact: {paviles, mgvalder, entrena, alindoso}@ing.uc3m.es.

Registers (CSRs), interrupts, and exceptions. The HARV processor's register file, program counter, and instruction register are hardened using Single-Error Correction and Double-Error Detection (SECDED). The control, Arithmetic-Logic Unit (ALU), and critical CSRs use Triple Modular Redundancy (TMR) for hardening. The SoC was evaluated in neutron, proton, and mixed-field irradiation campaigns, in which effective error mitigation was achieved.

In this work, we combine the recovery strategies explored in [18] with the SoC presented in [16], [17], as a joint effort to improve the SoC reliability. Therefore, we analyze the effectiveness of the key elements of the Macrosynchronized Lockstep (MSLS) architecture implemented in a fault-tolerant RISC-V SoC. In order to best exploit the system characteristics, our main contribution in this work is to use the error reporting of the SoC to trigger the MSLS recovery operations. Since the SoC detects several types of radiation-induced events (e.g., upsets in registers, controlling logic errors) in its internal structures, the rollbacks can be triggered for these error events. Hence, an additional processing core for the lockstep approach is dismissed and the potential non-correctable errors detected by the SoC can be potentially recovered with the MSLS protection at the software level.

This paper is organized as follows. Section II summarizes the related work in this field. The recovery strategy used in the proposed approach is presented in Section III. Section IV describes the implementation and overheads of the checkpoint and rollback operations in the fault-tolerant RISC-V processor. Section V details the experiments that have been performed to validate this approach and presents the experimental results for the radiation test campaign. Finally, Section VI summarizes the conclusions of this work.

II. Related Work

Nowadays, there is a wide variety of protection mechanisms against radiation-induced soft errors. A correct and widely used solution is the rollback-based recovery mechanism. Various approaches can be found in the literature that use this mechanism to achieve system recovery after the occurrence of errors. The error detection to trigger the rollback process is based, in many of these approaches, on lockstep techniques, taking advantage of these systems' ability to detect errors based on spatial redundancy.

In [19], a fault-tolerant reconfigurable system scheme that can be implemented in SRAM-based FPGA is proposed. The approach is capable of recovering from Single-Event Upsets (SEUs) through partial reconfiguration combined with rollback and roll-forward operations. These mechanisms rely on two Micro-Blaze cores in lockstep mode. In [20], another approach that uses a rollback mechanism at the software level and based on lockstep is presented. It is based on both software and hardware redundancy to detect and correct errors. In this case, the hardening technique utilizes a dual-core ARM Cortex-A9 processor. Other authors have also explored the use of different architectures at the same time to achieve fault tolerance. In [21], the authors propose a heterogeneous fault-tolerant architecture based on dual-core lockstep (DCLS) of different architectures. Two different versions are proposed, which combine, in each case, two different classes of processors: a hard-core processor with ARM architecture and a soft-core processor with RISC-V architecture. In both systems, the rollback mechanism is incorporated to deal with the occurrence of errors and mitigate failures by returning to an error-free state. An advantage of this system is the use of different microprocessors, which results in a design with diversity at the ISA level. However, as different architectures are used, the context of both microprocessors cannot be directly compared to determine if there are discrepancies between them, and, therefore, determine if an error has occurred.

Similarly, in [18], we presented another recovery technique aimed to be applied in SoCs, in which a high-end ARM multiprocessor hardened with a Macrosynchronized Lockstep (MSLS) was evaluated. The MSLS technique is based on detecting errors with a loosely-coupled lockstep architecture that triggers software rollbacks from execution context saved in safe memory. Another error mitigation technique that implements the rollback mechanism is presented in [22]. This hardening technique is based on the triple-core lockstep (TCLS). The proposal, unlike [18], is not based only on software modification but also uses verification and replication circuits at the hardware level. Although the proposed system reaches high levels of protection against soft errors (of around 98%), its main issue is the need to include an additional core, the Micro-Blaze, in addition to both cores of the ARM Cortex-A9 processor.

While the proposed approaches effectively implement the rollback mechanism to mitigate radiation-induced soft errors, they all employ more than one core. The use of multiple cores is driven by the need to detect errors through spatial redundancy. However, it is possible to implement rollback in single-core processors designed with error detection mechanisms, including soft-core processors. Numerous synthesizable soft-core processors have been developed for space applications, incorporating fault tolerance techniques to harden various circuitry components [23]–[26]. There are also many hardened RISC-V-based processors in the literature. RISC-V soft processors have emerged as a suitable alternative for applications utilized in harsh environments since designers are able to customize both processor cores and compilers. A more customized soft-core processor facilitates the adjustments on resource usage and power consumption if needed, an important requirement in the space domain.

In [27], the register file of the lowRISC RISC-V processor is duplicated and uses parity checking in both register files to protect them against SEUs. The redundant register performs all operations of the main register in a mirrored mode simultaneously. When an error is detected using the parity checking mechanism in the main register file, the error is masked with the information from the redundant (mirrored) register file. To assess the reliability of their proposed solution, the authors

conducted fault injection experiments on an FPGA. The results demonstrated a reduction in error propagation from 10.17% in the original version to zero in the modified version. In [15], the authors introduce the application of BL-TMR software to harden critical circuits within the Taiga RISC-V processor's netlist. This approach employs the Xilinx Vivado tool's netlist analysis to triplicate critical nodes. The results of the experiments conducted by the authors showed that the fault tolerance technique improved the mean time to failure by a factor of 24, primarily due to a 33-fold reduction in neutron cross-section. However, it led to an increase in logic resource usage and a reduction in operating frequency. On the other hand, in [28], a methodology is introduced to protect the processor's Arithmetic Logic Unit (ALU) from faults in the FPGA configuration memory. The technique proposes protection for only the most common operations. These operations may vary depending on the currently executing algorithm, but SRAM-based FPGAs can be easily reconfigured for each application. Several software applications are evaluated with TMR applied to only two operations of each algorithm. This partial TMR approach significantly enhanced the ALU's reliability with minimal overhead. In [29], the authors conducted an analysis of critical bits within a RISC-V processor implemented on an SRAM-based FPGA. Fault injection campaigns to assess the processor's suitability for space applications were performed. Two RISC-V implementations (unprotected and protected) based on the Rocket RISC-V were evaluated. The conducted experiments demonstrated a 4% enhancement in the reliability of a RISC-V processor implemented on an SRAM-based FPGA with Distributed TMR (DTMR). Given that the unprotected version already achieved a high level of correctness (96%), this modest increase proved sufficient to achieve 99.9% correctness in results.

Although these works enhance the reliability of RISC-V soft-core processors, none utilize recovery strategies such as the rollback mechanism, which can improve the system fault tolerance. In this work, we combine the checkpoint and rollback recovery strategies previously investigated in [18] with the SoC introduced in [16], [17], as a collaborative work aimed at enhancing the reliability of the SoC. Our primary contribution is the utilization of error reporting from a hardened RISC-V SoC to perform rollback recovery operations. Since the SoC is capable of detecting various types of radiation-induced events within its internal structures, rollbacks can be triggered in response to these error events. Consequently, the need for an additional processing core for the lockstep approach is eliminated, and the non-correctable errors detected by the SoC can potentially be recovered.

III. Recovery Strategy

The recovery strategy we propose to apply to the hardened RISC-V processor presented in [16], [17] is based on two distinct operations: checkpoint and rollback. Both are effectively implemented in the Macrosynchronized Lockstep (MSLS) architecture proposed in [18] and allow the processor to return to a previous error-free state in the execution when an error is detected. Being able to restore the microprocessor state implies the need to store the current state so that the system can access this information and use it when required in the restoration process. This approach can lead to unsuccessful recoveries in some cases and must be combined with other strategies (e.g., software restart, watchdog timers), supporting harsh environment applications, in which abnormal behavior and hangs are frequently observed. In our scenario, the processor will be working in a harsh environment and exceptions can be triggered because of radiation-induced effects. Being able to correctly detect the exception by the system and to try to restore regular execution is mandatory.

To implement the proposed recovery strategy, we use context switching. When function calls occur in a regular execution of a software application, context switching is used by default to jump from one function to another and then return correctly to the starting point. Processor context can be defined as the relevant information that characterizes the processor state, and it may include the content of the register file, control registers, cache memory, and others. In function calls, the context is stored on the stack and involves local variables and some registers of the register file. For that reason, we used the saved stack as a context to be stored in the safe memory. However, the more information is stored, the more complete the processor's context will be. Depending on the application and user requirements, global variables and other critical variables besides local variables and the register file should be considered.

In this section, the most significant aspects related to the recovery strategy applied to the hardened soft-core RISC-V-based SoC are presented. In subsection III-A, an overview of the hardened RISC-V processor is presented, whereas in subsections III-B, III-C and III-D, the checkpoint, rollback, and system restart strategies, respectively, are discussed.

A. Hardened RISC-V processor overview

The hardened RISC-V processor we previously designed and presented in [16], [17] uses the RV32I instruction set. Its design minimizes the logical resources required for processor implementation. Consequently, the processor employs a multi-cycle microarchitecture, thereby allowing for a reduction in the number of registers that require hardening. The resulting processor encompasses five core units: instruction fetch, instruction decode, execution, memory access, and write-back.

To make the processor fault-tolerant, we implemented several techniques at the architectural level. Our design specifically addressed soft errors within the processor organization. To protect the ALU and the control unit, we employed the Triple Modular Redundancy (TMR) technique. Consequently, these components were triplicated, with each of their outputs passing through a simple majority voting circuit. For the ALU, there is one voter for each of its two outputs: zero and result. Regarding the control logic, the figure offers a generic depiction of its implementation, featuring decoding/voting circuits for each of the

three output signals generated by this block. Furthermore, we expanded the width of all registers by seven bits and incorporated Hamming encoders and decoders. The encoder employs XOR logic to calculate parity bits for the data slated for registration. The decoder validates the parity of the encoded data, correcting single-bit upsets, and detecting double-bit upsets. Correction entails the inversion of the erroneous bit. Other error detection strategies were also used to monitor communication buses and peripherals, such as access timeouts and Cyclic Redundancy Checks (CRCs).

Besides the basic protection of the SoC, further work has improved the ability of the system to provide awareness of the errors recognized by the hardening structures of the SoC [30]. The error observability evaluated in that work is the main hardware mechanism employed for the proposed recovery strategy, in which the software is notified of errors through the usage of exceptions.

B. Context Saving: Checkpoint

The checkpoint is the mechanism that performs the context saving of the processor in a safe memory. Since local variables are stored on the stack for context switching during function calls, as well as the register file, the stack can be used for checkpoint operation. The Frame Pointer (FP) and Stack Pointer (SP) special registers may be used as references for copying the program stack partially or totally. This allows getting a copy of all the local variables, or many belonging to specific program functions. Also, global variables or some variables considered critical can be included in the checkpoint. In general, the programmer can decide how large is the data set to be protected in the hardened system. It is important to note that saving a large amount of data during checkpoint enhances the probability of effectiveness in recovery, but also increases recovery time (checkpoint and rollback) and requires a larger safe memory, increasing time and memory overhead. At the same time, it can decrease the whole system's performance. Therefore, the programmer must establish a trade-off between these parameters. Only relevant data should be stored, and it is quite important to select the data the programmer wants to protect, which is application-dependent.

Another important aspect is the granularity of checkpoints. The decision of how frequently the checkpoint is made affects the overhead in execution time and memory size. When checkpoints are performed frequently, it implies a higher overhead, but the gap to a functioning saved state is smaller. Checkpoints can be application-dependent, done after user-structured blocks of code, or they can be done periodically via interrupts to be more transparent and simple to use.

C. Context Restoration: Rollback

The rollback is the operation performed when an error is detected and requires recovery handling. It restores the processor context to a previous state, that is, data that has been stored in the safe memory is restored to continue the execution from the last checkpoint. If this checkpoint is successfully saved and contains a valid context, the system can return execution from this error-free state. The validity of a context can be checked using a checksum method. Several consecutive rollbacks can be performed: if a rollback process is not able to restore the system, another rollback is called, pointing to an older checkpoint. Several consecutive rollbacks could have the potential to restore a higher number of errors. A maximum number of consecutive rollbacks (rollback depth) must be set, depending on the number of checkpoints established and available in our hardened system. When the rollback depth is reached, more severe actions should be performed.

D. System restart

Sometimes a system restart is needed because of abnormal behavior or hangs. In harsh environments, these situations are common, and the usage of a watchdog timer could improve the system's reliability. We must study the system's available watchdogs that make it possible to detect and recover abnormal execution states. In the case of presenting several choices, we can establish a watchdog hierarchy. Many high-end processors have several watchdog timers that can control execution and trigger restart if the system is not responding correctly. The number of watchdogs utilized and how frequently they are restarted should be decided. Usually, an additional safety margin is added to the expected execution time.

Besides the watchdog timer resets for hangs, the system should be restarted after unsuccessful attempts of rollback by using a software reset. In some situations, the software reset will decrease the time to restart compared to the watchdog timer reset. For instance, if the maximum selected rollback depth is reached without success, the system will be restored by triggering a software reset and not waiting for a hang occurrence.

IV. Implementation

In this section, we detail the decisions and characteristics of our proposed design based on the concepts introduced in section III. Also, we present and briefly discuss the performance and resource overheads consequent to this solution.

A. *Proposed Design*

The checkpoint operation was implemented through a periodic Interrupt Service Routine (ISR), as shown in the flow diagram in Fig. 1. When the interrupt is triggered, the context of the SoC is transferred to a safe memory. For this work, the saved context consists of the processor stack that includes all the registers in the register file and the local variables of the application. For that purpose, we used BRAM memory blocks with SECDED protection. In order to add robustness, during the writing process to the BRAM, a CRC of 32-bit words (CRC32) is performed. Hence, while these values are extracted from the BRAM during the rollback operation, the CRC32 verification is performed again, and the value obtained is compared with the one previously calculated and stored during the checkpoint process. If the values match, we proceed to restore the processor context to the last stored checkpoint. However, if the CRC32 values do not match, restoration to checkpoint N-1 will be attempted.

The rollback operation, like the checkpoints, is also performed using an ISR. The SoC has built-in error detection, which is reported through exceptions. When any internal structure is affected, the SoC outputs the error report and informs the system ability to correct it. Thus, we defined that the rollback operations must be performed only when uncorrectable or external errors are detected. For example, this is the case for a double-bit upset in a register or a configuration memory corruption, respectively. It is worth mentioning that, this error classification is determined by the corrections enabled in the SoC, and despite that, reporting is always enabled.

Additional restart protection has been implemented. This allows restarting the system in case of critical errors. If the rollback depth is reached without any successful rollback, then a soft reset is performed, and the FPGA is reprogrammed. For our experiments, we have used a rollback depth equal to three, as in [18]. Since abnormal behaviors and hangs must also be managed, a system Watchdog Timer (WDT) is used. The WDT is refreshed inside the checkpoint and rollback operations. Its value has been set to allow proper operation with a margin to accommodate execution time variations, but the value is configurable and can be tuned according to the application requirements.

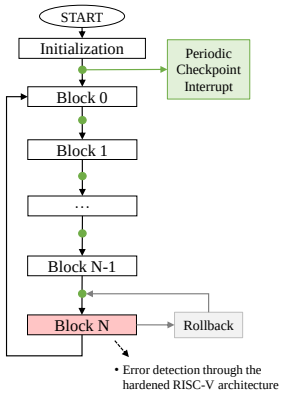


Fig. 1. Flow diagram of the software application.

In order to support this implementation in an SRAM-based FPGA, we extended HARV-SoC to include awareness of the errors in the configuration memory of the FPGA. For that, we included a new error handler that interfaces with a primitive structure provided by Xilinx. This primitive structure is called FRAME_ECCE2, and uses an interface to report the results from the built-in FPGA scrubbing, which reports one, two, and *n* bit errors and CRC checking errors. In Fig. 2, we present a block diagram that shows the newly added error handler and its interface with the primitive Xilinx interface. This extra reporting provides further insight into the errors observed at the architecture level for the software application, allowing the system to perform required actions according to those reports.

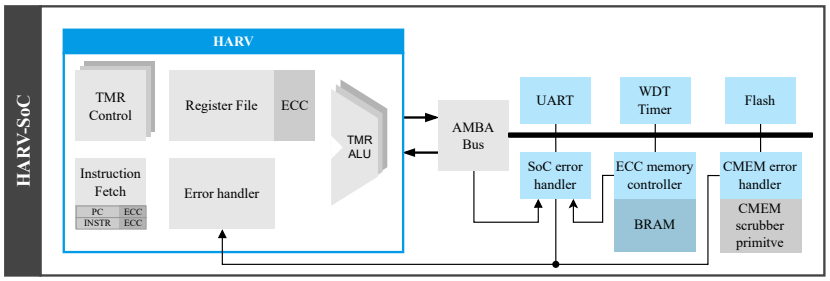


Fig. 2. Block diagram of the proposed hardened RISC-V with rollback capabilities.

B. Overheads

As previously described, the software recovery relies on the hardware observability provided by the HARV-SoC, using its fully hardened configuration and the built-in (hard-core) FPGA scrubber. Then, in comparison with the standard hardened HARV-SoC implementation, the proposed design required only an additional interface to allow the scrubber reporting registers to be accessible in the AMBA bus with negligible area overhead. Regarding the checkpoint memory, we used a 4KB block (in BRAM) with enough safety margins for the benchmark execution, allowing us to save the required three checkpoints. For the performance overhead, both the checkpoints and rollbacks degrade the execution time, the former having the most significant impact due to its periodicity, which the programmer defines, while the latter is rare. Our benchmark execution lasts approximately thirty seconds, performing five checkpoints during this period and adding a 2.9 % execution time overhead. Finally, each checkpoint and rollback operation take 171 ms and 223 ms, respectively.

Concerning the implementation of HARV-SoC, there is a resource usage overhead for the implementation of the techniques outlined in Subsection III-A and Fig. 2, including the hardening capabilities within HARV, error handlers, ECC controllers, and CMEM scrubber. The resource usage of the non-hardened version is of 3811 LUTs and 3065 FFs, with an estimated maximum operating frequency (Fmax) of 50.6 MHz. The hardened version, on the other hand, is implemented with 6888 LUTs and 3928 FFs, and has an estimated Fmax of 32.8 MHz. However, both versions are capable of running at 50 MHz due to HARV's multi-cycle characteristics.

C. Discussion

The presented implementation of hardware with a software recovery solution is a hybrid scheme for improving the reliability of a RISC-V SoC based on a COTS FPGA device. This scheme exploits the fault detection provided by the hardened architecture of the SoC (e.g., ECC in memories and registers, TMR in control and ALU, bus access timeouts for peripherals) to trigger software recovery using checkpoints and rollbacks without the higher cost penalty of the full duplication (or triplication) of the SoC and with enhanced reliability in comparison to software-only strategies. Notably, for this implementation, we extended the base HARV-SoC observation by integrating the built-in FPGA scrubber available to trigger software recovery. Despite being device-specific, the scrubber integration reduces potential functional failures and silent execution corruption.

The proposed strategy offers tailoring of base parameters for providing adequate coverage and effectiveness for specific applications. The developer should decide which data structures are included in the checkpoints. The base design considers saving the minimal context information (i.e., the software stack) but allows the addition of any critical variables and structures of interest. Besides that, the checkpoint depth could be increased to accommodate more rollback attempts. This decision directly impacts the memory size required for saving these checkpoints. Besides that, the developer should define the periodicity, which impacts the performance overhead and overall responsiveness of the software. For instance, in the implementation presented in this work, the performance and memory overhead had minimal impact on the benchmark execution and left margins for more aggressive usage of the checkpoints to achieve higher reliability with moderate overhead.

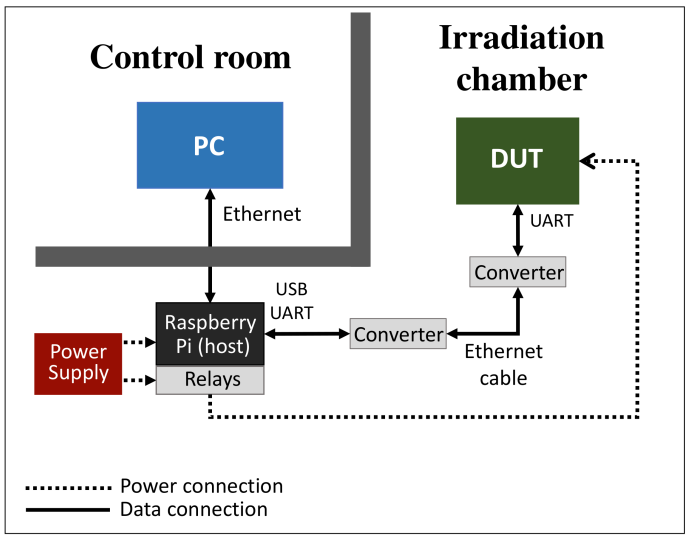
It is important to highlight that the proposed hybrid hardening strategy has a highly coupled hardware and software relation. Consequently, the implemented software is highly dependent on our target SoC, which could be a challenge for portability and flexibility for application development. We intend to demonstrate the feasibility, capabilities, and reliability of such systems, presenting alternative solutions for COTS utilization in applications with harsh environments.

V. Results

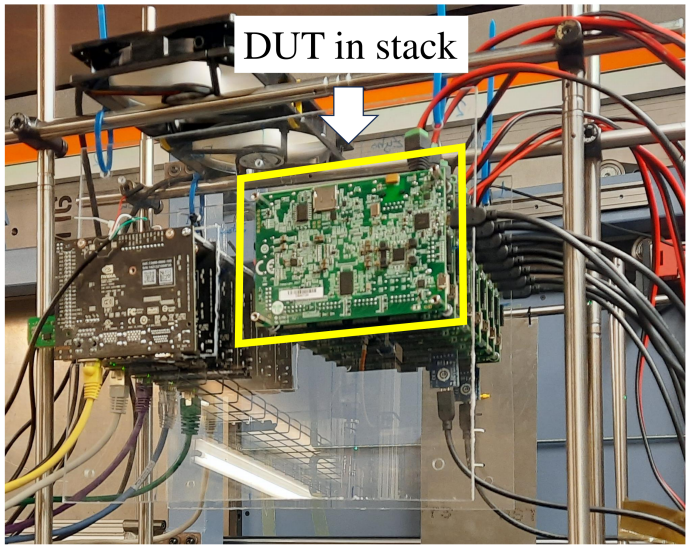
To evaluate the proposed system for radiation environments, we proceeded with an experimental approach, conducting an irradiation campaign in ChipIR beamline, part of the ISIS Neutron and Muon Source, at the Rutherford Appleton Laboratory, UK. The facility has a neutron spectrum representative of the atmospheric environment, generating up to 5×10^6 $cm^{-2}s^{-1}$ for energies with $E_n>10$ MeV [31]. In this work, the irradiation on the device reached an accumulated total fluence of 6.08×10^{10} n/cm^2 at the end of the test campaign. It is worth mentioning that, the reported accumulated fluence follows guideline recommendations provided in [32], [33] for reporting statistically accurate SEE results for neutron irradiation. In the following subsections, the details of the experimental setup (Subsection V-A) and the results of the test campaign (Subsections V-B, V-C, and V-D) are exposed.

A. Experimental Setup

The experimental setup consisted of a Zybo Z7 development board containing a Xilinx Zynq-7010 All Programmable System-On-Chip (APSoC) device, with a dual-core ARM Cortex-A9 microprocessor and an SRAM-based FPGA. To host our SoC, we used the SRAM-based FPGA contained in the Zynq-7010 device. The board is accessed through a serial interface for reporting results. In Fig. 3a, a diagram of the experimental setup is illustrated, while in Fig. 3b, the actual mounting in the ChipIR facility is shown. To optimize beam time utilization, multiple Zybo Z7 boards were arranged in a stack. One of the boards was used to assess the proposed approach, while the rest were utilized for other experiments.



(a)



(b)

Fig. 3. Experimental setup (a) diagram and (b) mounting at ChipIR facility.

Differently from the experiments performed in [17] with the SoC focusing on flash-based FPGAs, the usage of an SRAM-based FPGA imposes characteristics that are at the same time beneficial and concerning for the experiment, i.e., with higher device sensitivity, more events are generated, hence additional preparation is required for a proper test. Notably, the configuration memory required additional protection and a non-volatile memory for bitstream retention. Using an external flash available on the board, we prepared a procedure for self-reconfiguration of the bitstream from the flash to the configuration memory after a power cycle, improving the test flow and mitigating the impact of error accumulation. Also, the configuration memory itself was protected, using the FPGA's partial reconfiguration and scrubbing capabilities. Although, it is important to mention that the authors did not use the Soft Error Mitigation (SEM) core provided by the vendor, but implemented a customized solution using the device's internal blocks. This strategy was adopted to allow better integration and correlation between the device and the SoC error reporting, in which the SoC is aware of device errors and can actually trigger rollbacks depending on the hardening options.

In order to improve workload homogeneity, we used EEMBC's CoreMark™ [34] as the main stimuli for the processor execution. Designed for measuring the processors' performance in embedded applications, the benchmark is composed of four algorithms: list processing, matrix manipulation, state machines, and Cyclic Redundancy Check (CRC). The latest is used as a workload and provides a self-checking mechanism for the inner steps of the benchmark execution. CoreMark is split into several iterations and outputs a final summarizing score. It is not necessary to modify the CoreMark software application since context saving is performed through periodic interrupts. Thus, the proposed approach can be extended to any software application.

Although the user does not need to modify the application, the user must have some knowledge of the memory usage by the application to decide the amount of information to store at each checkpoint. The benchmark was executed throughout the experiment. A fully hardened design was used, in which all errors correctable by the SoC are handled and only the remaining errors trigger rollbacks, as a second layer of protection since it disturbs the execution flow.

B. Experimental Results: Hardware

The irradiation campaign was very effective in stressing the system and generating several radiation-induced faults. In total, the system was able to perform 162 runs, which are defined as the execution between external power cycles (independently of its case), i.e., from power on to power off. Each run may be composed of several inner executions split per watchdog or software restarts, as part of the recovery strategies, whereas the external power cycles are triggered only when the system stops responding.

In order to evaluate the system's endurance, we performed a Mean Fluence To Failure (MFTF) analysis. This result allows us to see the endurance of the system under neutron irradiation with a very high fluence, in which we can extrapolate the result to the targeted environments. For that, we created three error categories:

1) CoreMark error: when the benchmark result has errors, but the system continues operational.
2) Hangs: when the SoC stops responding (e.g., the processor in execution loop, serial interface issues) and the watchdog triggers a system restart.
3) FPGA failures: when a double bit upset in configuration memory is encountered or if a CRC error is encountered. They have a higher potential to affect the execution.

Subsequently, the failure category Processor Failure represents the joined results for Hang and Coremark Error, which are failures within the SoC; and All, combining the failures from the SoC and FPGA. Even though FPGA failures are not part of the SoC reliability, we included them to show their impact on the SoC. Table I presents the MFTF results.

TABLE I
MFTF SUMMARY BY FAILURE TYPE.

Failure Reason	MFTF [n/cm²]	Cross-section [cm²]
Coremark Error	1.46×10^{8}	1.37×10^{-10}
Hang	1.00×10^{9}	5.49×10^{-10}
FPGA failure	2.04×10^{8}	2.09×10^{-9}
Processor failure	8.35×10^{8}	6.86×10^{-10}
All	3.60×10^{8}	2.78×10^{-9}

Also, we considered two application fields with distinct radiation environments to provide specific estimations for Mean Time To Failure (MTTF) based on the target environments considering all failure types: 1) terrestrial, with reference to New York sea level, and 2) avionics, at a typical cruising altitude of 12 Km. According to [35], the neutron flux above 10 MeV for terrestrial applications (1) results in approximately 20 $n/cm^2/h$ and 300 times higher for avionics (2). We obtained the MTTF metric using the MFTF with the group *All* (i.e., including all the failure types) and the estimated fluence in the target environments. Therefore, for the first environment, the expected MTTF is about 7335 years. For the second, it is estimated in 24 years. These results give a good margin for the operation of most commercial systems.

Besides the failure analysis, we investigated radiation-induced events within the HARV-SoC in more detail, evidencing the base hardware hardening effectiveness and the remaining opportunities for the software recovery strategy.

C. Experimental Results: Software

Regarding the implemented recovery strategies, in Table II, we focus on the rollbacks and their effectiveness. The rollback types were defined as follows:

1) Effective: when the system resumes normal execution with subsequent checkpoints and could be at the first attempt or after two or three (which is the rollback depth) consecutive rollbacks.
2) Failed: when the rollback did not have an effect. The failed attempt category includes the rollbacks failed at first, second, and third attempt, as well as executions terminated before a single checkpoint was set and, consequently, a software reset was performed.

The results demonstrate the positive impact of the rollback operation in the hardening approach. The rollback was effective in 45.09% of cases considering all the cases when a rollback was needed, and the 44.51% was achieved in the first attempt (98.72% of the effective rollbacks). As can be observed, of the effective rollbacks, only 1.28% are due to cases where a second attempt was necessary, whereas, for the rollback depth (3 consecutive rollbacks), no effective rollback occurred. Therefore, an appropriate strategy considering these results would be to modify the rollback depth to 1. This way, unnecessary rollback attempts would be avoided, and the area and time overheads would be reduced. The rollback failed in 54.91% of the cases,

TABLE II
SOFTWARE RECOVERY ANALYSIS.

Rollback effectiveness		#Attempts	Distribution
Effective (45.09%)	1st rollback	77	98.72%
	2nd rollback	1	1.28%
	3rd rollback	0	-
Failure (54.91%)	1st attempt	41	43.16%
	2nd attempt	17	17.89%
	3rd attempt	21	22.11%
	Before checkpoint	16	16.84%

TABLE III
ERROR DETECTION, CORRECTION, AND PROPAGATION ANALYSIS USING HARV-SOC OBSERVABILITY.

HFTE type	#Detected	#HARV Corrected	#Effective Rollbacks	#Propagated Errors[1]	#Failure[2]	Failure XS[2] [cm²/device]
Undetectable CMEM error	195	0	33	150	3	4.94×10^{-11}
CMEM single-bit upset	74	0	20	59	4	6.58×10^{-11}
Load access fault	71	2	1	50	0	-
CMEM double-bit upset	36	36	n.a.[3]	33	0	-
Unknown/broken	31	0	28	16	0	-
ALU TMR error	13	13	n.a.[3]	0	0	-
Store access fault	11	3	0	9	1	1.65×10^{-11}
Memory single-bit upset	5	5	n.a.[3]	4	0	-
Memory stuck single-bit upset	1	1	n.a.[3]	0	0	-
Memory double-bit upset	1	0	0	1	0	-
Register file double-bit upset	1	0	1	0	0	-
Register file single-bit upset	1	1	n.a.[3]	1	0	-

[1]Error propagation classification is defined for errors that were not corrected by the SoC nor the rollback techniques, and caused an execution error; [2]Failures are defined as instances in which a power cycle is required or when the benchmark yields an error in the result; [3]When rollbacks are not triggered because the errors are handled by different methods (i.e. processor error correction, soft resets, or FPGA reset).

which implies that a software reset was needed. If the error occurs before having a checkpoint available, it is also inevitable to perform a software reset. Only in 9.25% of the total rollbacks, this was necessary (16.84% of the failed rollbacks). We should highlight that, although performing a software reset is a severe action, it is an effective method to recover the system. With these results, we could also corroborate the assumption that having multiple rollback levels could enhance recoverability, but more than the defined rollback depth, no significant improvements are obtained.

D. Experimental Results: Classification by HFTE

Despite the previously presented classifications, we also performed an analysis based on the detected errors by the HARV-SoC, presented in Table III. In these results, we present the values for each different detected HFTE (HARV Fault Tolerance Exception), which are the number of HFTEs detected (#Detected), followed by the number of those that were actually corrected by the HARV-SoC (#HARV Corrected), the number of HFTEs that were recovered through rollbacks (#Effective Rollbacks), the number of those that actually propagated to the execution (#Propagated Errors), and the amount that finally resulted in an execution failure, characterized by a power cycle requirement or Coremark error report (#Failure). Finally, we present the failure cross section for each different type of HFTE.

In these results, we noticed that the most common error (195 errors detected) was due to undetectable errors in the configuration memory (Undetectable CMEM error), which were reported to the application in real-time as an unknown HFTE. Out of those, there were 33 instances in which the rollbacks were effective and managed to reinstate the execution, but still, 150 led to error propagations, and 3 led to failures in the execution. Following this, single-bit upsets in the configuration memory (CMEM single-bit upset) were reported 74 times by the scrubber. When these errors appear, there is an attempt to rollback to ensure that the execution is not affected by the wrong configuration. When these errors happened, in only around 27% of the occurrences, the rollbacks were effective in restoring the execution, and 80% had its error propagated, leading to a failure in 4 instances. There were 36 occurrences of double-bit upset in the configuration memory, which are not correctable by the scrubber. All of these errors triggered an FPGA reset reconfiguration due to its uncorrectable nature, and because of that, it never led to failures.

Load access faults were detected 71 times, only corrected by HARV when they were due to a peripheral access timeout. When not correctable by HARV, a rollback was attempted, effective only once in the execution. Although the rollback could

not recover this error, it still did not lead to failures. For the other types, there were few that led to error propagations and even fewer that led to failures. The only failure was a store access fault in which rollbacks were attempted, but even with the effective rollback, it resulted in a Coremark error.

VI. Conclusions

In this work, we proposed and evaluated a hybrid approach for the radiation hardening of a RISC-V SoC by combining software and hardware strategies. A recovery mechanism based on checkpoint and rollback operations was implemented. To validate the proposal, an irradiation test campaign using atmospheric-like neutrons was performed. The results showed the effectiveness of the proposed approach, achieving a significant reduction in error propagation and presenting 45.09 % of effective rollbacks. Therefore, this work shows that application software could benefit from enhanced fault observability embedded in the SoC architecture, enabling the design of systems capable of reacting to radiation-induced effects and avoiding critical faults to propagate. Also, using a custom configuration memory protection strategy, we improved fault awareness at the SoC level, reduced FPGA resource utilization, and avoided the use of additional communication interfaces for reporting. In future work, we envision extending the recovery coverage by efficiently saving other aspects of the execution context and exploiting opportunities in other software domains, such as operating systems.

References

[1] M. Vozoff and J. Couluris, "SpaceX products-advancing the use of space," in *AIAA SPACE 2008 Conf. Expo.*, San Diego, CA, USA, Sept. 2008, Art. no. 7836.

[2] D. Rutishauser, R. Ramadorai, J. Prothro, T. Fleming, and P. Fidelman, "NASA and Blue Origin Collaborative Assessment of Precision Landing Algorithms and Computing," in *AIAA Scitech 2021 Forum*, Jan. 2021, Art. no. 0377.

[3] J. C. McDowell, "The Low Earth Orbit Satellite Population and Impacts of the SpaceX Starlink Constellation," *The Astrophysical J. Lett.*, vol. 892, no. 2:L36, Apr 2020.

[4] V. Verma, F. Hartman, A. Rankin, M. Maimone, T. Del Sesto, O. Toupet, E. Graser, S. Myint, K. Davis, D. Klein, J. Koch, S. Brooks, P. Bailey, H. Justice, M. Dolci, and H. Ono, "First 210 solar days of Mars 2020 Perseverance Robotic Operations - Mobility, Robotic Arm, Sampling, and Helicopter," in *Proc. IEEE Aerosp. Conf. (AERO)*, Big Sky, MT, USA, Mar. 2022, pp. 156–169.

[5] H. F. Grip, J. Lam, D. S. Bayard, D. T. Conway, G. Singh, R. Brockers, J. H. Delaune, L. H. Matthies, C. Malpica, T. L. Brown, A. Jain, A. M. S. Martin, and G. B. Merewether, "Flight control system for nasa's mars helicopter," in *AIAA Scitech 2019 Forum*, San Diego, CA, USA, Jan. 2019, Art. no. 1289.

[6] W. Riedler, K. Torkar, H. Jeszenszky, J. Romstedt, H. S. C. Alleyne, H. Arends, W. Barth, J. Biezen, B. Butler, P. Ehrenfreund *et al.*, "MIDAS–the micro-imaging dust analysis system for the Rosetta mission," *Space Sci. Rev.*, vol. 128, pp. 869–904, Feb. 2007.

[7] T. Estlin, D. Gaines, B. Bornstein, S. Schaffer, V. Verma, R. C. Anderson, M. Burl, S. Chu, D. Blaney, L. De Flores *et al.*, "Developing autonomous science technology for the MSL rover mission," 2015. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/estlin-ijcai2015-autonomous.pdf

[8] A. Waterman and K. Asanovic, "The RISC-V Instruction Set Manual," *Volume I: User-Level ISA*, May 2017, version 2.2. [Online]. Available: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

[9] S. Di Mascio, A. Menicucci, E. Gill, G. Furano, and C. Monteleone, "Leveraging the Openness and Modularity of RISC-V in Space," *J. Aerosp. Inf. Syst.*, vol. 16, no. 11, pp. 454–472, Aug. 2019.

[10] ESA, "Trisat-R Nanosatellite," 2022. [Online]. Available: https://www.eoportal.org/satellite-missions/trisat-r#eop-quick-facts-section

[11] F. Gaisler, "GRLIB IP Core User's Manual Version," 2023, version 2023.2. [Online]. Available: https://www.gaisler.com/products/grlib/grip.pdf

[12] P. D. Schiavone *et al.*, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *Proc. 27th Int. Symp. Power Timing Model. Optim. Simul. (PATMOS)*, Thessaloniki, Greece, Sept. 2017, pp. 185–192.

[13] S. Nolting, U. Martinez-Corral, zipotron, L. Asplund, H. B. Andersen, T. Meissner, A. Wenzel, A. Charles, Gideon, J. Herbert, T. Ramsland, lab-mathias claussen, R. Corsi, H. Guzmán-Miranda, J. Pfau, M. Ludwig, Mario, P. Cotret, P. Dwivedi, T. G. Badger, M. Fischer, and gottschalkm, "stnolting/neorv32: v1.7.2," Jun. 2022.

[14] N.-J. Wessman, F. Malatesta, J. Andersson, P. Gomez, M. Masmano, V. Nicolau, J. L. Rhun, G. Cabo, F. Bas, R. Lorenzo, O. Sala, D. Trilla, and J. Abella, "De-RISC: the First RISC-V Space-Grade Platform for Safety-Critical Systems," in *Proc. IEEE Space Comput. Conf. (SCC)*, Laurel, MD, USA, Aug. 2021, pp. 17–26.

[15] A. E. Wilson and M. Wirthlin, "Neutron Radiation Testing of Fault Tolerant RISC-V Soft Processor on Xilinx SRAM-based FPGAs," in *Proc. IEEE Space Comput. Conf. (SCC)*, Pasadena, CA, USA, Aug. 2019, pp. 25–32.

[16] D. A. Santos, L. M. Luza, L. Dilillo, C. A. Zeferino, and D. R. Melo, "Reliability analysis of a fault-tolerant RISC-V system-on-chip," *Microelectron. Rel.*, vol. 125, Art. no. 114346, Oct. 2021.

[17] D. A. Santos, A. M. P. Mattos, L. M. Luza, C. Cazzaniga, M. Kastriotou, D. R. Melo, and L. Dilillo, "Neutron Irradiation Testing and Analysis of a Fault-Tolerant RISC-V System-on-Chip," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Austin, TX, USA, Oct. 2022, pp. 31–36.

[18] P. M. Aviles, A. Lindoso, J. A. Belloch, M. Garcia-Valderas, Y. Morilla, and L. Entrena, "Radiation Testing of a Multiprocessor Macrosynchronized Lockstep Architecture With FreeRTOS," *IEEE Trans. Nucl. Sci.*, vol. 69, no. 3, pp. 462–469, Mar. 2022.

[19] H.-M. Pham, S. Pillement, and S. J. Piestrak, "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1179–1192, Jun. 2013.

[20] Á. B. de Oliveira, L. A. Tambara, and F. L. Kastensmidt, "Applying lockstep in dual-core ARM Cortex-A9 to mitigate radiation-induced soft errors," in *Proc. 8th IEEE Latin Amer. Symp. Circuits Syst. (LASCAS)*. Bariloche, Argentina: IEEE, 2017, pp. 93–96.

[21] I. Marques, C. Rodrigues, A. Tavares, S. Pinto, and T. Gomes, "Lock-V: A heterogeneous fault tolerance architecture based on ARM and RISC-V," *Microelectron. Rel.*, vol. 120, Art. no. 114120, May 2021.

[22] S. Kasap, E. W. Wächter, X. Zhai, S. Ehsan, and K. D. McDonald-Maier, "Novel lockstep-based fault mitigation approach for SoCs with roll-back and roll-forward recovery," *Microelectron. Rel.*, vol. 124, Art. no. 114297, Sept. 2021.

[23] M. Hijorth, M. Aberg, N.-J. Wessman, J. Andersson, R. Chevallier, R. Forsyth, R. Weigand, and L. Fossati, "GR740: Rad-Hard Quad-Core LEON4FT System-on-Chip," *DASIA 2015-Data Syst. Aerosp.*, 2015.

[24] C. Hulme, H. Loomis, A. Ross, and R. Yuan, "Configurable fault-tolerant processor (CFTP) for spacecraft onboard processing," in *Proc. IEEE Conf. Aerosp.*, vol. 4, Big Sky, MT, USA, Mar. 2004, pp. 2269–2276.

[25] M. J. Wirthlin, A. M. Keller, C. McCloskey, P. Ridd, D. Lee, and J. Draper, "SEU mitigation and validation of the LEON3 soft processor using triple modular redundancy for space processing," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2016, pp. 205–214.

[26] R. C. Goerl, P. R. Villa, L. B. Poehls, E. A. Bezerra, and F. L. Vargas, "An efficient EDAC approach for handling multiple bit upsets in memory array," *Microelectron. Rel.*, vol. 88, pp. 214–218, Sept. 2018.
[27] A. Ramos, A. Ullah, P. Reviriego, and J. A. Maestro, "Efficient Protection of the Register File in Soft-Processors Implemented on Xilinx FPGAs," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 299–304, Feb. 2018.
[28] A. Ramos, R. G. Toral, P. Reviriego, and J. A. Maestro, "An ALU Protection Methodology for Soft Processors on SRAM-Based FPGAs," *IEEE Trans. Comput.*, vol. 68, no. 9, pp. 1404–1410, Sept. 2019.
[29] L. A. Aranda, N.-J. Wessman, L. Santos, A. Sánchez-Macián, J. Andersson, R. Weigand, and J. A. Maestro, "Analysis of the Critical Bits of a RISC-V Processor Implemented in an SRAM-Based FPGA for Space Applications," *Electronics*, vol. 9, no. 1, Art. no. 175, Jan. 2020.
[30] D. A. Santos, A. M. P. Mattos, D. R. Melo, and L. Dilillo, "Enhancing Fault Awareness and Reliability of a Fault-Tolerant RISC-V System-on-Chip," *Electronics*, vol. 12, no. 12, Art. no. 2557, Jun. 2023.
[31] 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 Trans. Emerg. Topics Comput.*, vol. 9, no. 1, pp. 104–108, Jan.-Mar. 2021.
[32] JEDEC Standard, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," JEDEC Solid State Technol. Assoc., Sept. 2021, JESD89B (Revision of JESD89A, October 2006). [Online]. Available: https://www.jedec.org/system/files/docs/JESD89B.pdf
[33] ESCC, "Single Event Effects Test Method and Guidelines," European Space Components Coordination, Oct. 2014, ESCC Basic Specification No. 25100, Issue 2. [Online]. Available: https://escies.org/escc-specs/published/25100.pdf
[34] S. Gal-On and M. Levy, *Exploring CoreMark a benchmark maximizing simplicity and efficacy*, Embedded Microprocessor Benchmark Consortium, 2012.
[35] IEC, "Process Management for Avionics—Atmospheric Radiation Effects. Accommodation of Atmospheric Radiation Effects via Single Event Effects within Avionics Electronic Equipment," Int. Electrotechnical Commission, Jan. 2016, IEC 62396-1:2016. [Online]. Available: https://webstore.iec.ch/publication/24053